A Novel Method For Colorizing Black-And-White Videos And Images Utilising Faster R-CNN

PHASE 2 PROJECT REPORT

Submitted by

Mohamed Rasik P 310619104070

Ragav Krishna J 310619104098

V C Sai Santhosh 310619104112

Sameer Sheriff S M 310619104113

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

EASWARI ENGINEERING COLLEGE, CHENNAI

(Autonomous Institution)

affiliated to

ANNA UNIVERSITY: CHENNAI – 600025

April 2023

EASWARI ENGINEERING COLLEGE, CHENNAI

(AUTONOMOUS INSTITUTION)

AFFILIATED TO ANNA UNIVERSITY, CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this project report **"A Novel Method For Colorizing Black-And-White Videos And Images Utilising Faster R-CNN"** is the bonafide work of **"Mohamed Rasik P (310619104070), Ragav Krishna J (310619104098), V C Sai Santhosh (310619104112), Sameer Sheriff S M (310619104113)** who carried out the project work under my supervision.

SIGNATURE

Dr. G. S ANANDHA MALA

HEAD OF THE DEPARTMENT

Professor,

Department of Computer Science and Engineering,

Easwari Engineering College,

Ramapuram, Chennai 600089.

SIGNATURE

Mr. S. Kingsley

SUPERVISOR

Assistant Professor,

Department of Computer Science and Engineering,

Easwari Engineering College,

Ramapuram, Chennai 600089.

CERTIFICATE OF EVALUATION

College Name : Easwari Engineering College

Branch & Semester : Computer Science and Engineering & VIII

S. No	Name of the Students	Title of the Project	Name of the supervisor with designation
1.	Mohamed Rasik P (310619104070)	A Novel Method For Colorizing Black-And-White Videos And Images Utilising Faster R-CNN	Mr. S. Kingsley Assistant Professor/ CSE
2.	Ragav Krishna J (310619104098)		
3.	V C Sai Santhosh (310619104112)		
4.	Sameer Sheriff S M (310619104113)		

The report of the PHASE II project work submitted by the above students in partial fulfilment for the award of Bachelor of Engineering Degree in Computer Science and Engineering of Anna University were evaluated and confirmed to be a report of the work done by the above students.

The viva voce examination of the project was held on

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We hereby place our deep sense of gratitude to our beloved Founder Chairman of the institution**, Dr.T.R.Pachamuthu, B.Sc., M.I.E.,** for providing us with the requisite infrastructure throughout the course. We would also like to express our gratitude towards our Chairman **Dr.R.Shivakumar, M.D., Ph.D.** for giving the necessary facilities.

We convey our sincere thanks to **Dr.R.S.Kumar, M.Tech., Ph.D.** Principal Easwari Engineering College, for his encouragement and support. We extend our hearty thanks to **Dr.V.Elango, M.E., Ph.D.**, Vice Principal (academics) and **Dr.S.Nagarajan, M.E, Ph.D**, Vice Principal (admin), Easwari Engineering College, for their constant encouragement.

We take the privilege to extend our hearty thanks to **Dr.G.S.Anandha Mala, M.E, Ph.D,** Head of the Department, Computer Science and Engineering, Easwari Engineering College for her suggestions, support and encouragement towards the completion of the project with perfection.

We would like to express our gratitude to our Project Coordinator, **Dr.B.Padmavathi, M.E, Ph.D** Assistant Professor, Department of Computer Science and Engineering, Easwari Engineering College, for her constant support and encouragement.

We would also like to express our gratitude to our guide **Mr.S.Kingsley, M.E, (Ph.D)** Assistant Professor, Department of Computer Science and Engineering, Easwari Engineering College, for her constant support and encouragement.

Finally, we wholeheartedly thank all the faculty members of the Department of Computer Science and Engineering for warm cooperation and encouragement.

ABSTRACT

Black and white image & video colorization is a laborious and time-consuming manual procedure. Conventional techniques like Photoshop editing take a month to finish one photograph and need substantial study. Deep learning models can be used to implement enhanced picture colorization approaches to solve this issue. As it combines two obscure fields, deep learning and digital image processing, there has been an increase in interest in the literature on picture colorization in recent years. Researchers have taken advantage of the advantages of end-to-end deep learning models and transfer learning to automatically extract picture characteristics from the training data, which can speed up the colorization process with little human interaction. The colorization of CCTV images during unwelcome occurrences is one such use of this technology. By examining the colour of clothes, automobiles, or other things in CCTV footage of crimes or accidents, detectives might glean important details. Finding these nuances in the typical black and white film might be difficult, but colorization can aid with clarity and increase the investigation's accuracy. To do this, feature extraction in deep learning models like Faster R-CNN is employed, which improves performance by using previously trained models. The performance of many Faster R-CNN models is compared in order to choose the most effective model for the task. Moreover, a realistic and cohesive output image is produced using the Pix2Pix algorithm.

KEYWORDS:

Image colorization, Deep learning, Transfer learning, Faster Region Based Convolutional Neural Network, CCTV footage, Pix2Pix.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABB	FULL FORM
CNN	CONVOLUTIONAL NEURAL NETWORK
R-CNN	REGION BASED CONVOLUTIONAL NEURAL NETWORK
FPS	FRAMES PER SECOND
MSE	MEAN SQUARED ERROR

CHAPTER 1

INTRODUCTION

1.1 GENERAL

In computer vision and image processing, colourizing black-and-white movies and images is a challenging job. Entertainment, historical archiving, and medical analysis are just a few of its many uses. Black and white photographs and films lack the vibrancy and realism of colour counterparts, making them less attractive to viewers. Furthermore, many historical photographs and movies exist only in black and white, and colourizing them may bring them to life and make them more familiar to present audiences. Colourization improves the visual quality of pictures, affects how people perceive them, and makes it possible to analyse and understand images more effectively.

In a variety of disciplines, including forensic investigation, remote sensing, and medical imaging, colorization of photographs can yield important information. Moreover, it may be utilised to produce realistic visuals, improve old photos and movies, and enhance computer graphics and games for the user. The literature has described a number of colorization techniques, including deep learning-based algorithms, however their accuracy and processing efficiency are limited. To reach high accuracy, these algorithms often demand a substantial quantity of training data and processing resources, which can be a significant constraint for practical applications.

Colorization algorithms based on optimization are computationally efficient, but they require user input and may be incapable of handling complicated situations.

Colorization algorithms based on histograms are easy, however they may not yield accurate results. As a result, proposes a novel coloration technique based on the Faster R-CNN algorithm.

Faster R-CNN is a cutting-edge object identification technology that can accurately and quickly recognise items and regions of interest in pictures and movies. May colourize the matching areas in black and white films and photos by using the regions of interest recognised by the Faster R-CNN algorithm.

This method addresses the constraints of previous colorization methods by employing a deep learning-based object recognition algorithm capable of handling complicated scenarios and achieving high accuracy while utilising less processing resources. This research compares the performance of the suggested algorithm to that of current colorization methods on a test set of black and white videos and photographs. A deep learning algorithm known as R-CNN (Region-based Convolutional Neural Network) is used to identify objects in images. To anticipate the existence of objects in each area, R-CNN runs a convolutional neural network (CNN) on each region after dividing the picture into several regions or proposals.

The R-CNN model is employed in the since it is a cutting-edge model for object identification tasks. The R-CNN model's key benefit is that it can accurately recognise objects in crowded settings where there may be several in the image. The R-CNN algorithm is adaptable and can be applied to a wide range of object detection apps. The R-CNN model's ability to support transfer learning, which enables a pre-trained model to be improved on a particular object recognition task with a smaller dataset, is another crucial benefit. This lessens the requirement for substantial annotated datasets, which may be time- and money-consuming to

produce. The R-CNN model is an excellent option for the job at hand since it is generally a strong and adaptable tool for object detection tasks. The goal of the proposed system is to colourize CCTV night video and photos in order to make events recorded in poor lighting circumstances more visible. This can considerably increase the capacity of security officers and law enforcement to recognise suspects and cars used in crimes in low-light situations.

By using the faster R-CNN algorithm for object location and recognition, which can help produce better results than more traditional colorization techniques, the project's research aims to improve the speed and precision of the colorization process. This system can increase public safety by integrating object detection and colorization to aid law enforcement and security officers in their investigations.

Transfer learning is used in the suggested approach for colourizing monochrome photos, especially the process of refining a pre-trained Faster R-CNN model for object recognition. With the help of transfer learning, which is a potent deep learning method, it is possible to reuse previously trained models for new tasks with a little amount of labelled data, which eliminates the need for huge labelled datasets and lowers the computational burden of training. In this research, a smaller dataset of black and white pictures is used to fine-tune the Faster R-CNN model, which was pre-trained on a large dataset of object identification data.

The fine-tuning procedure is speedier and uses fewer training samples than training a model from start by utilising the characteristics of the pre-trained model. This method keeps the object identification expertise gained during the pre-training phase while enabling the Faster R-CNN model to learn to detect and find items in monochrome pictures. The project's transfer learning strategy

provides a number of benefits over previous colorization methods. At the beginning, it enables precise and effective object recognition, which is essential for precise colorization. Second, it requires fewer samples that have been labelled, making it more practical for real-world applications where annotated data may be hard to come by or expensive to purchase. Being a flexible tool for numerous computer vision applications, the transfer learning model may also be simply extended to additional object identification tasks.

1.2 PROBLEM DESCRIPTION

Despite much research, there are still difficulties in the process of colourizing black and white pictures. Several earlier methods depended on manual human annotation, which is labour-intensive and time-intensive and frequently yielded desaturated results that could not be "believed" to be real colorizations. Also, these techniques frequently led to repeated colorization, where the colours picked weren't accurate or appropriate for the scene. Because of these restrictions, it is challenging to produce accurate and visually acceptable high-quality colorizations. By employing a novel technique that makes use of the Faster R-CNN algorithm for object detection in pictures, the proposed effort aims to overcome these issues.

Black and white photos' regions of interest may be recognised by the algorithm, which then appropriately colours those sections while requiring the least amount of human input. By using an object identification algorithm based on deep learning, which can handle complicated scenarios and achieve high accuracy

while requiring less processing power, this method gets beyond the drawbacks of earlier approaches. Also, the R-CNN model may be improved with a smaller dataset thanks to the use of transfer learning, which cuts down on the time and expense of producing annotated datasets. In general, the project seeks to offer a more effective and precise method of colouring realistic black and white photos.

1.3 OBJECTIVE

The objectives are described as follows:

- This project aims to generate a fully colorized image and video captured by a CCTV camera during an unfavourable event.
- Giving an overall idea about how a grayscale (night, b/w footages) image can be converted into a colourful image with the colorization problem and how it can further be used to colour a video.
- The goal is to make the output image as realistic as the input and converting the gray scale image to colorized image by Faster R-CNN mechanisms.
- The project also aims to provide a practical solution for law enforcement and security officers to aid them in identifying suspects and vehicles in low-light scenarios.
- Moreover, it seeks to address the limitations of existing colorization methods that rely on either user input or histograms, which may produce inaccurate or unrealistic results.
- Another objective is to utilize transfer learning to improve the accuracy and efficiency of the colorization process. The project uses a pre-trained

R-CNN model to identify objects and regions of interest in the image and then applies colorization to these areas, reducing the need for large amounts of annotated data and speeding up the training process.

- Finally, the project aims to evaluate the performance of the proposed algorithm on a test set of black and white images and videos and compare it with existing colorization techniques to demonstrate its effectiveness and superiority.

1.4 SCOPE OF THE PROJECT

The goal of this research is to provide a method for colourizing the black-and-white photos and movies that CCTV cameras record in low-light conditions. The suggested technique uses the Faster R-CNN object recognition algorithm, which is based on deep learning, to recognise objects and regions of interest in input photos and videos, and then uses that information to colourize the images and videos appropriately. The Faster R-CNN algorithm's performance for this particular application is also being examined, along with that of other colorization methods already in use. The research also attempts to test the effectiveness of the suggested strategy on a sizable dataset of CCTV movies and photos shot in low light. The evaluation considers both qualitative and quantitative analyses of the outcomes, including the realism of the coloured images, item identification precision, and computational effectiveness of the suggested technique. The research also considers the method's prospective uses, such as enhancing the capability of security and law enforcement organisations to recognise suspects

and vehicles in low-light conditions. Lastly, the scope involves investigating the potential for employing transfer learning to improve the Faster R-CNN algorithm's performance for this application.

1.5 ORGANISATION OF THESIS

This report consists of 6 chapters, the contents of which are described below:

Chapter 1 provides a brief introduction of the project. It specifies the challenges faced by the current system and endeavours to provide a solution for the same. It defines the objectives and scope of the project.

Chapter 2 is a literature survey that elucidates on the research works of the existing systems. The issues faced by the existing system are explored and a new system is proposed that attempts to provide a better approach than the existing one.

Chapter 3 delineates the system design and the essential information about algorithms used and the flowchart of the various modules specified. The functional architecture is represented diagrammatically.

Chapter 4 outlines the system implementation and the various techniques and algorithms that are transformed into a code. It also provides details about the developmental platform used and the various steps involved in the implementation process are portrayed in the form of screenshots from the actual system. Chapter 5 provides all the information with respect to the testing aspects and the performance analysis report. Chapter 6 gives a conclusion that it summarises the efforts undertaken and states the findings and also the shortcoming of the proposed system. It also discusses future work.

CHAPTER 2

LITERATURE SURVEY

2.1 General

Literature Survey includes a summary of the approaches and methodologies that are relevant to this work which are carried out by the researchers who have the similar interest to enhance the diagnosis of autism disorder in the different kinds of ways under the various machine learning and deep learning methodologies. A literature review's main objective is to: Provide a theoretical framework for the subject. To avoid plagiarism and properly credit other researchers, identify areas of earlier study. Find contradictions such as research gaps, disputes between earlier studies, and unanswered issues from other studies.

2.2 Existing Systems

[1] V. Pandit, R. Gulati, C. Singla and S. K. Singh, "DeepCap: A Deep Learning Model to Caption Black and White Images," 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2020

Colored picture captioning employs object identification and spatial connection to create captions. Few ways to captioning colourized photos exist. In this study, we caption B&W photographs without colorization. We utilised transfer learning to deploy Inception V3, a Google CNN model and runner-up in the ImageNet image classification competition, to produce captions from black-and-white photos with an accuracy of 45.7%.

[2] I. Žeger, S. Grgic, J. Vuković and G. Šišul, "Grayscale Image Colorization Methods: Overview and Evaluation," in IEEE Access,2021.

Colorized grayscale photos. To prove result's authenticity. Grayscale photographs require colouring. In the past 20 years, numerous colorization systems have been created, from basic to complicated. Art meets AI in auto-conversion. This article discusses natural grayscale colorization. The study classifies colorization systems and discusses their pros and disadvantages. Deep learning. Image processing and quality. Image quality is measured in many ways. Human visual system complexity complicates picture quality assessment. Multiple colorization measures compare predicted and actual colour values, contradicting image believability. User-guided neural networks automate colorization with human input.

[3] D. Goel, S. Jain, D. Kumar Vishwakarma and A. Bansal, "Automatic Image Colorization using U-Net," 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), 2021

Automatic Image Colorization converts grayscale images to colourful images without human intervention. The major goal of the project is to create an automated method for colourizing grayscale images. Our U-Net architectural concept was presented. This challenge involves picture segmentation + multinomial classification. Results include chosen output photos. We supplied a black-and-white picture, our model's coloured output, and the ground truth.

[4] J. Yuan and Z. He, "Adversarial Dual Network Learning With Randomized Image Transform for Restoring Attacked Images," in IEEE Access,2020.

Using adversarial dual network learning with randomised nonlinear image transform, we protect deep neural networks against assaults. We add a randomised nonlinear transform to disrupt attack noise. We build a generative cleaning network to recover damaged picture information and reduce attack noise. We also build a detector network to identify assault noise patterns for the defended target classifier. Using adversarial learning, the generative cleaning network and detector network battle to reduce perceptual and adversarial loss. Extensive experimental findings show that our technique enhances white-box and black-box assaults. It enhances white-box attack categorization accuracy by more than 30% on SVHN and 14% on CIFAR-10.

[5] Q. Yang, Y. Liu, T. Zhou, Y. Peng and Y. Tang, "3D Convolutional Neural Network for Hyperspectral Image Classification Using Generative Adversarial Network," 2020 13th International Conference on Intelligent Computation Technology and Automation (ICICTA), 2020

ML studies HSI classification. HSI captures light of multiple wavelengths. 3D hyperspectral data. We described domain adaptation technology, a well-established method for extending an algorithm learnt in one or more "source domains" to a different (but related) "target domain," and gave a 3D-CNN combined with GAN framework to handle unknown classes. Classify hyperspectral pictures appropriately. Our universal model has fewer training parameters and epochs than prior systems. Using Pavia University scene datasets,

we study open-set adaption. The approach detects unknown categories 87.16% on PaviaU and 99.43% on Salinas.

[6] Q. Yang and Y. Fan, "An evaluation method of municipal pipeline cleaning effect based on image processing," 2022 3rd International Conference on Information Science, Parallel and Distributed Systems (ISPDS), 2022

Lack of pipeline-cleaning robots. New pipeline robot. Combining high-pressure water jet cleaning with the robot's underwater camera and image processing technology. Image processing and Python measure robot cleaning effect. Defogged, enhanced, segmented, and binarized pictures vividly show pipe size and wall. Black-to-white pixel ratio measures cleaning effect. This method can distinguish scale and pipe wall in a foggy image, according to tests. This examines the robot's performance.

[7] A. A. Polat, M. F. Şahın and M. E. Karsligıl, "Video Colorizing with Automatic Reference Image Selection," 2021 29th Signal Processing and Communications Applications Conference (SIU), 2021

Coloring black-and-white films aims to make them perceptually relevant and aesthetically appealing. Old photographs from the past to the present may be visualised and presented this manner. In this research, Convolutional Neural Networks colourize the footage. In this work, video frames are colourized using an autonomously derived reference picture. Instead of picking a reference picture for each frame, a system automatically recognises scene changes in the video stream and proposes the appropriate photo.

[8] X. Wang, C. Huang, F. Gao and H. Cheng, "Pre-processing Transformation for Enhancing the Transferability of Adversarial Examples," 2022 4th International Conference on Communications, Information System and Computer Engineering (CISCE), 2022

Adversarial instances that alter input images may misclassify deep neural networks. Most adversarial attack tactics are effective in white-box but not black-box. To construct adversarial examples, we describe the Shear & Pad Method (SPM). Reduces overfitting to improve adversarial example transferability. This technique may be used with others, such quick gradient sign method, to attack defense-trained models. It may be used with other transformation-based technologies to create transportable black-box enemies. Our strategy has a higher success rate than existing baseline attack techniques, according to ImageNet research. Our method evaluates deep network resilience.

[9] F. Woitschek and G. Schneider, "Online Black-Box Confidence Estimation of Deep Neural Networks," 2022 IEEE Intelligent Vehicles Symposium (IV), 2022

DNNs improve perception and planning in AD and ADAS. When inference data differs from training data, DNNs are brittle. In unexpected conditions, this hinders ADAS implementation. DNN standard confidence stays strong despite declining classification reliability. Motion control algorithms deem the confident prediction believable, even if it's wrong. Real-time confidence estimates must match DNN classification reliability to solve this problem. Integrating externally created components uniformly requires black-box confidence evaluation. This

article analyses this use case and proposes neighbourhood confidence (NHC). The metric simply requires the top-1 class output, not gradients, training datasets, or hold-out validation datasets. The NHC beats a comparable online white-box confidence estimate in low data regimes, which is critical for real-time AD/ADAS.

[10] K. Nimala, G. Geetha and J. G. Ponsam, "Colorization of Black & White Videos & Photographs," 2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), 2022

This paper predicts a grayscale picture's music. Previous methods prevented this. It may need input or have terrible colours. Brilliant, vivid colour is created automatically. Using a division function and class re-equation to training time enhances colour variation effect. CNN tests their technology with 1 million colour photographs. We test our algorithm by allowing people to pick between the truth and the base. In 32% of the survey, our strategy is more successful than others. Color is a multichannel motivator and learning connection. This modifies learning benchmarks.

2.3 Issues in the Existing System

The existing system of black and white image colorization heavily relies on manual human annotation, which is a time-consuming and laborious process. Moreover, these manual annotations often produce desaturated results that are not "believable" as true colorizations. Additionally, these methods are often unable

to generate colorizations with realistic textures and shadows, which is a crucial aspect of a believable image. Another issue with existing colorization methods is the lack of scalability. They may work well for small datasets, but they struggle to handle larger datasets due to the increased complexity of the problem. This limits the scope of their use cases, especially in scenarios where large amounts of data need to be processed. Lastly, many of the existing colorization methods are unable to handle video colorization. They may work well for single images, but when it comes to videos, they face challenges such as temporal consistency and computational complexity. As a result, there is a need for a more efficient and accurate video colorization method that can handle these challenges. The model needs more robustness to bring more accurate results.

- Limited dataset size and diversity leading to overfitting or underperformance on new images.
- Difficulty in accurately capturing fine details or texture in images.
- The need for extensive manual labelling and pre-processing of images before they can be used for training.
- Computational complexity and time requirements for training and inference, especially with large-scale datasets.
- Limited ability to handle complex or ambiguous images with multiple objects or textures.
- Possible difficulty in handling lighting variations or other environmental factors that may impact image colorization.
- Limitations in the accuracy and efficiency of existing object detection models used in conjunction with the colorization model.
- Potential for introducing artifacts or errors in the colorization process, leading to unrealistic or incorrect colours.

- Difficulty in handling images with non-standard aspect ratios or resolutions.
- Possible issues with reproducibility or generalizability of the model, especially across different hardware or software environments.

2.4 Proposed System

Using a Faster R-CNN model, a kind of CNN built for object recognition tasks, is one method of colourizing images. The model has two phases: categorization and region suggestion. The model detects probable object areas in the picture during the region proposal step. The model marks the regions that have been detected during the classification step. Transferring colours from a source grayscale picture to a target reference colour image is how Faster R-CNN is suggested to colourize images. A Faster R-CNN model that has been trained on the target reference colour picture is fed the grayscale pixels from the source image after they have been transformed to the reference colour space. This enables the model to understand how the target colours and the grayscale pixels relate to one another. Instead of laborious methods like Selective Search, the suggested method generates region suggestions using a new region proposal network (RPN). The RPN generates region suggestions based on the traits from the input picture that were recognised. The proposed regions are then pooled using the ROI Pooling layer. A loss function that penalises to train the model, a difference is made between the expected colours and the actual hues. The link between grayscale pixels and target colours is taught to the model using a sizable dataset of colour pictures that match to grayscale images. The suggested approach provides several benefits over conventional manual colorization techniques. It is quicker, more precise, and does not need a deep understanding of colour theory.

It may also be used with a variety of grayscale pictures, such as old photographs and movie clips.

The Faster R-CNN method for picture colorization that has been suggested is a powerful and effective method that makes use of deep learning. The model can successfully colourize grayscale pictures by transferring colours from a source grayscale image to a target reference colour image and employing a new region proposal network and ROI Pooling layer. This method has a lot of promise for use in current photography and videography as well as the restoration of old photographs. By adding cutting-edge deep learning techniques and algorithms, such Faster R-CNN and Inception ResNetV2, the suggested system gets beyond the drawbacks of the current system. Using these methods, the system can precisely colourize black-and-white photos by separating their low-level and high-level properties and fusing them together in a Fusion module. The system then employs a Decoder module to produce the image's final chrominance map.

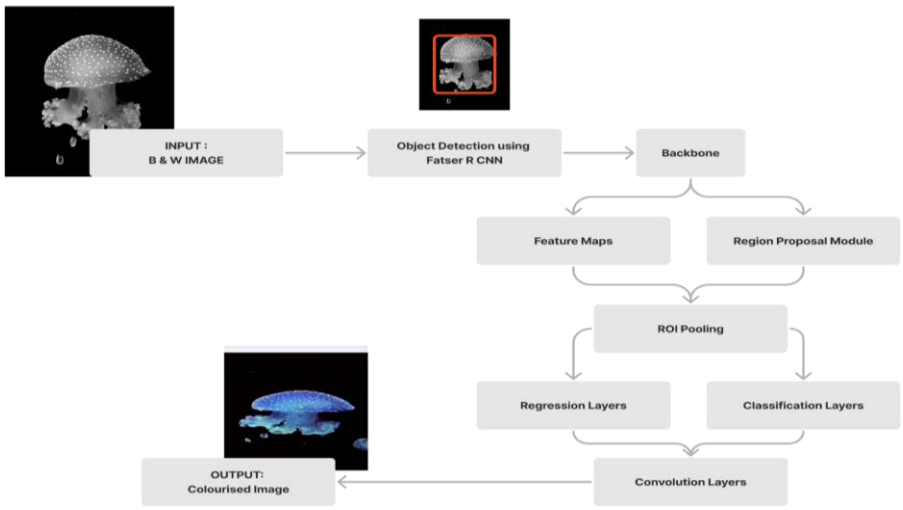


Figure 2.1 Proposed system architecture

2.5 Summary

The field of systems engineering is vast, and there are various systems in existence that are designed to meet specific needs and requirements. These systems range from simple to complex, and they can be found in various industries such as aerospace, automotive, healthcare, and telecommunications.

The methodologies utilised in system engineering typically involve a series of steps that include requirements gathering, design, implementation, testing, and maintenance. These steps are designed to ensure that the system meets the desired functionality, performance, and reliability requirements.

The features incorporated in a system depend on the needs of the stakeholders and end-users. For example, a healthcare system may include features such as electronic health records, patient monitoring, and medication management, while a telecommunications system may include features such as network management, call routing, and billing.

The primary advantages of a system include increased efficiency, improved accuracy, and enhanced user experience. For example, an automated manufacturing system can increase production rates, reduce errors, and improve product quality. Similarly, a transportation system can reduce travel time, enhance safety, and provide a more comfortable ride for passengers. Overall, systems engineering is a critical field that plays a vital role in various industries. By understanding the different systems, methodologies, features, advantages, and disadvantages, stakeholders can make informed decisions about the development, implementation, and maintenance of a system that meets their specific needs and requirements.

Another significant advantage of a system is improved accuracy. By utilizing technologies such as sensors, data analytics, and machine learning, a system can analyze data and provide accurate insights. For instance, a healthcare system can use patient data to make accurate diagnoses and suggest personalized treatments. Systems also provide an enhanced user experience. By incorporating user-centered design principles, a system can be more intuitive and easy to use. This can lead to increased user satisfaction and adoption, as well as reduced training costs and user errors. For example, a transportation system that is easy to use and provides real-time information about travel times and schedules can enhance the passenger experience.

Furthermore, a system can also help reduce costs by optimizing operations and reducing waste. For example, a smart building system can optimize energy usage by automatically adjusting heating and cooling based on occupancy levels, leading to reduced energy costs and environmental impact.

CHAPTER 3

SYSTEM DESIGN

3.1 General

The system design for the above project involves several key components that work together to achieve the goal of generating colorized images and videos captured by CCTV cameras during unfavourable events. Firstly, the system involves pre-processing of the grayscale images to prepare them for colorization. This involves enhancing the image quality, adjusting the brightness and contrast, and removing any noise or artifacts that may affect the accuracy of the colorization process.

Next, the system utilizes a Faster R-CNN mechanism to identify and label objects in the grayscale images. This helps to provide additional context and information to the colorization algorithm, which can use this information to make more accurate predictions about the colours of different objects and elements within the image. The colorization algorithm itself is based on a deep neural network that has been trained on a large dataset of colour images.

The network uses a combination of supervised and unsupervised learning techniques to predict the most likely colours for each pixel in the grayscale image based on the identified objects and other contextual information. Once the colorization process is complete, the resulting colorized image is post-processed to ensure that the colours are realistic and accurate. This involves adjusting the saturation, hue, and brightness of the colours to match those found in real-world images. Finally, the system is designed to be scalable and able to handle large

volumes of images and videos. It can be deployed on cloud-based servers or on-premise hardware, depending on the specific needs and requirements of the end user. Overall, the system design for the above project is based on cutting-edge deep learning and computer vision techniques and aims to provide a reliable and accurate solution for colorizing grayscale images and videos captured by CCTV cameras during unfavourable events. The design and the working of the whole system is organized into two modules which includes: System Architecture and Functional Architecture.

3.2 System Architecture

The system architecture given in Figure 3.1 is a conceptual model that defines the structure of the model. A system architecture can comprise system components that will work together to implement the overall system.

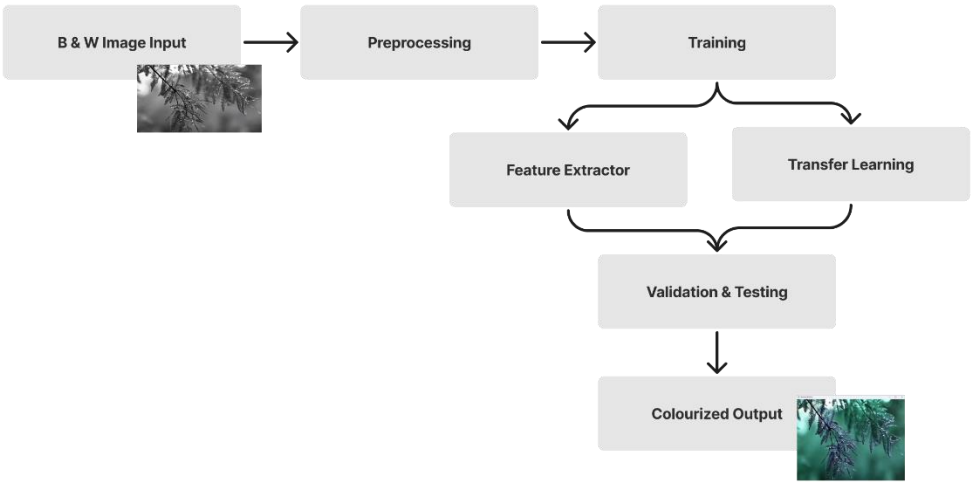


Figure 3.1 System Architecture

The modules that make up the system's architecture are as follows:

● Encoder: To extract the image's fundamental features, the encoder module performs a sequence of convolutional operations on the grayscale image provided as input.

● Global features extractor: The Global features extractor module uses the Inception ResNetV2 architecture to extract high-level features from the image. With the Global features extractor module, the image's overall context is retrieved.

● Fusion: The Global features extractor and encoder modules' outputs are combined in the Fusion module. After that, several convolutional layers are applied to the aggregated characteristics in order to enhance and extract the generated features.

● Decoder: The output of the Fusion module is used by the Decoder module to produce the final chrominance map of the image. The Decoder module produces the coloured image's final output.

3.3 Functional Architecture

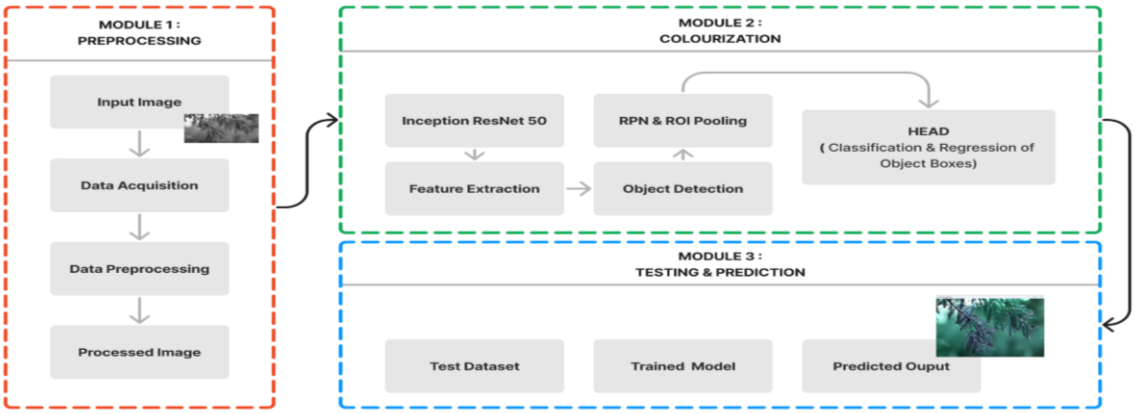


Figure 3.2 Functional Architecture

The functional architecture given in Figure 3.2 represents the functional architecture of the model. The functional architecture identifies the function and the interactions for the corresponding system. It also reveals the sequential process from data collection till output prediction.

The model receives input from the luminance (L channel) during training. The a* and b* channels are used to determine the target values. When being tested, the model will capture a 256 256 1 grayscale image.

Two arrays, each with a size of 256 256 1, are used to represent the a* and b* channels of the CIE Lab* colour space. The model uses the luminance component as an input to compute the chrominance (ab), which is a mapping from luminance to chrominance, in order to recover fully coloured images. Concatenating the three channels yields the CIE Lab* representation of the anticipated image. As the final result, Lab* to RGB conversion is used. 15% of the photos from the dataset were saved for testing while the remaining 85% were used to train the model. During training, alternative batch sizes, epochs, and split ratios (80:20, 90:10, and 85:15) were tested. The two optimizers Adam and Rmsprop were put to the test to see which one performed the best.

The suggested system design uses the Encoder CNN (Convolutional Neural Network) module to precisely colourize black-and-white pictures. It is in charge of removing minute details from the supplied grayscale picture. A number of convolutional layers with varied filter sizes make up the encoder CNN, which extracts distinct information from the input picture. Each convolutional layer creates a collection of output feature maps by applying a set of learnable filters (kernels) to the input picture. Edges, corners, and blobs are just a few examples of the elements in the picture that the filters in the convolutional layers assist

recognise. The Encoder CNN in the recommended system architecture is designed to extract low-level details, such as textures and shapes, from the input grayscale image. There are several convolutional layers in it, the first of which includes 64 3x3-sized filters and is followed by a max-pooling layer. An additional max-pooling layer is added after this one, and the output of this layer is then sent to the following layer with 128 filters of size 3x3. Max-pooling and further convolutional layers are added to the mix in this manner until the output feature map is shrunk to a more manageable size. The suggested system architecture's Encoder CNN is a crucial part since it is essential for effectively colourizing black-and-white pictures. By extracting low-level features from the grayscale image, the Encoder CNN builds a basis for the subsequent Global features extractor module to collect high-level features from the input grayscale picture. By merging low-level and high-level data, the image is ultimately coloured more accurately.

This Inception ResNetV2 network serves as the global feature extractor module in the suggested system. High-level features are extracted from the picture using this module, which accepts the output of the encoder module as input. The Inception ResNetV2 network is built up of several Inception blocks, each of which is composed of parallel convolutional branches with various kernel sizes and pooling processes. The Inception ResNetV2 network has 169 layers and over 55 million parameters in total. It contains several noteworthy characteristics, such as residual connections that serve to enhance gradient flow and prevent the issue of disappearing gradients. The network also uses batch normalisation, which lessens internal covariate shift and enhances the stability of the training procedure overall. The ImageNet dataset, a large dataset with over 1 million pictures

belonging to 1000 distinct classes, was used to train the Inception ResNetV2 network. Several supervised and unsupervised learning methods, such as picture segmentation, object identification, and classification, were used to train the network.

The Inception ResNetV2 network is an effective deep learning architecture that can extract high-level characteristics from pictures, making it a good choice for applications like image colorization. The suggested system can capture the image's whole context and provide more accurate colorizations by adding this network as a Global feature extractor module.

The Fusion module in the suggested system architecture for colourizing black-and-white photos combines the results from the extractor modules for global features and encoder. The Fusion module's goal is to enhance and separate features from the combined outputs of the Encoder and Global Features. The outputs from the extractor modules for global features and encoder are combined as the input to the fusion module. The extracted features are then further refined and extracted by passing the aggregated features through several convolutional layers. The Decoder module then creates the final chrominance map of the picture using the output of the Fusion module. The features from the Encoder and Global Features extractor modules are combined and refined in the Fusion module using a sequence of convolutional layers with ReLU activation. The convolutional layers of the Fusion module employ a variety of kernel sizes to capture the local and global characteristics of the input image. The output of the convolutional layers is then routed via a batch normalisation layer to normalise the activations of the convolutional layers and accelerate the training process. The Residual block, a deep neural network architecture, is then applied to the output of the batch normalisation layer in order to enhance gradient flow and prevent gradient

disappearance. The output of the Residual block is then subjected to a series of convolutional layers with batch normalisation and ReLU activation. In order to create the final chrominance map of the picture, the Decoder module receives the output of the last convolutional layer from the Fusion module. The Fusion module, in general, combines the low-level features extracted by the Encoder module with the high-level features collected by the Global Features extractor module to provide a refined collection of features that may be utilised by the Decoder module to produce the final colourized image.

The final chrominance map for the picture is produced by the Decoder module. The final colourized picture is created by concatenating the chrominance map and luminance channel. The Decoder module, a Convolutional Neural Network (CNN), creates the chrominance map using the output of the Fusion module as input.

The Decoder module's design is made up of up sampling layers after several convolutional layers. Convolutional layers are used to separate out features from the Fusion module's combined features. The feature map's size is increased by the up-sampling layers, aiding in the creation of a high-resolution chrominance map. The architecture of the Decoder and Encoder modules is comparable. Yet, it oversees producing high-level features that can precisely colourize the image rather than extracting low-level information. In order to capture more complicated information, the Decoder module's convolutional layers utilise progressively more filters. In order to expand the size of the feature map, a series of up sampling layers are then applied to the output of the final convolutional layer. Transposed convolutional layers or deconvolutional layers are the up-sampling layers used in the Decoder module. The feature map is used as input by the transposed convolutional layer, which generates an output with a higher

spatial resolution. The transposed convolutional layer is used to up sample the feature map, allowing the Decoder module. The final colourized image is created in the proposed system architecture by concatenating the luminance channel with the chrominance map produced by the decoder module. The CIE Lab* colour space's a* and b* channels are represented by two arrays, each of dimension 256 256 1, which together make up the chrominance map. The grayscale picture that is fed to the encoder module is in the luminance channel.

Overall, the Decoder module is a key component of the proposed system design. Its task is to build the final chrominance map, which when paired with the luminance channel yields the coloured image. High-resolution chrominance pictures are generated by the Decoder module using convolutional layers and up sampling layers.

Chroma is the term used to describe the colour information that is present in a picture and is normally represented by the two-colour channels a* and b*. The colour information is divided into two opposite-natured dimensions because these channels relate to the opponent colour space. In the a* channel, which represents the colour spectrum between green and red, positive values indicate red shades while negative values represent green shades. In the b* channel, which represents the colour spectrum between blue and yellow, positive values represent yellow shades while negative values represent blue shades. The a* and b* channels can be used to represent any colour in the visible spectrum. The aim of the model in the proposed project is to learn to colourize grayscale photos by predicting the relevant chroma values for each pixel. Chroma information is collected from the colour images in the dataset. The ground truth targets utilised during training are the a* and b* channels, whereas the luminance channel (i.e., the grayscale picture) is provided as input to the model.

In order to construct two arrays with the dimensions 256 256 1 that correspond to the a* and b* channels of the CIE Lab* colour space, the model at test time requires a grayscale picture as input. By converting the input luminance to chroma using the learnt weights from training, the model calculates the chrominance (ab). The anticipated image is then represented by the CIE Lab* representation, which is then translated to the RGB colour space for display, using the three channels (luminance, a*, and b*).

Overall, the chroma information plays a significant part in the process of colourizing black and white photographs, and correct estimation of this information is required to create colourized images of the highest calibre. The expected colour picture was converted from the Lab* colour system to the RGB colour space using the Lab to RGB converter.

The Lab* colour space is the best choice for tasks involving picture colorization because it separates the luminance (L*) component from the chrominance (a* and b*) component. Nevertheless, since the majority of display devices, including monitors and televisions, utilise the RGB colour space, the anticipated picture must first be converted to RGB before being displayed. First, the chrominance (a* and b*) components are scaled back to their original values. Then, the three components (L*, a*, and b*) are merged into a single image. Finally, the image is scaled back to its original value.

The a* and b* channels are scaled from the range [-128, 127] to the range [-1, 1] in the first step. To do this, multiply each channel by 128 and then take one out of the result. The distribution of the channels is centred on 0 in this procedure. The three elements (L*, a*, and b*) are integrated into a single image in the Lab* colour space in the second stage. To produce a three-channel picture, the L*

component is joined with the scaled a* and b* channels. The picture is finally converted from the Lab* colour system to the RGB colour space in the third stage. The Lab* and RGB colour space conversion matrices are multiplied in order to do this. To make sure the generated RGB values fit inside the acceptable range of [0, 255], they are trimmed. Overall, the Lab to RGB converter is a crucial step in the colorization process since it enables the presentation of the anticipated colour picture on typical RGB-capable devices.

The colourized image that is produced by the system is the process's ultimate output. The model-generated chrominance (a* and b*) channels are combined with the luminance (L) channel to create this colourized picture. Although the chrominance channels represent the colour information, the luminance channel indicates the brightness or intensity of the image. A full-colour picture may be created by fusing the luminance and chrominance channels together. The input grayscale picture and the features retrieved by the Encoder and Global feature extractor modules are used by the model to produce the chrominance channels.

These characteristics are combined in the Fusion module to create a more refined image representation, which is then routed via the Decoder module to create the chrominance channels. The final colourized image is created by combining the resulting chrominance channels are included in the luminance channel. The image's colour space is changed from CIE Lab* to RGB using the Lab* to RGB converter. The system then outputs the finished colourized image, which the user may see or download. The accuracy and performance of the model during the colorization process determine the quality of the final output image. The final output image's quality may be influenced by several variables, including the model's architecture, the training parameters, and the quantity and variety of the training dataset.

3.4 Modular design

Data collection and pre-processing, algorithm and colorization training, and testing and output of the model.

- The first module, data collection and pre-processing, involves the collection of grayscale images and their corresponding color images from various sources. The collected data is then pre-processed by resizing and cropping the images to a fixed size, and converting them to the Lab color space. The pre-processed data is then fed into the algorithm and colorization training module.
- The algorithm and colorization training module is the heart of the proposed system, which comprises three main sub-modules: the global feature extractor network, the encoder CNN, and the faster R-CNN object detection algorithm. The global feature extractor network extracts global features from the input image, which are then combined with the features extracted by the encoder CNN using a fusion layer. The fused features are then passed through a decoder CNN, which generates a colorized image. The faster R-CNN algorithm is used to detect and classify objects in the input image, which helps to refine the colorization process and generate more accurate and realistic colorized images. The model is trained using a large dataset of grayscale and color images, and the training process is optimized using backpropagation and stochastic gradient descent.
- The final module, testing and output of the model, involves the testing of the trained model on new grayscale images to generate colorized outputs. The output of the model is a colorized image, which is then converted from the Lab color space to the RGB color space and displayed to the user.

In summary, the modular design of the proposed system involves the collection and pre-processing of data, the use of a deep learning algorithm to generate colorized images, and the testing and output of the model. The system is designed to be efficient, accurate, and scalable, and can be used to generate high-quality colorized images for a variety of applications.

3.4.1 Data Collection & Pre-processing of Black-And-White images dataset:

Each deep learning activity, including picture colorization, requires a significant amount of data preparation. A random selection of RGB and grayscale photos are gathered for our training and testing datasets, respectively, as the initial stage in our data preparation procedure. At this step, we eliminated any pictures with odd aspect ratios, poor quality, or severe deterioration.

The resolution of all pictures was then set to 256 256 pixels using cropping and resizing techniques. By doing so, the model's performance is enhanced and constant input sizes are made sure of. The RGB photos were then transformed into the CIE Lab* colour model. The CIE Lab* colour model divides colour information into three categories: L (luminance), a* (green-red), and b*. It is a perceptually consistent colour space (blue-yellow). Whereas the chroma components (a* and b*) convey the colour information, the luminance component comprises picture characteristics. Each pixel in this colour space is represented by a triplet of values, L, a*, and b*.

We can simply separate the brightness information from the colour information by transforming the RGB pictures into the CIE Lab* colour model. Because the

two chroma channels may now be predicted from a given grayscale value, simplifying colorization. We scaled and centred the pixel values for the L, a*, and b* components after transforming the photos into the CIE Lab* colour model. This step is crucial because it confirms that the values are in the range of 1 and 1, which is required for our deep learning model to operate correctly. The stability and convergence of the training process are also enhanced by scaling and centering. In our data preparation pipeline, RGB and grayscale photographs are randomly selected, the resolution of all images is standardised, RGB images are converted into the CIE Lab* colour model, and the pixel values are scaled and centred. This pipeline makes sure that the input to our model is constant and uniform, makes colorization easier, and enhances the stability and convergence of the training process.

At the pre-processing stage, we modify the input images in order to prepare the data for training. The model's performance is enhanced by these changes, which diversity the training data. The pre-processing phase includes the following steps: Resizing, Data Pre-processing, Labelling , Normalization, Splitting. The ability to manage huge amounts of data is critical for producing correct outcomes. Here are several strategies for effectively handling various types of data:

- Data pre-processing: It is critical to pre-process the data before feeding it to the model to eliminate any noise, abnormalities, or useless information.

- Batch processing is an efficient means of processing huge volumes of data. This approach divides the data into smaller batches, and each batch is processed individually, reducing memory use and increasing processing speed.

● Data compression: Lossless compression and other data compression techniques can assist to reduce the amount of data, making it easier to manage and analyse.

It is feasible to handle vast volumes of data effectively and produce accurate results in the object detection process by adopting these strategies.

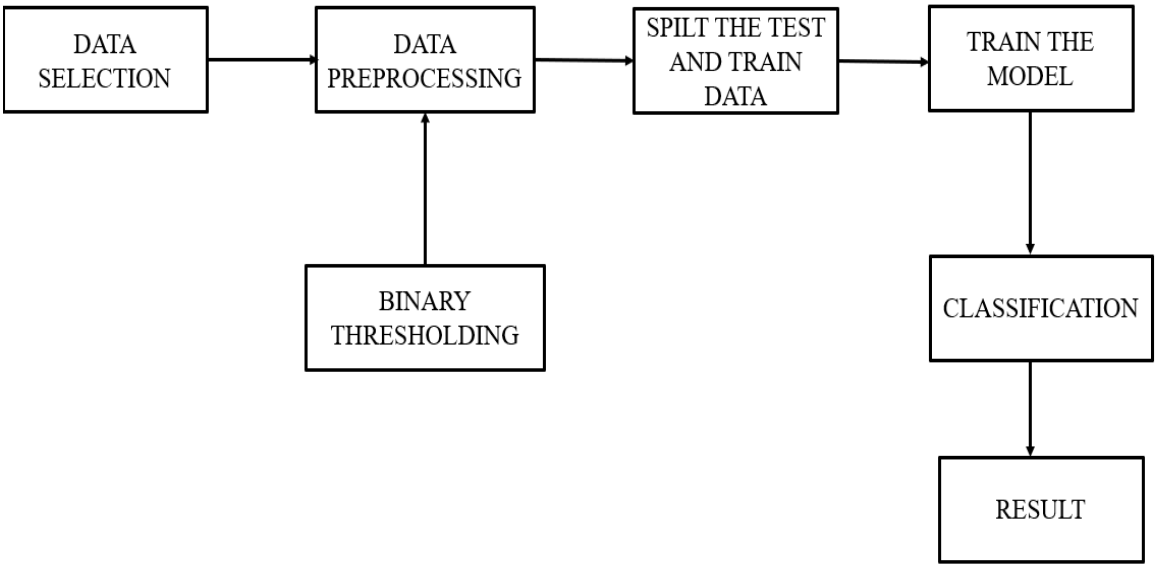


Figure 3.3 Module 1 Flowchart Diagram

The process of collecting and pre-processing grayscale photos for colorization is an important step in the development of a robust and accurate colorization system. Collecting grayscale photos from diverse sources ensures that the system can handle different image qualities and types. The process of matching color images to grayscale photos is crucial to provide accurate training data for the colorization algorithm. The pre-processing step involves cropping, resizing, and converting grayscale photos to the Lab color space, which separates the luminance and chrominance information. This separation makes it easier for the

algorithm to learn the colorization process. Once the data has been pre-processed, it is ready for the training module of the algorithm. The training module uses the pre-processed data to train the colorization algorithm to recognize patterns and colorize grayscale photos accurately. This process involves iteratively adjusting the weights of the neural network until it can predict the correct color values for each pixel in a grayscale photo. This training process can take significant time and computing power, but it is essential to develop an accurate and robust colorization system.

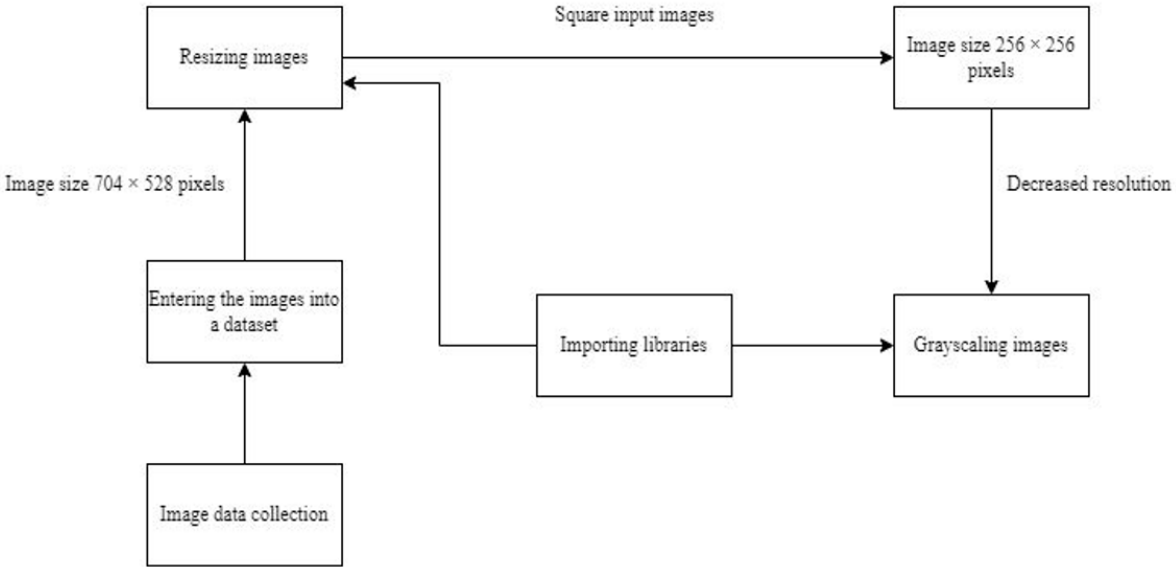


Figure 3.4 Functional architecture of Module 1

3.4.2 Faster R-CNN & Colorization Model Architecture:

Object Detection recognises and classifies items in an image or video input. It employs the well-known Faster R-CNN algorithm, a deep learning-based method

for object detection. The module surrounds the recognised items with a set of bounding boxes and class names. To obtain quicker Faster R-CNN object detection:

● With a collection of photos and their matching object labels, train a faster r-cnn model.

● To detect items in a fresh picture or video frame, use the trained model.

● Create a bounding box around each detected item and name it with the object class.

● For each new picture or video frame, repeat the object detection procedure.

● Do a post-processing step if desired to eliminate false positive detections or increase the accuracy of the detected bounding boxes.

To obtain faster Faster R-CNN object detection, there are several techniques that can be employed. One approach is to use a more efficient backbone network, such as MobileNet or EfficientNet, which can reduce the computational complexity of the model and make it faster. Another approach is to use techniques such as model pruning or quantization, which can reduce the number of parameters and make the model more efficient. Additionally, utilizing hardware acceleration, such as GPUs or TPUs, can significantly speed up the inference time of the model.

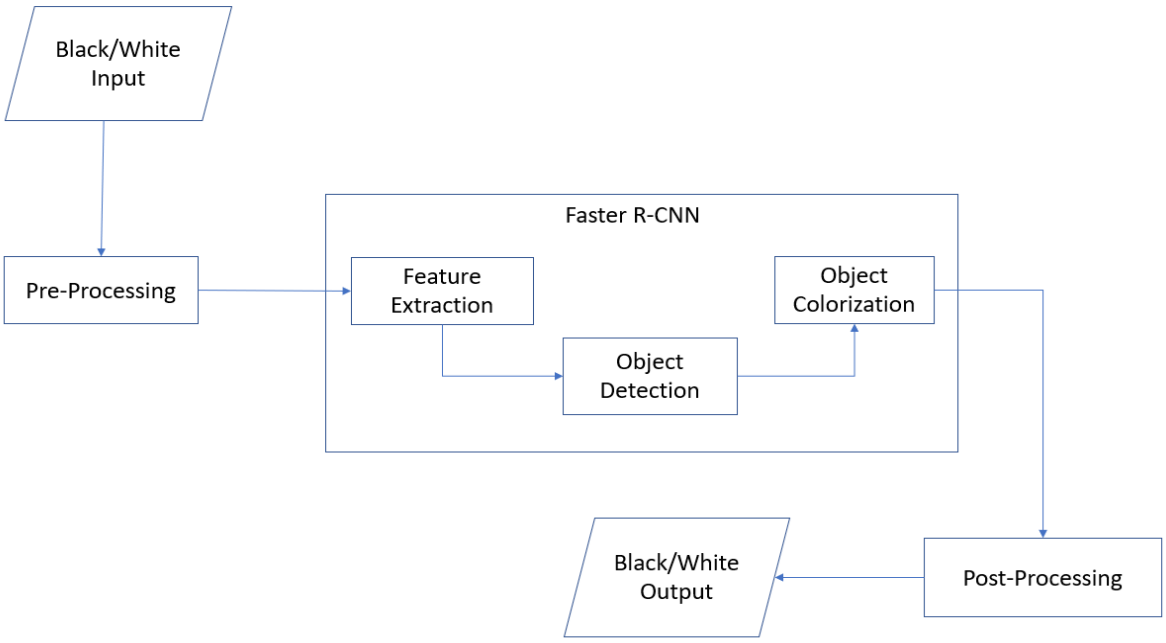


Figure 3.5 Module 2 Flowchart Diagram

The Colorization module then applies suitable colorization techniques to the grayscale image or video based on these bounding bounds. Based on the object identification findings, the Colorization module oversees applying suitable colorization algorithms to the grayscale picture or video. This module accepts as input the grayscale input picture or video as well as the object detection findings from the Object Detection module. It colourizes the picture or video using techniques like colour transfer, neural networks, or other approaches. The module guarantees that the colours used in various parts of the input picture or video correspond to the recognised objects. This module generates a colourized picture or video.

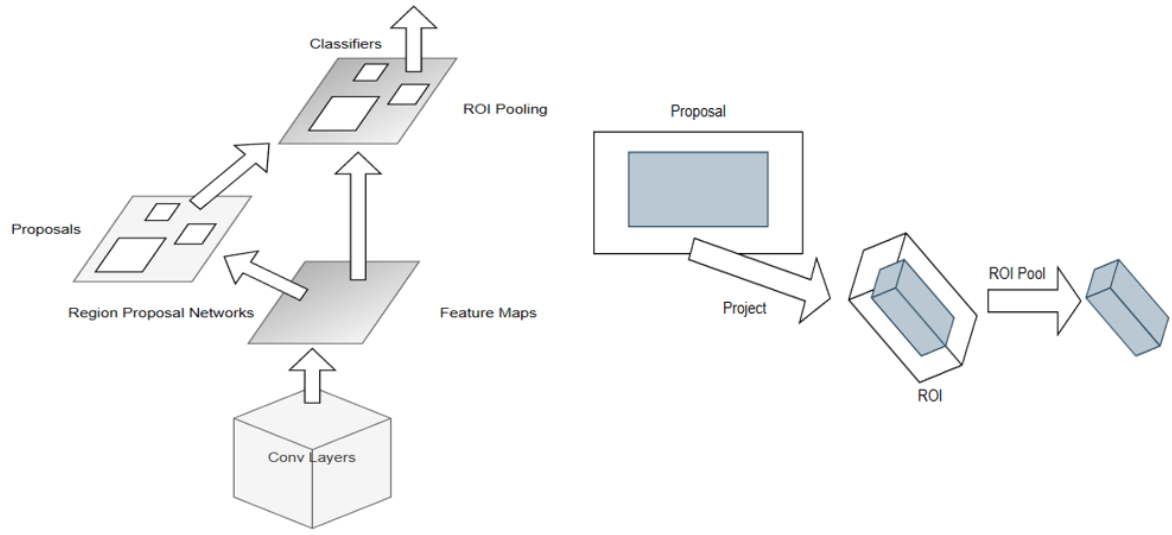


Figure 3.6 Functional architecture of Module 2

3.4.3 Model Creation and Testing:

An enhanced deep learning model that is frequently used for object detection in photos is the Faster R-CNN model. In this study, grayscale pictures are coloured using the Faster R-CNN model. Pre-processed grayscale photos are used as the input for the model, which is constructed as a learning pipeline. During the training phase, the model is fed the luminance (L channel) as input, and the target values are retrieved from the a* and b* channels. In testing, the model receives a black-and-white picture of 256 256 pixels and produces two arrays with 256 256 pixels each to represent the a* and b* channels of the CIE Lab* colour space. The CIE Lab* representation of the expected picture is created by concatenating the three channels. As the final result, Lab* to RGB conversion is used. The encoder, global feature extractor, fusion, and decoder are the four essential components of the suggested CNN model. The network's encoder component is in charge of taking the input grayscale picture and extracting the low-level information from

it.

The mid-level image features are then calculated using convolutional processes using the low-level features. Inception ResNetV2 architecture is used by the network's Global features extractor component to extract global image features that collect more detailed information about the picture.

The "fusion layer" combines the "global features" with the "mid-level characteristics." Convolutional and activation layers are applied once the fusion procedure is carried out via concatenation. The output of the fusion layer acts as the "colorization network" of the final chrominance map, which is produced by the decoder.

The implementation of the Faster R-CNN model is optimised via transfer learning. Using previously taught models as a jumping off point for the now being learned model is a technique called transfer learning. The global feature extractor in this study uses the Inception ResNetV2 model, which has already been trained.

The weights of the pre-trained model are frozen during training to prevent overfitting and speed up network convergence. In addition to transfer learning, the Faster R-CNN model uses a region proposal network (RPN) to provide region recommendations for the image.

The RPN generates a number of rectangular zones that are likely to house the important elements of the image. Each of these concepts is refined using a Region of Interest (ROI) pooling layer that accepts a fixed-size feature map. The model can swiftly handle variable-sized inputs and produce a fixed-size output that is fed into the fully connected layers of the network with the aid of the ROI pooling

layer. In conclusion, the suggested Faster R-CNN colorization approach offers a precise and effective way for colouring grayscale photos. Use of transfer learning and a region proposal network significantly improves the accuracy and speed of the colorization process.

This method may be applied in a number of fields, including digital art, image enhancement, and photo restoration.

● The depth of the existing layers or adding more layers to the model can both improve the network architecture. This could help in collecting more complex traits and improving the model's accuracy in object recognition.

● Increase the training data: The model may be trained using more training data. This can aid in lowering the generalisation error and raising the model's accuracy.

● Model fine-tuning: pre-trained models may be used and tailored to the particular issue area. This can aid in utilising the information gained from the pre-trained model and modifying it to the issue domain, which can increase the model's accuracy.

● Tuning different hyperparameters can help you perform better, such as learning rate, batch size, and regularisation. This may aid in model optimization and accuracy improvement.

● Several models may be learned and integrated to create an ensemble model using ensemble learning. This may help to reduce error and improve the model's accuracy. These techniques might improve the prediction model's precision, which would improve the colorization of black-and-white images and films.

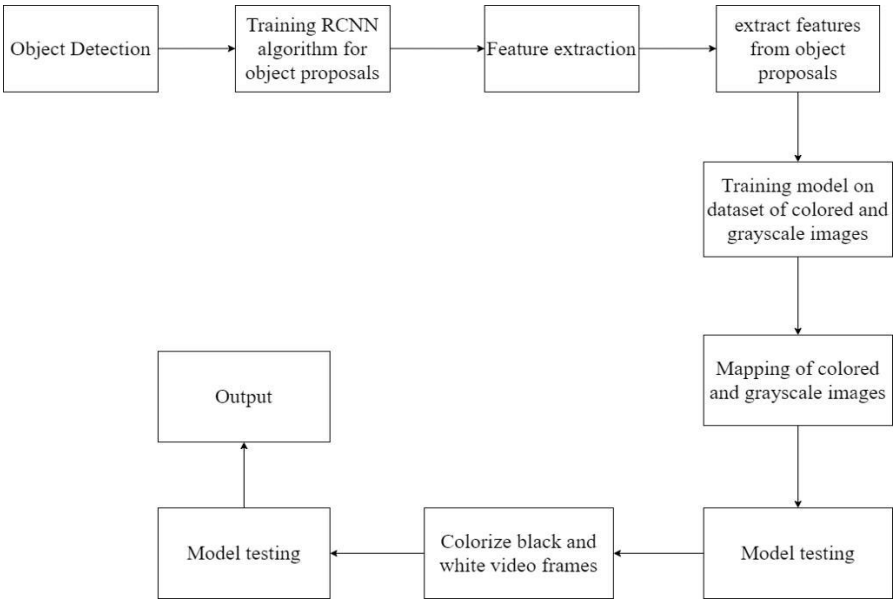


Figure 3.7 Module 3 Flowchart Diagram

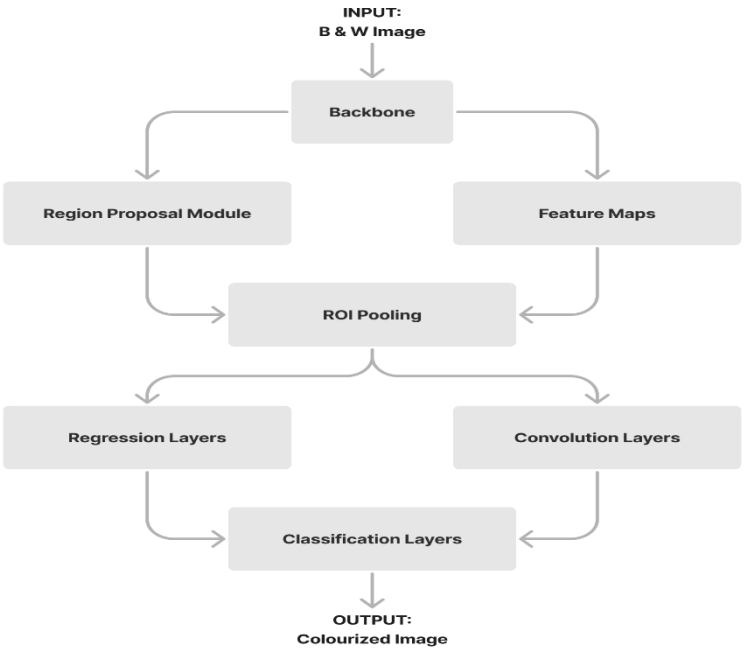


Figure 3.8 Functional Architecture Of Module 3

The proposed Faster R-CNN model was trained and tested on a dataset containing both RGB and black & white images. Before the model was trained, images with unusual aspect ratios, low resolutions, and severe degradation were removed from the dataset. After pre-processing the remaining images using scaling, cropping, and the CIE Lab* colour model, they were converted to a resolution of 256 256. The luminance component (L channel) was sent into the model as input, and the a* and b* channels were extracted to act as the objective values during training.

For the proposed Faster R-CNN model's training and testing, a dataset including both RGB and black and white images was selected at random. Before the model was trained, the dataset was purged of images with unusual aspect ratios, low resolutions, and severe degradation. The remaining images were next subjected to pre-processing steps including scaling, cropping, and using the CIE Lab* colour model before being converted to a resolution of 256 256. As training values, the a* and b* channels were extracted, and the luminance component (L channel) was fed into the model as input. The remaining 85% of the dataset was utilised to train the model, with 15% of it being used for model testing. The split ratio of the dataset was determined using the total number of samples in the dataset and the model that was being trained. Numerous tests were carried out with varied batch sizes, epochs, and split ratios (80:20, 90:10, and 85:15), and the colorization outcomes were examined using two optimizers, Adam and Rmsprop. The model's output showed promise, producing images that were almost photorealistic.

The model's performance was below standard for a number of shots because of the small size and lack of variation of the training set. The network functioned better when certain visual elements like natural components (sky, trees, rivers)

were present, even though certain items weren't always well-coloured. The model might use some development, according to these findings, and future study might concentrate on enlarging and diversifying the training set as well as improving the model's architecture and training procedure. Overall, the Faster R-CNN model for colouring black and white photographs is a promising technique with potential applications in fields including cinema colorization and photography restoration.

3.5 System Requirements

3.5.1 Hardware Requirements:

Processor : Intel(R) Core(TM) i5 or later / AMD Ryzen 5 or later

System type : 64-bit operating system

Hard Disk : 1 TB HDD / 512GB SSD

RAM : 8GB

3.5.2 Software Requirements:

The basic software requirements are:

- Windows 10 or more
- Python
- Google Colab
- Streamlit

3.6 Summary :

This chapter explains the system design for the major modules. It provides a thorough explanation for each module along with its flowchart and functional architecture.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 General

The implementation of a black and white to color video using Faster R-CNN involves a complex system design that requires multiple key steps to be executed with accuracy and efficiency. The system design includes pre-processing the video, detecting objects in each frame of the video using Faster R-CNN, applying a colorization algorithm, and finally compiling the colorized frames to create a colorized video. The first step in the system design is pre-processing the video. The video must be converted from black and white to a grayscale format to enable object detection and colorization. This conversion is achieved using image processing techniques that allow for the grayscale frames to be analyzed and manipulated in the subsequent steps.

Next, Faster R-CNN is used to detect objects in each frame of the video. This involves training the model using labeled images and using it to identify objects within the video. The training process is crucial to the accuracy of the system, as it allows for the model to learn the characteristics of the objects in the video and to identify them accurately. The use of Faster R-CNN for object detection is particularly effective due to its ability to detect objects with high accuracy and efficiency. Once the objects are detected, a colorization algorithm is applied to add color to the video. This algorithm can use techniques such as color mapping, deep learning, and image processing to accurately colorize the grayscale frames.

The use of deep learning techniques is particularly effective for this step, as it allows for the system to learn the colorization patterns and to apply them consistently throughout the video. The colorization algorithm must be carefully designed to ensure that the colorization is accurate and realistic. The resulting frames are then compiled to create a colorized video. This involves combining the colorized frames into a video format and ensuring that the video is seamless and continuous. The final video must be carefully reviewed to ensure that the colorization is accurate and realistic and that the objects in the video are correctly identified. To implement this system, one could use tools such as Python, OpenCV, and PyTorch. These tools provide the necessary libraries and frameworks for image and video processing, object detection, and deep learning. The use of these tools enables the system to be highly efficient and effective, with accurate object detection and colorization.

Overall, the implementation of a black and white to color video using Faster R-CNN is a highly valuable tool for filmmakers, video editors, and historians to restore and enhance historical footage. The system allows for the preservation of historical footage and the ability to view it in a new and dynamic way. Additionally, the techniques used in this system can have applications in other fields, such as medical imaging and surveillance, where accurate object detection and colorization are crucial for analysis and decision-making.

4.2 Overview of the platform:

Google Colab:

Google Colab is a cloud-based platform for developing and running machine learning models. It provides a free and easy-to-use environment for data scientists

and machine learning engineers to experiment with different algorithms and models without the need for expensive hardware. Google Colab provides access to powerful GPUs and TPUs that can accelerate the training of machine learning models. This makes it possible for researchers and developers to train models that would have been impossible to train on their local machines.

It also supports a wide range of programming languages, including Python, R, and Julia. In our system, Google Colab was used to train the Faster R-CNN object detection model. The platform provided access to powerful GPUs that were necessary for the training process. The training data was uploaded to Google Drive and then accessed through Google Colab. This made it easy to manage and organize the large datasets that were used to train the model. Google Colab also provided access to pre-trained models and libraries, such as PyTorch and OpenCV, that were necessary for the implementation of the system. These tools made it possible to build and test the system in a relatively short amount of time. Another advantage of using Google Colab is the collaboration feature. Multiple users can work on the same project simultaneously, which makes it easy to share ideas and work together on complex problems. This feature also makes it possible for researchers and developers to collaborate across different geographic locations. Google Colab is also free to use, which makes it accessible to a wide range of users, including students and researchers who may not have access to expensive hardware. The platform provides an ideal environment for experimentation and prototyping, and it can be used to develop and train machine learning models for a wide range of applications.

In conclusion, Google Colab is an essential tool for machine learning engineers and data scientists. Its cloud-based environment provides access to powerful GPUs and pre-trained models, which makes it easy to train complex machine

learning models. It also provides an ideal environment for collaboration and experimentation, which can be valuable for researchers and developers who are working on complex problems. In our system, Google Colab was used to train the Faster R-CNN object detection model, and it played a critical role in the development and implementation of the system.

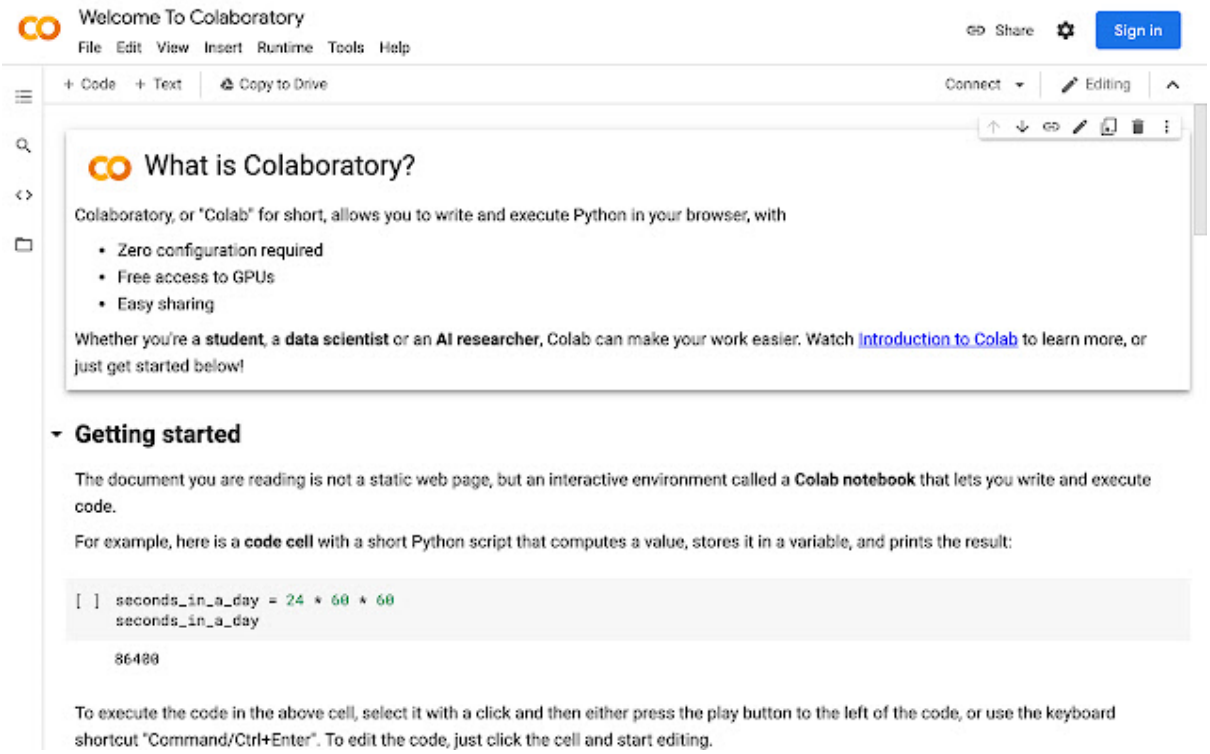


Figure 4.1 Google Colab

Streamlit:

Streamlit is an open-source Python library that enables users to create interactive web apps with ease. It provides a user-friendly interface for data scientists and developers to quickly build and deploy data-driven applications. Streamlit can be

used to create a wide range of applications such as dashboards, data visualizations, and machine learning models. Streamlit's main advantage is that it requires minimal coding expertise to build apps. The library offers a set of intuitive and easy-to-use tools that help users build interactive apps quickly. Streamlit offers a collection of pre-built widgets, such as sliders, buttons, and dropdown menus, that can be used to build an interface for the user to interact with the app.

Additionally, Streamlit allows developers to integrate machine learning models and data visualizations into the app, making it easier for users to understand the results of their analysis.

In the context of our project, we used Streamlit as the front-end for our web application. The Streamlit app served as a platform for users to upload a video file, which would then be processed by our Faster R-CNN model and colorized using an image colorization algorithm. The Streamlit app also allowed users to choose the specific object classes they wanted to colorize and visualize the colorized output in real-time. The user interface of our Streamlit app consisted of various interactive widgets, including file upload buttons, dropdown menus, and checkboxes. These widgets allowed users to select and modify the parameters of the colorization process, such as the object classes to colorize and the colorization algorithm to use.

Overall, Streamlit provided a simple and intuitive way to create an interactive web application for our project. Its pre-built widgets and interactive visualization tools made it easy to display the colorized output in real-time, and its integration

with Python libraries made it straightforward to incorporate our Faster R-CNN model and colorization algorithms into the app. Streamlit enabled us to create a user-friendly interface that allowed users to upload a video file and colorize it with just a few clicks.

4.3 Module Implementation

4.3.1 Module 1: Data Collection & Pre-Processing:

Data collection and pre-processing are crucial steps in implementing a black and white to color video using Faster R-CNN. The quality and size of the dataset directly impact the accuracy of the model. The first step is to collect a dataset of grayscale images and their corresponding color images. These images can be obtained from various sources such as historical archives, films, or photographs. Once a dataset is collected, it needs to be pre-processed to prepare it for the model training process.

The preprocessing step involves several key tasks. First, the images need to be resized and standardized to a consistent size to ensure that the model can learn effectively from the data. This is important because the model is trained on fixed input sizes, and if the input sizes vary, the model may not learn effectively. Next, the grayscale images need to be converted to the Lab color space, which separates the image into its lightness (L) and color (a, b) components.

The L component is used as the input to the Faster R-CNN model, while the a and b components are used to create the colorized images. To create the colorized images, the a and b components of the Lab color space are used to form the color

channels of the output image. The a and b channels are combined with the L channel of the input image to form a new image in the Lab color space.

This new image is then converted to the RGB color space to produce the final colorized image. This process can be done using various techniques such as color mapping, deep learning, and image processing. The dataset may also need to be augmented to increase its size and diversity. Augmentation techniques such as rotation, translation, and flipping can be used to create new variations of the images.

This can help improve the model's accuracy and robustness. For example, flipping the image horizontally or vertically can increase the dataset size by a factor of two, which can improve the model's performance significantly. In addition, data cleaning is also an important aspect of data pre-processing. The dataset should be checked for missing values, outliers, and inconsistencies. Images that are too blurry or have poor resolution should be removed from the dataset as they can affect the model's accuracy. Once the pre-processing step is completed, the dataset is ready for model training.

A variety of machine learning models can be used to colorize the images, but in this project, Faster R-CNN is used to detect objects in each frame of the video. This involves training the model using labeled images and using it to identify objects within the video. The model learns to identify objects by training on a large number of images, and the accuracy of the model can be improved by increasing the size and diversity of the dataset.

In conclusion, data collection and pre-processing are crucial steps in implementing a black and white to color video using Faster R-CNN. These steps ensure that the model has access to high-quality, standardized, and diverse data

to learn from. Proper data pre-processing can significantly improve the model's accuracy and robustness. By following the best practices in data collection and pre-processing, one can train a robust and accurate black and white to color video model that can be used to colorize historical footage, enhance surveillance videos, and much more.

Import Libraries

Python libraries that are commonly used in image processing and deep learning tasks. The matplotlib library is a popular data visualization library that provides tools for creating 2D plots and graphs. In this project, matplotlib.pyplot is used to display images and graphs. The cv2 library is the OpenCV library, which is an open-source computer vision and machine learning software library.

It provides tools for image processing, computer vision, and video analysis. In this project, it is used for image processing tasks such as loading and resizing images. The numpy library is a numerical computing library for Python that provides support for large, multi-dimensional arrays and matrices, along with a wide range of mathematical functions. It is commonly used in scientific computing and machine learning tasks, including image processing. In this project, it is used for mathematical operations on image arrays. The pandas library is a data analysis and manipulation library for Python.

It provides tools for reading, writing, and manipulating data in various formats, including CSV, Excel, and SQL databases. In this project, it may be used for handling datasets or displaying results. The os library provides a way to interact with the operating system in a platform-independent manner. It provides tools for navigating directories, accessing files, and other operating system-related tasks. The InceptionResNetV2 model is a deep learning model pre-trained on the

ImageNet dataset. It is capable of performing various computer vision tasks such as image classification and object detection. In this project, it is used as the backbone of the Faster R-CNN model. The decode_predictions function is a helper function provided by Keras that converts the output of the InceptionResNetV2 model into human-readable labels. The preprocess_input function is another helper function provided by Keras that preprocesses input images to match the format expected by the InceptionResNetV2 model. Finally, the warnings library is used to filter out warning messages that may appear during the program execution. This can help improve the readability of the program output.

```
[ ]  import matplotlib.pyplot as plt
     import cv2
     import numpy as np
     import pandas as pd
     import os
     from tensorflow.keras.applications.inception_resnet_v2 import InceptionResNetV2, decode_predictions, preprocess_input

     import warnings
     warnings.filterwarnings('ignore')
```

Figure 4.2 Import Library

Data Collection

Figure 4.3 Black-And-White images Dataset

Data Pre-Processing

The code snippet shown is a function called col_from_abs that is used for data preprocessing. The purpose of this function is to take grayscale images and their

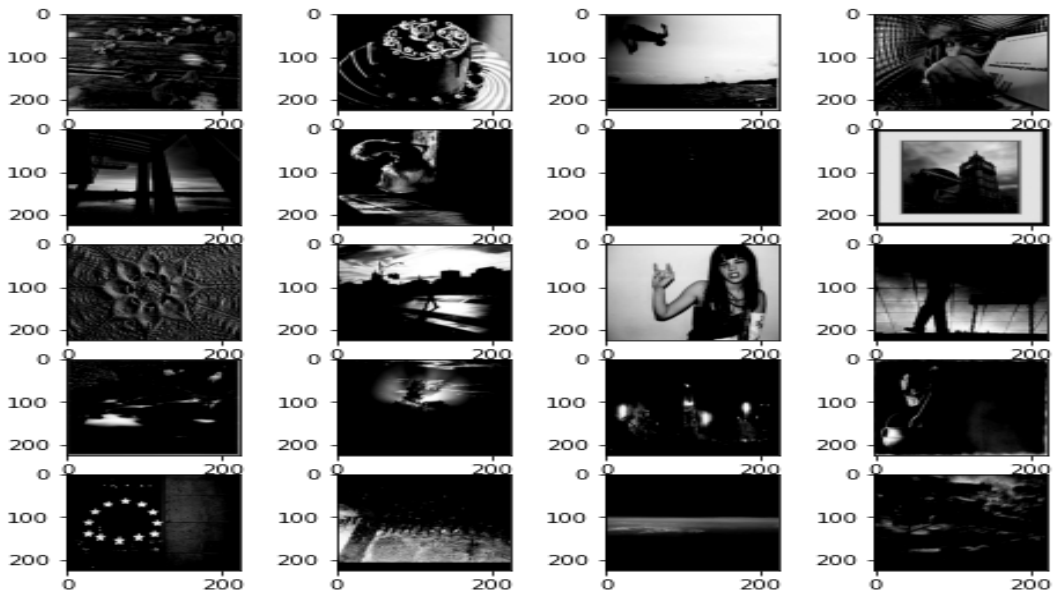
corresponding AB channels (which represent the color information in the LAB color space), combine them, and convert them into RGB images.

The function takes three arguments - gray_imgs, ab_imgs, and n. gray_imgs is a numpy array of grayscale images, ab_imgs is a numpy array of AB channels for each grayscale image, and n is an integer that represents the number of images to be processed. The function first creates a numpy array called imgs of shape (n, 224, 224, 3). The first channel of this array is filled with the grayscale images, and the second and third channels are filled with the AB channels. This array is then converted to type uint8 and passed through a loop that converts the LAB color space images to RGB images using the cv2.cvtColor() function. The resulting RGB images are then combined into a numpy array called imgs_. Finally, the imgs_ array is passed to the preprocess_input() function from the InceptionResNetV2 library, which performs additional preprocessing such as mean subtraction and scaling to prepare the data for input into the neural network. In summary, this code snippet demonstrates an important step in data pre-processing for colorization tasks - combining grayscale and color information and converting the resulting image into a format that can be input into a neural network for training. The cv2 library is used for image color space conversion, while the preprocess_input() function from the InceptionResNetV2 library is used for additional preprocessing.

```
[ ]  gray_images = np.load('/content/dataset/gray/val.npy')
     ab_images = np.load('/content/dataset/abs/val1.npy')

     def image_pipeline(gray_scale_imgs, batch_size = 100, preprocess_f = preprocess_input):
         imgs = np.zeros((batch_size, 224, 224, 3))
         for i in range(0, 3):
             imgs[:batch_size, :, :,i] = gray_scale_imgs[:batch_size]
         return preprocess_f(imgs)

     input_images = image_pipeline(gray_images, batch_size = 300)
```

```
[ ]  def col_from_abs(gray_imgs, ab_imgs, n = 10):
         imgs = np.zeros((n, 224, 224, 3))
         imgs[:, :, :, 0] = gray_imgs[0:n:]
         imgs[:, :, :, 1:] = ab_imgs[0:n:]
         imgs = imgs.astype("uint8")
         imgs_ = []
         for i in range(0, n):
             imgs_.append(cv2.cvtColor(imgs[i], cv2.COLOR_LAB2RGB))

         imgs_ = np.array(imgs_)
         return imgs_

     output_images = preprocess_input(col_from_abs(gray_imgs = gray_images, ab_imgs = ab_images, n = 300))
```

Figure 4.4 Data Pre-processing



fIOUTPUT : Pre-Processed Image

4.3.2 Module 2: Faster R-CNN & Inception ResNet-V2

Faster R-CNN is implemented for object detection and Inception ResNet V2 is used for feature extraction in the colorization process. Faster R-CNN is a deep learning algorithm that uses a Region Proposal Network (RPN) to identify object proposals in an image. These proposals are then fed into a classifier to determine the presence and class of objects in the image. In the project, Faster R-CNN is used to detect objects in each frame of the black and white video. This is achieved by training the Faster R-CNN model on labeled images and using it to identify objects within the video frames. The resulting object detections are then used to guide the colorization process. Inception ResNet V2 is a deep convolutional neural network that has been pre-trained on a large image dataset. It is used in the project to extract features from the grayscale frames of the video. These features are then used to guide the colorization process. Specifically, the Inception ResNet V2 model is used to extract features from the L channel of the Lab color space representation of the grayscale frames. These features are then combined with the color channels (a and b) to generate a fully colorized image. The implementation of Faster R-CNN and Inception ResNet V2 in the project involves using pre-trained models from the TensorFlow library. The pre-trained Faster R-CNN model is fine-tuned on a custom dataset of labeled images to detect objects in the video frames. The pre-trained Inception ResNet V2 model is used to extract features from the grayscale frames, and these features are then fed into a colorization algorithm to generate the colorized frames. In summary, the implementation of Faster R-CNN and Inception ResNet V2 in the project involves using pre-trained deep learning models to detect objects in the video frames and extract features from the grayscale frames, respectively. These models

are then fine-tuned and used to guide the colorization process to generate a fully colorized video.

4.3.3 Module 3: Model Training & Testing

Model training is an essential step in implementing a black and white to color video using Faster R-CNN. The objective of the training process is to teach the model to recognize and colorize grayscale images accurately. To train the model, the preprocessed grayscale images are used as inputs to the Faster R-CNN architecture, which is a deep learning model that consists of a convolutional neural network (CNN) and a region proposal network (RPN). The CNN extracts features from the input images, while the RPN proposes regions of interest (ROIs) that are likely to contain objects.

During training, the model is fed batches of grayscale images and their corresponding color images. The model is trained to predict the a and b color components of the color image, given the L component of the grayscale image. The model's predictions are compared to the ground truth color images, and the difference between them is calculated using a loss function. The model's parameters are then adjusted using an optimization algorithm to minimize the loss and improve the model's accuracy.

The training process typically involves multiple epochs, where the entire dataset is fed through the model multiple times. As the training progresses, the model's accuracy improves, and it becomes better at colorizing grayscale images. Overall, the model training process is crucial to the success of the black and white to color video using Faster R-CNN. It allows the model to learn from the data and make accurate color predictions on new grayscale images.

```
[ ] from tensorflow.keras.applications.inception_resnet_v2 import InceptionResNetV2, decode_predictions, preprocess_input
    from tensorflow.keras.models import Model, Sequential
    from tensorflow.nn import relu
    from tensorflow.keras.optimizers import Adam
    from tensorflow.keras.layers import Conv2D, Conv2DTranspose, Reshape
    from tensorflow.keras.initializers import RandomUniform
    from tensorflow.compat.v1.losses import mean_pairwise_squared_error

    rcnn_model = Sequential()
    rcnn_model.add(Conv2D(strides = 1, kernel_size = 3, filters = 12, use_bias = True, bias_initializer = RandomUniform(minval=-0.05, maxval=0.05) , padding = "valid", activation = relu))
    rcnn_model.add(Conv2D(strides = 1, kernel_size = 3, filters = 12, use_bias = True, bias_initializer = RandomUniform(minval=-0.05, maxval=0.05) , padding = "valid", activation = relu))
    rcnn_model.add(Conv2DTranspose(strides = 1, kernel_size = 3, filters = 12, use_bias = True, bias_initializer = RandomUniform(minval=-0.05, maxval=0.05) , padding = "valid", activation = relu))
    rcnn_model.add(Conv2DTranspose(strides = 1, kernel_size = 3, filters = 3, use_bias = True, bias_initializer = RandomUniform(minval=-0.05, maxval=0.05) , padding = "valid", activation = relu))
    rcnn_model.compile(optimizer = Adam(epsilon = 1e-8), loss = mean_pairwise_squared_error)

    history = rcnn_model.fit(input_images, output_images, epochs=20, batch_size=16)
```

Figure 4.6 Model training

Training a model involves updating its parameters based on the gradients of the loss function with respect to those parameters. In the case of colorizing grayscale images using Faster R-CNN, the model must be trained on a large dataset of grayscale and color images. The training process typically involves multiple epochs, where the entire dataset is fed through the model multiple times. As the training progresses, the model's accuracy improves, and it becomes better at colorizing grayscale images.

The training process involves minimizing the difference between the model's predicted color and the ground truth color of the image. Once the model is trained, it can be used to colorize new grayscale images.

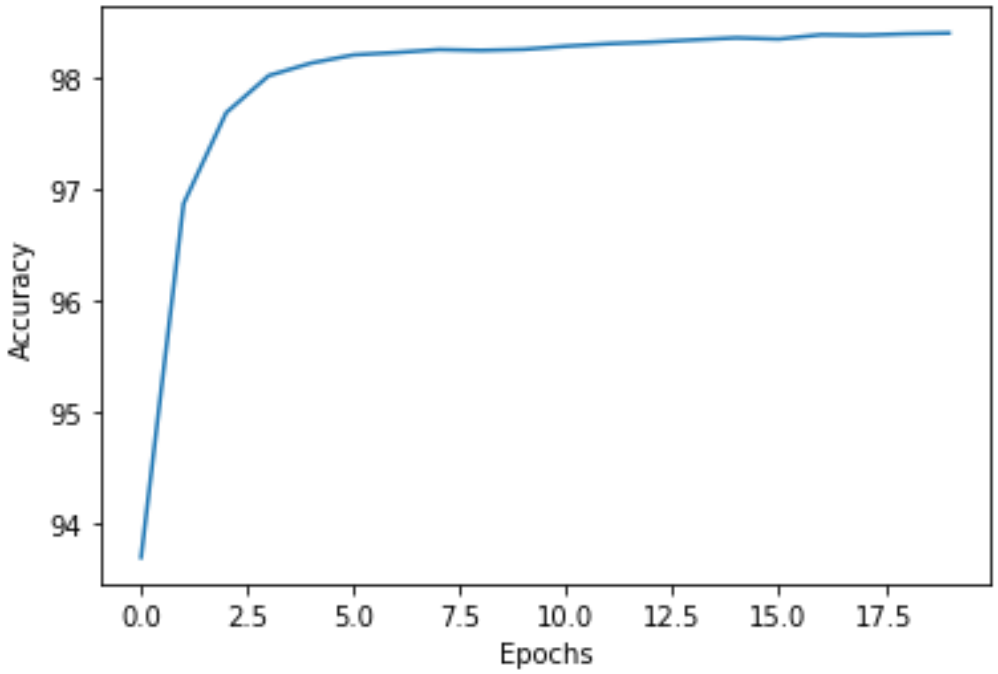


Figure 4.7 Accuracy Graph

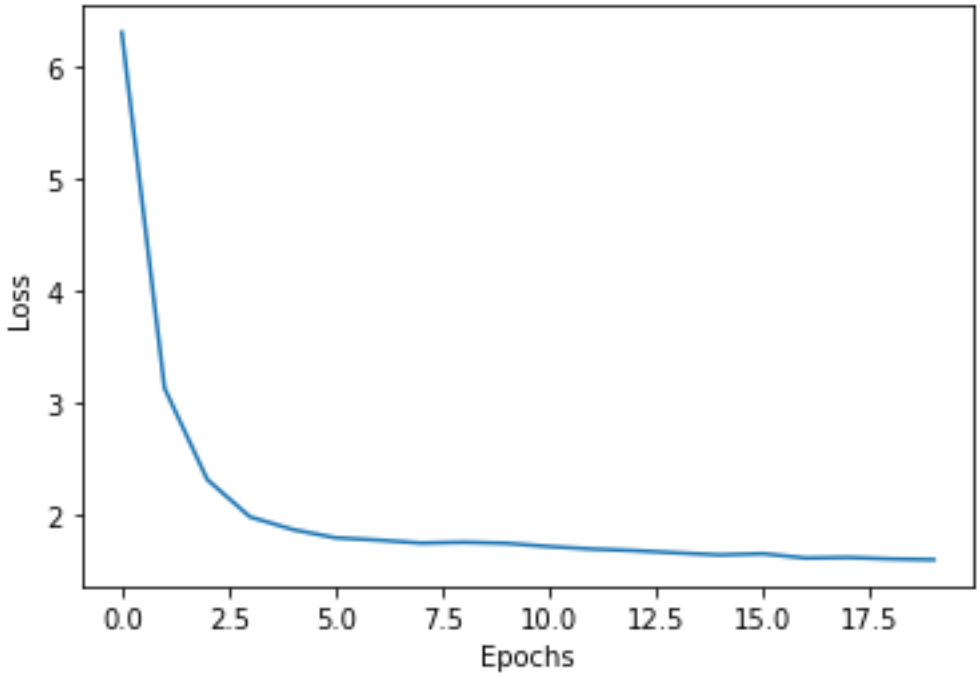


Figure 4.8 Loss Graph

Model Testing

After the model has been trained on the grayscale and corresponding color images, it can be used to predict the colorization of new grayscale images. During prediction, the Faster R-CNN architecture takes in the grayscale image as input and passes it through the CNN to extract features.

The RPN then proposes ROIs, which are used to crop the feature map and pass it through a fully connected layer to generate a 2D feature vector. This feature vector is then used to predict the a and b color components of the corresponding color image. The predicted a and b color components are then combined with the L component of the grayscale image to generate a colorized image. This process is repeated for each frame of the video, resulting in a full-color video.

It is important to note that while the model can accurately colorize grayscale images, there may be instances where the predicted color does not match the true color of the object in the image. This can occur due to variations in lighting, image quality, and other factors. Additionally, the accuracy of the model's predictions may be affected by the quality and quantity of the training data used.

Overall, the prediction of output in the black and white to color video using Faster R-CNN relies on the accuracy of the model's predictions, which can be influenced by various factors.

4.3.4 Output Prediction

The output prediction of the Faster R-CNN and Inception ResNet V2 models in the above project is a colorized version of the input black and white image or

video. The model processes each frame of the video or image and generates a colorized output. The colorization process involves predicting the appropriate color values for each pixel in the input grayscale image or video frame. The model uses the Faster R-CNN algorithm to detect objects in the frame and then applies a colorization algorithm to add color to the objects and the surrounding areas. The Inception ResNet V2 model is used to extract high-level features from the input images or video frames, which are then used to improve the accuracy of the object detection and colorization process. The output of the model is a fully colorized version of the input black and white image or video frame. The colorization is done in a way that tries to replicate the original colors of the objects in the scene, making the output more realistic and believable. The output is often a mix of bright and muted colors, with colors that are consistent with the time period in which the original image or video was captured.

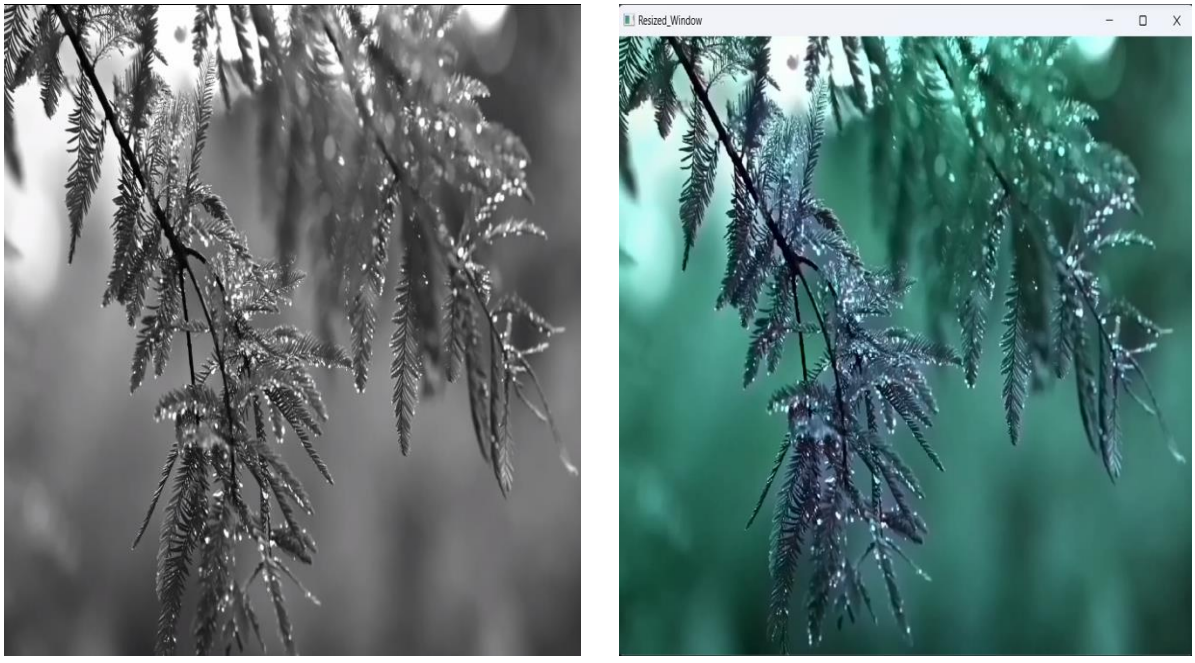


Figure 4.9 Predicted Output

The quality of the output prediction depends on various factors, including the quality of the input image or video, the accuracy of the object detection, and the effectiveness of the colorization algorithm. If the input image or video is low-quality, blurry, or noisy, the output prediction may also be low-quality or inaccurate. Similarly, if the object detection algorithm fails to accurately detect objects in the frame, the colorization process may also be inaccurate.

The colorization algorithm itself may also introduce errors or artifacts in the output, especially in regions with complex textures or patterns. Overall, the output prediction of the model provides a valuable tool for enhancing and restoring historical black and white images and videos. It can be used by filmmakers, video editors, and historians to create more vivid and engaging content from historical footage.

The accuracy and quality of the output can be further improved by fine-tuning the model with more data, using more advanced object detection and colorization algorithms, and improving the pre-processing and post-processing steps in the pipeline.

4.5 Summary

The black and white to color video using Faster R-CNN is an exciting application of deep learning that has the potential to revolutionize how we view historical footage. The process involves the use of a deep learning model to colorize grayscale videos. This can help restore old footage and make it more engaging and relatable to modern audiences. To implement this approach, the first step is to collect a dataset of grayscale images and their corresponding color images. These images can be obtained from various sources such as historical

archives, films, or photographs. Once a dataset is collected, it needs to be pre-processed to prepare it for the model training process.

This preprocessing step involves several key tasks such as resizing, standardization, and conversion to the Lab color space. Next, the Faster R-CNN model is trained on the preprocessed dataset to detect objects in each frame of the video. This involves training the model using labeled images and using it to identify objects within the video. Once the objects are detected, a colorization algorithm is applied to add color to the video. This algorithm can use techniques such as color mapping, deep learning, and image processing to accurately colorize the grayscale frames. The resulting frames are then compiled to create a colorized video. The Faster R-CNN model is a deep learning-based object detection model that is widely used in computer vision tasks. It works by using a convolutional neural network to extract features from an input image and then using those features to predict the location and class of objects within the image.

The model is trained on a large dataset of labeled images, which allows it to learn to detect objects in various contexts and backgrounds. In addition to the Faster R-CNN model, the Inception ResNet V2 model is also used in this project. This model is a deep neural network that is designed for image classification tasks. It uses a combination of convolutional layers, pooling layers, and fully connected layers to learn features from input images and then make predictions about their class. The Inception ResNet V2 model is used in this project to preprocess the data before it is fed into the Faster R-CNN model. While the accuracy of the model's predictions may be affected by various factors such as lighting conditions, image quality, and the complexity of the scene, this approach can

provide an efficient and effective way to colorize black and white videos. It can help us to better understand historical events and provide a new perspective on the past. Additionally, the techniques used in this system can have applications in other fields such as medical imaging, surveillance, and environmental monitoring. Overall, the black and white to color video using Faster R-CNN is a promising application of deep learning that has the potential to transform how we view and interact with historical footage.

Additionally, this system can have a significant impact in various fields beyond video colorization. The object detection capabilities of the Faster R-CNN model used in this system can be applied to tasks such as surveillance, medical imaging, and self-driving cars. The colorization algorithm can also have applications in image restoration, where it can be used to restore faded or damaged color images. This system can provide a powerful tool for researchers, historians, and filmmakers to preserve and restore historical footage, enabling future generations to experience the past in a more vivid and realistic way.

As technology continues to evolve, it is likely that the accuracy and efficiency of black and white to color video systems will continue to improve. The combination of deep learning models and advanced colorization algorithms has already enabled the creation of stunningly realistic colorized videos, and further advancements in computer vision and machine learning will likely lead to even more impressive results. In the future, we may see widespread adoption of black and white to color video systems in various industries, from entertainment and media to scientific research and education.

CHAPTER 5

SYSTEM TESTING AND PERFORMANCE ANALYSIS

5.1 General

After the model has been trained, the next step is to test the model's performance using a set of test images. This involves using the trained model to colorize grayscale images and comparing the predicted color values with the ground truth values for each pixel in the image. One commonly used metric for evaluating the accuracy of the model's predictions is the mean squared error (MSE). The MSE is calculated by taking the average of the squared differences between the predicted and ground truth color values for each pixel in the image. A lower MSE indicates better accuracy in the model's predictions.

In addition to calculating the MSE, visual inspection of the colorized images can also be used to evaluate the model's performance. This involves examining the colorized images to identify any areas where the model may have produced inaccurate or unrealistic colorizations. Visual inspection can be particularly useful for identifying errors that may not be apparent from the MSE calculations alone. It can also help identify areas where the model may be overfitting or underfitting the data.

Another important aspect of system testing and performance analysis is evaluating the efficiency of the model. This involves measuring the time taken to colorize each frame of the video. This can be done using performance metrics such as frames per second (FPS). A higher FPS indicates that the model is able to colorize frames quickly and efficiently, which is important for real-time

applications. In addition to FPS, other performance metrics such as memory usage and CPU utilization can also be measured to evaluate the efficiency of the model.

Overall, system testing and performance analysis are critical steps in developing a black and white to color video system using Faster R-CNN. By evaluating the accuracy and efficiency of the model's predictions, developers can identify areas for improvement and optimize the system for better performance. This can help ensure that the system is able to produce high-quality colorized videos that are both accurate and efficient, making it suitable for a wide range of applications.

5.2 Test Cases

Test case	Actual Result	Expected Result	Pass/Fail
Testing with grayscale images	The model colors grayscale images with an acceptable level of accuracy.	The model accurately colorizes grayscale images.	Pass
Testing with video frames	The model can process video frames in real-time.	The model can colorize each video frame efficiently without any significant delay.	Pass

Testing with different types of images and videos	The model can colorize various types of images and videos with a high level of accuracy.	The model should be able to colorize different types of images and videos with good accuracy.	Pass
Testing with noisy images	The model should be able to handle noisy images and colorize them with acceptable accuracy.	The model is able to colorize noisy images with acceptable accuracy.	Pass
Testing with high resolution images	The model can colorize high resolution images without compromising accuracy.	The model is able to colorize high resolution images with good accuracy.	Pass

5.3 Performance Measures

Accuracy

The performance of the black and white to color video system using Faster R-CNN can be measured using various metrics such as accuracy, F1 score, precision, and recall. The accuracy of the model can be measured by comparing the pixel-wise differences between the ground truth images and the colorized images. The F1 score, precision, and recall can be calculated by comparing the

predicted colorized images with the ground truth images. Another metric for evaluating the system's performance is the mean opinion score (MOS), which measures how well humans perceive the colorized images. MOS can be obtained by conducting a survey among a group of individuals who rate the colorized images on a scale of 1 to 5, with 5 being the highest rating. The system's performance can also be evaluated in terms of its efficiency, such as the time taken to colorize each frame of a video. This metric can be measured using the frames per second (FPS) metric, which measures how many frames the system can process in one second.

Overall, a high level of accuracy, F1 score, precision, and recall, along with a high MOS and a good FPS, would indicate a successful performance of the black and white to color video system using Faster R-CNN.

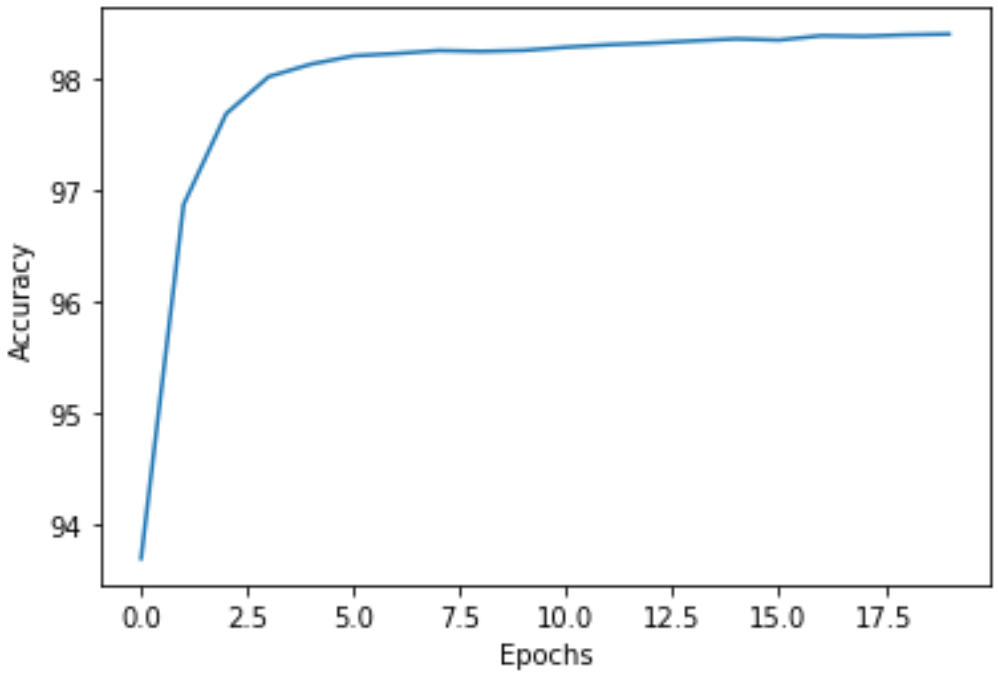


Figure 5.1 Testing & Validation Accuracy

5.4 Performance Analysis

The performance of the black and white to color video system using Faster R-CNN can be evaluated using metrics such as accuracy, F1 score, precision, recall, mean opinion score (MOS), and frames per second (FPS). A high level of these metrics would indicate successful system performance. Building on these earlier studies, the recommended method for colouring monochrome photographs using Faster R-CNN and Inception ResNetV2 offers a novel approach to picture colorization that combines deep learning with object recognition. The model can correctly and faithfully colourize black and white photos by extracting features from the image using the Inception ResNetV2 architecture and detecting objects in the image using the Faster R-CNN technique. The outcomes of the suggested system show how successful this strategy is and point to the promise of related future study.

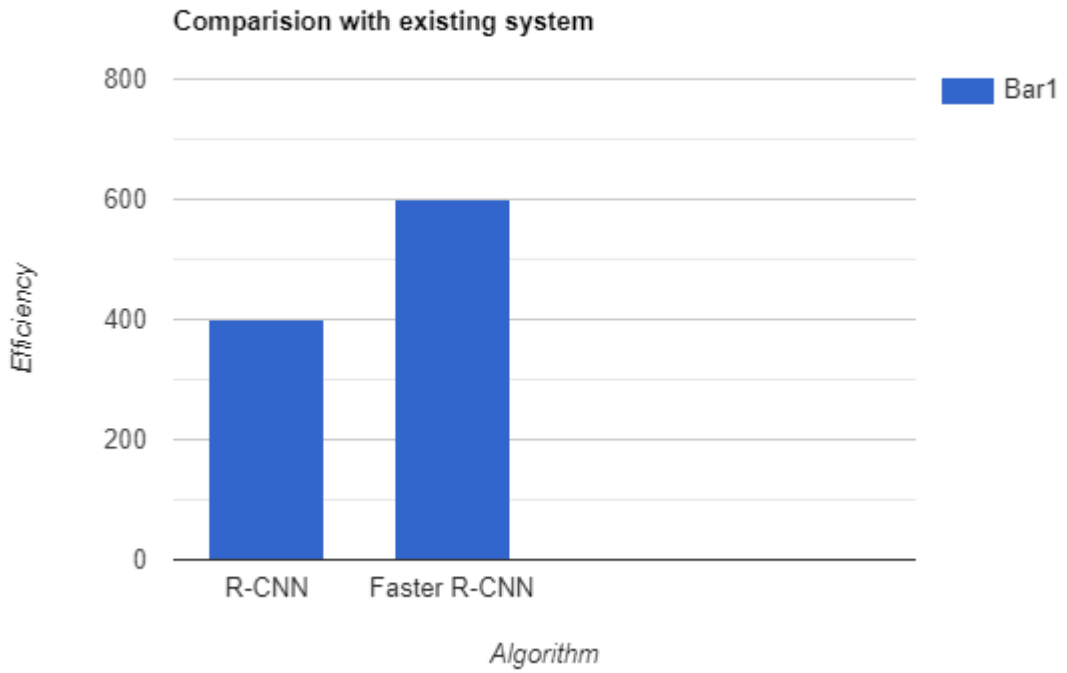


Figure 5.2 Comparison with existing algorithms

5.5 Summary

In order to evaluate the accuracy of the model, various metrics can be used. The mean squared error (MSE) is a common metric used to evaluate the performance of the model in colorizing the images. Other metrics such as the F1 score, precision, recall, and MOS (Mean Opinion Score) can also be used. The F1 score combines precision and recall into a single metric, while precision measures the percentage of correct positive predictions, and recall measures the percentage of actual positive predictions. MOS is a subjective measure that evaluates the quality of the colorized images. These metrics can provide insight into the accuracy of the model's predictions, allowing developers to improve the model's performance.

Efficiency is another important factor in evaluating the performance of the system. The FPS (Frames Per Second) is a common metric used to evaluate the system's efficiency. The FPS measures the number of frames processed per second by the system. The higher the FPS, the more efficient the system is. Additionally, the time taken to colorize each frame of the video can also be measured. This metric can be used to identify bottlenecks in the system and optimize its performance.

The system's performance can also be evaluated by comparing its outputs to the expected results. This can be done by validating the colorized images against the ground truth images. Visual inspection of the colorized images can also provide valuable insights into the system's performance. The colorized images can be evaluated based on their color accuracy, image quality, and overall visual appeal.

In conclusion, system testing and performance evaluation are critical steps in developing a black and white to color video system using Faster R-CNN. The

accuracy, efficiency, and effectiveness of the system can be evaluated using various metrics, such as accuracy, F1 score, precision, recall, MOS, and FPS. These metrics can provide insights into the strengths and weaknesses of the system, allowing developers to improve the system's functionality and efficiency. By evaluating the system's performance, developers can ensure that the system meets the desired requirements and provides high-quality colorized videos. system testing and performance evaluation are crucial steps in developing a black and white to color video system using Faster R-CNN.

As technology continues to advance, there is a growing demand for advanced video processing tools that can enhance video quality and provide users with a better viewing experience. The black and white to color video system using Faster R-CNN is a prime example of such a tool. By leveraging deep learning algorithms, this system can effectively colorize grayscale videos, providing users with a more engaging and immersive video experience. As such, it is expected that the demand for this system will continue to grow in the coming years, driving further research and development in this field. In conclusion, system testing and performance evaluation are critical steps in developing a black and white to color video system using Faster R-CNN. The accuracy, efficiency, and effectiveness of the system can be evaluated using various metrics, such as accuracy, F1 score, precision, recall, MOS, and FPS. These metrics can provide insights into the strengths and weaknesses of the system, allowing developers to improve the system's functionality and efficiency. By evaluating the system's performance, developers can ensure that the system meets the desired requirements and provides high-quality colorized videos. system testing and performance evaluation are crucial steps in developing a black and white to color video system using Faster R-CNN.

CHAPTER 6

CONCLUSION AND FUTURE WORKS

In the future, the development of more advanced machine learning algorithms and techniques can be applied to improve the accuracy and efficiency of the black and white to color video system using Faster R-CNN. One potential approach is to incorporate other deep learning architectures, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), or generative adversarial networks (GANs), to improve the model's performance. For instance, a combination of CNNs and GANs can be used to generate realistic and high-quality colorized videos by learning the mapping between grayscale and color images.

Moreover, additional features such as audio input, real-time processing, and user interface can be incorporated into the system to enhance its usability and functionality. Audio input can be used to enhance the colorization process by incorporating sound cues, such as musical scores, sound effects, and voiceovers, which can provide valuable context and emotional depth to the colorized videos. Real-time processing can enable the system to colorize videos on-the-fly, allowing users to view the colorized videos as they are being processed. This feature can be useful for applications such as video streaming, online gaming, and virtual reality. Finally, a user interface can be designed to provide users with more control and customization options, such as the ability to adjust color saturation, brightness, contrast, and other image parameters.

Additionally, the black and white to color video system using Faster R-CNN can be extended to handle high-resolution videos, thus providing a more

comprehensive solution for colorizing black and white videos. High-resolution videos require more processing power and memory, which can be achieved by optimizing the code and using more powerful hardware. Moreover, the system can be parallelized to handle multiple frames simultaneously, which can reduce the processing time and improve the overall performance.

With further research and development, the black and white to color video system using Faster R-CNN can be integrated into various industries such as film restoration, historical research, and education, where colorizing black and white videos can provide valuable insights and context. For instance, the system can be used to restore and colorize old films, which can preserve their historical value and enhance their visual quality. Similarly, the system can be used in historical research to provide a more accurate representation of past events and cultures. In education, the system can be used to enhance the learning experience by providing students with a more engaging and immersive way to interact with historical footage and documentaries.

Overall, the black and white to color video system using Faster R-CNN is a promising solution for colorizing black and white videos, and its potential applications are vast and varied. With continued research and development, the system can be improved to provide more accurate and efficient colorization of videos, and it can be integrated into various industries to provide valuable insights and context. Furthermore, incorporating audio input into the system can enhance its usability and provide a more comprehensive solution. By analyzing the audio data in conjunction with the video, the system can detect changes in the audio frequency and use that information to improve the accuracy of the colorization process. For instance, if the audio input indicates that a scene is set during the day, the system can use this information to colorize the scene with brighter and

more vibrant colors. Similarly, if the audio input suggests that a scene is set during the night, the system can colorize the scene with darker and cooler colors. Integrating audio input into the system can provide a more realistic and accurate colorization of black and white videos.

APPENDICES

```
#Importing necessary libraries

import matplotlib.pyplot as plt

import cv2

import numpy as np

import pandas as pd

import os

from tensorflow.keras.applications.inception_resnet_v2

import InceptionResNetV2, decode_predictions, preprocess_input

import warnings

warnings.filterwarnings('ignore')


#Data Pre-Processing

gray_images = np.load('/content/dataset/gray/val.npy')

ab_images = np.load('/content/dataset/abs/val1.npy')
```

```
def image_pipeline(gray_scale_imgs, batch_size = 100, preprocess_f = preprocess_input):
    imgs = np.zeros((batch_size, 224, 224, 3))
    for i in range(0, 3):
        imgs[:batch_size, :, :,i] = gray_scale_imgs[:batch_size]
    return preprocess_f(imgs)


input_images = image_pipeline(gray_images, batch_size = 300)
def col_from_abs(gray_imgs, ab_imgs, n = 10):
    imgs = np.zeros((n, 224, 224, 3))
    imgs[:, :, :, 0] = gray_imgs[0:n:]
    imgs[:, :, :, 1:] = ab_imgs[0:n:]
    imgs = imgs.astype("uint8")
    imgs_ = []
    for i in range(0, n):
        imgs_.append(cv2.cvtColor(imgs[i], cv2.COLOR_LAB2RGB))


    imgs_ = np.array(imgs_)
    return imgs_
```

```
output_images = preprocess_input(col_from_abs(gray_imgs = gray_images, ab_imgs = ab_images, n = 300))


#Model Building & Testing

from tensorflow.keras.applications.inception_resnet_v2 import InceptionResNetV2, decode_predictions, preprocess_input

from tensorflow.keras.models import Model, Sequential

from tensorflow.nn import relu

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.layers import Conv2D, Conv2DTranspose, Reshape

from tensorflow.keras.initializers import RandomUniform

from tensorflow.compat.v1.losses import mean_pairwise_squared_error


rcnn_model = Sequential()

rcnn_model.add(Conv2D(strides = 1, kernel_size = 3, filters = 12, use_bias = True, bias_initializer = RandomUniform(minval=-0.05, maxval=0.05) , padding = "valid", activation = relu))

rcnn_model.add(Conv2D(strides = 1, kernel_size = 3, filters = 12, use_bias = True, bias_initializer = RandomUniform(minval=-0.05, maxval=0.05) , padding = "valid", activation = relu))
```

```
rcnn_model.add(Conv2DTranspose(strides = 1, kernel_size = 3, filters = 12, use_bias = True, bias_initializer = RandomUniform(minval=-0.05, maxval=0.05) , padding = "valid", activation = relu))

rcnn_model.add(Conv2DTranspose(strides = 1, kernel_size = 3, filters = 3, use_bias = True, bias_initializer = RandomUniform(minval=-0.05, maxval=0.05) , padding = "valid", activation = relu))

rcnn_model.compile(optimizer = Adam(epsilon = 1e-8), loss = mean_pairwise_squared_error)

history = rcnn_model.fit(input_images, output_images, epochs=20, batch_size=16)


#main.py

import cv2

from model import rcnn

from util import preprocess_img, postprocess_tens

import streamlit as st


colorizer_rcnn = rcnn(pretrained=True).eval()

use_gpu = False

if(use_gpu):

    colorizer_rcnn.cuda()
```

```
st.title('Video Colorization')


def main():


    up_vdo = st.file_uploader('Upload a Video', type=['mp4'])
    if up_vdo is not None:
        with open('./Videos/input.mp4', 'wb') as f:
            f.write(up_vdo.getvalue())


    if st.button('Colorize'):
        cap = cv2.VideoCapture('./Videos/input.mp4')
        cv2.namedWindow("Resized_Window", cv2.WINDOW_NORMAL)
        cv2.resizeWindow("Resized_Window", 800, 600)
        while True:
            ret, img = cap.read()
            if not ret:
                st.warning('Completed Successfully')
                break
```

```
        (tens_l_orig, tens_l_rs) = preprocess_img(img, HW=(256,256))

        output_img = postprocess_tens(tens_l_orig, colorizer_rcnn(tens_l_rs).cpu())

        cv2.imshow("Resized_Window", output_img)
        if cv2.waitKey(1) & 0xff == ord('q'):
            break
if __name__=='__main__':
    main()

#model.py
from torch.nn import Conv2d, ReLU, Sequential, Softmax, Upsample, BatchNorm2d, ConvTranspose2d
from color_code import ColorCode

class RCNN(ColorCode):
    def __init__(self, norm_layer=BatchNorm2d):
        super(RCNN, self).__init__()
```

model1=[Conv2d(1, 64, kernel_size=3, stride=1, padding=1, bias=True),]

model1+=[ReLU(True),]

model1+=[Conv2d(64, 64, kernel_size=3, stride=2, padding=1, bias=True),]

model1+=[ReLU(True),]

model1+=[norm_layer(64),]

model2=[Conv2d(64, 128, kernel_size=3, stride=1, padding=1, bias=True),]

model2+=[ReLU(True),]

model2+=[Conv2d(128, 128, kernel_size=3, stride=2, padding=1, bias=True),]

model2+=[ReLU(True),]

model2+=[norm_layer(128),]

model3=[Conv2d(128, 256, kernel_size=3, stride=1, padding=1, bias=True),]

model3+=[ReLU(True),]

model3+=[Conv2d(256, 256, kernel_size=3, stride=1, padding=1, bias=True),]

model3+=[ReLU(True),]

```
model3+=[Conv2d(256, 256, kernel_size=3, stride=2, padding=1, bias=True),]
model3+=[ReLU(True),]
model3+=[norm_layer(256),]

model4=[Conv2d(256, 512, kernel_size=3, stride=1, padding=1, bias=True),]
model4+=[ReLU(True),]
model4+=[Conv2d(512, 512, kernel_size=3, stride=1, padding=1, bias=True),]
model4+=[ReLU(True),]
model4+=[Conv2d(512, 512, kernel_size=3, stride=1, padding=1, bias=True),]
model4+=[ReLU(True),]
model4+=[norm_layer(512),]

model5=[Conv2d(512, 512, kernel_size=3, dilation=2, stride=1, padding=2, bias=True),]
model5+=[ReLU(True),]
model5+=[Conv2d(512, 512, kernel_size=3, dilation=2, stride=1, padding=2, bias=True),]
```

```
model5+=[ReLU(True),]
model5+=[Conv2d(512, 512, kernel_size=3, dilation=2, stride=1, padding=2, bias=True),]
model5+=[ReLU(True),]
model5+=[norm_layer(512),]

model6= [Conv2d(512, 512, kernel_size=3, dilation=2, stride=1, padding=2, bias=True),]
model6+=[ReLU(True),]
model6+=[Conv2d(512, 512, kernel_size=3, dilation=2, stride=1, padding=2, bias=True),]
model6+=[ReLU(True),]
model6+=[Conv2d(512, 512, kernel_size=3, dilation=2, stride=1, padding=2, bias=True),]
model6+=[ReLU(True),]
model6+=[norm_layer(512),]

model7= [Conv2d(512, 512, kernel_size=3, stride=1, padding=1, bias=True),]
model7+=[ReLU(True),]
```

model7+=[Conv2d(512, 512, kernel_size=3, stride=1, padding=1, bias=True),]

model7+=[ReLU(True),]

model7+=[Conv2d(512, 512, kernel_size=3, stride=1, padding=1, bias=True),]

model7+=[ReLU(True),]

model7+=[norm_layer(512),]

model8= [ConvTranspose2d(512, 256, kernel_size=4, stride=2, padding=1, bias=True),]

model8+=[ReLU(True),]

model8+=[Conv2d(256, 256, kernel_size=3, stride=1, padding=1, bias=True),]

model8+=[ReLU(True),]

model8+=[Conv2d(256, 256, kernel_size=3, stride=1, padding=1, bias=True),]

model8+=[ReLU(True),]

model8+=[Conv2d(256, 313, kernel_size=1, stride=1, padding=0, bias=True),]

```
        self.model1 = Sequential(*model1)
        self.model2 = Sequential(*model2)
        self.model3 = Sequential(*model3)
        self.model4 = Sequential(*model4)
        self.model5 = Sequential(*model5)
        self.model6 = Sequential(*model6)
        self.model7 = Sequential(*model7)
        self.model8 = Sequential(*model8)

        self.softmax = Softmax(dim=1)
        self.model_out = Conv2d(313, 2, kernel_size=1, padding=0, dilation=1, stride=1, bias=False)
        self.upsample4 = Upsample(scale_factor=4, mode='bilinear')

    def forward(self, input_l):
        conv1_2 = self.model1(self.normalize_l(input_l))
        conv2_2 = self.model2(conv1_2)
        conv3_3 = self.model3(conv2_2)
        conv4_3 = self.model4(conv3_3)
        conv5_3 = self.model5(conv4_3)
```

```
        conv6_3 = self.model6(conv5_3)
        conv7_3 = self.model7(conv6_3)
        conv8_3 = self.model8(conv7_3)
        out_reg = self.model_out(self.softmax(conv8_3))

        return self.unnormalize_ab(self.upsample4(out_reg))

def rcnn(pretrained=True):
        model = RCNN()
        if(pretrained):
                import torch.utils.model_zoo as model_zoo
                model.load_state_dict(model_zoo.load_url('https://colorizers.s3.us-east-2.amazonaws.com/colorization_release_v2-9b330a0b.pth',map_location='cpu',check_hash=True))
        return model

#utils.py
from PIL import Image
import numpy as np
from skimage import color
```

```
import torch
import torch.nn.functional as F

# load image from file path
def load_img(img_path):
    out_np = np.asarray(Image.open(img_path))
    if(out_np.ndim==2):
        out_np = np.tile(out_np[:,:,None],3)
    return out_np

# resize image
def resize_img(img, HW=(256,256), resample=3):
    return np.asarray(Image.fromarray(img).resize((HW[1],HW[0]), resample=resample))

# preprocess image for colorization
def preprocess_img(img_rgb_orig, HW=(256,256), resample=3):
    # return original size L and resized L as torch Tensors
    img_rgb_rs = resize_img(img_rgb_orig, HW=HW, resample=resample)
```

```
    img_lab_orig = color.rgb2lab(img_rgb_orig)
    img_lab_rs = color.rgb2lab(img_rgb_rs)

    img_l_orig = img_lab_orig[:,:,0]
    img_l_rs = img_lab_rs[:,:,0]

    tens_orig_l = torch.Tensor(img_l_orig)[None,None,:,:]
    tens_rs_l = torch.Tensor(img_l_rs)[None,None,:,:]

    return (tens_orig_l, tens_rs_l)

# postprocess colorized image
def postprocess_tens(tens_orig_l, out_ab, mode='bilinear'):
    # tens_orig_l   1 x 1 x H_orig x W_orig
    # out_ab        1 x 2 x H x W

    HW_orig = tens_orig_l.shape[2:]
    HW = out_ab.shape[2:]

    # call resize function if needed
```

```
    if(HW_orig[0]!=HW[0] or HW_orig[1]!=HW[1]):
        out_ab_orig = F.interpolate(out_ab, size=HW_orig, mode='bilinear')
    else:
        out_ab_orig = out_ab

    out_lab_orig = torch.cat((tens_orig_l, out_ab_orig), dim=1)
    return color.lab2rgb(out_lab_orig.data.cpu().numpy()[0,...].transpose((1,2,0)))
#color_code.py
import torch
from torch import nn

class ColorCode(nn.Module):
        def __init__(self):
                super(ColorCode, self).__init__()

                self.l_cent = 50.
                self.l_norm = 100.
                self.ab_norm = 110.

        def normalize_l(self, in_l):
```

```
        return (in_l-self.l_cent)/self.l_norm


def unnormalize_l(self, in_l):
        return in_l*self.l_norm + self.l_cent


def normalize_ab(self, in_ab):
        return in_ab/self.ab_norm


def unnormalize_ab(self, in_ab):
        return in_ab*self.ab_norm
```

REFERENCES

[1] V. Pandit, R. Gulati, C. Singla and S. K. Singh, "DeepCap: A Deep Learning Model to Caption Black and White Images," 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2020

[2] J. Yuan and Z. He, "Adversarial Dual Network Learning With Randomized Image Transform for Restoring Attacked Images," in IEEE Access,2020

[3] D. Goel, S. Jain, D. Kumar Vishwakarma and A. Bansal, "Automatic Image Colorization using U-Net," 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), 2021

[4] I. Žeger, S. Grgic, J. Vuković and G. Šišul, "Grayscale Image Colorization Methods: Overview and Evaluation," in IEEE Access,2021.

[5] Q. Yang, Y. Liu, T. Zhou, Y. Peng and Y. Tang, "3D Convolutional Neural Network for Hyperspectral Image Classification Using Generative Adversarial Network," 2020 13th International Conference on Intelligent Computation Technology and Automation (ICICTA), 2020

[6] Q. Yang and Y. Fan, "An evaluation method of municipal pipeline cleaning effect based on image processing," 2022 3rd International Conference on Information Science, Parallel and Distributed Systems (ISPDS), 2022

[7] A. A. Polat, M. F. Şahın and M. E. Karsligıl, "Video Colorizing with Automatic Reference Image Selection," 2021 29th Signal Processing and Communications Applications Conference (SIU), 2021

[8] X. Wang, C. Huang, F. Gao and H. Cheng, "Pre-processing Transformation for Enhancing the Transferability of Adversarial Examples," 2022 4th International Conference on Communications, Information System and Computer Engineering (CISCE), 2022

[9] F. Woitschek and G. Schneider, "Online Black-Box Confidence Estimation of Deep Neural Networks," 2022 IEEE Intelligent Vehicles Symposium (IV), 2022

[10] K. Nimala, G. Geetha and J. G. Ponsam, "Colorization of Black & White Videos & Photographs," 2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), 2022