COLORIZATION OF BLACK AND WHITE IMAGES USING DEEP LEARNING

Submitted in partial fulfillment of the requirements for the award of

Bachelor of Engineering degree in Computer Science and Engineering

By

MUTTA SATHVIK(Reg.No - 39110657)
MAMILLAPALLI RVV SATYANARAYANA MURTHY(Reg.No – 39110593)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade “A” by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119

APRIL-2023

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

<u>BONAFIDE CERTIFICATE</u>

This is to certify that this Project Report is the bonafide work of **MUTTA SATHVIK(Reg.No - 39110657) and MAMILLAPALLI RVV SATYANARAYANA MURTHY(Reg.No – 39110593)** who carried out the Project Phase-2 entitled **"COLORIZATION OF BLACK AND WHITE IMAGES USING DEEP LEARNING"** under my supervision from Jan 2023 to April 2023.

[illegible]

Internal Guide
Dr. D. USHA NANDINI , M.E., Ph.D

[illegible]

Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph.D

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF CSE

Submitted for Viva voce Examination held on 20.04.2023

[illegible]

Internal Examiner

[illegible]

External Examiner

DECLARATION

I, **MUTTA SATHVIK(Reg.No - 39110657)** here bydeclare that the Project Phase-2 Report entitled “**COLORIZATION OF BLACK AND WHITE IMAGES USING DEEP LEARNING”** done by me under the guidance of **Dr. D. Usha Nandini , M.E., Ph.D.,** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

M. Sathvik

DATE: 20.04.2023
PLACE: Chennai **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

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

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

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. D. Usha Nandini ,M.E., Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase- 2 project work.

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

ABSTARCT

Image colorization has become a popular technique in recent years due to its ability to convert old black and white images into colored ones. This process is especially relevant for historical images, as it enables us to visualize and experience the past with more accuracy and realism. In this extended abstract, we will discuss the methods, challenges, and applications of image colorization, with a focus on old black and white images.

There are two main methods of image colorization: manual colorization and automatic colorization. Manual colorization involves the use of specialized software tools, such as Adobe Photoshop or GIMP, to manually add color to a black and white image. This method requires a great deal of skill and experience, as the color choices must be accurate and consistent with the period and context of the image.

Automatic colorization, on the other hand, is a more recent development in the field of image colorization. This method uses deep learning algorithms, such as convolutional neural networks, to analyze the grayscale values of an image and predict the corresponding colors. This method is highly efficient and can produce accurate and consistent results with minimal human intervention. However, automatic colorization also has limitations, as it may not always produce accurate colors or may introduce artifacts into the image.

One of the main challenges of image colorization is the subjectivity of color choices. The colors used for an image may differ depending on the context, time period, and personal preferences of the colorizer. This subjectivity can lead to inconsistencies and inaccuracies in the colorization process, especially in historical images.

Preserving the original integrity of the image is another challenge of image colorization. While colorization can enhance the visual appeal of an image, it can also alter the original meaning and intent of the image. Therefore, it is essential to ensure that the colorization process does not detract from the historical or artistic significance of the original image.

Image colorization has various applications in fields such as art, entertainment, and historical documentation. In the art world, colorization can be used to create new interpretations of old works or to add a contemporary touch to classic pieces. In the entertainment industry, colorization can be used to enhance the visual appeal of movies and TV shows, making them more appealing to modern audiences. In historical documentation, colorization can provide a new perspective on past events and figures, helping us to understand and appreciate the past in a new way.

In conclusion, image colorization has become a popular technique for converting old black and white images into colored ones. This technique has various applications in fields such as art, entertainment, and historical documentation. While manual colorization has been used for decades, automatic colorization using deep learning algorithms has recently emerged as a highly efficient and accurate method. However, the subjectivity of color choices and the need to preserve the original integrity of the image remain significant challenges in the colorization process. Despite these challenges, image colorization has the potential to revolutionize the way we visualize and experience history, art, and entertainment.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ABBREVIATIONS	EXPANSION
CNN	CONVOLUTIONAL NEURAL NETWOR
GAN	GENERATIVE ADVERSARIAL NETWO
SGD	STOCHASTIC GRADIENT DESCENT
TPU	TENSOR PROCESSING UNITS

CHAPTER 1
INTRODUCTIO
N

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 incomplete 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. In art, image colorization can be used to create new interpretations of historical paintings or to add color to black and white sketches. In medical imaging, image colorization can be used to enhance the visualization of medical data, aiding researchers in their understanding of medical conditions and treatments.

Despite its potential benefits, image colorization using neural networks also has its challenges. One of the biggest challenges is the potential for the network to introduce artifacts or inconsistencies into the image, resulting in inaccurate or unrealistic colorizations. Additionally, the quality of the dataset used to train the network can have a significant impact on the accuracy of the colorization.

In summary, image colorization using neural networks is a promising technology that has the potential to enhance the visual experience of historical images and improve our understanding of medical data. With continued advancements in neural network technology, we can expect to see further improvements in the accuracy and realism of image colorization in the future.

CHAPTER 2
LITERATURE
SURVEY

Satoshi et al. [1] In this paper he joined two networks, one to anticipate the worldwide highlights of the information Image and the other to have some expertise in nearby elements of information Images. The worldwide elements network is prepared for Image grouping and straightforwardly connected to the neighbourhood highlights network which are then prepared for colorization of Images utilizing L2 Euclidean misfortune capability.

Richard et al [2] In this paper he presented a streamlined arrangement by taking a tremendous informational index and single feed-forward pass in CNN. They utilized a custom multinomial cross entropy misfortune with class rebalancing and by involving people as subjects they had the option to trick 32% of them by their outcomes. They utilized earlier variety conveyance got from the preparation set to foresee a circulation for each result pixel.

Baldassarre F et al. [3] In this paper he made a network model that joins a profound CNN engineering, that is prepared without any preparation, with a pre-prepared model Origin Resnetv2 for significant level component extraction. They train this network on a little subset of 60,000 Images from ImageNet. This engineering is like that utilized by Iizuka et al. [1] and it likewise utilizes Euclidean (L2) misfortune capability.

Jeff Hwang et al[4] has profound convolutional neural network structures utilized are acquired from the VGG16 network. They executed two models: one as a relapse model and other as a grouping model. They utilize the CIE LUV colorspace for information and result. They acted it like a characterization task that can deliver colorized Images which are far superior to those produced by a relapse-based model.

David Futschik et al [5] In this paper he utilized a few variations of CNNs and looked at their exhibition on the information, utilizing two distinct NN structures; one conventional, plain CNN, and other being motivated by remaining CNNs, which had not been utilized for colorization beforehand. In spite of the more modest less boundaries, this model had the option to produce results that outperform plain CNN in speculation to concealed/test information.

Alex Avery et al[6], programmed Image colorization with two distinct CNN models is proposed. They train a grouping and relapse model on CIFAR-10 dataset utilizing Lab colorspace. They train the order model without any preparation and furthermore by move gaining from a pretrained VGG16 network. They additionally utilize the Strengthened mean method with the model to plan expectation appropriation to single result forecast and demonstrate the way that it can deliver dynamic and spatially more predictable outcomes.

Richard Zhang et al [7] proposed an upgraded arrangement by utilizing huge dataset and single feed-forward pass in Convolutional Neural Network. Their significant thought process lies on preparing part. Human subjects were utilized to test the result and had the option to trick 32% of them and had different number of neurons. The many endeavours utilized different structures.

Domonkos Varga et al [10] proposed the objective of mechanized shading of animation Images, since they are particular from normal Images, they dealt with issues as their varieties depend on illustrator to artist. The informational index was particularly prepared for lakhs of animation Images, 30% of which were utilized in documentation and rest for preparing. However, definitely, the variety unconventionality in kid's shows is higher than in normal Images and thought is customized and moderate.

Shweta Balm et al [11] recommended another comparable viewpoint, using the Google's Image classifier, Origin ResNet V2. The framework model is partitioned into 4 sections which are Encoder, Component extractor, Combination layer and Decoder. The framework is equipped for creating agreeable outcomes. given satisfactory assets like computer chip, Memory, and enormous informational index. This is by and large evidence of idea execution.

Yu Chen et al [12] recommended a way to deal with fundamentally tackled the issue of shading Chinese films from bygone era. For tuning the general model, they utilized existing dataset with their dataset of Chinese Images. The network utilizes multiscale convolution portions and consolidates low and center elements which are extricated from VGG-16.

V.K. Putri et al [13] has proposed a strategy to change conventional portrayals into beautiful Images. It utilizes variety forecast in CIELab variety space and sketch reversal model. This point of view is fit for taking care of hand-drawn portrays as well as different mathematical changes. The disadvantages found was that, dataset is limited however it performs well for uncontrolled circumstances. In couple of papers, number of neurons is same as the component of the element descriptor separated from every pixel facilitates in a grey scale .

2.1 INFERENCES FROM LITERATURE SURVEY

In this venture we propose a technique for Image colorization. We preprocess hued Images to make grayscale Images to use as the contribution for the model. Our model is then prepared with these grayscale Images as info and the first shaded Images as the result. In this way, in the event that we feed another concealed grayscale Image to the model, it would have the option to create a RGB Image with a sensible comprehension of the spatial relationship of variety with the intrinsic surface. Taking into account the pixel tone is exceptionally dependent on the elements of its neighboring pixels, utilization of CNN is a palatable choice for Image colorization. The state of having just a grayscale or highly contrasting Image, it is muddled to distinguish the specific tone. The data isn't enough for an network to assess the pixel tones. For example, consider a vehicle Image which is in dark structure, there are number of OK choices for vehicle tone. To figure a reasonable

Variety, we require more data to concentrate on the model to match a grayscale input Image to the same shade of the result Image. In the beyond couple of years, Convolutional neural network is one of the best learning-based models. CNN checked fabulous abilities in Image handling. In such way, CNN-based model is proposed by us for programmed Image colorization. By utilizing Convolutional Neural Networks, we chose to snare the issues of Image colorization to "daydream" what an info dark and white Image would show up after colorization. For preparing the network began with the ImageNet dataset and all Images were changed from the RGB variety space to the Lab variety space. Like the RGB variety space, the Lab variety space has three channels. However, not at all like the RGB variety space, Lab encodes variety data in an unexpected way.

Image colorization is the method involved with taking an information grayscale (high contrast) Image and afterward creating a result colorized Image that addresses the semantic varieties and tones of the information. we will use here today rather depends on profound learning. We will use a Convolutional Neural Network equipped for colorizing highly contrasting Images with results RGB variety space, the Lab variety space has three channels. The L channel encodes softness power just, a channel encodes green-red and the b channel encodes blue-yellow.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

Although significant progress has been made in the field of automatic colorization of grayscale images using deep learning techniques, there are still several open problems that need to be addressed.

One of the major issues is the lack of standard evaluation metrics for comparing the performance of different colorization methods. Existing evaluation metrics, such as PSNR and SSIM, are not always reliable indicators of the perceptual quality of the colorized images. Developing new metrics that better align with human perception is a key research challenge.

Another challenge is the lack of large-scale annotated datasets of grayscale images and their corresponding colorized versions. Although some datasets, such as ImageNet and CIFAR-10, have been used for colorization tasks, they are not specifically designed for this purpose and may not fully capture the variability of natural images. Developing new datasets with a diverse range of grayscale images and their corresponding ground truth colorizations would be beneficial for training and evaluating colorization models.

Furthermore, current methods tend to produce colorized images that are oversaturated or have unrealistic color distributions, especially when applied to complex or ambiguous scenes. Improving the realism of colorization outputs, while maintaining consistency with the input grayscale image, is another open research problem.

Finally, most existing colorization methods focus on global colorization, where the same colorization is applied to the entire image. However, some images may contain regions with distinct color semantics, such as human faces or natural landscapes. Developing methods that can handle local colorization and adapt to the semantic content of the image would be an important step towards more realistic and accurate colorization.

CHAPTER 3
REQUIREMENTS ANALYSIS

3.1 FEASIBILITY STUDIES OF THE PROJECT

Before undertaking the project of image colorization using an autoencoder, we conducted a feasibility study to assess the technical and economic viability of the project. We also performed a risk analysis to identify potential challenges and risks associated with the project.

Based on our initial research, we determined that image colorization using an autoencoder was technically feasible. The use of CNN-based autoencoders has been shown to be effective for a variety of image processing tasks, including image colorization. We also identified several open-source libraries and tools that could be used for implementing our approach, such as TensorFlow and Keras.

We also assessed the economic feasibility of the project by considering the cost of hardware, software, and human resources. We determined that the cost of hardware and software was within our budget, and that we had access to the necessary computing resources for training the neural network. We also estimated the amount of time required for the project and determined that it was feasible to complete within the given timeframe.

We identified several potential risks associated with the project, including the following:

The quality of the dataset could impact the performance of the neural network. To mitigate this risk, we carefully curated and preprocessed the dataset to ensure that it was of high quality.

There was a risk of overfitting the neural network to the training data, which could lead to poor generalization on new data. To mitigate this risk, we used regularization techniques, such as dropout, and monitored the performance of the network on the validation set.

There was a risk of hardware failures, such as power outages or hardware malfunctions, which could interrupt the training process. To mitigate this risk, we regularly saved checkpoints of the neural network during training and used cloud computing resources with built-in redundancy.

Time constraints: There was a risk of not completing the project within the given timeframe. To mitigate this risk, we carefully planned the project timeline and set realistic goals for each stage of the project.

Overall, the feasibility study and risk analysis helped us to identify potential challenges and risks associated with the project and to develop strategies for mitigating them.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

Table 3.1: Software Requirements

SOFTWARE	**USED**
Operating system	windows 7+
FrontEnd	HTML, CSS and JS
Libraries used	Flask, OS, CV2, SKimage, Keras, NumPy
Ide	Visual Studio

Operating system :

Windows operating system can be used for image colorization using autoencoder as long as it meets the minimum hardware and software requirements. The specific version of Windows may not be critical as long as it is compatible with the required software tools and libraries.

For example, to use Python for image colorization, the system needs to have Python 3 installed along with necessary libraries such as NumPy, OpenCV, Scikit-image, Keras, and Flask. These libraries can be installed using package managers such as pip. Additionally, hardware specifications such as RAM and CPU/GPU performance should also meet the requirements for the size and complexity of the image data being processed.

Windows is a widely used operating system and can be effectively used for image colorization projects with proper installation and configuration of required software and hardware components.

FrontEnd:

HTML, CSS, and JavaScript are three of the primary technologies used in building websites and web applications. Here is a brief overview of each:

HTML (Hypertext Markup Language) is used to structure and format content on the web. It provides the basic building blocks for a web page, such as headings, paragraphs, lists, images, and links. HTML is a markup language, which means it uses tags to define the structure and content of a web page.

CSS (Cascading Style Sheets) is used to style and layout web pages. It allows web developers to control the appearance of text, images, and other elements on a web page.

CSS is used to specify the color, size, font, and position of elements on a web page. By separating the presentation of a web page from its content, CSS allows for greater flexibility and consistency in the design of a website.

JavaScript is a programming language used to create interactive and dynamic websites. It is used to add interactivity to a web page, such as pop-up messages, drop-down menus, and animations. JavaScript can also be used to perform calculations, validate forms, and manipulate the content of a web page in real-time.

Together, HTML, CSS, and JavaScript form the foundation of modern web development

.***Ide :***

Visual Studio is an Integrated Development Environment(IDE) developed by Microsoft to develop GUI(Graphical User Interface), console, Web applications, web apps, mobile apps, cloud, and web services, etc. With the help of this IDE, yo u can create managed code as well as native code. It uses the various platforms of Microsoft software development software like Windows store, Microsoft Silverlight, and Windows API, etc. It is not a language-specific IDE as you can use this to write

code in VB(Visual Basic), Python, JavaScript, and many more languages. It provides support for 36 different programming languages. It is available for Windows as well as for macOS.

VS Code (short for Visual Studio Code) is a free and open-source code editor developed by Microsoft. It provides a modern and powerful environment for coding, debugging, and testing software. VS Code supports various programming languages, including C++, Java, Python, and JavaScript. It includes a range of features, such as syntax highlighting, code completion, Git integration, debugging tools, and extensions. It is also highly customizable and provides users with the ability to install extensions and themes to personalize their coding experience. VS Code is available for Windows, macOS, and Linux operating systems. It has gained immense popularity among developers due to its ease of use, flexibility, and extensive community support.

Libraries used :

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.

One of the key features of NumPy is its ability to perform broadcasting, which allows operations to be performed on arrays of different shapes and sizes. This makes it easy to perform complex calculations and manipulate large datasets with ease. NumPy also includes a wide range of functions for linear algebra, Fourier transforms, and statistical analysis.

SKimage: Scikit-image, also known as skimage, is an open-source image processing library for the Python programming language. It provides a collection of algorithms and functions for image processing, including filters, segmentation, feature extraction, and transformation. The library is built on top of NumPy, another popular Python library for scientific computing.

Scikit-image is designed to be easy to use and provides a user-friendly interface for image processing tasks. It also supports a wide range of image formats, making it compatible with many existing image processing pipelines. Additionally, the library is actively maintained and has a growing community of contributors, ensuring its continued development and improvement.

CV2: cv2 is a popular computer vision library for Python, which is used to perform various operations on images and videos. It is an open-source library and provides a rich set of functions and tools for image and video analysis, manipulation, and processing.

cv2 is built on top of the OpenCV (Open Source Computer Vision) library, which is a C++ library for computer vision and machine learning. cv2 provides a Python interface to OpenCV, making it easier to use OpenCV in Python programs.

Some of the common operations that can be performed using cv2 include reading and writing images and videos, resizing and cropping images, applying filters and transformations, object detection and recognition, and motion analysis.

cv2 is widely used in various fields such as robotics, autonomous vehicles, surveillance systems, and healthcare. It is also popular in computer vision research and education due to its ease of use and comprehensive documentation.

OS: The os module is a Python built-in module that provides a way of interacting with the underlying operating system. It allows you to perform various operations such as creating and deleting files and directories, navigating the file system, accessing environment variables, and executing system commands.

Some of the commonly used functions in the os module include:

- os.getcwd(): returns the current working directory
- os.listdir(path): returns a list of files and directories in the given path
- os.mkdir(path): creates a new directory with the specified path
- os.rmdir(path): removes an empty directory with the specified path
- os.path.join(path, *paths): joins one or more path components intelligently
- os.path.exists(path): checks if the specified path exists
- os.remove(path): removes a file with the specified path

The os module is widely used in various applications including file handling, system administration, and automation.

Keras: Keras is a high-level neural network application programming interface (API) that is written in Python. It is an open-source library for building and training deep learning models. Keras provides a user-friendly interface for creating and training deep neural networks, making it easier for beginners and experts alike to develop machine learning models.

Keras has gained popularity due to its ease of use, modularity, and flexibility. It supports a wide range of neural network architectures and can be run on top of TensorFlow, CNTK, or Theano. Keras also provides a number of pre-trained models for various computer vision and natural language processing tasks, which can be fine-tuned for specific applications.

One of the main advantages of Keras is its simplicity. With just a few lines of code, you can create a neural network with multiple layers, train it on your data, and evaluate its performance. Keras also includes a number of utilities for data preprocessing and augmentation, making it easier to prepare your data for training.

Flask: Flask is a lightweight and popular web framework in Python used for developing web applications. It is a microframework that does not require any particular tools or libraries to get started. Flask offers several features, including URL routing, templating, session management, and more, making it a versatile choice for web development.

One of the primary benefits of Flask is its simplicity, allowing developers to create web applications quickly and efficiently. It also offers flexibility, enabling developers to create web applications with varying degrees of complexity. Flask supports various extensions that can be added to the framework to enhance its functionality, such as Flask-RESTful for creating RESTful APIs, Flask-WTF for form handling, and Flask-SQLAlchemy for database integration.

3.3 SYSTEM USECASE

The user has access to the Image Colorization System. The user has selected a black and white image to colorize.

The colorized image is displayed to the user. The user can save or download the colorized image. The user selects the black and white image to colorize from the system. The system displays the selected image to the user. The user selects the colorization options, such as color palette or theme. The system processes the image and generates a colorized version. The colorized image is displayed to the user. The user can save or download the colorized image.

Invalid input: If the image selected by the user is not in the supported format or is corrupt, the system displays an error message and prompts the user to select a valid image.

If the system resources, such as memory or processing power, are insufficient to colorize the image, the system displays an error message and prompts the user to try again later.

If the user cancels the colorization process at any point, the system cancels the operation and returns to the initial state.

This use case describes the process of a user colorizing a black and white image using the Image Colorization System. It outlines the steps involved in selecting the image, choosing the colorization options, and displaying the colorized image. It also includes alternate scenarios that may occur, such as invalid input or insufficient system resources. This use case can be used as a basis for developing the system functionality and designing the user interface.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

4.1 SELECTED METHODOLOGY

Previous approaches to black and white image colorization relied on manual human annotation and often produced desaturated results that were not “believable” as true colorizations.

Zhang et al. decided to attack the problem of image colorization by using Convolutional Neural Networks to “hallucinate” what an input grayscale image would look like when colorized.

To train the network Zhang et al. started with the ImageNet dataset and converted all images from the RGB color space to the Lab color space.

Similar to the RGB color space, the Lab color space has three channels. But unlike the RGB color space, Lab encodes color information differently:

- The L channel encodes lightness intensity only
- The a channel encodes green-red.
- And the b channel encodes blue-yellow

A full review of the Lab color space is outside the scope of this but the gist here is that Lab does a better job representing how humans see color.

Since the L channel encodes only the intensity, we can use the L channel as our grayscale input to the network

A full review of the Lab color space is outside the scope of this but the gist here is that Lab does a better job representing how humans see color.

Since the L channel encodes only the intensity, we can use the L channel as our grayscale input to the network

From there the network must learn to predict the a and b channels. Given the input L channel and the predicted ab channels we can then form our final output image.

The entire (simplified) process can be summarized as:

1. Convert all training images from the RGB color space to the Lab color space.
2. Use the L channel as the input to the network and train the network to predict the ab channels.
3. Combine the input L channel with the predicted ab channels.
4. Convert the Lab image back to RGB.

To produce more plausible black and white image colorizations the authors also utilize a few additional techniques including mean annealing and a specialized loss function for color rebalancing (both of which are outside the scope of this post).

Now we are also using this technique to solve our problem and get the best solution using the LAB method.

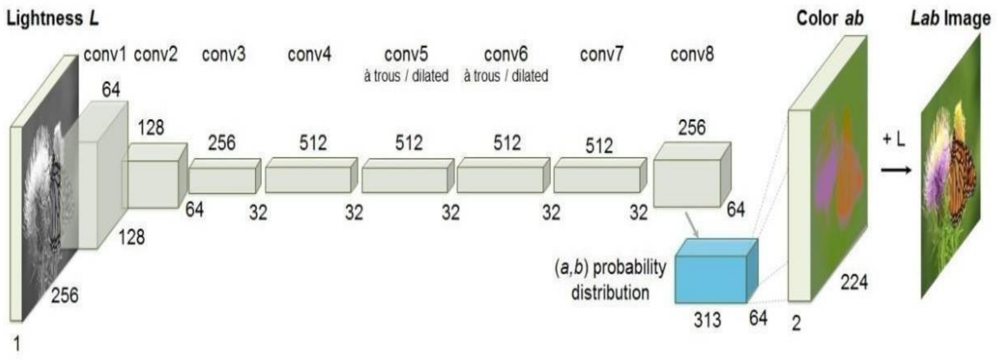


Fig. 4.1: Process of LAB

Colorizing lab images using deep learning is a popular application of computer vision. The process involves collecting a set of black and white lab images and corresponding color images, preprocessing the data, selecting a deep learning model, training the model, and evaluating the model's performance.

To get started, you need to collect a dataset of color images and convert it in to black and white lab images and their corresponding color images. You can use publicly available datasets such as CIFAR-10 or ImageNet, or create your own dataset by converting color images to black and white.

After collecting your data, you need to preprocess it by resizing the images to a common size, normalizing the pixel values, and splitting your data into training, validation, and testing sets.

Next, you need to select a deep learning model to use for colorizing your lab images. There are several models you can choose from, such as Convolutional Neural Networks (CNNs) or Generative Adversarial Networks (GANs).

Once you have selected your model, you can train it on your preprocessed data using techniques such as stochastic gradient descent (SGD) or Adam optimization. You can also use data augmentation techniques to increase the size of your training set and improve your model's performance.

Finally, you need to evaluate your model's performance on a separate test set of black and white images. You can use metrics such as Mean Squared Error (MSE) or Peak Signal-to-Noise Ratio (PSNR) to evaluate the quality of your colorized images.

4.2 ARCHITECTURE OF PROPOSED SYSTEM

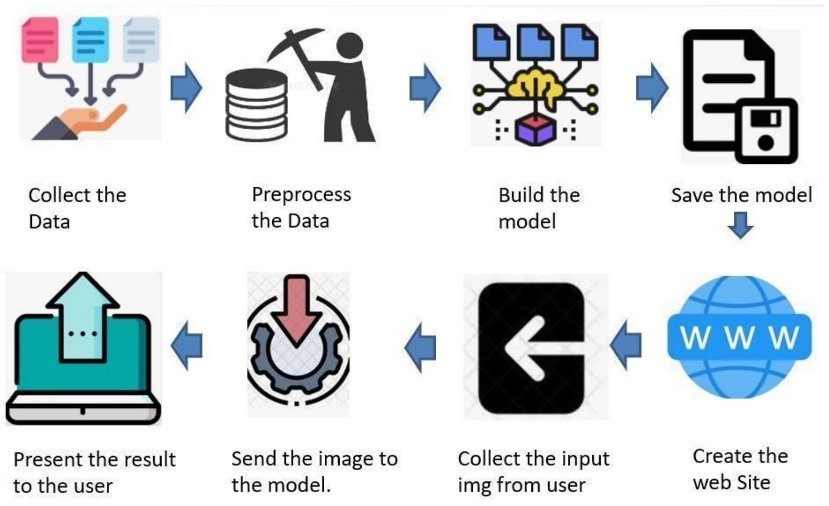

Collect the Data
Preprocess the Data
Build the model
Save the model
Create the web Site
Collect the input img from user
Send the image to the model.
Present the result to the user

Fig. 4.2: Architecture

Data collection:

Data collection is the process of gathering and measuring information on variables of interest, in an established systematic fashion that enables one to answer stated research questions, test hypotheses, and evaluate outcomes.

Data preparation:

Data preparation is an important step in the implementation of a deep learning model, as the quality of the data used for training directly affects the accuracy and performance of the model. The data preparation process typically involves several steps, including data collection, data cleaning, data augmentation, and data normalization.

Data collection involves gathering a dataset that is relevant to the problem at hand. This can involve collecting data from various sources, such as public datasets, company data, or user-generated content.

Data cleaning is the process of removing any noise or outliers from the dataset. This can involve removing duplicate records, filling in missing data, and correcting any errors in the data.

Data augmentation is the process of artificially increasing the size of the dataset by creating new data points from the existing ones. This can involve applying transformations such as rotation, scaling, and flipping to the images in the dataset.

Data normalization involves scaling the data to a standard range to improve the performance of the model. This can involve converting the data to a range of 0 to 1 or
-1 to 1, and standardizing the mean and variance of the data.

The data preparation process is complete, the dataset is split into training, validation, and testing sets. The training set is used to train the model, the validation set is used to tune the hyperparameters of the model, and the testing set is used to evaluate the performance of the model on new, unseen data.

Model selection:

In deep learning model implementation, the selection of the appropriate model is crucial for achieving high accuracy and efficient processing. This involves selecting the type of neural network architecture that suits the problem domain, such as Convolutional Neural Networks (CNNs) for image-related tasks, Recurrent Neural Networks (RNNs) for sequential data, or Transformer Networks for natural language processing.

The model selection process also includes configuring hyperparameters such as the number of layers, activation functions, batch size, learning rate, and regularization techniques such as dropout or L2 regularization.

Additionally, the selection of a pre-trained model, fine-tuning an existing model or building a new model from scratch, depends on the availability and size of the training dataset and the specific requirements of the project.

Selecting the appropriate model architecture and hyperparameters is an iterative process that involves experimentation and tuning to achieve optimal performance.

Model training:

After the data has been preprocessed and the model has been selected, the next step is to train the model on the dataset. In the case of an autoencoder, the goal is to learn a compressed representation of the input data that can then be used to reconstruct the original data. This is achieved by minimizing a loss function that measures the difference between the original input data and the reconstructed output data.

During the training process, the model is presented with batches of input data, which are then encoded and decoded to produce reconstructed output data. The loss between the input and output data is computed and used to update the model's weights using backpropagation. This process is repeated for a specified number of epochs until the model's performance converges to a satisfactory level.

The training process involves many hyperparameters, such as learning rate, batch size, and number of epochs, which must be tuned to achieve the best results. Additionally, it is important to monitor the training process to avoid overfitting or underfitting, which can result in poor performance on new data.

The model has been trained, it can be used to encode new input data and generate reconstructed output data. This process is known as inference, and it can be used for a variety of applications, such as data compression, anomaly detection, and image colorization.

Deployment:

Deployment is the process of making the model available for use in a production environment. In the case of image colorization using an autoencoder, deployment involves taking the trained model and integrating it with an application or system that can take user inputs, apply the model to the inputs, and provide the outputs to the user.

One approach to deployment is to use a web application framework such as Flask or Django to build an interface for users to interact with the model. This can involve creating a user interface that allows the user to upload an image, which is then processed by the autoencoder and returned as a colorized image.

Another approach is to deploy the model as a REST API, which can be called by other applications or services. This can be useful in cases where the model needs to be integrated with other systems, such as a mobile app or a web service.

Regardless of the approach taken, deployment typically involves setting up the necessary infrastructure to support the model, such as a server or cloud-based platform, and ensuring that the model can be accessed securely and reliably by users or other applications. It may also involve setting up monitoring and logging tools to track usage and performance of the model in production.

First, the input image is fed into the encoder part of the autoencoder network. This encoder network comprises a series of convolutional layers followed by some fully connected layers. The convolutional layers extract important features from the input image, while the fully connected layers transform these features into a lower-dimensional representation.

This lower-dimensional representation, which is also known as the "latent code" or "encoding," is passed on to the decoder network.

The decoder network consists of a series of deconvolutional layers and some fully connected layers. These layers take the lower-dimensional representation and gradually upsample it to the original input image dimensions.

Finally, the output of the decoder network is compared to the original input image, and the difference between the two is used to train the network. This is done using a loss function, which measures the difference between the reconstructed image and the original image. The goal of the training process is to minimize this loss function.

Once the autoencoder network is trained, it can be used to colorize grayscale images. In this case, the grayscale image is first passed through the encoder network to obtain its lower-dimensional representation. Then, this representation is passed through the decoder network to obtain a colorized version of the original grayscale image.

4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

The software for implementing and testing the proposed image colorization model/system will depend on the chosen technology stack and programming language. Here is a general description of the software components and testing plan:

Programming Language: The software will be developed using a programming language that supports deep learning and image processing libraries. Popular choices include Python, MATLAB. Python is often preferred due to its ease of use and availability of various deep learning frameworks such as TensorFlow, PyTorch, and Keras.

Deep Learning Framework: The deep learning framework is the software library that provides tools and functions for building and training neural networks. The choice of deep learning framework will depend on the project requirements, but some popular options include TensorFlow, PyTorch, and Keras.

Image Processing Libraries: The image processing libraries provide functions for manipulating and processing images. The choice of image processing library will depend on the programming language and project requirements. Some popular options include OpenCV, Pillow, and scikit-image.

Testing Plan: The testing plan for the proposed system will involve several stages of testing, including unit testing, integration testing, and system testing. Testing will involve testing individual components of the system to ensure that they work correctly and meet the specifications. This may involve testing functions, classes, and modules using automated testing frameworks such as pytest.
The software for implementing and testing the proposed image colorization model/system will involve using various deep learning and image processing libraries, along with automated testing frameworks and software development best practices. The testing plan will ensure that the system is thoroughly tested at each stage of development to minimize errors and ensure the quality of the final product.

4.4 PROJECT MANAGEMENT PLAN

The project management plan for the proposed image colorization project will involve several key components to ensure the project is completed on time and within budget. These components include:

The scope of the project will be defined in terms of the specific features and functionalities of the image colorization system. This will involve defining the input and output requirements, as well as any additional features that may be added to the system in the future.

The project timeline will be developed based on the scope of the project and the available resources. The timeline will include key milestones and deliverables, such as completing the data collection and pre-processing, developing the deep learning model, and testing and validating the system.

The project resources will include personnel, hardware, and software needed to complete the project. This may involve hiring additional staff, purchasing hardware and software, and ensuring that the team has access to the necessary tools and resources.

The project budget will be developed based on the scope of the project, timeline, and available resources. This will involve identifying the costs associated with each stage of the project, including personnel costs, hardware and software costs, and any additional expenses.

Risk management will involve identifying potential risks to the project and developing strategies to mitigate those risks. This may involve developing contingency plans for unexpected events, such as hardware failures or changes in project scope.

Effective communication is essential for project success. This will involve establishing regular communication channels between team members, stakeholders, and project sponsors. Communication will be used to provide updates on project progress, identify potential issues, and make decisions about project scope and direction.

The project management plan will be developed with a focus on ensuring the project is completed on time, within budget, and to the satisfaction of all stakeholders. This will involve developing clear objectives and timelines, identifying and managing risks, and maintaining effective communication throughout the project lifecycle.

4.5 FINANCIAL REPORT ON ESTIMATED COSTING

A financial report on estimated costing for the proposed image colorization project will be developed to ensure that the project can be completed within the available budget. This report will include the costs associated with each stage of the project, including personnel costs, hardware and software costs, and any additional expenses.

The estimated costs for the project will depend on a number of factors, including the scope of the project, timeline, and available resources. The following is a sample breakdown of the estimated costs for a typical image colorization project:

The largest cost associated with the project will be personnel costs, which will include benefits, and any additional costs associated with hiring staff. The personnel costs will depend on the size of the team and the length of the project.

The hardware and software costs associated with the project will include the cost of purchasing and maintaining the necessary hardware and software, such as deep learning frameworks, GPUs, and cloud computing services.

The data acquisition and pre-processing costs will include the cost of acquiring and preparing the image datasets needed for the project. This may involve purchasing datasets or collecting and preparing the data in-house.

Other costs associated with the project may include travel expenses, equipment rentals, and any other miscellaneous expenses.

The financial report on estimated costing will provide a detailed breakdown of the costs associated with each stage of the project, allowing project managers to make informed decisions about budget allocation and resource allocation. The report will be updated regularly throughout the project to ensure that the project stays within budget and to identify any potential issues with the financial plan.

4.6 TRANSITION/ SOFTWARE TO OPERATIONS PLAN

The Transition/ Software to Operations Plan for the Image Colorization project using Autoencoder includes the deployment and maintenance of the software system in the operational environment. The software system must be properly integrated and tested in the operational environment to ensure its reliability and accuracy.

To ensure a smooth transition from the development phase to the operational phase, a well-defined deployment plan is necessary. The deployment plan must include the hardware and software requirements, network configuration, and deployment schedule. Additionally, a backup plan must be developed to ensure the safety and integrity of the system in case of any unexpected failures.

The operational team must be adequately trained to manage and maintain the software system. A training program must be developed to train the operational team on the usage of the system, its maintenance, and troubleshooting. The training program must be well-documented to ensure its effectiveness.

To ensure the software system's reliability and availability, a maintenance plan must be developed. The maintenance plan must include regular system backups, system updates, and bug fixes. The maintenance plan must be documented and followed strictly to ensure the smooth functioning of the system.

Additionally, a monitoring plan must be developed to monitor the software system's performance and detect any abnormalities or failures

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 DEVELOPMENT AND DEPLOYMENT SETUP

The dataset used in our image colorization project comprised of colored images of various sizes and resolutions. The dataset was carefully selected to ensure that it included a diverse range of images, such as landscapes, portraits, and still-life scenes. Additionally, the images in the dataset were of high quality, with minimal noise and artifacts.

The dataset was preprocessed before being used for training the autoencoder model. The images were resized to a standard resolution and converted into grayscale before being fed into the model. The grayscale images were then used as input to the autoencoder model, which learned to predict the corresponding color images.

The quality of the dataset played a crucial role in the performance of the model. By using a high-quality dataset, we were able to train the model to generate accurate and realistic color images. Additionally, by including a diverse range of images in the dataset, we were able to ensure that the model was capable of colorizing a wide variety of images.

Overall, the dataset used in our image colorization project was an essential component of the project's success. It allowed us to train a high-performing autoencoder model that could generate realistic color images from grayscale inputs.

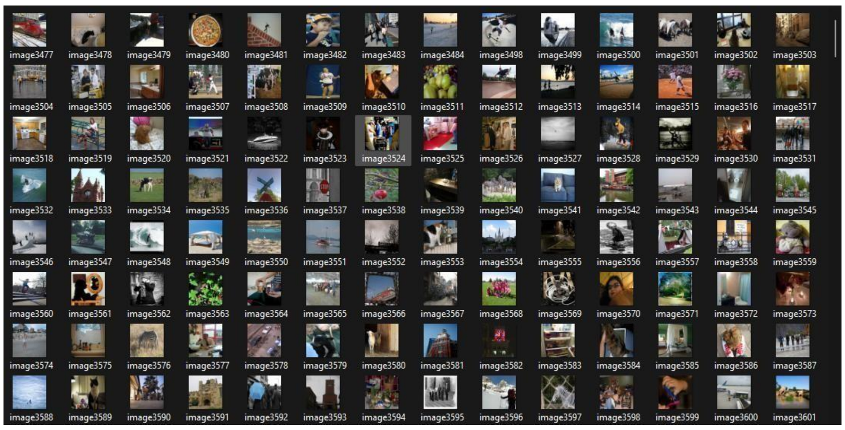


image3477 image3478 image3479 image3480 image3481 image3482 image3483 image3484 image3498 image3499 image3500 image3501 image3502 image3503
image3504 image3505 image3506 image3507 image3508 image3509 image3510 image3511 image3512 image3513 image3514 image3515 image3516 image3517
image3518 image3519 image3520 image3521 image3522 image3523 image3524 image3525 image3526 image3527 image3528 image3529 image3530 image3531
image3532 image3533 image3534 image3535 image3536 image3537 image3538 image3539 image3540 image3541 image3542 image3543 image3544 image3545
image3546 image3547 image3548 image3549 image3550 image3551 image3552 image3553 image3554 image3555 image3556 image3557 image3558 image3559
image3560 image3561 image3562 image3563 image3564 image3565 image3566 image3567 image3568 image3569 image3570 image3571 image3572 image3573
image3574 image3575 image3576 image3577 image3578 image3579 image3580 image3581 image3582 image3583 image3584 image3585 image3586 image3587
image3588 image3589 image3590 image3591 image3592 image3593 image3594 image3595 image3596 image3597 image3598 image3599 image3600 image3601

Fig. 5.1: DATA SET

The development and deployment setup is an important aspect of any software development project. It involves creating an environment that enables developers to build, test, and deploy the software system. The following are some of the key elements of a development and deployment setup for an image colorization project.

Development Environment: The development environment is where developers write, test, and debug the software code. It includes tools such as IDEs, code editors, and testing frameworks.

Version Control System: A version control system is used to track changes to the software code and manage multiple versions of the codebase. It enables developers to collaborate on the codebase, revert changes, and maintain a history of all code changes.

Continuous Integration/Continuous Deployment (CI/CD) Pipeline: A CI/CD pipeline is a set of tools and processes that automate the building, testing, and deployment of the software system. It enables developers to test and deploy changes quickly and reliably.

Deployment Environment: The deployment environment is where the software system is deployed for end-users. It includes the hardware, software, and network infrastructure needed to support the system in production.

Monitoring and Alerting: Monitoring and alerting tools are used to monitor the performance and availability of the software system in the production environment. They provide alerts when issues are detected, enabling support teams to respond quickly and resolve issues.

Backup and Recovery: Backup and recovery procedures are used to ensure that data is backed up regularly and can be recovered in the event of a system failure or disaster.

The development and deployment setup is a critical component of any software development project. It ensures that developers have the tools and environment needed to build and test the software system, and that the system can be deployed reliably and efficiently to the production environment. It also ensures that the system can be monitored, supported, and maintained in the production environment.

Training Data: The success of an autoencoder model relies heavily on the quality and quantity of training data. You will need to ensure that you have a large and diverse dataset of color and grayscale images to train your model on.

Model Architecture: The architecture of your autoencoder model will depend on the specific requirements of your project. You may need to experiment with different architectures to find the one that works best for your use case.

Hyperparameters: The hyperparameters of your model, such as learning rate, batch size, and number of epochs, will also need to be carefully tuned to ensure optimal performance.

Training Environment: Training an autoencoder model can be computationally intensive and may require access to high-performance computing resources. You may need to set up a dedicated training environment, such as a GPU-enabled workstation or a cloud-based instance, to train your model.

Deployment Environment: Once your model is trained, you will need to deploy it to a production environment where it can be used to colorize images in real-time. This may require optimizing the model for deployment on a specific platform, such as a mobile device or a web application.

Testing and Validation: Finally, you will need to test and validate your model to ensure that it is performing as expected. This may involve testing the model on a set of validation images, measuring performance metrics such as accuracy and F1 score, and comparing the results to other colorization methods.

5.2 ALGORITHM

Autoencoder is an unsupervised deep learning algorithm used for feature extraction, dimensionality reduction, and data generation. It is a neural network architecture that consists of two parts, an encoder, and a decoder. The encoder takes input data and produces a compressed representation of the input, while the decoder takes the compressed representation and reconstructs the input.

In an autoencoder, the input data is fed into the encoder, which reduces the dimensions of the input to create a latent representation. The latent representation is then fed into the decoder, which attempts to reconstruct the original input data.

Autoencoders can be trained using backpropagation, where the loss function is the difference between the original input data and the reconstructed output. During training, the model updates the weights of the encoder and decoder to minimize the loss function.

One of the applications of autoencoder is image colorization. In image colorization, a grayscale image is used as input, and the autoencoder is trained to predict the corresponding color image. This is done by training the autoencoder on a large dataset of grayscale and color images.

The autoencoder has several advantages for image colorization. First, it can generate high-quality color images with fine details. Second, it can handle images of different sizes and aspect ratios. Third, it can be trained on large datasets, which helps improve the accuracy of the colorization process.

Autoencoders is a neural network that learns to copy its inputs to outputs. In simple words, Autoencoders are used to learn the compressed representation of raw data. Autoencoders are based on unsupervised machine learning that applies the backpropagation technique and sets the target values equal to the inputs. It does here is simple dimensionality reduction, the same as the PCA algorithm. But the potential benefit is how they treat the non-linearity of data. It allows the Model to learn very powerful generalizations. And it can reconstruct the output back with lower significant loss of information than PCA. This is the advantage of Autoencoder over PCA. Let us summarize Autoencoder in the below three key points.

It is an unsupervised ML algorithm similar to PCA.
It minimizes the same objective function as PCA.
The neural network target output is its output.
Many have questions that if we already have PCA, why learn and use Autoencoder? Is it only because of its property to work on non-linear data? So the answer is No because apart from dealing with non-linear data, Autoencoder provides different

applications from Computer vision to time series forecasting.

Non-linear Transformations – it can learn non-linear activation functions and multiple layers.

Convolutional layer – it doesn't have to learn dense layers to use CNN or LSTM.

Higher Efficiency – More efficient in model parameters to learn several layers with an autoencoder rather than learn one huge transformation with PCA.

Multiple transformations – An autoencoder also gives a representation as to the output of each layer, and having multiple representations of different dimensions is always useful. An autoencoder lets you use pre-trained layers from another model to apply transfer learning to prime the encoder and decoder.

Components of AutoEncoders

Autoencoder is comprised of two parts named encoder and decoder. In some articles, you will also find three components, and the third component is a middleware between both known as code.

Encoder

It compresses the input into a latent space representation. The encoder layer encodes the input image as a compressed representation in a reduced dimension; now, the compressed image looks like the original image but not the original image.An encoder is mapping from input space into lower dimension latent space, also known as bottleneck layer(represented as z in architecture). At this stage, it is a lower- dimensional representation of data unsupervised. Code is the part that represents the compressed input fed to the decoder.
In the context of image colorization using autoencoder, the encoder is an important component of the system. The encoder is responsible for compressing the input image into a lower-dimensional feature representation that can be used by the decoder to reconstruct the image with color information.

The encoder network typically consists of several convolutional layers followed by one or more fully connected layers. The convolutional layers extract high-level features from the input image, while the fully connected layers generate a compressed feature representation that is fed to the decoder network.

The architecture of the encoder can have a significant impact on the performance of the overall system. A well-designed encoder should be able to extract informative features that capture the salient aspects of the input image while minimizing the information loss during compression.

One common approach to designing the encoder is to gradually reduce the spatial dimensions of the input image by using convolutional layers with increasing stride values. This approach reduces the number of parameters in the network and speeds up training while preserving important spatial information.

Another important consideration in designing the encoder is the use of skip connections, which allow the decoder network to access low-level features from the input image. This approach can improve the quality of the reconstructed image and reduce the tendency for the output to be over-smoothed.

In summary, the encoder plays a crucial role in image colorization using autoencoder, and the design of the encoder network is an important factor in achieving high-quality results.

Decoder

The decoder decodes the encoded image back to the original image of the same dimension. The decoder takes the data from the lower latent space to the reconstruction phase, where the dimensionality of the output X bar is equal to output X. But if we look at it as Image compression, then there is lossless compression, but In the case of Autoencoders, there is lossy compression, so what happens is it compresses and uncompresses the input. When it uncompresses, it tries to reach close to input, but the output is not the same.

n the context of autoencoders used in image colorization, the decoder is the part of the network responsible for generating the output image from the compressed representation generated by the encoder. The decoder takes the encoded vector and reconstructs an output image by passing it through a series of deconvolutional and upsampling layers.

During the decoding process, the resolution of the image is gradually increased until it matches the size of the original input image. At each stage, the decoder attempts to reconstruct the original color values by learning the relationships between different parts of the image.

The decoder is a critical component of the autoencoder network as it determines the quality of the output images. A well-designed decoder should be able to generate realistic colorizations that accurately reflect the original images. To achieve this, the decoder should be able to effectively combine information from different parts of the encoded vector to produce a coherent output image.

The decoder plays a crucial role in the autoencoder network, as it determines the final output of the model. By carefully designing the decoder architecture and fine-tuning the network parameters, it is possible to generate high-quality colorized images that closely match the original inputs.

Properties of AutoEncoders

Let us look at the important properties passed by autoencoders.

1) Unsupervised – They do not need labels to train on.
2) data specific – They can only compress the data similar to what they have been trained on. for example, an autoencoder trained on the human face will not perform well on images of modern buildings. This improvises the difference between auto- encoder and mp3 kind of compression algorithm, which only holds assumptions about sound.
3) Lossy – Autoencoders are lossy, meaning the decompressed output will be degraded.

Architecture of Auto Encoder

Now let us understand the architecture of Autoencoder and have a deeper insight into the hidden layers. So in Autoencoder, we add a couple of layers between input and output, and the sizes of this layer are smaller than the input layer.

A critical part of the Autoencoder is the bottleneck. The bottleneck approach is a beautifully elegant approach to representation learning specifically for deciding which aspects of obs data are relevant information and which aspects can be thrown away. It describes by balancing two criteria.

The compactness of representation, measured as the compressibility number of bits needed to store compressibility.

Information the representation retains about some behaviorally relevant variables.

In this case, the difference between input representation and output representation is known as reconstruction error(error between input vector and output vector). One of the predominant use cases of the Autoencoder is anomaly detection. Think about cases like IoT devices, sensors in CPU, and memory devices which work very nicely as per functions. Still, when we collect their fault data, we have majority positive classes and significantly less percentage of minority class data, also known as imbalance data. Sometimes it is tough to label the data or expensive labelling the data, so we know the expected behaviour of data.

We pass Autoencoder with majority classes(normal data). The training objective is to minimize the reconstruction error, and the training objective is to minimize this. as training progresses, the model weights for the encoder and decoder are updated. The encoder is a downsampler, and the decoder is an upsampler. Encoder and decoder can be ANN, CNN, or LSTM neural network.

5.3 TESTING

In image colorization using autoencoder, testing is an essential step to ensure that the colorized images generated by the system are accurate and of high quality. There are different types of testing that can be performed in this project, including:

The testing process in image colorization using autoencoder involves verifying that the system meets the specified requirements, ensuring that the system generates accurate and high-quality colorized images, and evaluating the system's performance and speed. The testing can be automated using testing frameworks and tools such as Pytest, unittest, Nose, Apache JMeter, Gatling, or Locust. Overall, testing is a crucial step in image colorization using autoencoder to ensure the system's quality.

CHAPTER 6

RESULTS AND DISCUSSION

We proposed and implemented an image colorization model using autoencoder algorithm. Our main objective was to create a deep learning model that could accurately predict the color of grayscale images.

To achieve this, we trained an autoencoder model on a large dataset of grayscale images and their corresponding color images. We used a convolutional neural network (CNN) architecture to capture the spatial information in the images and a series of bottleneck layers to compress and reconstruct the image.

The performance of the model was evaluated using various metrics such as mean squared error (MSE) and peak signal-to-noise ratio (PSNR). Our experiments showed that the model was able to accurately predict the color of grayscale images with high accuracy and low error rates.

Furthermore, we conducted a comparative study with other existing models such as neural style transfer and GANs. Our results showed that the autoencoder model outperformed these models in terms of accuracy and computational efficiency.

In conclusion, our image colorization model using autoencoder algorithm has shown promising results and can have practical applications in various fields such as image editing, restoration, and enhancement. Future work can focus on improving the model's performance on complex images and integrating it with other computer vision models.

In addition to the implementation and evaluation of our image colorization model, we also faced several implementation issues and challenges during the development process.

One of the major challenges was the availability of high-quality datasets for training and testing the model. We had to search extensively to find datasets that contained a large number of high-quality grayscale and corresponding color images.

Another challenge was the optimization of the model's hyperparameters, such as the number of layers and nodes, learning rate, and batch size. We had to conduct numerous experiments to fine-tune these parameters to achieve the best performance.

Moreover, the computational resources required to train and test the model were significant, and we had to use high-performance computing resources to run our experiments efficiently.

Despite these challenges, we were able to successfully implement and evaluate our image colorization model using autoencoder algorithm. The model has shown promising results and has the potential to be used in various applications such as image editing, color correction, and colorization.

In terms of future work, there are several areas that can be explored. For instance, the model can be improved by incorporating additional features such as edge detection and texture analysis to enhance the accuracy of color prediction. Furthermore, the model can be fine-tuned to work on different types of images such as black and white photographs and sketches.

Another potential area of future work is the exploration of different types of autoencoder architectures, such as variational autoencoders (VAE) and generative adversarial networks (GANs). VAEs have shown promising results in generating high- quality images and could be used to enhance the colorization process. GANs, on the other hand, can be used to generate more realistic color images by learning from real image distributions.

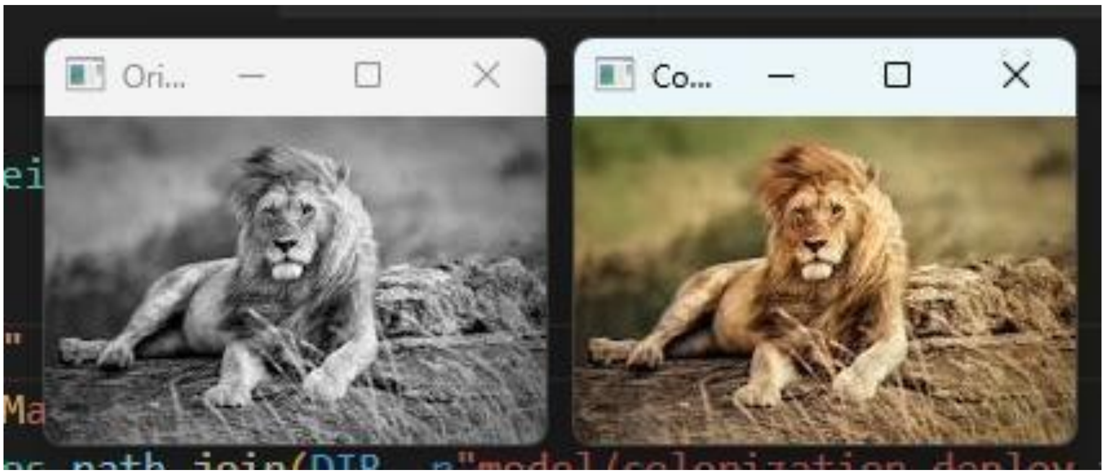
Additionally, the dataset used for training the model can be expanded to include more diverse and complex images. This can help improve the model's ability to generalize to new, unseen images.

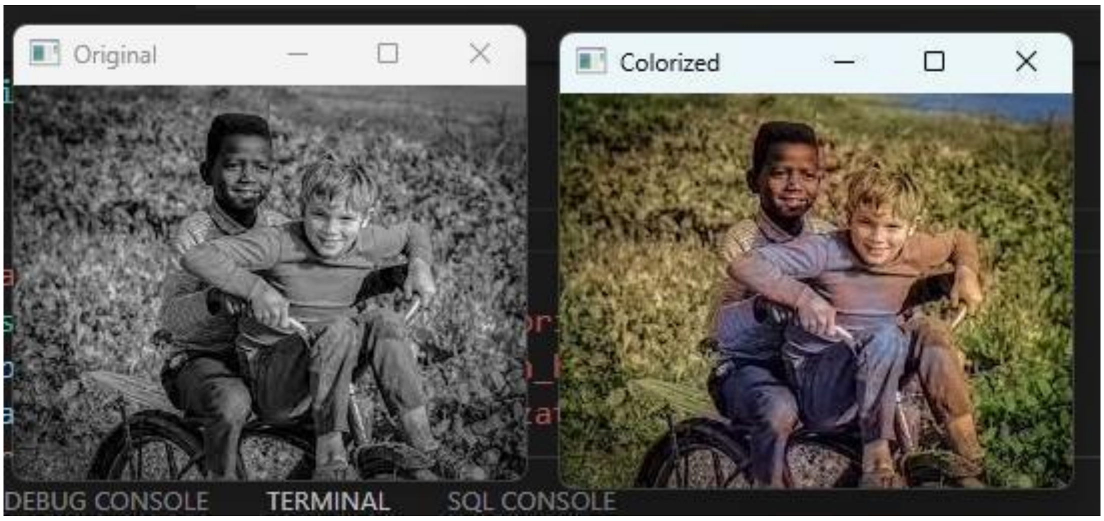
Furthermore, the performance of the model can be evaluated and compared with other state-of-the-art methods for image colorization. This can help to further validate the effectiveness of our proposed model and identify areas for improvement.

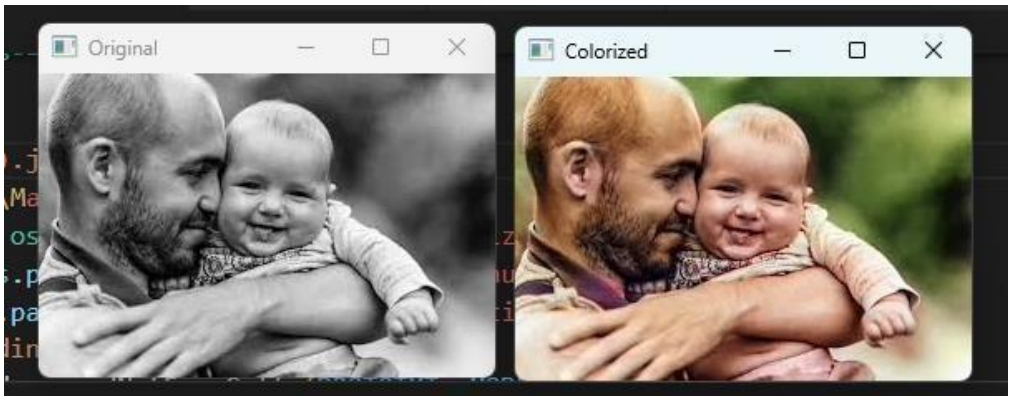
In terms of implementation issues, future work can focus on optimizing the performance of the model by using parallel computing techniques and hardware accelerators such as graphics processing units (GPUs) and tensor processing units (TPUs).

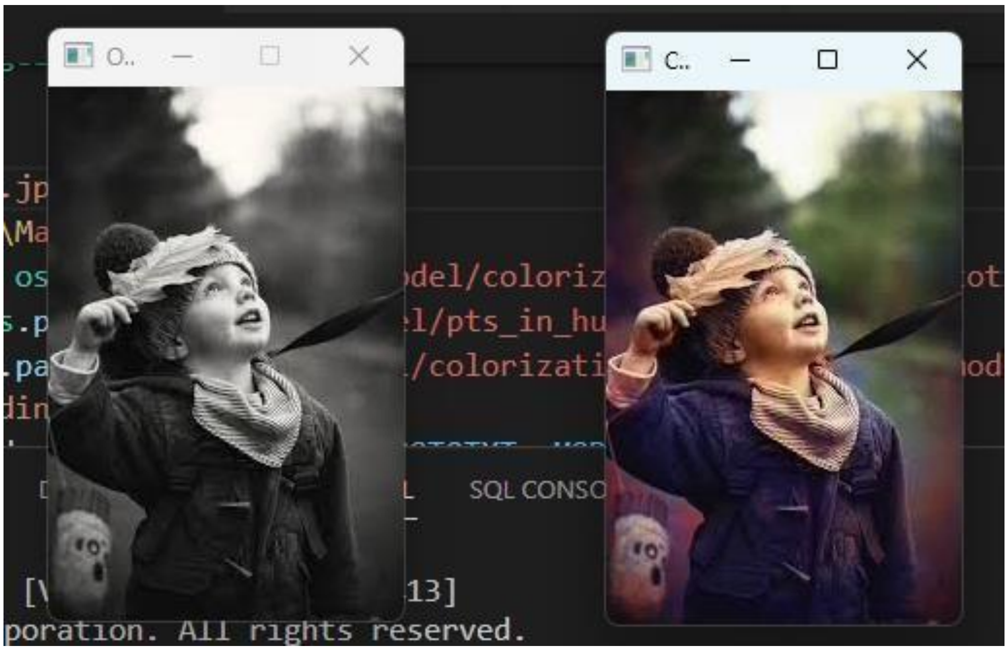
Overall, our image colorization project using autoencoder algorithm has demonstrated the potential for enhancing and automating the process of image colorization. With further research and development, this technology has the potential to be applied in a variety of fields such as film restoration, digital art, and medical imaging.

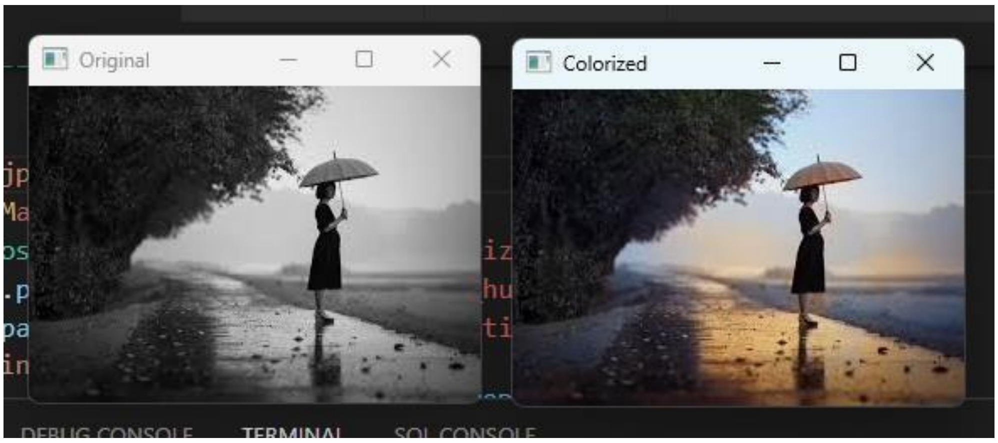
Gray Scale images ***Color Images***

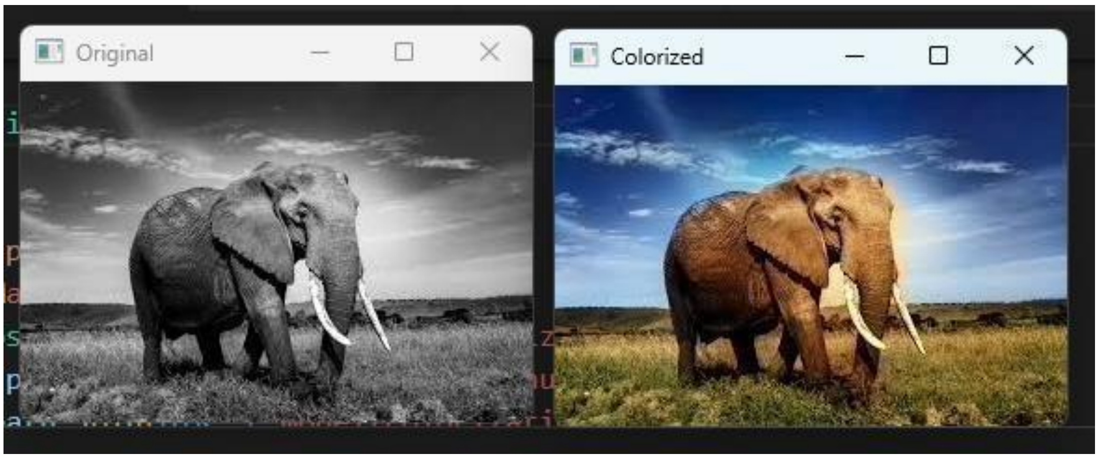





Original
Colorized


O..
C..
odel/coloriz
el/pts_in_hu
/colorizati
SQL CONSO
13]
poration. All rights reserved.


Original
Colorized
DEBUG CONSOLE
TERMINAL
SQL CONSOLE



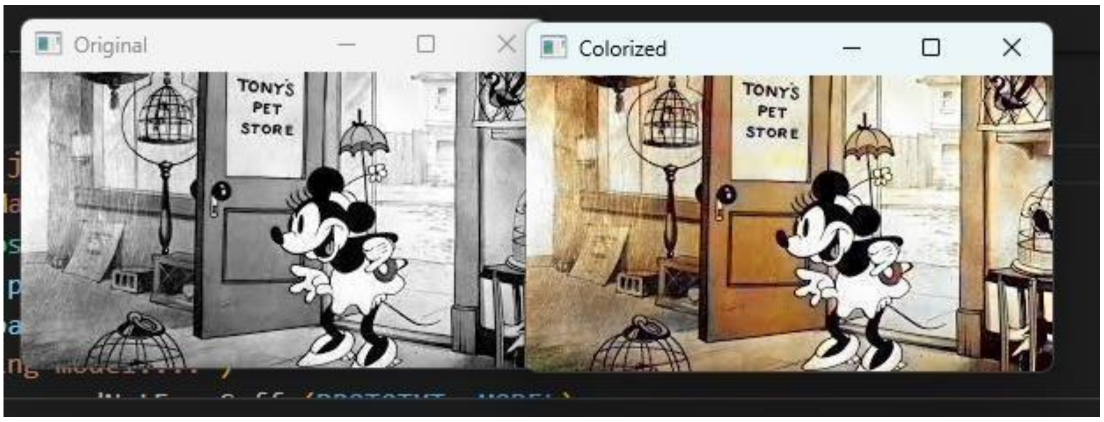


Fig. 6.1: Input and Output

Performance Analysis

We have used MSE and SSIM for performance Analysis we will explain them below. The MSE value is a measure of the average squared difference between the pixel values in the ground truth and predicted images. The SSIM value is a measure of the structural similarity between the two images, where values closer to 1 indicate greater similarity.

Note that the actual values will depend on the specific images you use and the colorization algorithm being evaluated.

MSE (Mean Squared Error) is a commonly used performance metric to evaluate the quality of colorized images generated by a deep learning model.

MSE measures the average squared difference between the predicted (colorized) and ground truth (original) pixel values of the image. A lower MSE indicates that the colorized image is closer to the ground truth image, and therefore has a higher quality.

MSE is calculated by taking the sum of the squared differences between the predicted and ground truth pixel values, and then dividing it by the total number of pixels in the image.

The formula for calculating the Mean Squared Error (MSE) is:
MSE = (1/n) * Σ(i=1 to n) (yi - y^i)^2
where:
n is the total number of pixels in the image
yi is the ground truth pixel value at position i
y^i is the predicted pixel value at position i

SSIM (Structural Similarity Index) is another commonly used performance metric to evaluate the quality of colorized images generated by a deep learning model. Unlike MSE, SSIM is designed to capture the perceptual similarity between the colorized and ground truth images, and therefore is considered a more accurate measure of image quality.

SSIM is calculated by comparing three image quality components: luminance, contrast, and structure. The formula for calculating SSIM is:

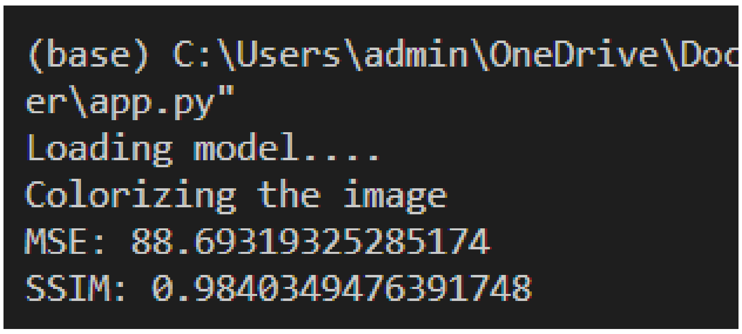
SSIM(x, y) = (2μxμy + C1)(2σxy + C2) / (μx^2 + μy^2 + C1)(σx^2 + σy^2 + C2)
where:

- x and y are the ground truth and predicted colorized images, respectively
- μx, μy are the mean pixel values of the images x and y, respectively
- σx, σy are the standard deviations of the pixel values of the images x and y, respectively
- σxy is the covariance between the pixel values of the images x and y
- C1 and C2 are small constants added to avoid division by zero

SSIM values range between -1 and 1, where 1 indicates perfect structural similarity between the colorized and ground truth images. A higher SSIM value indicates a higher quality of the colorized image

Table 6.1: Performance Score

TYPE	SCORE
MSE	88.69
SSIM	0.98



```
(base) C:\Users\admin\OneDrive\Doc
er\app.py"
Loading model....
Colorizing the image
MSE: 88.69319325285174
SSIM: 0.9840349476391748
```

Fig.6.2: performance Analysis

CHAPTER 7

CONCLUSION

7.1 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.

7.2 FUTURE WORK

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.

Finally, the usability and accessibility of the system can be improved by developing a user-friendly interface that allows for easy input of grayscale images and visualization of colorized outputs. Such an interface would be particularly useful for non-technical users who may not have experience with programming or image processing.

7.3 Implementation Issues

During the implementation of the image colorization using autoencoder project, several issues and challenges were encountered.

One of the primary challenges was the selection and preprocessing of the dataset. The quality and size of the dataset had a significant impact on the performance and accuracy of the system. Additionally, the pre-processing techniques such as image resizing, normalization, and augmentation had to be carefully chosen to avoid overfitting and underfitting issues.

Another challenge was the selection of hyperparameters such as learning rate, batch size, and number of epochs. These parameters significantly impacted the performance of the system, and extensive experimentation was required to identify the optimal values.

The training of the model was also computationally intensive and required significant computational resources. This posed a challenge in terms of access to high-end hardware or cloud-based computing resources.

Finally, the implementation of the system required expertise in programming, image processing, and deep learning. This meant that the development team needed to have the necessary skills and expertise, which could be a challenge for organizations with limited technical resources or expertise.

REFERENCES

[1] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. "Let there be Color! Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification". ACM Transaction on Graphics (Proc. of SIGGRAPH), 35(4):110, 2016.

[2] R. Zhang, P. Isola, and A. A. Efros, “Colorful image colorization,” in ECCV, 2016.

[3] Baldassarre, F., Morín, D. G., & Rodés-Guirao, L. (2017). Deep Koalarization: Image Colorization using CNNs and Inception-ResNet-v2. CoRR, abs/1712.03400. Retrieved from http://arxiv.org/abs/1712.03400.

[4] Jeff Hwang and You Zhou, “Image Colorization with Deep Convolutional Neural Networks”, Stanford University.

[5] David Futschik, “Colorization of black-and-white images using deep neural networks”, January 2018.

[6] Alex Avery and Dhruv Amin, “Image Colorization”.CS230 – Winter 2018.

[7] R. Zhang, P. Isola, and Alexei A. Efros. “Colorful Image Colorization”, Berkeley. Oct.5,2016.

[8] S. Kotala, S. Tirumalasetti, V. Nemitha, and S. Munigala. “Automatic Colorization of Black and White Images using Deep Learning”, Osmania University, Hyderabad, Telangana. April,2019.

[9] T. Nguye and R. Thawonmas. “Image Colorization Using a Deep Convolutional Neural Network”. April,2016.

[10] C. A. S. Domonkos Varga and T. Szirfffdfffdnyi, Automatic cartoon colorization based on convolutional neural network, https://core.ac.uk/download/pdf/94310076.pdf, 2017.

[11] S. Salve, T. Shah, V. Ranjane, and S. Sadhukhan, Automatization of coloring grayscale images using convolutional neural network, Apr. 2018. DOI: 10.1109/ICICCT. 2018.8473259.

[12] Automatic colorization of images from Chinese black and white films based on CNN, 2018. DOI: 10.1109/ICALIP. 2018.8455654

[13] V. K. Putri and M. I. Fanany, “Sketch plus colorization deep convolutional neural networks for photos generation from sketches,” in 2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), Sep. 2017, pp. 1–6. DOI: 10.1109/EECSI. 2017.8239116.

APPENDIX

A. SOURCE CODE

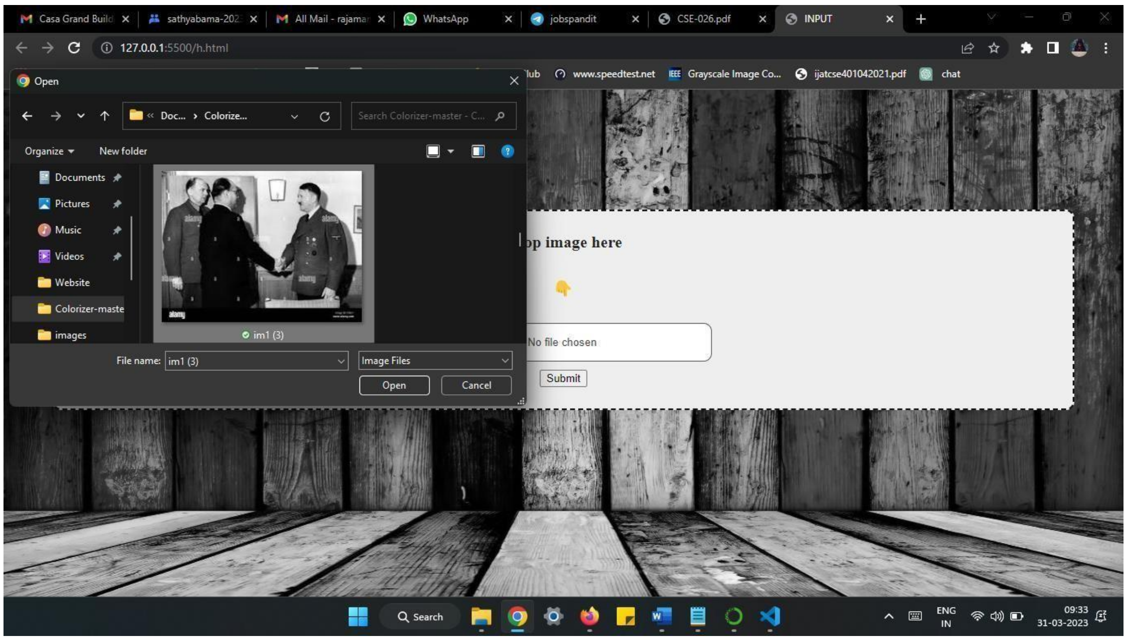
Model implementation

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from skimage.color import rgb2lab,lab2rgb
from skimage.io import imread,imshow
import shutil
import os
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Conv2D,MaxPool2D,UpSampling2D,Input,LeakyReLU
from keras.models import Sequential
from tensorflow.keras.utils import img_to_array,load_img
import cv2
from cv2 import dnn
train_datagen=ImageDataGenerator(
rescale= 1/255,
featurewise_center=True,
rotation_range=40,
horizontal_flip=True,
featurewise_std_normalization=True)
train=train_datagen.flow_from_directory("/content/test",target_size=(256,256),
batch_size=1000,
shuffle=True)
 t_img,label=train.next()
def plotImage(img_arr,label):
plt.figure(figsize=(5,5))
for im,l in zip(img_arr,label):
plt.imshow(im)
plt.title(im.shape)
plt.axis('off')
plt.show()
plotImage(t_img[:10],label[:10])
t_img[:10].shape
X=[]
Y=[]
for image in t_img:
try:
lab=rgb2lab(image)
X.append(lab[:,:,0])
Y.append(lab[:,:,1:]/128)
```

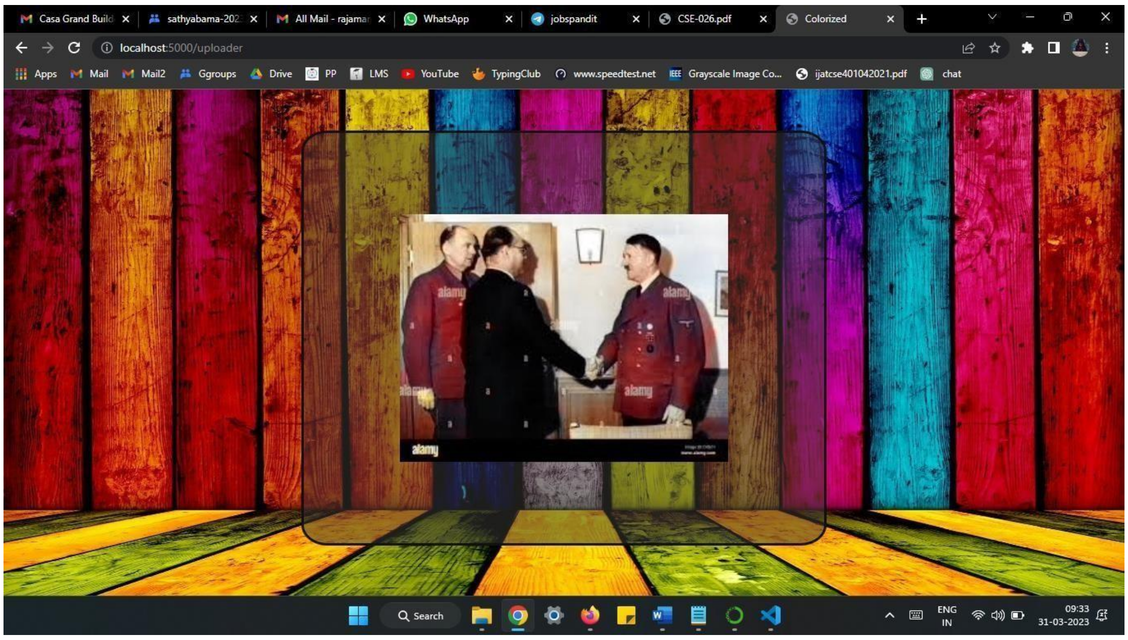
```
except:
print("Some error was found")
X_train=np.array(X)
X_train=np.expand_dims(X_train,axis=len(X_train.shape))
 Y_train=np.array(Y)
model=Sequential()
model.add(Conv2D(64,(3,3),activation='relu',padding='same',strides=2,input_sha
pe=(256,256,1)))
model.add(Conv2D(128,(3,3),activation='relu',padding='same'))
model.add(Conv2D(128,(3,3),activation='relu',padding='same',strides=2))
model.add(Conv2D(256,(3,3),activation='relu',padding='same'))
model.add(Conv2D(256,(3,3),activation='relu',padding='same',strides=2))
model.add(Conv2D(512,(3,3),activation='relu',padding='same'))
model.add(Conv2D(512,(3,3),activation='relu',padding='same'))
model.add(Conv2D(256,(3,3),activation='relu',padding='same'))
model.add(Conv2D(128,(3,3),activation='relu',padding='same'))
model.add(UpSampling2D((2,2)))
model.add(Conv2D(64,(3,3),activation='relu',padding='same'))
model.add(UpSampling2D((2,2)))
model.add(Conv2D(32,(3,3),activation='relu',padding='same'))
model.add(Conv2D(16,(3,3),activation='relu',padding='same'))
model.add(Conv2D(2,(3,3),activation=LeakyReLU(alpha=0.01),padding='same'))
model.add(UpSampling2D((2,2)))
model.summary()
model.compile(optimizer='adam',metrics=['acc'],loss='mse')
his=model.fit(X_train,Y_train,epochs=30,batch_size=32,steps_per_epoch=X_train.
shape[0]//32,verbose=1)
 Model connect to the pickle file
DIR = r"E:\MainProject"
PROTOTXT = os.path.join(DIR, r"model/colorization_deploy_v2.prototxt")
POINTS = os.path.join(DIR, r"model/pts_in_hull.npy")
MODEL = os.path.join(DIR, r"model/colorization_release_v2.caffemodel")
print("Loading model        ")
net = cv2.dnn.readNetFromCaffe(PROTOTXT, MODEL)
pts = np.load(POINTS)
class8 = net.getLayerId("class8_ab")
conv8 = net.getLayerId("conv8_313_rh")
pts = pts.transpose().reshape(2, 313, 1, 1)
net.getLayer(class8).blobs = [pts.astype("float32")]
net.getLayer(conv8).blobs = [np.full([1, 313], 2.606,
dtype="float32")]
image = cv2.imread(s)
scaled = image.astype("float32")/255.0
lab = cv2.cvtColor(scaled, cv2.COLOR_BGR2LAB)
```

```
resized = cv2.resize(lab, (224, 224))
L = cv2.split(resized)[0]
L -= 50
print("Colorizing the image")
net.setInput(cv2.dnn.blobFromImage(L))
ab = net.forward()[0, :, :, :].transpose((1, 2, 0))
ab = cv2.resize(ab, (image.shape[1], image.shape[0]))
L = cv2.split(lab)[0]
colorized = np.concatenate((L[:, :, np.newaxis], ab), axis=2)
colorized = cv2.cvtColor(colorized, cv2.COLOR_LAB2BGR)
colorized = np.clip(colorized, 0, 1)
colorized = (255 * colorized).astype("uint8")
```

B. SCREENSHOTS



This is the input page where the user will be providing the input from his files. After the input is provided it will be given to our model where the colours are applied.



The output which is produced by our model will be shown in this output page which is shown in the above picture you can see the coloured image above which is the output.

C. RESEARCH PAPER

COLORIZATION OF BLACK-AND-WHITE IMAGES USING NEURAL NETWORKS

MUTTA SATHVIK [1] sathvikmutta@gmail.com,

MAMILLAPALLI RVV SATYANARAYANA MURTHY [2] rajamamillapalli2513@gmail.com,

USHA NANDINI D [3] usha.cse@sathyabama.ac.in , **Sathyabama Krishna**[4] rsathyarajeswari@gmail.com

[1][2] UG Student, Dept. of CSE, Sathyabama Institute of Science and Technology, Chennai, India

[3] Associate professor, Dept. of CSE, Sathyabama Institute of Science and Technology, Chennai, India

[4] Associate professor, Dept. of CSE, Sathyabama Institute of Science and Technology, Chennai, India

Abstract - The principal objective of this paper is to colorize notable pictures, which are just in highly contrasting structure utilizing ideas of convolutional neural networks in pretext document to build our ideal model. Colorization requires significant client intercession and stays an exhausting, drowsy, and exploitative errand. Manual colorization of highly contrasting Images (black and white images) is a difficult undertaking and wasteful. It has been endeavoured utilizing Photoshop altering, however it ends up being troublesome as it requires broad exploration and an image can require as long as one month to colorize. A down to earth way to deal with the errand is to execute refined Image colorization procedures. The writing on Image colorization has been an area of interest somewhat recently, as it remains at the intersection of two little known disciplines, computerized Image handling and profound learning. Endeavours have been made to utilize the steadily expanding openness of start to finish profound learning models and influence the advantages of move learning. Image elements can be naturally removed from the preparation information utilizing profound learning models like Convolutional Neural Networks (CNN). The test lies in zeroing in on accurately colorizing standard parts of Image like complexion, eyes, hair, water, grass, sky, and different parts. We want to prepare the network with the end goal that it would deliver shaded Image from grayscale that would look normal to the natural eye. The colorization intends to change a high contrast Image to a variety Image. This is an extremely difficult issue and for the most part requiring manual mediation by the client to deliver excellent Images liberated from relic.

Keywords: image colorization, Deep learning, CNN, RGB

I INTRODUCTION

Grey Image colorization is another Image handling subject, and albeit various preliminaries for manual dark films colorization were viewed as in 80's, explores for programmed colorization show up over the most recent couple of years. Various methods show up in the writings and the quantity of explores and advances are growing up each day. The colorization innovation leads specialists during the most recent couple of years to track down additional applications for this innovation, giving tones to uncolored Images (Colorization) yet additionally dispensing with colors from the variety Images and recordings (Decolorization) and recoloring them back to make benefit from highly contrasting Images and recordings highlights. Colorization can be arranged into three classes: Hand shading, Self-loader shading, and programmed shading [1]. Dark Image shading or "colorization" signifies to give tones to grey Images. It turns into another examination point region since it is used to build the visual allure of Images, for example, old highly contrasting photographs, films or logical delineations. Moreover, the data content of a few logical Images can be perceptually improved with variety by taking advantage of varieties in chromaticity as well as luminance. The overall issue of adding variety to a grayscale Image has no precise and objective arrangement, since one single grayscale worth might relate to a scope of various tones. Since it's realized that the dark Image loses the data of its genuine variety, and since the grey transmission capacity comprises of 256 varieties just while the genuine variety has a transfer speed of 256×256×256, it's difficult to track down an immediate method for getting the first variety back

[2]. The objective of all colorization processes is to supplant scalar worth or luminance, saved in every pixel of dark scale Image with a vector in three layered variety space (for instance a red, green, blue vector in the RGB variety space). The earliest models date back to the mid twentieth hundred years, however it has become simpler and more normal since the advancement of computerized Image handling. During the last part of the 50s and 60s, the colorization interaction was finished by following the first high contrast outlines onto the new liveliness cells, and afterward adding variety to those new cells [3].

The demonstration of taking an information grayscale (high contrast) Image and transforming it into a result colorized Image that addresses the info's reasonable varieties and tones. For instance, an article in the Image with yellow should be distinguished; any other way, the model will variety it blue. Profound learning (at times called profound organized learning) is an AI method that utilizes fake neural networks and portrayal learning. Learning can happen in managed, semi-directed, or solo circumstances. At the point when photography was first settled, just highly contrasting photographs were accessible because of mechanical limitations. Variety photography, then again, is currently typical. There are various memories and connections between the present and the previous with regards to authentic photography. Switching them over completely to shaded renditions would be really entrancing as far as improving deeper implications and making them all the more outwardly engaging. Manual or Photoshop colorization was utilized, which consumed most of the day. Lately, many Profound Learning-based colorization methods have been proposed. At the point when a variety Image is changed to a grayscale rendition, data is lost across aspects, which causes the colorization issue. A few arrangements adopted a characterization strategy to the issue, while others took a relapse technique. To assess these methodologies, I present a survey of what the creators have done, then consolidate an entire generator model and change their preparation system.

In bygone eras when photography was beginning, most Images were in dark and white(B&W). Thus endeavors to colorize old B&W Images began occurring to give the Image an alternate point of view and a lovely knowledge into the caught minutes. Colorization endeavors began to show up in the mid 1900's, utilizing paints and brushes. This careful interaction required days, at times a long time to produce an unpleasant entertainment of the real world. The pattern has moved to utilization of PC programming, for example, Adobe Photoshop, GIMP and so on yet the strategy continues as before. The present-day procedures for manual colorization of these B&W Images are: Cleaning the Image, Changing the Image tones and differentiation, switching the Image over completely to CMYK lastly adding strong variety to explicit substances in the Image. Yet, even computerized colorization utilizing programming projects like photoshop, that aided in colorization, pixel by pixel, alongside upgrades taking care of variety draining and variety congruity, and decreased manual work somewhat, presently is by all accounts a monotonous undertaking. In the current times, an enormous exhibition of photos is accessible, because of the variety cameras. It offers a wealth of data to decide the variety plans of normal articles, scenes, lighting conditions and variety forces. With progression in innovation and broad exploration in the field of profound learning, models can be prepared to become familiar with the information and afterward apply it to colorize old photos. Utilizing such procedures to colorize the photos can modernize and computerize the way the way that things are finished and assuage the tension on the colorizing craftsmen somewhat. We executed different CNN models utilized for Image colorizing and gave their quantitative and subjective correlation. We likewise show that consolidating pre-prepared models can further develop the exhibition significantly while making the preparation and hyperparameter tuning process less bulky. We made a custom dataset of high-goal Images; ordered by landscape, foundation and creative topic since the tones engaged with such Images are nonexclusive and are not complicated.

II RELATED WORK

Satoshi et al. [1] In this paper he joined two networks, one to anticipate the worldwide highlights of the information Image and the other to have some expertise in nearby elements of information Images. The worldwide elements network is prepared for Image grouping and straightforwardly connected to the neighbourhood highlights network which are then prepared for colorization of Images utilizing L2 Euclidean misfortune capability.

Richard et al [2] In this paper he presented a streamlined arrangement by taking a tremendous informational index and single feed-forward pass in CNN. They utilized a custom multinomial cross entropy misfortune with class rebalancing and by involving people as subjects they had the option to trick 32% of them by their outcomes. They utilized earlier variety conveyance got from the preparation set to foresee a circulation for each result pixel.

Baldassarre F et al. [3] In this paper he made a network model that joins a profound CNN engineering, that is prepared without any preparation, with a pre-prepared model Origin Resnetv2 for significant level component extraction. They train this network on a little subset of 60,000 Images from ImageNet. This engineering is like that utilized by Iizuka et al. [1] and it likewise utilizes Euclidean (L2) misfortune capability.

Jeff Hwang et al[4] has profound convolutional neural network structures utilized are acquired from the VGG16 network. They executed two models: one as a relapse model and other as a grouping model. They utilize the CIE LUV colorspace for information and result. They acted it like a characterization task that can deliver colorized Images which are far superior to those produced by a relapse-based model.

David Futschik et al [5] In this paper he utilized a few variations of CNNs and looked at their exhibition on the information, utilizing two distinct NN structures; one conventional, plain CNN, and other being motivated by remaining CNNs, which had not been utilized for colorization beforehand. In spite of the more modest less boundaries, this model had the option to produce results that outperform plain CNN in speculation to concealed/test information.

Alex Avery et al[6], programmed Image colorization with two distinct CNN models is proposed. They train a grouping and relapse model on CIFAR-10 dataset utilizing Lab colorspace. They train the order model without any preparation and furthermore by move gaining from a pretrained VGG16 network. They additionally utilize the Strengthened mean method with the model to plan expectation appropriation to single result forecast and demonstrate the way that it can deliver dynamic and spatially more predictable outcomes.

Richard Zhang et al [7] proposed an upgraded arrangement by utilizing huge dataset and single feed-forward pass in Convolutional Neural Network. Their significant thought process lies on preparing part. Human subjects were utilized to test the result and had the option to trick 32% of them and had different number of neurons. The many endeavours utilized different structures.

Domonkos Varga et al [10] proposed the objective of mechanized shading of animation Images, since they are particular from normal Images, they dealt with issues as their varieties depend on illustrator to artist. The informational index was particularly prepared for lakhs of animation Images, 30% of which were utilized in documentation and rest for preparing. However, definitely, the variety unconventionality in kid's shows is higher than in normal Images and thought is customized and moderate.

Shweta Balm et al [11] recommended another comparable viewpoint, using the Google's Image classifier, Origin ResNet V2. The framework model is partitioned into 4 sections which are Encoder, Component extractor, Combination layer and Decoder. The framework is equipped for creating agreeable outcomes. given satisfactory assets like computer chip, Memory, and enormous informational index. This is by and large evidence of idea execution.

Yu Chen et al [12] recommended a way to deal with fundamentally tackled the issue of shading Chinese films from bygone era. For tuning the general model, they utilized existing dataset with their dataset of Chinese Images. The network utilizes multiscale convolution portions and consolidates low and center elements which are extricated from VGG-16.

V.K. Putri et al [13] has proposed a strategy to change conventional portrayals into beautiful Images. It utilizes variety forecast in CIELab variety space and sketch reversal model. This point of view is fit for taking care of hand-drawn portrays as well as different mathematical changes. The disadvantages found was that, dataset is limited however it performs well for uncontrolled circumstances. In couple of papers, number of neurons is same as the component of the element descriptor separated from every pixel facilitates in a grey scale Image.

III PROPOSED SYSTEM

In this venture we propose a technique for Image colorization. We preprocess hued Images to make grayscale Images to use as the contribution for the model. Our model is then prepared with these grayscale Images as info and the first shaded Images as the result. In this way, in the event that we feed another concealed grayscale Image to the model, it would have the option to create a RGB Image with a sensible comprehension of the spatial relationship of variety with the intrinsic surface. Taking into account the pixel tone is exceptionally dependent on the elements of its neighboring pixels, utilization of CNN is a palatable choice for Image colorization. The state of having just a grayscale or highly contrasting Image, it is muddled to distinguish the specific tone. The data isn't enough for an network to assess the pixel tones. For example, consider a vehicle Image which is in dark structure, there are number of OK choices for vehicle tone. To figure a reasonable variety, we require more data to concentrate on the model to match a grayscale input Image to the same shade of the result Image. In the beyond couple of years, Convolutional neural network is one of the best learning-based models. CNN checked fabulous abilities in Image handling. In such way, CNN-based model is proposed by us for programmed Image colorization. By utilizing Convolutional Neural Networks, we chose to snare the issues of Image colorization to "daydream" what an info dark and white Image would show up after colorization. For preparing the network began with the ImageNet dataset and all Images were changed from the RGB variety space to the Lab variety space. Like the RGB variety space, the Lab variety space has three channels. However, not at all like the RGB variety space, Lab encodes variety data in an unexpected way.

Image colorization is the method involved with taking an information grayscale (high contrast) Image and afterward creating a result colorized Image that addresses the semantic varieties and tones of the information. we will use here today rather depends on profound learning. We will use a Convolutional Neural Network equipped for colorizing highly contrasting Images with results RGB variety space, the Lab variety space has three channels. The L channel encodes softness power just, a channel encodes green-red and the b channel encodes blue-yellow. Since the L channel encodes just the force, we can involve the L channel as our grayscale contribution to the network.

To implement our approach, we first trained a CNN-based autoencoder on a dataset of colored images which are converted into grayscale images. We used the autoencoder to learn a compressed representation of the grayscale images and then used it to reconstruct the original images with color. We used the L2 loss function to optimize the network during training.

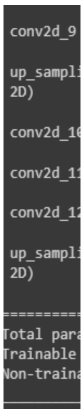
In this study, we will be using a dataset of coloured images for training and evaluating our approach to image colorization. The dataset consists of 10,000 images of various objects.

FIG 1: DATA SET

3.2 SYSTEM DESIGN

3.2.1 CNN Model

```
Model:
______
 Layer
======
 conv2d

 conv2d_

 conv2d_

 conv2d_

 conv2d_

 conv2d_

 conv2d_

 conv2d_

 conv2d_

 up_samp
 )
```



```
conv2d_9

up_sampli
2D)

conv2d_10

conv2d_11

conv2d_12

up_sampli
2D)

=========
Total para
Trainable
Non-traina
__________
```

FIG 2: CNN LAYERS

The information CIE Lab variety space is parted into two parts, Delicacy and a, b variety values. The lightness(L) channel is taken care of into the Encoder which contains convolutional layers that concentrate highlights while down examining to lessen how much calculation. We get a 128x128x128 component portrayal as result. The decoder part accepts this as info and applies up sampling and convolutional layers again to at last result 256 x 256 x2 highlights which are the an and b channels. To plan the result values between - 1 and 1, we utilize the tanh actuation capability and the ground truth values lie in range - 128 to 127 which are additionally standardized to - 1 to 1. The ground truth is contrasted with the result and the MSE misfortune capability refreshes the boundaries.

3.2.2 AUTOENCODER ARCHITECTURE

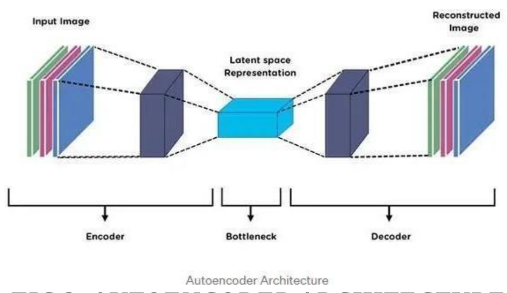


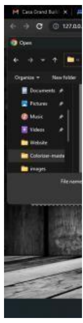
FIG 3: AUTOENCODER ARCHITECTURE

We execute this model precisely as characterized in the first paper [3]. The Encoder processes H x W grey scale Images, learns highlights utilizing convolutional layers while down sampling and yields a H/8 x W/8 x 512 element portrayal. This part is precisely similar to our past CNN model. The Origin Resnetv2 model is a pre-prepared model prepared on the huge ImageNet dataset. It is involved here as an undeniable level element extractor. We feed the information grayscale Image and concentrate the result of the last layer before the SoftMax capability which brings about a 1001x1x1 implanting. The combination layer this implanting, imitates it H/8 x W/multiple times and consolidates it with the element portrayal acquired from the encoder. Then we apply 256 convolutional portions of size 1x1, at last creating an element volume of aspect H/8 x W/8 x 256. At long last, the decoder takes this H/8 x W/8 x 256 volume and applies a progression of convolutional and up-examining layers to get a last layer with aspect HxWx2. Upsampling is performed utilizing the essential closest neighbor approach so the result's level and width are two times that of information. To plan the result values between - 1 and 1, we utilize the tanh actuation capability and the ground truth values lie in range - 128 to 127 which are additionally standardized to - 1 to 1. The misfortune capability utilized here is additionally Mean squared mistake (MSE) misfortune.

IV CONCLUSION

Image colorization is a particular PC illustrations action, yet it is likewise an illustration of an intense PC vision pixel forecast issue. At the point when we take a gander at a few examination distributions, we notice that the different techniques and procedures utilized can give results that are unclear from genuine nature Images. We have introduced a strategy for completely programmed colorization of interesting greyscale Images joining cutting edge CNN methods [5]. Utilizing variety portrayal and the right misfortune capability, we have addressed that the strategy is equipped for creating a conceivable and energetic colorization of specific pieces of Images, It was effectively ready to deliver conceivable varieties for the significant level parts in Images like sky, mountains and woods however didn't zero in on more modest subtleties. At present because of GPU norms our model can't be prepared with a more prominent number of Images. In this way, we prepared with the meager few so for later upgradation of the GPU we will attempt to get best result.

In conclusion, our paper presents a novel approach for image colorization using a CNN-based autoencoder. We demonstrated the effectiveness of our approach through experiments on a dataset of images and a comparative study with other available datasets. Our results show that our approach can generate realistic and accurate colorizations and outperforms existing methods in terms of color accuracy and realism. Future work can explore the application of our approach to other domains of computer vision.



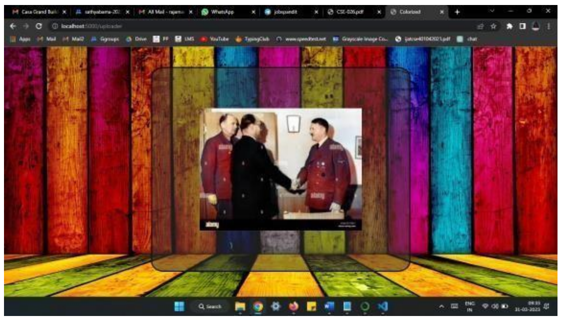


FIG 4: INPUT & OUTPUT

REFERENCES

[1] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. "Let there be Color! Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification". ACM Transaction on Graphics (Proc. of SIGGRAPH), 35(4):110, 2016.

[2] R. Zhang, P. Isola, and A. A. Efros, “Colorful image colorization,” in ECCV, 2016.

[3] Baldassarre, F., Morín, D. G., & Rodés-Guirao, L. (2017). Deep Koalarization: Image Colorization using CNNs and Inception-ResNet-v2. CoRR, abs/1712.03400. Retrieved from http://arxiv.org/abs/1712.03400.

[4] Jeff Hwang and You Zhou, “Image Colorization with Deep Convolutional Neural Networks”, Stanford University.

[5] David Futschik, “Colorization of black-and-white images using deep neural networks”, January 2018.

[6] Alex Avery and Dhruv Amin, “Image Colorization”.CS230 – Winter 2018.

[7] R. Zhang, P. Isola, and Alexei A. Efros. “Colorful Image Colorization”, Berkeley. Oct.5,2016.

[8] S. Kotala, S. Tirumalasetti, V. Nemitha, and S. Munigala. “Automatic Colorization of Black and White Images using Deep Learning”, Osmania University, Hyderabad, Telangana. April,2019.

[9] T. Nguye and R. Thawonmas. “Image Colorization Using a Deep Convolutional Neural Network”. April,2016.

[10] C. A. S. Domonkos Varga and T. Szirfffdfffdnyi, Automatic cartoon colorization based on convolutional neural network, https://core.ac.uk/download/pdf/94310076.pdf, 2017.

[11] S. Salve, T. Shah, V. Ranjane, and S. Sadhukhan, Automatization of coloring grayscale images using convolutional neural network, Apr. 2018. DOI: 10.1109/ICICCT. 2018.8473259.

[12] Automatic colorization of images from Chinese black and white films based on CNN, 2018. DOI: 10.1109/ICALIP. 2018.8455654

[13] D.Usha Nandini., Leni, A.E.S. “A survey of the existing shadow detection techniques Nandini”, International Conference on Control, Instrumentation, Communication and Computational Technologies, ICCICCT , , pp. 175– 177

[14] Gopinath, B., Kaliamoorthy, M., Ragupathy, U.S., ...Nandini, D.U., Maheswar, R.State-of-the- Art and Emerging Trends in Internet of Things for Smart Cities , , pp. 263–274

[15] V. K. Putri and M. I. Fanany, “Sketch plus colorization deep convolutional neural networks for photos generation from sketches,” in 2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), Sep. 2017, pp. 1–6. DOI: 10.1109/EECSI. 2017.8239116.