VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI-590018, KARNATAKA

A project report on

"OLD SCHOOL IMAGE AND VIDEO COLORIZATION USING DEEP LEARNING"

Submitted in partial fulfillment of the requirement for the degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE & ENGINEERING

Submitted by:

ABHINAV PRAKASH	**1TJ19CS001**
ANKIT AMAN	**1TJ19CS008**
BISHWAJIT PAUL	**1TJ19CS014**
CHARAN KUMAR K	**1TJ19CS015**

Under the guidance of:

Ms. SONIA DAS

Assistant Professor

Department of Computer Science and Engineering,
TJIT, Bengaluru

T. JOHN INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University)
#86/1, Gottigere, Bannerghatta Road, Bengaluru-560083

2022-2023

Affiliated to Visvesvaraya Technological University)
Approved by AICTE, Govt. of India, New Delhi.
No. 88/1, Gottigere, Bannerghatta Road, Bengaluru-560083

CERTIFICATE

This is certified that the project work entitled **"OLD SCHOOL IMAGE AND VIDEO COLORIZATION USING DEEP LEARNING"** carried out by **ABHINAV PRAKASH (1TJ19CS001), ANKIT AMAN (1TJ19CS008), BISHWAJIT PAUL (1TJ19CS014), CHARAN KUMAR K (1TJ19CS015),** bonafide student of **T. John Institute of Technology, Bangalore** in partial fulfillment for eight semesters of **Bachelor of Engineering** in **Computer Science & Engineering** of **Visvesvaraya Technological University, Belagavi** during the year **2022-2023**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the Bachelor of Engineering Degree.

Signature of Guide	Signature of HOD	Signature of Principal
Ms. Sonia Das	**Ms. Suma R**	**Dr. P. Suresh Venugopal**
Assistant Professor	Associate Professor & Head	Principal, TJIT,
Dept. of CSE, TJIT, Bengaluru.	Dept. of CSE, TJIT, Bengaluru.	Bengaluru.

Name of Examiners	**Signature of Examiners**
1._____	_____
2._____	_____

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of several individuals whose professional guidance and encouragement helped me in the successful completion of this project report work.

We take this opportunity to express our gratefulness to everyone who has extended their support for helping me in the project completion. First and foremost, we thank God, the Almighty for his abundant grace and blessings.

First and foremost, we thank **Dr. Thomas P. John**, Chairman of the T. John Group of Institutions, **Dr. T. D. Kemparaju**, Campus Director, T. John Group of Institutions, and **Dr. P. Suresh Venugopal**, Principal, T. John Institute of Technology for giving us this opportunity to study in this prestigious institute and also providing us with the best of facilities. We would like to show our greatest appreciation to **Dr. Srinivasa H P**, Vice-Principal, and **Ms. Suma R**, Associate Professor, HOD, Dept. of CSE

We would like to thank our Project Guide **Ms. Sonia Das, Assistant Professor**, Dept. of CSE for constantly guiding us throughout the Project. Her overall direction, encouragement, inspiration, and guidance have been responsible for the successful completion of the project.

We would like to thank all faculty members of the Department of Computer Science & Engineering, TJIT, and my friends for their constant support and encouragement. Finally, we would like to thank our parents for their constant support and encouragement.

ABHINAV PRAKASH	**1TJ19CS001**
ANKIT AMAN	**1TJ19CS008**
BISHWAJIT PAUL	**1TJ19CS014**
CHARAN KUMAR K	**1TJ19CS015**

DECLARATION

We, hereby declare that the project entitled **"OLD SCHOOL IMAGE AND VIDEO COLORIZATION USING DEEP LEARNING"** has been carried out and submitted by us in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science & Engineering**, from **Visvesvaraya Technological University, Belagavi** during the academic year **2022- 2023,** under the guidance of **Ms. Sonia Das,** Assistant Professor, Department of Computer Science & Engineering, T John Institute of Technology, Bengaluru.

We further declare that, to the best of our knowledge and belief, the work reported here is accepted and satisfied and has not been submitted previously for the award of any degree or diploma by us to any other university or institution.

Name	USN	Signature
1) ABHINAV PRAKASH	1TJ19CS001	_____
2) ANKIT AMAN	1TJ19CS008	_____
3) BISHWAJIT PAUL	1TJ19CS014	_____
4) CHARAN KUMAR K	1TJ19CS015	_____

ABSTRACT

Colorization Old-school image and video colorization using deep learning is a technique that employs neural networks to automatically add color to black-and-white or sepia-toned images and videos. This approach uses the power of deep learning algorithms to analyze the context and characteristics of the input images or videos and predict and generate the corresponding color information.

The process of old-school image and video colorization using deep learning involves training a neural network model on a large dataset of colored and grayscale images or videos. The model learns to identify patterns and relationships between the grayscale and colored versions of the same images or videos and uses this knowledge to predict the colors of new images or videos.

One of the key advantages of old-school image and video colorization using deep learning is its ability to capture and replicate the nuances of color and lighting in different scenes. This technique can accurately recreate the colors of objects and environments, and produce natural-looking results that closely resemble the original color photographs or footage.

There are several challenges associated with old-school image and video colorization using deep learning, including the need for high-quality training data, the complexity of the neural network architecture, and the time and computational resources required for training and inference. However, ongoing advancements in deep learning and computer vision are improving the accuracy and efficiency of this technique, making it a valuable tool for various applications, including film restoration, historical preservation, and visual effects in the entertainment industry.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Old-school image and video colorization is the process of adding color to black-and-white or sepia-toned images and videos to make them look more realistic and vivid. This technique has been used for decades through manual coloring methods, but the recent advancements in deep learning have enabled the development of automated techniques that can quickly and accurately colorize images and videos.

Deep learning is a subfield of machine learning that uses artificial neural networks to learn from data and make predictions or decisions. Deep learning algorithms have revolutionized image and video processing, enabling the development of sophisticated techniques that can extract and analyze complex features in visual data.

Old-school image and video colorization using deep learning is a popular application of this technology, as it can generate high-quality, colorized images and videos that are faithful to the original scene. This technique has numerous applications, including historical preservation, film restoration, and visual effects in the entertainment industry.

Consider the grayscale photographs in Figure 1. At first glance, hallucinating their colors seems daunting, since so much of the information (two out of the three dimensions) has been lost. Looking more closely, however, one notices that in many cases, the semantics of the scene and its surface texture provide ample cues for many regions in each image: the grass is typically green, the sky is typically blue, and the ladybug is most definitely red. Of course, these kinds of semantic priors do not work for everything, e.g., the croquet balls on the grass might not, in reality, be red, yellow, and purple (though it's a pretty good guess). However, for this paper, our goal is not necessarily to recover the actual ground truth color but rather to produce a plausible colorization that could potentially fool a human observer. Therefore, our task becomes much more achievable: to model enough of the statistical dependencies between the semantics and the textures of grayscale images and their color versions in order to produce visually compelling results.

Given the lightness channel L, our system predicts the corresponding a and b color channels of the image in the CIE Lab colorspace. To solve this problem, we leverage large-scale data. Predicting color has the nice property that training data is practically free: any color photo can be used as a training example, simply by taking the image's L channel as input and its ab channels as the supervisory signal. Others have noted the easy availability of training data, and

previous works have trained convolutional neural networks (CNNs) to predict color on large datasets. However, the results from these previous attempts tend to look desaturated. One explanation is that use loss functions that encourage conservative predictions. These losses are inherited from standard regression problems, where the goal is to minimize Euclidean error between an estimate and the ground truth.

We instead utilize a loss tailored to the colorization problem. As pointed out by, color prediction is inherently multimodal many objects can take on several plausible colorizations. For example, an apple is typically red, green, or yellow, but unlikely to be blue or orange. To appropriately model the multimodal nature of the problem, we predict a distribution of possible colors for each pixel. Furthermore, we re-weight the loss at training time to emphasize rare colors. This encourages our model to exploit the full diversity of the large-scale data on which it is trained. Lastly, we produce a final colorization by taking the annealed mean of the distribution. The end result is colorizations that are more vibrant and perceptually realistic than those of previous approaches.

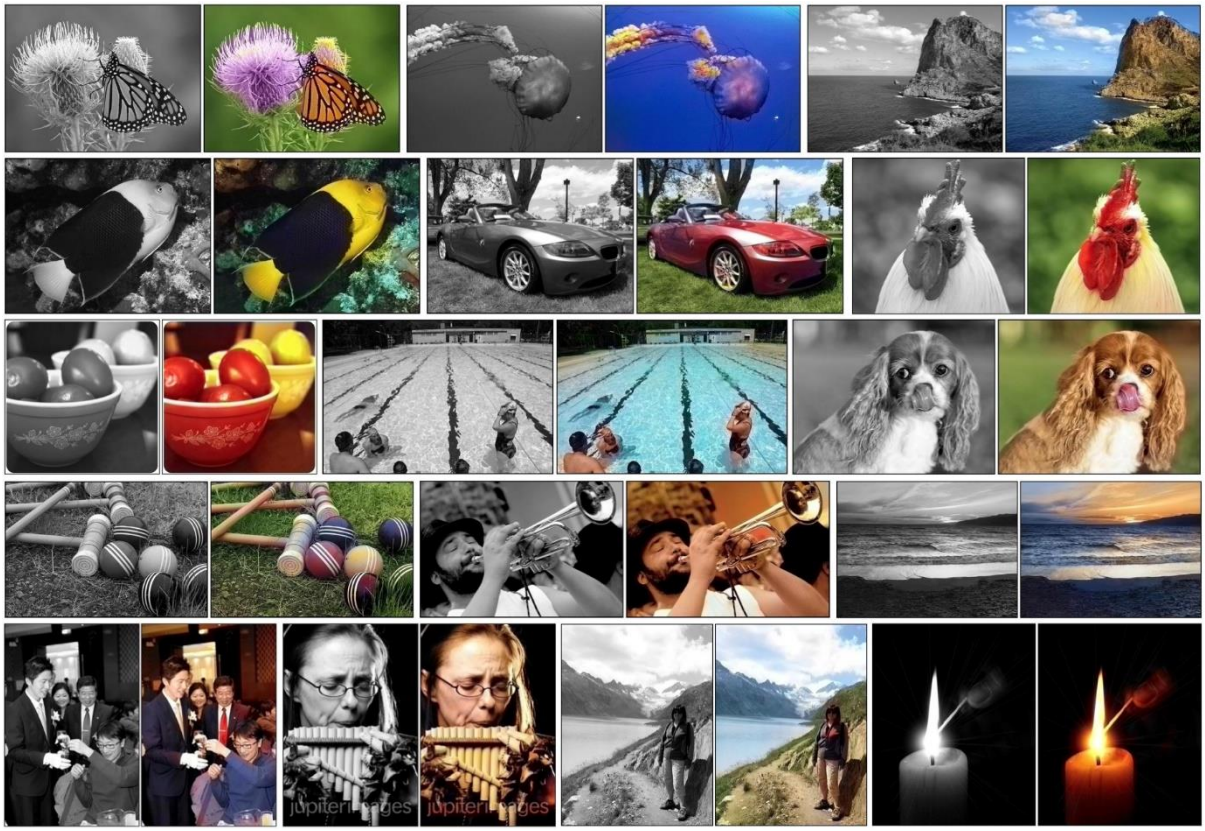


Fig 1.1: Grayscale photos and output colorizations

1.1DEFINITION

Old school image and video colorization refers to the process of adding color to black and white or sepia-toned images and videos to create a more realistic and vivid appearance. This technique has been used for many years, typically through manual methods such as painting or tinting, but has recently been automated using deep learning algorithms. The process involves analyzing the characteristics and context of the grayscale image or video and using this information to generate corresponding color information. The goal is to produce a colorized version that closely resembles the original scene, with accurate colors and lighting. This technique has many applications in areas such as film restoration, historical preservation, and visual effects in the entertainment industry.

1.2MOTIVATION

The motivation behind old school image and video colorization is to breathe new life into historic photographs and film footage, making them more accessible and engaging to modern audiences. By adding color to black and white or sepia-toned images and videos, this technique can help bridge the gap between the past and the present, allowing people to better visualize historical events and scenes in a more immersive and realistic way.

Additionally, old school image and video colorization can be used to preserve and restore historic footage, which may have been degraded over time due to wear and tear or exposure to the elements. By colorizing the footage, it becomes easier to identify and interpret details that may have been lost or obscured in the original black and white version.

The automation of old school image and video colorization using deep learning algorithms has made the process more accessible and efficient, allowing it to be applied to a wider range of images and videos. This has opened up new opportunities for using this technique in various fields, including film restoration, historical preservation, and entertainment.

CHAPTER 2

LITERATURE REVIEW

he objective of conducting a literature survey on old school image and video colorization is to gather existing knowledge, research, and techniques related to the topic.

2.1. Image Colorization

Authors: Richard Zhang, Phillip Isola, Alexei A. Efros

- We train a CNN to map from a grayscale input to a distribution over quantized color value outputs using the architecture.
- we focus on the design of the objective function, and our technique for inferring point estimates of color from the predicted color distribution.[1]

2.2. A Unified Framework for Multi-Modal Colorization with Transformer

Authors: ZHITONG HUANG, NANXUAN ZHAO, JING LIAO

- We propose the first unified framework UniColor to support colorization in multiple modalities, including both unconditional and conditional ones, such as stroke, exemplar, text, and even a mix of them.
- Rather than learning a separate model for each type of condition, we introduce a two-stage colorization framework for incorporating various conditions into a single model.[2]

2.3. Automatic Image and Video Colourisation using Deep Learning

Authors: Brian Sam Thomas, Rajat Dogra, Bhaskar Dixit, Aditi Raut

- They design and build a Convolutional Neural Network (CNN)- Long Short-Term Memory (LSTM) that accepts a black and white image as an input and generates a colourised version of the image as its output.
- The colouring information is stored by the LSTM, and provided for colouring the next images.
- This system colours an image solely on images it has learnt from in the past, with no

direct human intervention.[3]

2.4. Two-stage Sketch Colorization

Authors: LVMIN ZHANG, CHENGZE LI, TIEN-TSIN WONG, CHUNPING LIU

- Our framework colorizes a sketch with two stages. The first stage renders the sketch into a rough color draft image. To refine the color draft, we introduce a second stage to identify mistakes and refine them, in order to obtain the final result.
- They build an interactive software based on our model for evaluation. Users can iteratively edit and refine the colorization.[4]

2.5. Palette-based Photo Recoloring

Authors: Huiwen Chang, Ohad Fried, Yiming Liu, Stephen DiVerdi, Adam Finkelstein

- Our approach specifies both the colors to be manipulated and the modifications to these colors via a color palette a small set of colors that digest the full range of colors in the image.
- They perform a study showing that with our tool untrained users can produce similar results to those of expert Photoshop users.[5]

2.6. Image Colorization with Deep Convolutional Neural Networks

Authors: Jeff Hwang, You Zhou

- They present a convolutional-neural-network-based system that faithfully colorizes black and white photographic images without direct human assistance
- They explore various network architectures, objectives, color spaces, and problem formulations.[6]

2.7. ImageNet Classification with Deep Convolutional Neural Networks

Authors: Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton

- They trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes.

- On the test data, we achieved top-1 and top-5 error rates of 36.5% and 16.0% which is considerably better than the previous state-of-the-art.[7]

2.8. Image Colorization Using Similar Images

Authors: Raj Kumar Gupta, Alex Yong-Sang Chia, Deepu Rajan, Ee Sin Ng, Huang Zhiyong

- They present a new example-based method to colorize a gray image.
- The user needs only to supply a reference color image which is semantically similar to the target image.
- They extract features from these images at the resolution of superpixels, and exploit these features to guide the colorization process.[8]

2.9. Deep Residual Learning for Image Recognition

Authors: Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun

- They present a residual learning framework to ease the training of networks that are substantially deeper than those used previously.
- They explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions
- They provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth.[9]

2.10. Learning Representations for Automatic Colorization

Authors: Gustav Larsson, Michael Maire, and Gregory Shakhnarovich

- Our approach leverages recent advances in deep networks, exploiting both low-level and semantic representations.
- As many scene elements naturally appear according to multimodal color distributions, we train our model to predict per-pixel color histograms.
- This intermediate output can be used to automatically generate a color image, or further manipulated prior to image formation.[10]

Table 2.1: Literature Survey

Sl. No.	TITLE	YEAR	AUTHOR	FACTS
1	Image Colorization	2016	Richard Zhang, Phillip Isola, Alexei A. Efros	It shows results on legacy black and white photographs from renowned photographers Ansel Adams and Henri Cartier-Bresson, along with a set of miscellaneous photos.
2	Unified Framework for Multi-Modal Colorization with Transformer	2022	ZHITONG HUANG, NANXUAN ZHAO, JING LIAO	It presents a method for unifying multi-modal conditions, including stroke, exemplar, and text, by taking hint points as an intermediate representation. Especially, we introduce a novel CLIP-based method for text-to-hint-point conversion
3	Automatic Image and Video Colourisation using Deep Learning	2018	Brian Sam Thomas, Rajat Dogra, Bhaskar Dixit, Aditi Raut	A network that has demonstrated excellence with the vast number of classes present in the ImageNet dataset would serve well as the foundational basis of our network. The reason this works well is because Image subject matter implies colour palette, almost always.
4	Two-stage Sketch Colorization	2018	LVMIN ZHANG, CHENGZE LI, TIEN-TSIN WONG,	Users can iteratively edit and refine the colorization. We evaluate our learning model and the interactive system through an extensive user study. Statistics shows that our method outperforms the state-of-art

			CHUNPING LIU	techniques and industrial applications in several aspects including, the visual quality, the ability of user control, user experience, and other metrics.
5	Palette-based Photo Recoloring	2015	Huiwen Chang, Ohad Fried, Yiming Liu, Stephen DiVerdi, Adam Finkelstein	Our approach specifies both the colors to be manipulated and the modifications to these colors via a color palette a small set of colors that digest the full range of colors in the image. Given an image, it generates a suitable palette.
6	Image Colorization with Deep Convolutional Neural Networks	2016	Jeff Hwang, You Zhou	It takes a statistical-learning-driven approach towards solving this problem. It also designs and build a convolutional neural network (CNN) that accepts a black-and-white image as an input and generates a colorized version of the image as its output.
7	ImageNet Classification with Deep Convolutional Neural Networks	2012	Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton	It trained the models using stochastic gradient descent with a batch size of 128 examples, momentum of 0.9, and weight decay of 0.0005.
8	Image Colorization Using Similar Images	2012	Raj Kumar Gupta, Alex Yong-Sang Chia, Deepu Rajan, Ee Sin	It evaluate the method on a diverse range of images comprising portrait, painting, landscape as well as on images containing deformable and rigid foreground objects.

			Ng, Huang Zhiyong	
9	Deep Residual Learning for Image Recognition	2015	Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun	Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously.
10	Learning Representations for Automatic Colorization	2016	Gustav Larsson , Michael Maire , and Gregory Shakhnarovich	Colorization of grayscale images is a simple task for the human imagination. A human need only recall that sky is blue and grass is green; for many objects, the mind is free to hallucinate several plausible colors.

CHAPTER 3

PROBLEM STATEMENT

3.1 EXISTING SYSTEM

Old-school image and video colorization is an intriguing area of research that aims to bring new life to historical visual content by adding colors. Over the years, several systems and techniques have been developed to automate and enhance the process of colorizing black and white or grayscale images and videos. Here, we provide a brief overview of some notable existing systems in this field:

3.1.1 Colorful Image Colorization: Proposed by Zhang et al. in 2016, this system employs deep learning techniques, specifically convolutional neural networks (CNNs), to automatically colorize grayscale images. It learns from a large dataset of colored images to predict plausible colorizations, bringing vibrant hues to previously monochromatic visuals.

3.1.2 DeOldify: DeOldify, an open-source project initiated by Jason Antic, has gained significant popularity for its impressive results in colorizing and restoring old photos and videos. It utilizes a combination of generative adversarial networks (GANs) and historical references to produce realistic and visually appealing colorizations, often surpassing the capabilities of other methods.

3.1.3 Automatic Colorization of Grayscale Images: Developed by Iizuka et al. in 2016, this system leverages deep learning and user guidance to automatically colorize grayscale images. It employs a CNN-based approach where users provide hints or inputs to guide the colorization process, resulting in accurate and customizable colorizations.

3.1.4 Video Colorization: Colorizing videos is a more complex task due to the temporal coherence that needs to be maintained across frames. Vondrick et al. proposed a technique in 2018 that learns temporal color coherence from large-scale video datasets. By utilizing deep learning methods, this system propagates colors across video frames, ensuring consistent and visually pleasing colorizations throughout the video sequence.

3.1.5 Style-Based Generative Adversarial Networks (GANs): StyleGAN and its variants have also been employed in the domain of old school image and video colorization. These models excel at capturing the style and content of an image, enabling the generation of diverse and realistic colorizations.

It is important to note that the field of old school image and video colorization is dynamic, with ongoing research and advancements. New techniques and systems may have emerged since the time of writing, requiring further exploration and investigation. Conducting a comprehensive literature survey will provide a more detailed understanding of the state-of-the-art and help researchers and practitioners stay up to date with the latest developments in this exciting field.

3.1.6 Advantages

- **Automation and Efficiency:** Existing systems on old school image and video colorization utilize advanced algorithms and deep learning techniques to automate the colorization process. This automation saves time and effort compared to manual colorization methods, making it more efficient for colorizing large volumes of images or videos.
- **Restoration and Enhancement:** The use of existing systems can help restore and enhance old-school images and videos, bringing them closer to their original appearance or improving their visual quality. These systems can fill in missing details, reduce noise, and enhance the overall aesthetic appeal of the colorized content.
- **Realism and Accuracy:** Many existing systems aim to produce realistic and accurate colorizations by leveraging historical references or training on large datasets of colored images. By considering color patterns, cultural context, and visual cues, these systems strive to create colorizations that closely resemble the original colors, contributing to a more authentic representation of the historical visuals.
- **Customization and Control:** Some existing systems allow users to provide inputs or guidance during the colorization process. This feature gives users control over the colorization results, allowing for customization based on personal preferences or historical knowledge. Users can adjust color tones, highlight specific details, or ensure color consistency, resulting in more tailored colorizations.

3.1.7 Disadvantages

- **Subjectivity and Interpretation:** Existing systems rely on algorithms and training data, which may introduce biases or subjective interpretations of color choices. The colorization process involves making assumptions and decisions that may not align with the original artistic intent or historical accuracy, potentially leading to misrepresentations or inaccuracies.

- **Dependence on Training Data:** The performance and accuracy of existing systems heavily rely on the quality and diversity of the training data they are trained on. Insufficient or biased training data can result in colorizations that do not accurately represent the intended colors or cultural context of the lllllllll visuals.
- **Challenges with Complex Scenes:** Existing systems may face challenges when colorizing complex scenes or images/videos with intricate details. Fine textures, intricate patterns, or areas with ambiguous color information can pose difficulties for automated algorithms, leading to less accurate or plausible colorizations.
- **Technical Limitations:** Existing systems may have technical limitations, such as limitations in computational resources or hardware requirements. High-quality colorization often demands significant computational power, which may restrict access or hinder real-time colorization for certain applications or users with limited resources.

It is important to consider these advantages and disadvantages when using existing systems on old school image and video colorization. Understanding the limitations and potential biases associated with these systems can help researchers, practitioners, and viewers critically evaluate the colorization results and ensure responsible use of these technologies.

3.2 PROPOSED SYSTEM

A proposed system for old school image and video colorization using deep learning would involve several components:

3.2.1 Data Acquisition: The first step is to gather a large and diverse dataset of black and white or sepia-toned images and videos. This dataset would be used to train the deep learning model.

3.2.2 Pre-processing: The raw data would then be pre-processed to prepare it for use in the deep learning model. This may involve resizing the images or videos, converting them to grayscale, and normalizing the pixel values.

3.2.3 Model Training: A deep learning model would then be trained on the pre-processed dataset. There are several approaches to model training, such as using convolutional neural networks (CNNs) or generative adversarial networks (GANs). The objective of the training process is to teach the model to recognize patterns and relationships between grayscale and color images or videos.

3.2.4 Colorization: Once the model is trained, it can be used to colorize new grayscale images or videos. This is done by feeding the grayscale input into the trained model, which outputs the corresponding colorized version.

3.2.5 Postprocessing: The colorized images or videos may require some postprocessing to refine the results and remove any artifacts or errors. This may involve smoothing the color gradients or adjusting the color balance.

3.2.6 Evaluation: Finally, the quality of the colorized images or videos can be evaluated using metrics such as PSNR or SSIM, and compared to the ground truth or original color versions.

The proposed system for old school image and video colorization using deep learning is an automated and efficient approach that can produce high-quality results. It has applications in areas such as film restoration, historical preservation, and entertainment, and can help bring historic imagery to life in a more vivid and engaging way.

3.2.7 Advantages

- Colourise Turing test.
- Does not require any external hardware implementation.
- System generated results prone to less error.
- Reduces human effort and intervention.
- Accuracy in prediction.
- Drastically reduces time compared to manual detection.
- Flexible and Fast.
- Easy to use.

3.2.8 Disadvantages

- The accuracy result is not up to the expected level.
- Required accuracy is not obtained.
- The produced image may not be exactly like the desired image.
- The proposed system is not effective in over saturated images
- The shadows may get decreased.

CHAPTER 4

OBJECTIVE OF THE PROJECT

The objective of "old school image and video colorization" is to add color to black and white or grayscale images or videos. This can be done manually or with the use of software and algorithms that attempt to automatically determine the correct colors for the image or video.

One reason for colorizing images and videos is to make them more visually appealing and engaging. Black and white images and videos can be perceived as dull and lacking in contrast, so adding color can help to bring them to life and make them more interesting to look at.

Another reason for colorizing images and videos is to restore or preserve historical and cultural information. Many images and videos from the past were captured in black and white, and adding color can help to give a sense of what these scenes may have looked like at the time they were captured. This can be particularly important for historical documents and artifacts, as it can help to provide a more accurate representation of the time period.

Overall, the main objective of image and video colorization is to enhance the visual appeal and cultural value of these media.

4.1 AIM

- The aim of old school image and video colorization is to add color to black and white images or videos.
- This can be done for a variety of reasons, including to restore images or videos that have been damaged or faded over time, to make historical images or videos more accessible to modern audiences, or simply to add a creative touch to a work of art.
- The goal is to produce a color version of the image or video that is as true to the original as possible, while also taking into account the artistic intentions of the creator.
- In some cases, the goal of colorization is to produce a result that is as close as possible to the original, fully-colored version of the image or video.
- In other cases, the goal may be to produce a stylized or artistic version of the image or video, with colors that are enhanced or modified in some way.

CHAPTER 5

SYSTEM REQUIREMENTS

Old school image and video colorization systems require specific system requirements to ensure effective and efficient performance. These requirements include hardware resources such as a powerful CPU or GPU, sufficient RAM, and storage space for handling large datasets and complex algorithms. Software frameworks and libraries, such as deep learning frameworks like TensorFlow or PyTorch and image processing libraries like OpenCV, are essential for implementing the colorization algorithms. Acquiring and preparing suitable datasets of old school images and videos is crucial, along with the availability of training and inference infrastructure to support computationally intensive tasks. The choice of colorization algorithms and models, whether based on convolutional neural networks (CNNs) or generative adversarial networks (GANs), should align with the system's capabilities. User interface and interaction should be designed to allow users to input grayscale images or videos, customize colorization parameters, and visualize the results. Validation and evaluation mechanisms, including benchmark datasets and metrics, ensure the quality and accuracy of the colorization outputs. Scalability and performance considerations are important for handling large-scale datasets and real-time colorization requirements. Lastly, ethical considerations should be addressed, including proper documentation, cultural sensitivity, and responsible handling of historical content.

5.1 HARDWARE REQUIREMENTS

It is a list of physical components of a computer system which are used for project implementation and it gives the short description of the computer system configuration used, required amount of memory, quality of display and the other peripheral devices needed for project.

- Processor: Minimum Intel Core i3
- Memory: 4GB RAM
- Disk Space: 8GB Hard Disk Memory
- Input Device: Standard Keyboard and Mouse
- Output Device: High-Resolution Monitor

5.2 SOFTWARE REQUIREMENT

It is a list of the most necessary software components and packages which are used in project implementation. In a few words describes the types of operating system that is required and its version along with other required details.

- Operating System: Windows 6/8/9/10/11
- Programming language: Python
- Library: OpenCV, numpy, imutils
- Simulation tool: Anaconda Navigator IDE 3.6.4.(Jupyter Notebook) and Visual Studio Code.

5.3 FUNCTIONAL REQUIREMENT

Functional requirements cover the requirements that the user and system need. These requirements are a list of necessary functionalities that should there with respect to system and user.

- Image and Video Input: The system should be able to accept grayscale or black and white images and videos as input for colorization. It should support various file formats commonly used for images and videos.
- Colorization Algorithm: The system should implement a colorization algorithm or a combination of algorithms capable of accurately and realistically adding color to the input images or videos. The algorithm should be able to handle different types of images and videos, including those with complex scenes or fine details.
- Preprocessing and Enhancement: The system may incorporate preprocessing and enhancement techniques to prepare the input images or videos before colorization. This may include noise reduction, contrast adjustment, and other image or video enhancement methods to improve the colorization results.
- Training and Learning: If the system utilizes machine learning or deep learning techniques, it should include functionality for training the models on relevant datasets. This may involve providing labeled or annotated data to train the colorization algorithm effectively.

5.4 NON- FUNCTIONAL REQUIREMENT

Non-functional requirements cover all the remaining requirements which are not covered by the functional requirements. They specify criteria that judge the operation of a system, rather than specific behaviours.

- Performance: The system should exhibit efficient performance, processing images and videos in a timely manner. It should be capable of handling large datasets and real-time colorization requirements without significant delays or performance degradation
- Accuracy and Quality: The colorization outputs should be of high quality and accurately represent the intended colors. The system should minimize artifacts, inconsistencies, or distortions in the colorized images and videos, ensuring a realistic and visually pleasing result.
- Scalability: The system should be scalable, able to handle varying workloads and datasets of different sizes. It should adapt to increasing demands without compromising performance or functionality, allowing for efficient colorization of a growing number of images and videos.
- Robustness and Reliability: The system should be robust and resilient, capable of handling unexpected situations, errors, or disruptions without losing data or compromising the colorization process. It should recover gracefully from failures and ensure the integrity of the colorization results.

CHAPTER 6

SYSTEM DESIGN

The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and helps in defining overall system architecture. Design is a creative process where a good feature design is the key to successful system. System design is defined as “The process in which one can apply a variety of techniques and principles for the reason of defining a system or process in adequate detail to allow its physical considerate”. A variety of design features are followed to develop the system. The design specification describes the features of the system, the components or elements of the system and their appearance to end-users.

Design is a process of converting in to the computer-based format from the user-oriented inputs. The aim of designing input data is to make the automation as simple and error free as much as possible. To provide a good input design for the application, simple data input and selection features are adopted. The requirements of input design such as user friendliness, consistent format and interactive dialogue for giving the right message and help for the user at right time are also considered for the development of the project. Input design is a part of whole system design which requires very careful attention. Repeatedly the collection of input data is the costliest part of the system, which desires to be route through number of modules.

The project design can be explained in two types:

- High level design provides an outer structure of an entire system, identifying all of its elements at some level of abstraction.
- Low level design provides the detailed design of each of these elements.

6.1 Low-Level Design

The Low-Level Design (LLD) is the complete description of the high-level design it enables real logic for each and every system component. One can say it is a design at code level. Each applications document of low-level design should grant a complete and detailed blueprint of the design for the software that will be developed in the plan, including the classes, associations between classes, member and nonmember functions that are involved.

6.1.1 Data-Flow Diagram

The DFD is straightforward graphical formalism that can be utilized to speak to a framework as far as the info information to the framework, different preparing did on this information and the yield information created by the framework. A DFD model uses an exceptionally predetermined number of primitive images to speak to the capacities performed by a framework and the information stream among the capacities.

The principle motivation behind why the DFD method is so famous is most likely in light of the way that DFD is an exceptionally basic formalism- It is easy to comprehend and utilization. Beginning with the arrangement of abnormal state works that a framework performs, a DFD display progressively speaks to different sub capacities. Actually, any various levelled model is easy to get it. The human personality is such that it can without much of a stretch see any progressive model of a framework in light of the fact that in a various levelled model, beginning with an extremely straightforward and unique model of framework, distinctive points of interest of a framework are gradually presented through the diverse orders.

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

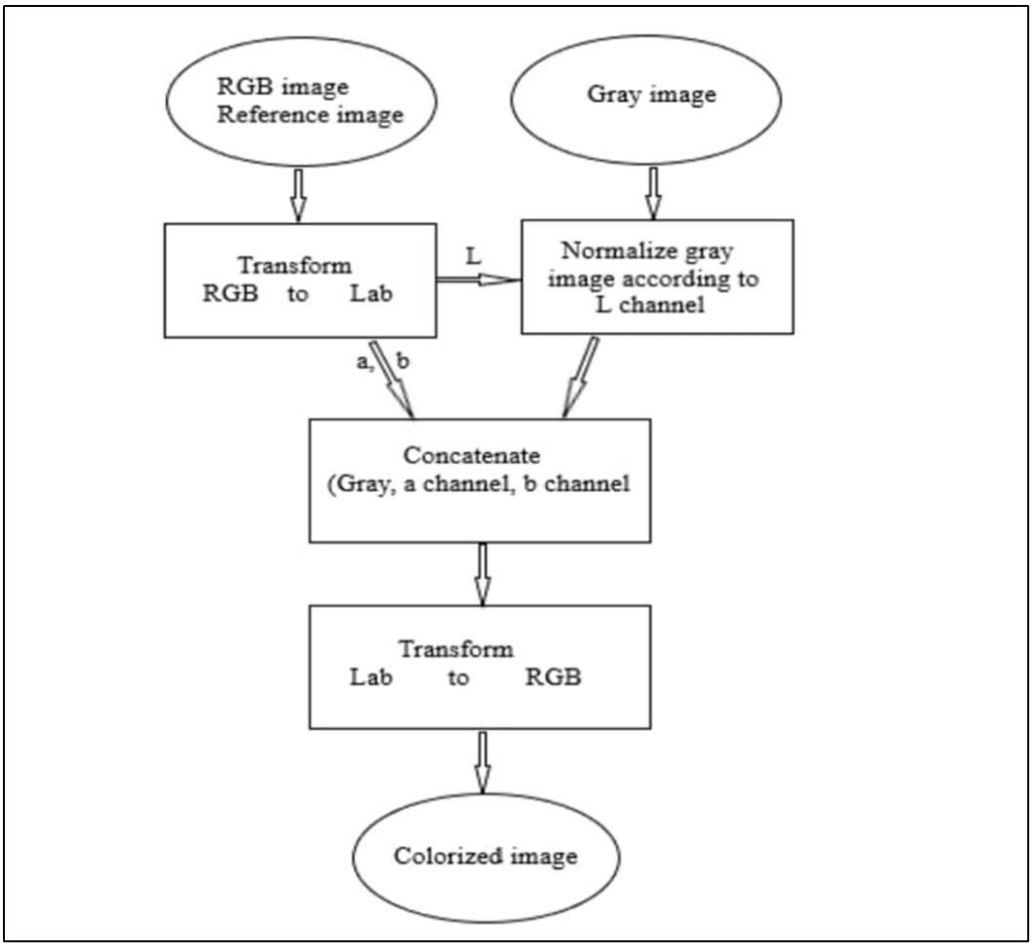


Fig. 6.1: Data-Flow Diagram of Monochromatic Image Colorization

6.1.2 Activity Diagram

Activity diagram is defined as UML diagram that focus on the execution and flow of the behaviour of a system instead of implementation. It is also called as object-oriented flow chart. Activity diagrams consists of activities that are made up of actions which apply to behavioral modelling technology. Activity diagrams are pictorial representations of workflows of step-by-step activities and actions with support for iteration, concurrency and choice. Activity diagrams show the entire flow of control.

The activity diagram is an UML diagram that describes the system's dynamic aspects. In fact, it is a flowchart that regulates the flow every event. The event can be described as the operation of the system. The control flow shall be taken between operations. Activity diagrams are in the form of flowchart. Typical techniques of flowchart have lack of constructs for expressing concurrency. However, the symbols of join and split only resolve this for simple cases in activity diagrams. The exact meaning of the model is not that much clear when they are arbitrarily combined with loops or decisions.

Activity diagrams are organized from shapes which are limited number, connected by arrows. The most important shape types:

- Decisions represented by diamonds.
- Start or end of concurrent activities represented by bars.
- The initial state or start of the workflow represented by black circle.
- Final or end state represented by an encircled black circle.
- Arrows flows from the start to the end and symbolize the flow in which activities happen.

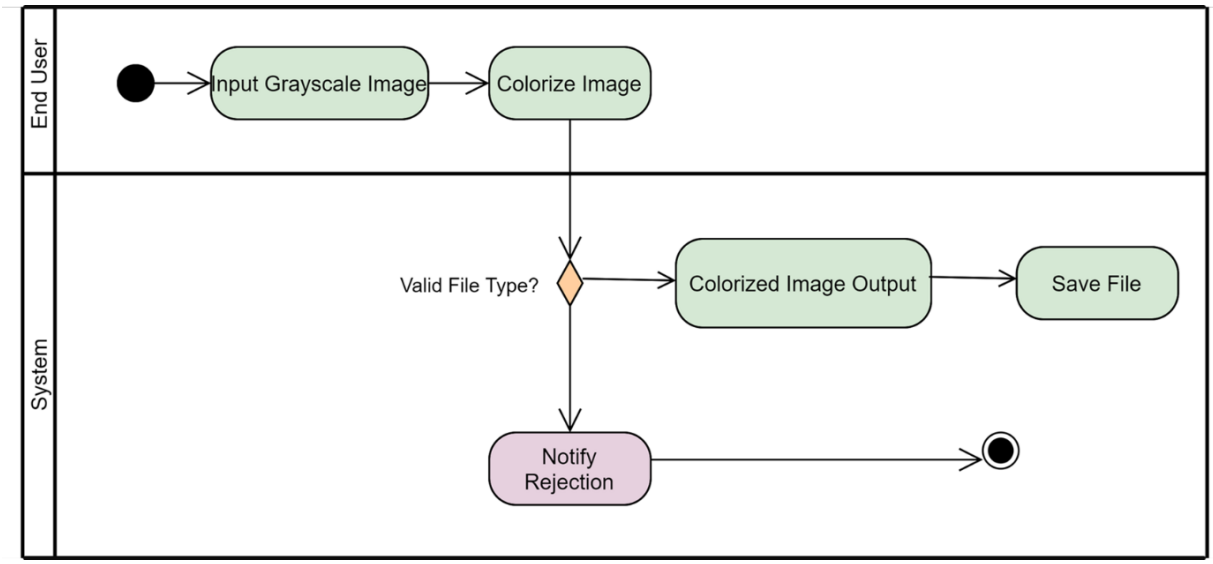


Fig. 6.2: Activity Diagram of Monochromatic Image Colorization

6.1.3 Sequence Diagram

Sequence diagrams are sometimes referred to as descriptions of actions or activities. It shows the messages shared between them in the order in which they occur as parallel vertical lines (lifelines), different processes or objects that exist continuously as horizontal arrows.

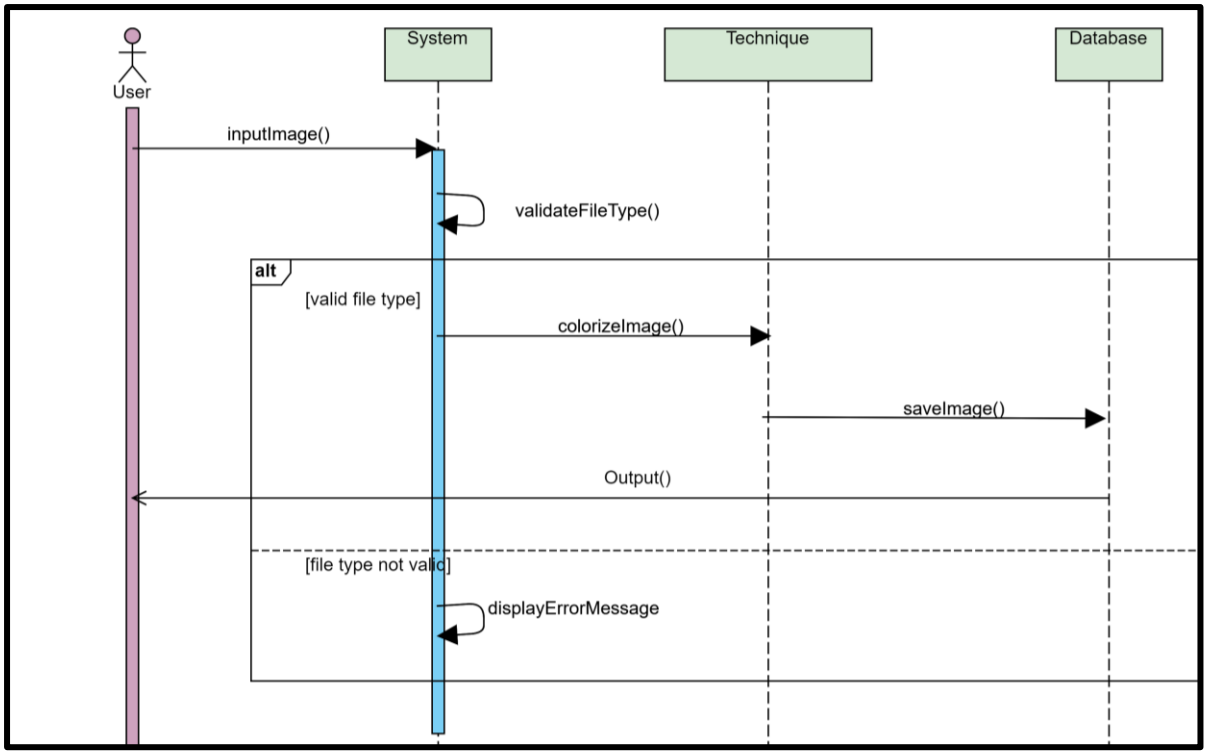


Fig. 6.3: Sequence Diagram of Monochromatic Image Colorization

6.1.4 Use-Case Diagram

A use case chart is a kind of behavioural graph made from a Use-case examination. Its object is to present a graphical diagram of the usefulness gave by a framework regarding performers, their objectives (spoke to as utilization cases), and any conditions between those utilization cases. Use case chart gives us the data about how that clients and utilization cases are connected with the framework. Use cases are used amid prerequisites elicitation and examination to speak to the usefulness of the framework. Use cases concentrate on the conduct of the framework from an outside perspective. A use case depicts a capacity gave by framework that yields an obvious result for a performer. A performing artist portrays any element that collaborates with the system. The performers are outside the limit of the framework, while the use cases are inside the limit of the framework. On-screen characters are spoken to with stick figures, use cases with ovals, and the limit of the framework with a container encasing the use cases.

A use case diagram describes the user's communication with the process. The relationship of user with different use cases is depicted. A use case diagram will explain various types of program users and use cases and is often accompanied by other diagram forms as well.

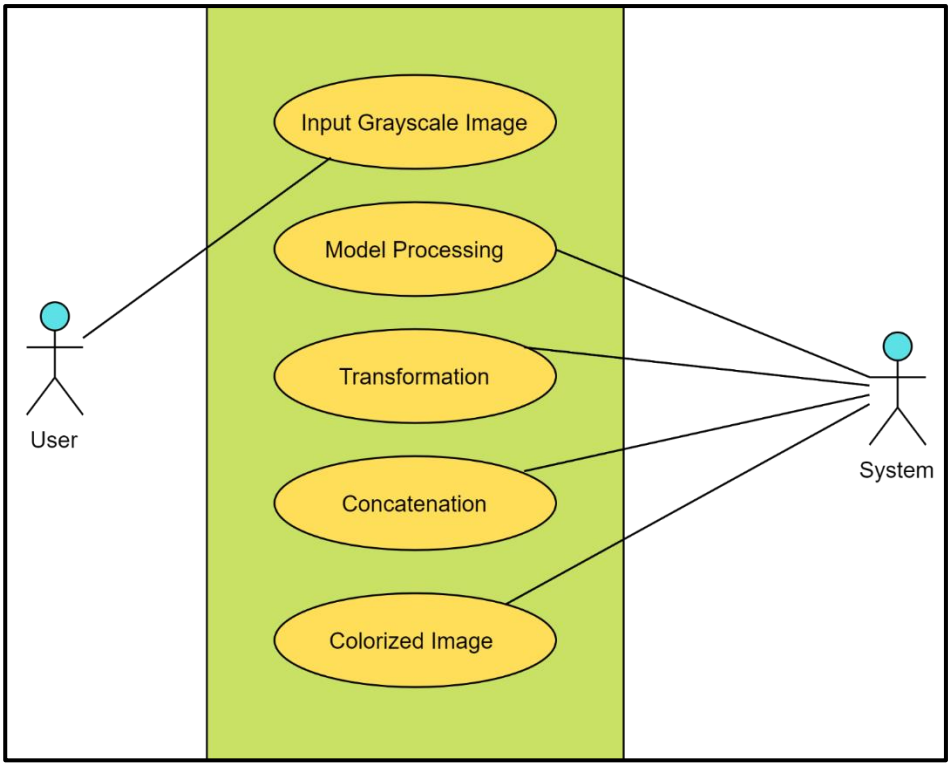


Fig. 6.4: Use-Case Diagram of Monochromatic Image Colorization

6.2 High-Level Design

The High-Level Design (LLD) design provides an overview of a whole system, identifying each and every element at some level of abstraction. High Level Design covers the system architecture and database design as overall system design. It explains the associate among variety of functions and modules of the system.

6.2.1 System Modules

The proposed system consists of 4 modules. They are:

- **Input:** Input will be a monochromatic image.
- **Model Processing:** It is done at Convolutional Layer. Here, all training images are converted from the RGB color space to the Lab color space. The convolutional layer converts the image into numerical values, allowing the neural network to interpret and extract relevant patterns.
- **Concatenation:** It is done at UpSampling Layer. UpSampling layer restores the size of image. Using the pre-trained model, the ab channels can be predicted. Combine the input L channel with the predicted ab channels. Having the trained model, we can use it to colorize a new B&W photo, where this photo will be the input of the model or the

component "L". The output of the model will be the other components "a" and "b", that once added to the original "L", will return a full colorized photo as shown here:

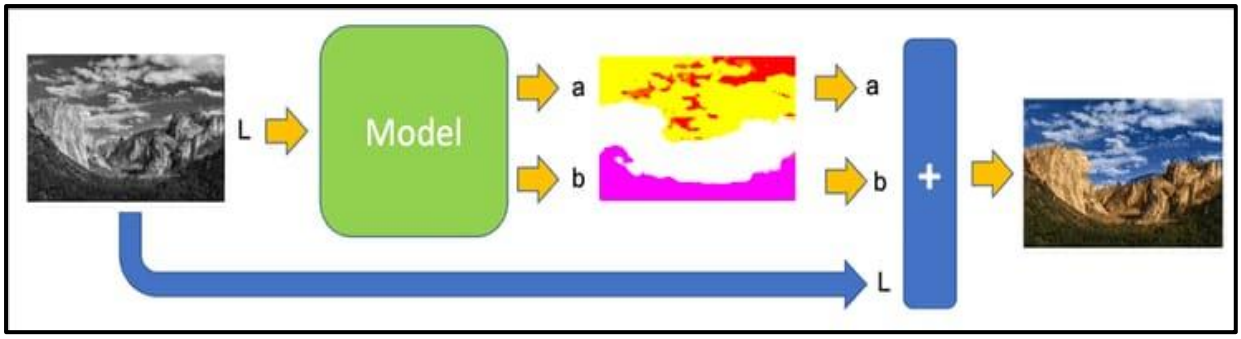


Fig. 6.5 System Design Modules

The entire (simplified) process can be summarized as:

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

6.2.2 System Architecture

The architectural configuration procedure is concerned with building up a fundamental basic system for a framework. It includes recognizing the real parts of the framework and interchanges between these segments. The beginning configuration procedure of recognizing these subsystems and building up a structure for subsystem control and correspondence is called construction modeling outline and the yield of this outline procedure is a portrayal of the product structural planning. The proposed architecture for this system is given below. It shows the way this system is designed and brief working of the system.

A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. A system architecture can consist of system components and the sub-systems developed, that will work together to implement the overall system.

Our architecture is based on the model which is a convolutional neural network based on Lab color space. And a pretrained data – ImageNet dataset, using Caffe model. Images on a

computer are typically represented in the RGB color model, in which the color of each pixel is determined by a 3-tuple ρ = (r, g, b) denoting the red, green, and blue components of that color respectively.

Like RGB, Lab is another color space. It is also three channel color space like RGB where the channels are:

- L channel: This channel represents the Lightness intensity
- a channel: This channel represents green-red
- b channel: This channel represents blue-yellow

  Steps for the process of the model is as:

- The grayscale part of the image is only encoded in L channel.
- The images are input to the VGG16 model. The model has two parts encoder responsible for features extraction and decoder for recreating network.
- The output generated is n dimensional vectors and it is flattened before passing to the clustering.

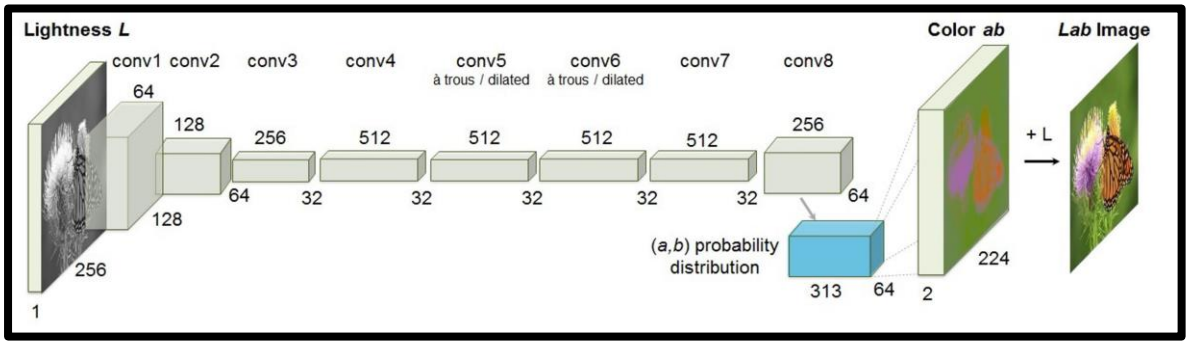


Fig. 6.6: VGG-16 model architecture

CHAPTER 7

IMPLEMENTATION

The implementation of an old-school image and video colorization system using deep learning involves several steps. First, a dataset of black-and-white or sepia-toned images and videos is acquired. The raw data is then pre-processed by resizing, converting to grayscale, and normalizing pixel values. A deep learning model, such as a convolutional neural network (CNN) or a generative adversarial network (GAN), is selected and trained on the pre-processed dataset to minimize the difference between the colorized output and the ground truth color image. The trained model is validated and can be used to colorize new grayscale images or videos. Post-processing is applied to refine the results, and the quality of the colorized images or videos is evaluated using metrics such as PSNR or SSIM. The implementation of an old-school image and video colorization system using deep learning requires significant computational resources and expertise but can produce high-quality colorization results useful for film restoration, historical preservation, and entertainment.

7.1 Models Used

We are using RGB model, CIELAB model, and VGG-16 model. We are also using pre-trained model- Caffe Model.

7.1.1 RGB Model:

The RGB color model is an additive color model in which red, green and blue light are added together in various ways to reproduce a broad array of colors. The name of the model comes from the initials of the three additive primary colors, red, green, and blue.

The main purpose of the RGB color model is for the sensing, representation, and display of images in electronic systems, such as televisions and computers, though it has also been used in conventional photography. Before the electronic age, the RGB color model already had a solid theory behind it, based in human perception of colors.

7.1.2 CIELAB Model:

The CIELAB color space (also known as CIE L*a*b* or sometimes abbreviated as simply "Lab" color space) is a color space defined by the International Commission on Illumination

(CIE) in 1966. It expresses color as three numerical values, L* for the lightness and a* and b* for the green–red and blue-yellow color components.

The color space L * a * b * was created after the theory of opposing colors, where two colors cannot be green and red at the same time, or yellow and blue at the same time. CIELAB was designed to be perceptually uniform with respect to human color vision, meaning that the same amount of numerical change in these values corresponds to about the same amount of visually perceived change.

Unlike the RGB color model, Lab color is designed to approximate human vision. It aspires to perceptual uniformity, and its L component closely matches human perception of lightness. The L component is exactly what is used as input of the AI model, that was train to estimate the remained components, "a" and "b".

7.1.3 VGG-16 Model:

The ImageNet dataset contains images of fixed size of 224*224 and have RGB channels. This model processes the input image and outputs values in vector.

It has limitations as:

- It is very slow to train (the original VGG model was trained on Nvidia Titan GPU for 2-3 weeks).
- The size of VGG-16 trained ImageNet weights is 528 MB. So, it takes quite a lot of disk space and bandwidth which makes it inefficient.
- 138 million parameters lead to exploding gradients problem.

7.1.4: Caffe Model:

Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by Berkeley AI Research (BAIR) and by community contributors. It is a pre-trained model stored in the Caffe framework’s format that can be used to predict new unseen data.

Caffe is used as it has advantages as:

- Expressive architecture encourages application and innovation.
- Extensible code fosters active development.
- Community: Caffe already powers academic research projects,

7.2 Algorithm Used

We are using CNN Algorithm for our proposed model.

7.2.1 About CNN Algorithm:

- A convolutional neural network, or CNN, is a deep learning neural network sketched for processing structured arrays of data such as portrayals.
- CNN are very satisfactory at picking up on design in the input image, such as lines, gradients, circles, or even eyes and faces. This characteristic that makes convolutional neural network so robust for computer vision.
- The strength of a convolutional neural network comes from a particular kind of layer called the convolutional layer.
- CNN contains many convolutional layers assembled on top of each other, each one competent of recognizing more sophisticated shapes.
- With three or four convolutional layers it is viable to recognize handwritten digits and with 25 layers it is possible to differentiate human faces.

7.2.2 CNN Design:

- The construction of a convolutional neural network is a multi-layered feed-forward neural network, made by assembling many unseen layers on top of each other in a particular order.
- It is the sequential design that give permission to CNN to learn hierarchical attributes.
- In CNN, some of them followed by grouping layers and hidden layers are typically convolutional layers followed by activation layers.
- The pre-processing needed in a ConvNet is kindred to that of the related pattern of neurons in the human brain and was motivated by the organization of the Visual Cortex.

7.2.3 CNN Process:

The Artificial Intelligent (AI) approach is implemented as a feed-forward pass in a CNN (" Convolutional Neural Network") at test time and is trained on over a million color images. In other words, millions of color photos were decomposed using Lab model and used as an input feature ("L") and classification labels ("a" and "b"). For simplicity let's split in two: "L" and "a+b" as shown in the block diagram:

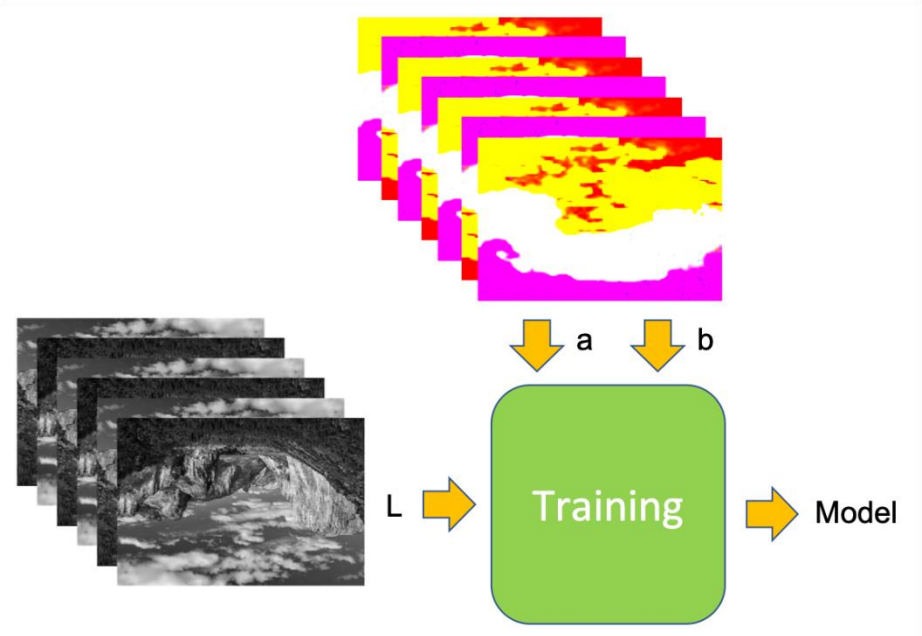


Fig. 7.1: CNN (Deep Learning Method) Process

7.3 Programming Language - Python

It is an object-oriented programming language. The processing happens during the runtime, and this is performed by the interpreter. Python's simple to learn and easy to use is an advantage and thus makes it developer friendly. It is easier to read and understand as the syntax is conventional. The code can be executed line by line using the interpreter. Python can support multiple platforms like Linux, UNIX, windows, Macintosh, and so on. The paradigms of Object-oriented programming are supported by python. The functions such as polymorphism, operator overloading and multiple inheritance is supported python.

Features of python are:

- Easy to Code: Python is a very high-level programming language, yet it is effortless to learn.
- Easy to Read: Python code looks like simple English words.
- Free and Open-Source: It is completely free to use, even for commercial purposes. It can also be freely modified and re-distributed.
- Interpreted: When a programming language is interpreted, it means that the
- Portable: Python is portable in the sense that the same code can be used on different machines. As such, there is no need to write a program multiple times for several platform.

- Object-Oriented and Procedure-Oriented: A programming language is object-oriented if it focuses design around data and objects, rather than functions and logic. On the contrary, a programming language is procedure-oriented if it focuses more on functions (code that can be reused). One of the critical Python features is that it supports both object-oriented and procedure-oriented programming.
- Extensible: A programming language is said to be extensible if it can be extended to other languages. Python code can also be written in other languages like C++, making it a highly extensible language.
- Expressive: Python needs to use only a few lines of code to perform complex tasks.
- Support for GUI: A user can easily interact with the software using a GUI. Python offers various toolkits which allows for GUI's easy and fast development.

7.4 SAMPLE CODES FOR IMAGE COLORIZATION

The provided code snippet imports two packages: numpy and cv2.

7.4.1 Import Liabraries

The numpy package is a fundamental library for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. In the context of computer vision and image processing, numpy is commonly used to manipulate and process pixel values in images and videos. It allows for efficient storage, indexing, and computation of numerical data, making it an essential package in many computer vision workflows.

The cv2 package is an abbreviation for OpenCV (Open Source Computer Vision Library). OpenCV is a widely-used open-source library for computer vision and image processing tasks. It offers a vast array of functions and algorithms to perform tasks such as image and video I/O (input/output), image filtering and enhancement, object detection and tracking, and more. By importing cv2, we gain access to these powerful capabilities provided by OpenCV, enabling us to perform various computer vision tasks efficiently and effectively.

Final_Gray2Color_Image.py > ...

```
1   # importing the packages
2   import numpy as np
3   import cv2
```

7.4.2 Pre-trained model for colorization

In the given code snippet, we are loading a pre-trained model for colorization using the OpenCV library.

First, the message "Pre-Trained Model Loading" is printed to indicate the start of the model loading process.

Next, the cv2.dnn.readNetFromCaffe() function is used to load the pre-trained model. This function takes two arguments: the path to the proto-txt file (model/colorization_deploy_v2.prototxt) and the path to the pre-trained caffemodel file (model/colorization_release_v2.caffemodel). The prototxt file defines the architecture of the network, while the caffemodel file contains the learned parameters and weights of the model.

The loaded model is assigned to the variable net. This variable represents the neural network model that will be used for colorization.

Additionally, the code snippet loads the pts_in_hull.npy file using np.load(). This file contains the ab color space quantization bins. The ab color space represents the color information in the image. The pts variable is assigned the loaded numpy array, which will be used later in the colorization process.

By loading the pre-trained model and the quantization bins, we are now ready to perform colorization on images using this model.

```
#------------------------------Part1: Pre-processing---------------------------------------#

#Step1-Loading our pre-trained model
print("Pre-Trained Model Loading")
net = cv2.dnn.readNetFromCaffe('model/colorization_deploy_v2.prototxt', 'model/colorization_release_v2.caffemodel')
pts = np.load('model/pts_in_hull.npy')
```

7.4.3 Loading the cluster centers

In this step, we are loading the cluster centers used for ab channel quantization and rebalancing in the colorization process.

The code snippet defines two layer IDs, class8 and conv8, which correspond to the layers in the pre-trained model responsible for handling the ab channels.

Next, the pts array is reshaped and transposed to match the required dimensions for the layer blobs. The pts array contains the ab color space quantization bins and is reshaped to have dimensions (2, 313, 1, 1). This shape represents two channels (a and b), with 313 bins for each channel.

Then, we access the layer with the ID class8_ab using net.getLayer(class8) and assign the

reshaped pts array as the layer's blobs. This sets the cluster centers used for quantization in the ab channels of the colorization model.

Similarly, we access the layer with the ID conv8_313_rh using net.getLayer(conv8) and assign a 1x313 array filled with the value 2.606 as the layer's blobs. This value is used for rebalancing the color distribution in the ab channels during the colorization process.

By setting the blobs of these layers, we ensure that the pre-trained model is properly initialized with the required cluster centers and rebalancing factors for accurate colorization of images.

```
#Step2-Load centers for ab channel quantization used for rebalancing(cluster centers as 1*1 convoltions)
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")]
```

7.4.4 Preparing the input image

In this step, we are preparing the input image for the colorization process by converting it to LAB color space.

First, the code snippet reads an image file named 'mark_twain.jpg' using the cv2.imread() function. The image is loaded as a BGR (Blue-Green-Red) color image.

Next, the image is scaled to a floating-point representation in the range [0, 1] by dividing each pixel value by 255.0. This scaling operation ensures that the pixel values are normalized to the range expected by the colorization model, which typically works with inputs in the range [0, 1] Then, the cv2.cvtColor() function is used to convert the scaled image from the BGR color space to the LAB color space. The LAB color space consists of three channels: L (lightness), a (green-red), and b (blue-yellow). The L channel represents the lightness intensity of the image, while the a and b channels represent the color information. Converting the image to the LAB color space separates the color information from the lightness information, which facilitates the subsequent colorization process.

After converting the image to the LAB color space, we can proceed with the colorization algorithm using the LAB image as input.

```
#Step3
image = cv2.imread('images/mark_twain.jpg') #take grayscale input image
scaled = image.astype("float32") / 255.0 #scaling in range[0,1]
lab = cv2.cvtColor(scaled, cv2.COLOR_BGR2LAB) #RGB to LAB color space
```

7.4.5 Preprocessing steps

In this step, we perform some preprocessing steps on the LAB image before feeding it into the colorization network.

First, the cv2.resize() function is used to resize the LAB image to the required input dimensions for our network, which are (224, 224) in this case. Resizing the image ensures that it matches the expected input size of the colorization network.

Next, we use the cv2.split() function to split the resized LAB image into its individual channels. Since we are interested in the L channel (representing the lightness), we extract it by accessing the first channel in the resulting list of channels.

After obtaining the L channel, we perform mean subtraction by subtracting 50 from each pixel value in the L channel. Mean subtraction is a common preprocessing step in computer vision tasks to normalize the input data. By subtracting the mean value (50 in this case), we center the pixel values around zero. This helps improve the convergence and stability of the neural network during training and inference.

By resizing the image and performing mean subtraction on the L channel, we have preprocessed the input image in a way that is suitable for the subsequent colorization process.

```
#Step4
resized = cv2.resize(lab, (224, 224)) #resize it to the required input dimensions for our network
L = cv2.split(resized)[0] #take the channel L
L -= 50 #mean subtraction
```

7.4.6 Preprocessed L channel as input

In this step, we pass the preprocessed L channel as input to the colorization network and extract the predicted AB channel.

First, the message "Passing Input L channel into the network to predict AB channel" is printed to indicate the start of the process.

Next, the preprocessed L channel is passed into the network using the cv2.dnn.blobFromImage() function. This function creates a 4-dimensional blob from the L channel image, which is then set as the input to the neural network using net.setInput().
After setting the input, the network is forward-passed using net.forward(). This performs the inference process, where the input L channel is propagated through the network, and the AB channel is predicted.

The output of the forward pass is obtained as a 4-dimensional array, where the dimensions represent the batch size, channels, height, and width of the output. In this case, the output shape is (1, 2, 224, 224) since we have one image, two channels (AB), and a resolution of 224x224.
To extract the predicted AB values, the array is indexed as [0, :, :, :], which selects the first image's AB channels. Then, the transpose() function is used to rearrange the dimensions from (2, 224, 224) to (224, 224, 2), which represents the AB channel predictions in the correct order.

By passing the L channel through the network and extracting the predicted AB channel, we have obtained the colorization output for the input image.

```
#Step5
print("Passing Input L channel into the network to pridict AB channel")
net.setInput(cv2.dnn.blobFromImage(L)) #passing L input into the network
ab = net.forward()[0, :, :, :].transpose((1, 2, 0)) #forward it  and extract the predicted ab value
```

7.4.7 Predict the AB channel

In this step, we resize the predicted AB channel to match the dimensions of the original input image.
First, the ab array, which contains the predicted AB channel values obtained from the colorization network, is resized using the cv2.resize() function. The cv2.resize() function takes two arguments: the array to be resized (ab) and the desired output dimensions (specified as the shape of the original input image using image.shape[1] for width and image.shape[0] for height).
Resizing the ab array ensures that its dimensions match the dimensions of the original input image, allowing for the proper alignment and combination of the lightness (L) channel with the predicted AB channel during the colorization process.
By resizing the predicted AB channel to match the dimensions of the original input image, we ensure that the colorization output will be correctly aligned and can be combined with the lightness channel to produce the final colorized image.

```
#Step6
ab = cv2.resize(ab, (image.shape[1], image.shape[0])) #resize as our input image
```

7.4.8 Post Processing

In this code snippet, we perform the final steps of the colorization process and display the input and output images.

Step 1: We extract the L channel from the LAB image using cv2.split(). The L channel represents the lightness information of the image. Then, we concatenate the L channel with the predicted AB channel, ab, to form a colorized output image. The np.concatenate() function is used for concatenation, and axis=2 specifies that we are concatenating along the channel dimension.

Step 2: The colorized image, which is currently in the LAB color space, is converted back to the RGB color space using cv2.cvtColor(). This step is necessary to obtain the final colorized output in the RGB format.

Step 3: To ensure that the pixel values are within the valid range of [0, 1], we use the np.clip() function to remove any values that fall outside this range. This step helps to avoid any artifacts or unexpected results in the output image.

Step 4: The colorized image is scaled back to the range of [0, 255] by multiplying it with 255 and converting the data type to uint8 using the .astype("uint8") method. This step brings the pixel intensities back to the range suitable for display and saving as an image.

Finally, we display the original input image and the colorized output image using cv2.imshow(). The input image is shown with the title "Input Image", and the colorized output image is shown with the title "Output Image". The cv2.waitKey(0) function is used to wait for a key press before closing the image windows.

At the end, the message "Task Done" is printed to indicate the completion of the colorization process.

```
#Step1
L = cv2.split(lab)[0] #take the L channel value
colorized = np.concatenate((L[:, :, np.newaxis], ab), axis=2) #concatenate with ab to form a colorized output image

#Step2
colorized = cv2.cvtColor(colorized, cv2.COLOR_LAB2BGR) #LAB-->RGB color space
colorized = np.clip(colorized, 0, 1)  #remove that fall outside the range[0,1]

#Step3
colorized = (255 * colorized).astype("uint8") #brining back the pixel intensity back to the range[0,255]

#Step4
cv2.imshow("Input Image", image)
cv2.imshow("Output Image", colorized)
cv2.waitKey(0)

print("Task Done")
```

7.5 SAMPLE CODES FOR VIDEO COLORIZATION

7.5.1 Import Libraries for video

The code snippet imports several packages necessary for video processing tasks.

- The imutils.video package is imported to utilize the VideoStream class, which simplifies the process of capturing and reading frames from video streams, such as webcams or video files.
- The numpy package is imported as np to work with large, multi-dimensional arrays and matrices efficiently. It is commonly used for manipulating pixel values and performing calculations on video frames.
- The imutils package provides utility functions for image and video processing tasks. It includes functions for resizing, rotating, and performing other common operations on images and video frames.
- The time package is imported to handle timing-related operations. It offers functions for measuring and controlling time, which can be useful for synchronization and performance measurement in video processing tasks.

Lastly, the cv2 package, which stands for OpenCV (Open Source Computer Vision Library), is imported to access the functions and capabilities of OpenCV for video processing. OpenCV is a popular computer vision library that provides numerous functions and algorithms for tasks like reading and writing video files, performing image transformations, and applying filters.

```
#Part1-import packages
from imutils.video import VideoStream
import numpy as np
import imutils
import time
import cv2
```

7.5.2 Enter a choice

In this part of the code, the user is prompted to enter a choice between using a webcam or a video file for further processing.

First, the messages "Enter 0 for Webcam" and "Enter 1 for Video File" are displayed to provide the available options.

The user's choice is then captured by reading an integer input using int(input()) and assigning it to the variable x.

If the user enters 1, it indicates that they want to use a video file. In this case, the variable webcam is set to False, indicating that a video file will be used for processing.

If the user enters 0, it indicates that they want to use a webcam. In this case, the variable webcam is set to True, indicating that the webcam will be used for processing.

After assigning the appropriate value to the webcam variable, the message "Enter 'E' for Exit" is displayed to inform the user about the option to exit the program.

This part of the code allows the user to choose between using a webcam or a video file based on their input, making the program more flexible and adaptable to different scenarios.

```
#Part2-To check...is it webcam or video file
print("Enter 0 for Webcam ")
print("Enter 1 for Video File ")
x=int(input())
if x==1:
        webcam=False
elif x==0:
        webcam=True

print("Enter 'E' for Exit ")
```

7.5.3 Option to choose webcam or video file

In this part of the code, the program checks whether the user has chosen to use a webcam or a video file, based on the value of the webcam variable.

Part 3: If the webcam variable is set to True, it indicates that the user has chosen to use a webcam. In this case, the message "Webcam video's information loading" is printed to indicate the start of the webcam video processing. The VideoStream class from the imutils.video module is used to initialize the webcam video stream. The src=0 argument specifies the index of the webcam device to be used (usually 0 for the default webcam). The video stream is then started by calling the start() method. A delay of 2 seconds is introduced using time.sleep(2.0) to allow the webcam to initialize properly. The variable W is set to 640, which represents the width of the video frame for further processing.

Part 4: If the webcam variable is set to False, it indicates that the user has chosen to use a video file. In this case, the message "Video file's information loading" is printed to indicate the start of the video file processing. The cv2.VideoCapture() function is used to initialize the video capture object, and the video file path 'images/pexels-free-videos-854102-1920x1080-

25fps.mp4' is passed as an argument to open the video file. The width of the video frame is obtained using vs.get(cv2.CAP_PROP_FRAME_WIDTH), and the value is assigned to the variable W for further processing.

By checking the value of the webcam variable and initializing the appropriate video stream or video capture object, the program ensures that the correct video source is used for further processing, whether it is a webcam or a video file.

```
#Part3-if webcam
if webcam:
    print("Webcam vedio's information loading")
    vs = VideoStream(src=0).start()
    time.sleep(2.0)
    W=640

#Part4- else video
else:
    print("Video file's informations loading")
    vs = cv2.VideoCapture('images/pexels-free-videos-854102-1920x1080-25fps.mp4')
    W=int(vs.get(cv2.CAP_PROP_FRAME_WIDTH))
```

7.5.4 Pre-Trained Model

In this part of the code, the program loads the pre-trained model and sets up the necessary configurations for using GPU acceleration.

Part 5: The message "Loading model's information" is printed to indicate the start of the model loading process. The cv2.dnn.readNetFromCaffe() function is used to load the pre-trained model from two files: 'model/colorization_deploy_v2.prototxt' and 'model/colorization_release_v2.caffemodel'. These files contain the architecture and weights of the colorization model respectively. The net variable is assigned with the loaded model.

Next, the program sets the preferable backend and target for the neural network using net. setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA),net.setPreferableTarget(cv2.dnn. DNN_TARGET_CUDA) respectively. These lines configure the OpenCV deep neural network module to use CUDA for GPU acceleration if available. This helps to improve the processing speed and efficiency of the model.

The variable pts is assigned with the cluster centers, which are loaded from the 'model/pts_in_hull.npy' file. These cluster centers represent the color quantization information used for rebalancing the AB channel predictions during the colorization process.

Part 6: The program adds the cluster centers to the network for the AB channel predictions. The layer IDs for the "class8_ab" and "conv8_313_rh" layers are obtained using net.getLayerId(). The pts array is then reshaped to match the expected shape of (2, 313, 1, 1) and assigned back to the pts variable. The blobs attribute of the "class8_ab" and "conv8_313_rh" layers is updated with the cluster center information by assigning the pts array and a numpy array filled with 2.606, respectively.

By loading the pre-trained model, configuring GPU acceleration if available, and adding the cluster centers, the program sets up the necessary components for the colorization process.

```
#Part5-load the model
print("Loading model's informations")
net = cv2.dnn.readNetFromCaffe('model/colorization_deploy_v2.prototxt', 'model/colorization_release_v2.caffemodel')
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)
net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)
pts = np.load('model/pts_in_hull.npy')

#Part6-add the cluster centers
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")]
```

7.5.5 Process each frame of the video source

In this part of the code, the program enters a loop to process each frame of the video source (webcam or video file) and display the input, grayscale, and colorized output frames.

Part 6: The loop starts with the condition while True, indicating that it will continue indefinitely until explicitly broken. Within the loop, the next frame is read from the video source using vs.read(). If the source is a webcam (webcam variable is True), the frame is assigned directly to the frame variable. If the source is a video file (webcam variable is False), the frame is obtained from the second element of the returned tuple (frame[1]), as the first element represents a boolean indicating whether the frame was successfully read.

The program then resizes the frame using imutils.resize() to the width specified by the W variable. The frame is scaled by dividing it by 255.0, converting the pixel values to the range [0, 1]. The scaled frame is then converted from the BGR color space to the LAB color space using cv2.cvtColor().

The LAB frame is resized to (224, 224) dimensions and the L channel is extracted using cv2.split() to obtain the lightness information. The pixel values of the L channel are reduced by 50 through element-wise subtraction (L -= 50).

Next, the L channel is passed into the network for colorization. The L channel is first converted into a blob using cv2.dnn.blobFromImage() and set as the input to the network using net.setInput(). The network processes the input and produces a prediction for the AB channel. The AB channel prediction is obtained from the network output, transposed to the shape (height, width, channels), and assigned to the ab variable.

The predicted AB channel is resized to match the dimensions of the original frame. The L channel is obtained again from the LAB frame using cv2.split(). The L channel and the predicted AB channel are concatenated along the channel axis to form the colorized output image.

The colorized output image is converted back to the BGR color space using cv2.cvtColor(). The pixel values are clipped to the range [0, 1] using np.clip() to ensure they fall within the valid range. The pixel values are then scaled back to the range [0, 255] by multiplying by 255 and converting the data type to uint8.

The input frame, grayscale frame, and colorized output frame are displayed using cv2.imshow(). The grayscale frame is obtained by converting the input frame to grayscale using cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY). The program waits for 50 milliseconds for a key press using cv2.waitKey(50). If the key pressed is 'E' (as defined by ord("E")), the loop is broken and the program proceeds to the next section.

Part 8: If a webcam is used, the vs.stop() method is called to release the webcam resources.

Part 9: If a video file is used, the vs.release() method is called to release the video file resources.

Part 10: Finally, all open windows are closed using cv2.destroyAllWindows().

This section of the code ensures that frames from the video source are continuously processed, and the input, grayscale, and colorized output frames are displayed until the 'E' key is pressed or the video ends.

```
#Part7-loop to get the frames from video
while True:
    #take the next frame from the video
    frame = vs.read()
    frame = frame if webcam else frame[1]

    # end it if it is last frame
    if not webcam and frame is None:
        break

    frame = imutils.resize(frame, width=W) #resize
    scaled = frame.astype("float32") / 255.0 # scale it [0,1]
    lab = cv2.cvtColor(scaled, cv2.COLOR_BGR2LAB) #convert the frame from the BGR to Lab

    resized = cv2.resize(lab, (224, 224))# resize
    L = cv2.split(resized)[0] #split channels
    L -= 50 #extract the 'L'

    #  channel through the network which will  channel values
    net.setInput(cv2.dnn.blobFromImage(L)) #pass the L
    ab = net.forward()[0, :, :, :].transpose((1, 2, 0)) #predict ab

    ab = cv2.resize(ab, (frame.shape[1], frame.shape[0])) #resize
    L = cv2.split(lab)[0] #grab the 'L'
    colorized = np.concatenate((L[:, :, np.newaxis], ab), axis=2) #concatenate L and ab

    colorized = cv2.cvtColor(colorized, cv2.COLOR_LAB2BGR)
    colorized = np.clip(colorized, 0, 1) #clip
    colorized = (255 * colorized).astype("uint8") #again convert it

    #show the output
    cv2.imshow("Input Video", frame)
    cv2.imshow("Grayscale Video", cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY))
    cv2.imshow("Output Video", colorized)
    key = cv2.waitKey(50) & 0xFF

    # press E to exit
```

```
#Part8-stop webcam
if webcam:
    vs.stop()
#Part9-stop vedio file
else:
    vs.release()

#Part10-close windows
cv2.destroyAllWindows()
```

CHAPTER 8

TESTING

The purpose of testing image and video colorization in old school style is to apply color to grayscale images or videos in a way that resembles the traditional hand-coloring techniques used before the advent of digital technology. This process adds an artistic and nostalgic touch to the visuals, creating a vintage or retro aesthetic.

Old school image and video colorization can be used for various purposes, including:

- Restoration: Colorizing old black and white photographs or videos can help bring them to life and make them more relatable to contemporary viewers. It allows us to visualize historical events, people, and places in a more vivid and realistic manner.
- Artistic Expression: Applying color to grayscale images or videos can be a creative way to reinterpret or enhance the original content. It provides opportunities for artistic expression, allowing photographers, filmmakers, and artists to experiment with different color palettes and styles.
- Visual Storytelling: Color has a significant impact on visual storytelling. By adding color to specific elements or areas of an image or video, we can guide the viewer's attention, highlight important details, or evoke specific emotions. Old school colorization techniques can be used to enhance the narrative and create a unique visual experience.
- Educational Purposes: Colorized historical images and videos can aid in educational contexts, making them more engaging and accessible to students. By seeing historical figures, events, or locations in color, learners can develop a deeper connection and understanding of the past.

Overall, testing image and video colorization in an old school style allows for artistic exploration, historical visualization, and creative reinterpretation of visual content. It combines traditional techniques with modern technology, providing a unique and visually appealing result.

8.1 Types of Testing

When it comes to testing old school image and video colorization techniques, there are several types of testing that can be conducted to evaluate the effectiveness and quality of the colorization process. Here are some common types of testing:

- Visual Inspection: Visual inspection involves carefully examining the colorized images or videos to assess the overall quality and accuracy of the colorization. Testers can compare the colorized output with the original grayscale input or reference color images/videos to check for color consistency, appropriate color choices, and the level of realism achieved.
- User Feedback: Gathering user feedback is essential to understand how the colorized images or videos are perceived by the target audience. Conducting user surveys, interviews, or focus groups can help collect opinions, preferences, and suggestions regarding the colorization process, the choice of colors, and the overall aesthetic appeal.
- Accuracy Evaluation: This type of testing involves quantitatively measuring the accuracy of the colorization process. Metrics such as color difference (e.g., Delta E) can be used to compare the colorized output with the original colored images or videos. The lower the color difference value, the more accurate the colorization is considered to be.
- Comparative Testing: Comparative testing involves comparing different colorization techniques or algorithms to determine their strengths and weaknesses. Multiple colorization methods can be applied to the same set of grayscale images or videos, and the results can be compared based on factors such as color accuracy, preservation of details, computational efficiency, and visual appeal.
- Benchmark Datasets: Using benchmark datasets is another way to evaluate the performance of old school colorization techniques. These datasets consist of a collection of grayscale images or videos with corresponding ground truth color references. By comparing the colorized output with the ground truth, testers can assess the accuracy and quality of the colorization process.
- Computational Efficiency: Testing the computational efficiency of old school colorization methods is important, especially when dealing with large-scale image or video datasets. Performance measurements such as processing time and resource usage

(CPU, memory) can be evaluated to ensure that the colorization process is efficient and feasible for real-world applications.

Overall, a combination of visual inspection, user feedback, quantitative evaluation, and comparative analysis can provide a comprehensive assessment of old school image and video colorization techniques, helping to refine and improve the colorization process.

8.2 Performance Testing

Performance testing in old school image and video colorization refers to evaluating the efficiency and speed of the colorization process. It aims to assess how well the colorization algorithm or technique performs in terms of processing time, resource utilization, and scalability. Here are some aspects to consider in performance testing:

- Processing Time: Performance testing involves measuring the time taken by the colorization process to convert grayscale images or videos into colorized versions. This includes the time required for pre-processing steps, such as resizing or converting color spaces, as well as the actual colorization algorithm. Testers can measure the processing time for individual images or frames and calculate average processing time per unit to assess the efficiency of the colorization process.
- Resource Usage: Performance testing also involves monitoring the utilization of system resources, such as CPU (Central Processing Unit), memory, and disk space. Testers can measure the CPU usage during the colorization process to determine if it is within acceptable limits. Excessive memory usage or disk space requirements can also be indicators of inefficient resource management.
- Scalability: Colorization algorithms should be tested for scalability, which refers to their ability to handle large-scale image or video datasets without significant degradation in performance. Testers can increase the size or complexity of the input dataset and observe if the colorization process maintains reasonable processing time and resource usage levels.
- Optimization Techniques: Performance testing can also involve evaluating different optimization techniques or strategies to improve the speed and efficiency of the colorization process. For example, testers can explore parallel processing approaches, GPU acceleration, or algorithmic optimizations to speed up the colorization process and reduce resource utilization.

- Benchmarking: Comparing the performance of different colorization algorithms or techniques is crucial. Testers can benchmark the processing time and resource usage of different methods using the same input dataset and measure the differences in performance. This helps identify the most efficient and effective colorization approach for a given set of requirements.
- Real-time Performance: In some cases, real-time colorization is required, such as in live video applications. Performance testing can involve assessing the ability of the colorization process to operate in real-time, with minimal latency or delay. Testers can measure the frame rate achieved during colorization and ensure that it meets the desired real-time requirements.

By conducting performance testing in old school image and video colorization, it is possible to identify bottlenecks, optimize algorithms, and select the most efficient techniques for achieving high-quality colorization results in a timely manner.

CHAPTER 9

RESULTS

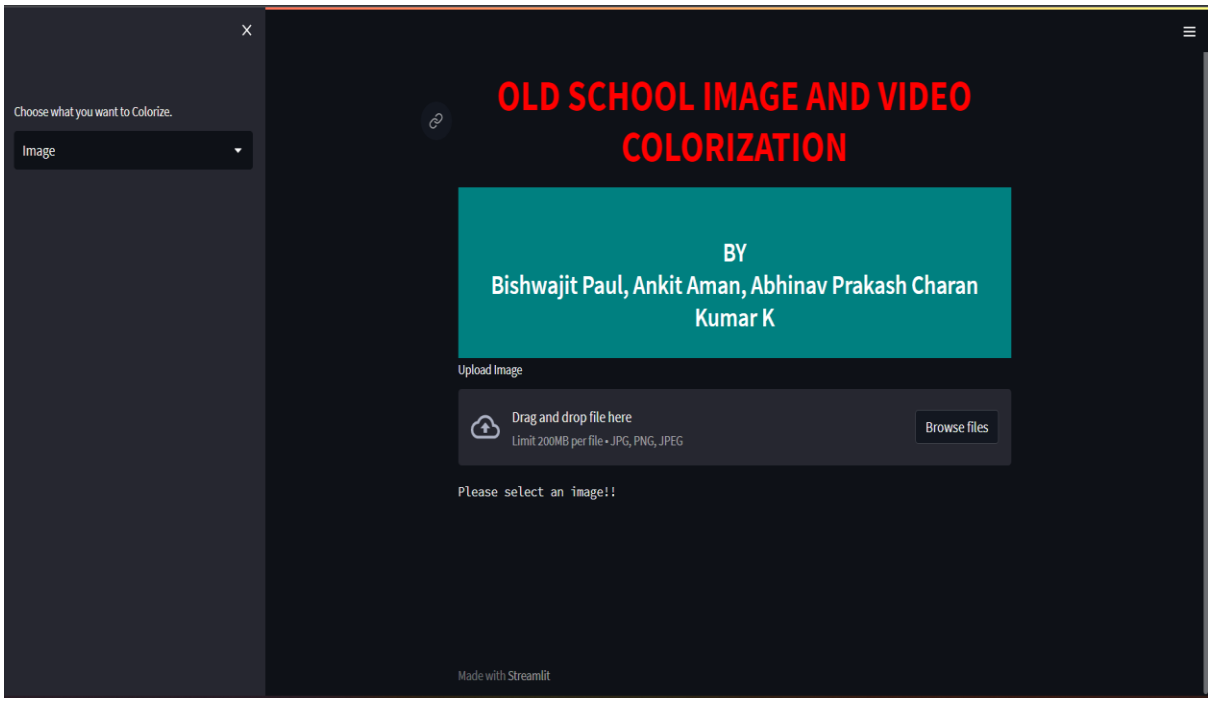


Fig 9.1 Webb.py

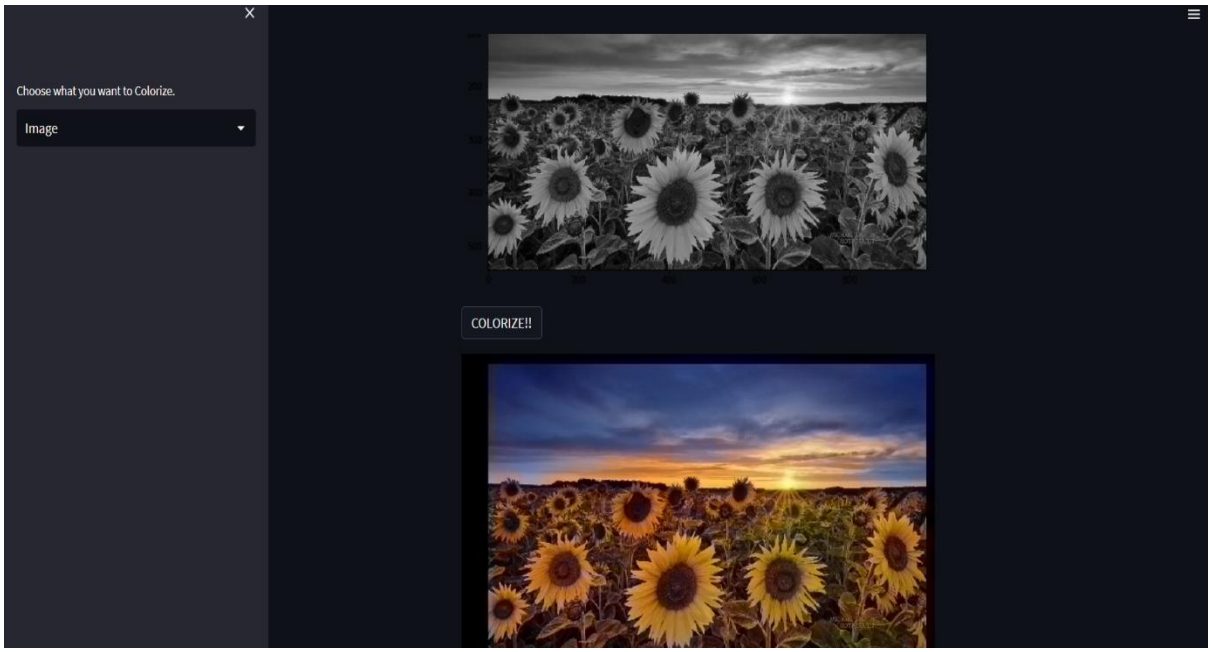


Fig 9.2: Colorize Image

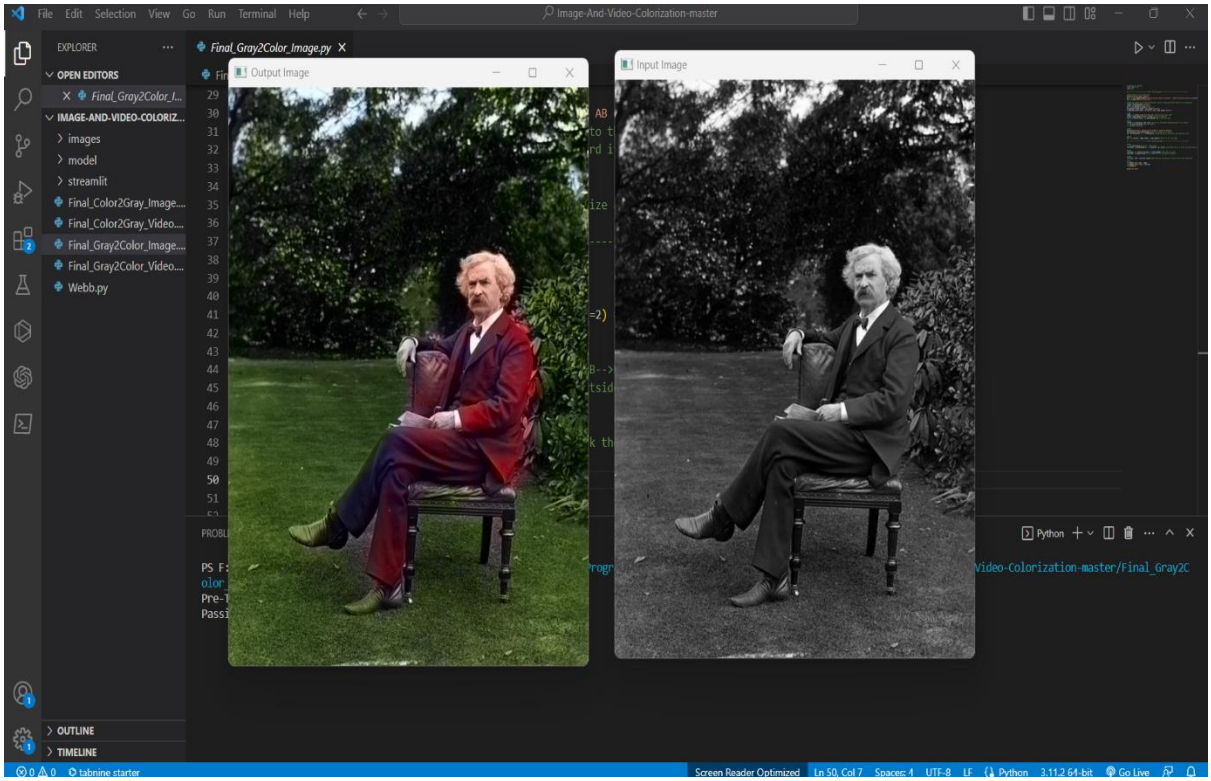
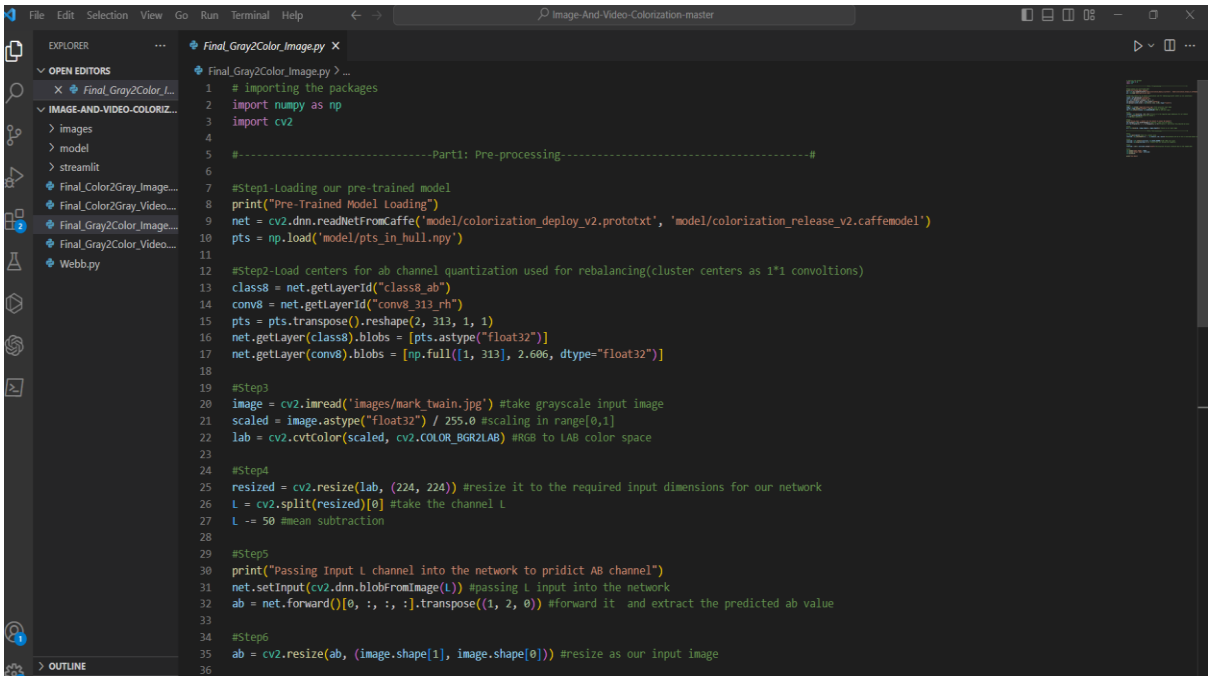


Fig 9.3: Colorize the Image



```python
# importing the packages
import numpy as np
import cv2

#--------------------------------Part1: Pre-processing----------------------------------------#

#Step1-Loading our pre-trained model
print("Pre-Trained Model Loading")
net = cv2.dnn.readNetFromCaffe('model/colorization_deploy_v2.prototxt', 'model/colorization_release_v2.caffemodel')
pts = np.load('model/pts_in_hull.npy')

#Step2-Load centers for ab channel quantization used for rebalancing(cluster centers as 1*1 convoltions)
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")]

#Step3
image = cv2.imread('images/mark_twain.jpg') #take grayscale input image
scaled = image.astype("float32") / 255.0 #scaling in range[0,1]
lab = cv2.cvtColor(scaled, cv2.COLOR_BGR2LAB) #RGB to LAB color space

#Step4
resized = cv2.resize(lab, (224, 224)) #resize it to the required input dimensions for our network
L = cv2.split(resized)[0] #take the channel L
L -= 50 #mean subtraction

#Step5
print("Passing Input L channel into the network to pridict AB channel")
net.setInput(cv2.dnn.blobFromImage(L)) #passing L input into the network
ab = net.forward()[0, :, :, :].transpose((1, 2, 0)) #forward it  and extract the predicted ab value

#Step6
ab = cv2.resize(ab, (image.shape[1], image.shape[0])) #resize as our input image
```

Fig 9.4 Code

CONCLUSION

Image colorization is the process of adding color to a grayscale image or video. There are several methods for colorizing images, including manual techniques, semi-automatic approaches, and fully automatic methods.

Manual image colorization involves an artist or graphic designer manually adding color to the image using image editing software, such as Photoshop. This method can produce high-quality results, but it is time-consuming and requires a skilled artist.

Semi-automatic image colorization methods involve using algorithms to generate a colorization that is then refined by a human artist. This can produce good results, but it still requires human intervention and is therefore somewhat time-consuming.

Fully automatic image colorization methods use algorithms to generate a colorization without any human intervention. These methods have improved significantly in recent years, but they still have limitations and may not produce results that are as visually appealing as those produced by manual or semi-automatic methods.

In general, image colorization is an active area of research, and there is still much room for improvement in terms of the accuracy and speed of automatic methods. However, it is an important task in many applications, including digital restoration of old photographs, improving the accessibility of grayscale documents and images for people with color vision deficiency, and enhancing the visual appeal of videos and images.

Old school image and video colorization techniques offer a unique way to transform grayscale visuals into colorful representations with a nostalgic and artistic touch. These techniques aim to recreate the traditional hand-coloring methods used before the digital era, providing a vintage and retro aesthetic to the images and videos.

In this process, grayscale images or videos are converted to the LAB color space, where the L channel represents the grayscale information. The colorization is achieved by predicting the AB channels using pre-trained models and cluster centers. The AB channels are then combined with the L channel to create the final colorized output.

Testing old school image and video colorization involves various aspects. Visual inspection is crucial to assess the quality and accuracy of the colorization process, ensuring that the colors are consistent and realistic. User feedback helps gauge the perception and preferences of the

target audience. Accuracy evaluation and comparative testing assist in quantitatively measuring the accuracy of the colorization and comparing different techniques or algorithms. Additionally, performance testing focuses on evaluating the efficiency, processing time, resource usage, and scalability of the colorization process.

Old school image and video colorization techniques serve multiple purposes. They can be used for restoration, bringing old black and white visuals to life and making historical content more relatable. They offer a means of artistic expression, enabling photographers, filmmakers, and artists to experiment with color palettes and styles. These techniques also have educational value, providing a visually engaging way to learn about historical events and figures. Moreover, the results of colorization can be used for storytelling purposes, guiding the viewer's attention and evoking specific emotions.

By combining the nostalgia of the past with modern technology, old school image and video colorization techniques provide a creative and visually appealing way to enhance and reinterpret grayscale visuals. Whether it is for restoration, artistic expression, education, or storytelling, these techniques offer a unique way to breathe life into grayscale images and videos and evoke a sense of history and nostalgia.

REFERENCES

[1] Richard Zhang, Phillip Isola, Alexei A. Efros. “Colourful Image Colourisation” In University of California, Berkeley, October 2016. http://richzhang.github.io/Colourisation/

[2] ZHITONG HUANG, NANXUAN ZHAO, JING LIAO. “A Unified Framework for Multi-Modal Colorization with Transformer” In University of Hong Kong, China, September 2022. https://luckyhzt.github.io/unicolor.

[3] B. S. Thomas, R. Dogra, B. Dixit and A. Raut, "Automatic Image and Video Colourisation using Deep Learning," 2018 International Conference on Smart City and Emerging Technology (ICSCET), *2018*, pp. 1-4, doi: 10.1109/ICSCET.2018.8536308.

[4] LvMin Zhang, Chengze Li, Tien-Tsin Wong, Yi Ji, and ChunPing Liu. 2018. Two-stage Sketch Colorization. ACM Trans. Graph. 36, 6, Article 261 (November 2018), 14 pages. https://doi.org/10.1145/3262126.3265090.

[5] Huiwen Chang, Ohad Fried, Yiming Liu,Stephen DiVerdi, Adam Finkelstein. “Palette-based Photo Recoloring” ACM Transactions on Graphics (Proc. SIGGRAPH), July 2015.

[6] Jeff Hwang, You Zhou. “Image Colourisation with Deep Convolutional Neural Networks” In Stanford University Research Projects, 2016.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks.” In Advances in neural information processing systems, pages 1096–1105, 2012.

[8] Gupta, R.K., Chia, A.Y.S., Rajan, D., Ng, E.S., Zhiyong, H. “Image Colourisation using similar images.” In Proceedings of the 20th ACM International Conference on Multimedia, ACM (2012) 369-368.

[9] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition.” In arXiv preprint arXiv:1512.03385, 2015.

[10] Larsson, G., Maire, M., Shakhnarovich, G.: Learning representations for automatic colorization. European Conference on Computer Vision (2016).

PUBLISHED PAPER

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 2 Feb. 2023, pp: 525-529 www.ijaem.net ISSN: 2395-5252

Old-school Image and Video colorization using Deep learning

Sonia Das [1], Abhinav Prakash[2], Ankit Aman[3], Bishwajit Paul[4], Charan Kumar K[5]

[1]*Asst. Professor, Department of Computer Science & Engineering, T John Institute of Technology, Bengaluru, India.*

[2,3,4,5]*Students, Department of Computer Science & Engineering, T John Institute of Technology, Bengaluru, India*

Date of Submission: 03-02-2023 Date of Acceptance: 17-02-2023

ABSTRACT - Colorization of old images and videos using deep learning involves using machine learning techniques to automatically add color to black and white or grayscale images and videos. There are several approaches to using deep learning for colorization, including using convolutional neural networks (CNNs) trained on large datasets of color images to predict the colors of specific pixels in the input image or video. One way to approach the problem is to use an autoencoder architecture, where the model is trained to reconstruct a colored version of the input image or video from its grayscale version. Another approach is to use a generative adversarial network (GAN), where the model is trained to generate realistic-looking colored versions of the input image or video. To train a deep learning model for colorization, a dataset of color images is typically used. The model is then trained to predict the colors of the pixels in the input image based on the colors of the pixels in the corresponding regions of the training Images. The model can then be used to colorize new images or videos by predicting the colors of the pixels in the input image or video based on the patterns and relationships learned from the training data.

I. INTRODUCTION

Old-school image and video colorization is the process of adding color to black-and-white images or videos using deep learning techniques. This is done by training neural networks on large datasets of colored images so that the networks can learn to predict the color of an image based on its grayscale input. This technology has improved significantly over recent years and has been used for a wide range of applications, from restoring old film footage to creating new and unique art styles. The use of deep learning has enabled high-quality colorization results to be achieved with much less manual effort than previously possible.

The process of colorization typically involves feeding the grayscale image through a deep convolutional neural network, which outputs a colored version of the image. The network is trained on large datasets of colored images, which allows it to learn the relationships between the grayscale input and the corresponding color output. The network can then be used to colorize new grayscale images, either by fine-tuning specific datasets or by applying the trained network directly to the input image. For video colorization, the process is similar, but the network must be able to handle sequential inputs and outputs, as the color of each frame is dependent on the previous frames. Recurrent neural networks (RNNs) or long short-term memory (LSTM) networks are often used to handle this. In conclusion, the use of deep learning has revolutionized the field of old-school image and video colorization. With its ability to learn complex relationships between grayscale inputs and color outputs, deep learning has enabled high-quality colorization results to be produced with much less manual effort than was previously possible. Additionally, deep learning-based colorization has opened up new possibilities for preserving and revitalizing historical images and videos. With its ability to automatically add color to old black-and-white media, this technology can bring new life to historical artifacts, making them more engaging and accessible to a wider audience. Furthermore, deep learning-based colorization has also found use in the entertainment industry, where it has been used to add color to classic movies and TV shows, allowing viewers to experience these works in a new way. Despite its advances, deep learning-based colorization is imperfect and still faces challenges, such as accurately handling complex scenes and colorizing certain textures, like hair and

DOI: 10.35629/5252-0502525529 |Impact Factorvalue 6.18| ISO 9001: 2008 Certified Journal

CERTIFICATE OF PUBLICATION

Certificate of Publication

IJAEM

This is to confirm that

Sonia Das , Abhinav Prakash , Ankit Aman, Bishwajit Paul , Charan Kumar K

Published following article

Old school Image and Video colorization using Deep learning

Volume 5, Issue 02, pp: 525-529

www.ijaem.net

A Peer Reviewed Journal

International journal of Advances in Engineering and Management (IJAEM)

ISSN: 2395-5252

Publication Head

CONFERENCE PAPER

KAS Knowledge As a Service

T. JOHN INSTITUTE OF TECHNOLOGY

Approved by All India Council for Technical Education (AICTE)
& Affiliated to Visvesvaraya Technological University (VTU)
#86/1, Gottigere P. O., Bannerghatta Road, Bangalore - 560083
Tel: 080 - 40250555, 080 - 49030555 | Fax: 080 49030509
Email: tjitprincipal@tjohngroup.com | Website: www.tjohncollege.com

Indian Journal of Science and Technology

CERTIFICATE OF PRESENTATION

This is to certify that Dr./Mr./Mrs. Abhinav Prakash Prof./ ASP./AP./RS./Students
Department of CSE / T John Institute Of Technology has presented a paper titled
Old-School Image And Video Colorization Using Deep Learning in the

International Conference on Advances in Modern Science and Technology (ICAMSnt 2023)
Organised by T. John Institute of Technology, Bangalore, Karnataka, India, In Association with KAS Innovative India, Chennai
on 21 - 22 April 2023

Convenor
Dr. John T Mesia Dhas

202304IKSOFT21222791

Principal
Dr. P. Suresh Venugopa

KAS Knowledge As a Service

T. JOHN INSTITUTE OF TECHNOLOGY

Approved by All India Council for Technical Education (AICTE)
& Affiliated to Visvesvaraya Technological University (VTU)
#86/1, Gottigere P. O., Bannerghatta Road, Bangalore - 560083
Tel: 080 - 40250555, 080 - 49030555 | Fax: 080 49030509
Email: tjitprincipal@tjohngroup.com | Website: www.tjohncollege.com

Indian Journal of Science and Technology

CERTIFICATE OF PRESENTATION

This is to certify that Dr./Mr./Mrs. Ankit Aman Prof./ ASP./AP./RS./Students
Department of CSE / T John Institute Of Technology has presented a paper titled
Old-School Image And Video Colorization Using Deep Learning in the

International Conference on Advances in Modern Science and Technology (ICAMSnt 2023)
Organised by T. John Institute of Technology, Bangalore, Karnataka, India, In Association with KAS Innovative India, Chennai
on 21 - 22 April 2023

Convenor
Dr. John T Mesia Dhas

202304IKSOFT21222793

Principal
Dr. P. Suresh Venugopa

KAS Knowledge As a Service

T. JOHN INSTITUTE OF TECHNOLOGY

Approved by All India Council for Technical Education (AICTE)
& Affiliated to Visvesvaraya Technological University (VTU)
#86/1, Gottigere P. O., Bannerghatta Road, Bangalore - 560083
Tel: 080 - 40250555, 080 - 49030555 | Fax: 080 49030509
Email: tjitprincipal@tjohngroup.com | Website: www.tjohncollege.com

Indian Journal of Science and Technology

CERTIFICATE OF PRESENTATION

This is to certify that Dr./Mr./Mrs. **Bishwajit Paul** Prof./ ASP./AP./RS./Students
Department of CSE / T John Institute Of Technology has presented a paper titled
Old-School Image And Video Colorization Using Deep Learning in the

International Conference on Advances in Modern Science and Technology (ICAMSnt 2023)

Organised by T. John Institute of Technology, Bangalore, Karnataka, India, In Association with KAS Innovative India, Chennai on 21 - 22 April 2023

Convenor
Dr. John T Mesia Dhas

202304IKSOFT21222792

Principal
Dr. P. Suresh Venugopa

KAS Knowledge As a Service

T. JOHN INSTITUTE OF TECHNOLOGY

Approved by All India Council for Technical Education (AICTE)
& Affiliated to Visvesvaraya Technological University (VTU)
#86/1, Gottigere P. O., Bannerghatta Road, Bangalore - 560083
Tel: 080 - 40250555, 080 - 49030555 | Fax: 080 49030509
Email: tjitprincipal@tjohngroup.com | Website: www.tjohncollege.com

Indian Journal of Science and Technology

CERTIFICATE OF PRESENTATION

This is to certify that Dr./Mr./Mrs. **Charan Kumar K** Prof./ ASP./AP./RS./Students
Department of CSE / T John Institute Of Technology has presented a paper titled
Old-School Image And Video Colorization Using Deep Learning in the

International Conference on Advances in Modern Science and Technology (ICAMSnt 2023)

Organised by T. John Institute of Technology, Bangalore, Karnataka, India, In Association with KAS Innovative India, Chennai on 21 - 22 April 2023

Convenor
Dr. John T Mesia Dhas

202304IKSOFT21222790

Principal
Dr. P. Suresh Venugopa