AUTOMATED IMAGE COLORIZATION

A PROJECT REPORT

Submitted by

ASWATHY DEVI M L(MCT19MCA001)

to

The APJ Abdul Kalam Technological University

In partial fulfillment of the requirements for the award of

the Degree of

Master of Computer Application

Department of Computer Application

MOHANDAS COLLEGE OF ENGINEERING AND TECHNOLOGY ANAD,NEDUMANGAD,

THIRUVANATHAPURAM-695544

2022

DECLARATION

I undersigned hereby declare that the project report "AUTOMATED IMAGE COLORIZATION", submitted for partial fulfillment of the requirements for the award of degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala is a bona fide work done by me under supervision of Prof. Sreeja K. This submission represents my ideas in my own words and where ideas or words of others have been included; I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data, idea, fact ,or source in my submission .I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

PLACE: Trivandrum **Aswathy Devi M L**

DATE:

MOHANDAS COLLEGE OF ENGINEERING AND TECHNOLOGY ANAD,NEDUMAGAD, THIRUVANATHAPURAM-695544

CERTIFICATE

This is to certify that the report entitled "**AUTOMATED IMAGE COLORIZATION"** submitted by **ASWATHY DEVI M L (**MCT19MCA001**)** to the APJ Abdul Kalam Technological University in partial fulfilment of the requirements for the award of the Degree of **Master of Computer Applications** is a bona fide record of the project work carried out by her under my/our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose

Internal Supervisor(s) Project Coordinator

Head of the Department External Examiner

ACKNOWLEDGMENT

The success and outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have get this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

At the outset, I thank God Almighty for standing by me throughout the project and making it possible for met complete the project with in the stipulated time.

I wish to record my deep sense of gratitude to our Director , Dr. ASHALATHA THAMPURAN, and Principal , Dr. S .SHEELA , for their extensive support and guidance through out the course of my project.

I owe my deep gratitude to my project guide Prof. Subha Ramachandran and HOD Prof. Sreeja K who took keen interest on my project work and guided me all along till the completion of my project work by providing all the necessary information for developing a good system.

I am thankful to and fortunate enough to get constant encouragement, support and guidance from all Teaching staffs of MCA ,which helped main completing the project work successfully. Also, I would like to extend my sincere esteems to all staff in laboratory for their timely support.

Finally, I wish to express my sincere gratitude to all my friends, who directly directly contributed to this venture and support me.

ASWATHY DEVI M L

ABSTRACT

Image colorization task is to recover a plausible color version of the image from its grayscale version. In this work ,we explore machine learning technique that colorize grayscale images in an automated way, and further apply convolutional neural network with pre-trained EfficientNet to produce realistic colorization. The model is trained on places dataset and evaluated in both subjective and quantitative ways. To explore its capability of generalization, we also collect real world black white photos and observe model performance on them.

We present a convolutional-neural-network-based system that faithfully colorizes black and white photographic images without direct human assistance. We explore various network architectures, objectives, color spaces, and problem formulations. The final classification-based model we build generates colorized images that are significantly more aesthetically-pleasing than those created by the baseline regression-based model, demonstrating the viability of our methodology and revealing promising avenues for future work.

Table of Contents

1. Introduction

Automated colorization of black and white images has been subject to much research within the computer vision and machine learning communities. Beyond simply being fascinating from an aesthetics and artificial intelligence perspective, such capability has broad practical applications ranging from video restoration to image enhancement for improved interpretability. Here, we take a statistical-learning-driven approach towards solving this problem.

Image colorization is the process of assigning colors to a grayscale image to make it more aesthetically appealing and perceptually meaningful. These are recognized as sophisticated tasks than often require prior knowledge of image content and manual adjustments to achieve artifact-free quality. Also, since objects can have different colors, there are many possible ways to assign colors to pixels in an image, which means there is no unique solution to this problem. Nowadays, image colorization is usually done by hand in Photoshop. Many institutions use image colorization services for assigning colors to grayscale historic images. There is also for colorization purposes in the documentation image. However, using Photoshop for this purpose requires more energy and time. One solution to this problem is to use machine learning / deep learning techniques. Recently, deep learning has gained increasing attention among researchers in the field of computer vision and image processing. As a typical technique, **convolutional neural network (CNNs)** have been well-studied and successfully applied to several tasks such as image recognition, image reconstruction, image generation, etc

In recent years, CNNs have emerged as the de facto standard for solving image classification problems, achieving error rates lower than 4% in the ImageNet challenge .CNNs owe much of their success to their ability to learn and discern colors, patterns, and shapes within images and associate them with object classes. We believe that these characteristics naturally lend themselves well to colorizing images since object classes, patterns, and shapes generally correlate with color choice.

At first glance, hallucinating their colors seems daunting, since so much of the information (two out of the three dimensions) has been lost. Looking more closely, however, one notices that in many cases, the semantics of the scene and its surface texture provide ample cues for many regions in each image: the grass is typically green, the sky is typically blue, and the ladybug is most definitely red. Of course, these kinds of semantic priors do not work for everything, e.g., the croquet balls on the grass might not, in reality, be red, yellow, and purple (though it’s a pretty good guess). However, for this

project, our goal is not necessarily to recover the actual ground truth color, but rather to produce a plausible colorization that could potentially fool a human observer. Therefore, our task becomes much more achievable: to model enough of the statistical dependencies between the semantics and the textures of grayscale images and their color versions in order to produce visually compelling results.

Given the lightness channel L, our system predicts the corresponding a and b color channels of the image in the CIE Lab color space. To solve this problem, we leverage large-scale data. Predicting color has the nice property that training data is practically free: any color photo can be used as a training example, simply by taking the image's L channel as input and its ab channels as the supervisory signal. Others have noted the easy availability of training data, and previous works have trained convolutional neural networks (CNNs) to predict color on large datasets .However, the results from these previous attempts tend to look desaturated. One explanation is that use loss functions that encourage conservative predictions. These losses are inherited from standard regression problems, where the goal is to minimize Euclidean error between an estimate and the ground truth.

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

We additionally explore colorization as a form of self-supervised representation learning, where raw data is used as its own source of supervision. The idea of learning feature representations in this way goes back at least to autoencoders More recent works have explored feature learning via data imputation, where a held-out subset of the complete data is predicted. Our method follows in this line, and can be termed a cross-channel encoder. We test how well our model performs in generalization tasks, and concurrent self-supervision algorithms, and find that our method performs surprisingly well, achieving state-of-the-art performance on several metrics. Our contributions in this project are in two areas. First, we make progress on the graphics problem of automatic image colorization by (a) designing an appropriate objective function that handles the multimodal uncertainty of the colorization problem and captures a wide diversity of colors, (b) introducing a novel framework for testing colorization algorithms, potentially applicable to other image synthesis tasks, and (c) setting a new high-water mark on the task by training on a million color photos. Secondly, we introduce the colorization task as a competitive and

straightforward method for self-supervised representation learning, achieving state-of-the-art results on several benchmarks.

1.1 Grayscale Images

In digital images, grayscale means that the value of each pixel represents only the intensity information of the light. Such images typically display only the darkest black to the brightest white. In other words, the image contains only black, white, and gray colors, in which gray has multiple levels. In grayscale images, the value of each pixel is related to the number of bits of data used to represent the pixel.

The value of the gray image is usually represented by 8 bits, that is, the combination of eight binary numbers represents the pixel value of a pixel. Therefore, the value range of pixels is 0–255 (0b00000000-0b11111111, "0b" means the following number is in binary format), with a total of 256 grayscale levels. If a 16-bit number is used to represent the pixel value of a pixel, the value range will be 0–65,535, with a total of 65,536 grayscale levels.

1.2 Prior Work on Colorization.

Colorization algorithms mostly differ in the ways they obtain and treat the data for modeling the correspondence between grayscale and color. Non-parametric methods, given an input grayscale image. Parametric methods, on the other hand, learn prediction functions from large datasets of color images at training time, posing the problem as either regression onto continuous color space [1,2,22] or classification of quantized color values.Our method also learns to classify colors, but does so with a larger model, trained on more data, and with several innovations in the loss function and mapping to a final continuous output.

1.3 Coloring Problem

Gray image coloring or "colorization means to give colors to gray images.it becomes a new research point area since it is utilized to increase the visual appeal of images such as old black and white photos.In addition, the information content of some scientific images can be perceptually enhanced with color by exploiting variation in chromaticity as well as luminance.To illustrate the problem of coloring, there are two definitions to describe the gray value as an equation of the three basic components of RGB color model(red,green and blue)

1.Intensity(most common used):

Gray=(Red+Green+Blue)/3

2.Luminance

Gray = 0.299 Red+0.587 Green+0.114 Blue

These two equations are not reversible, that means, any gray value can't be converted back to its red, green, and blue components. Since the possible number of colorsare(256×256×256), there exist 256×256 combinations of completely different colors for each gray value from 0 to 255.

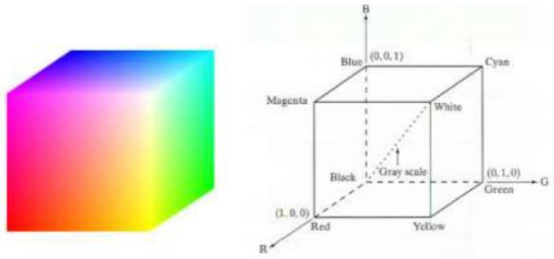
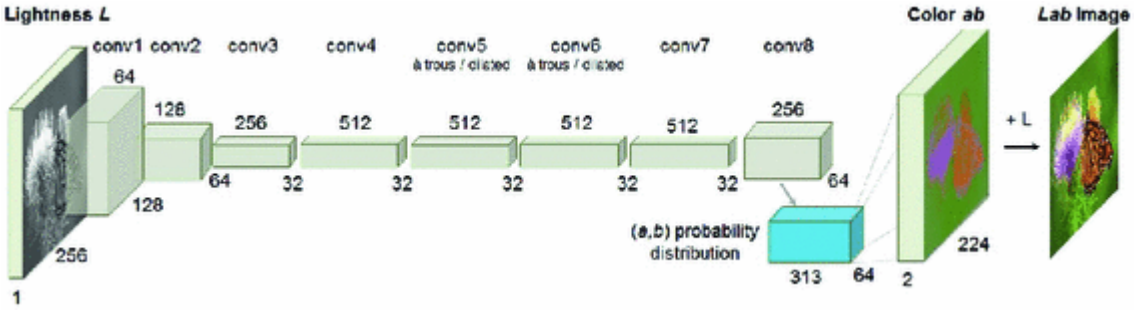


Figure 1. 1: Normalized RGB cube model

Figure 1. 2 Original RGB image (499, 554 bytes)

2.Definition of Colorization Technique



Our network architecture. Each conv layer refers to a block of 2 or 3 repeated conv and ReLU layers, followed by a BatchNorm layer. The net has no pool layers. All changes in resolution are achieved through spatial down sampling or up sampling between conv blocks

2.1 Objective Function

Given an input lightness channel X∈RH×W×1X∈RH×W×1, our objective is to learn a mapping Yˆ=F(X)Y^=F(X) to the two associated color channels Y∈RH×W×2Y∈RH×W×2, where *H*, *W* are image dimensions. (We denote predictions with a ·ˆ·^ symbol and ground truth without.) We perform this task in CIE *Lab* color space. Because distances in this space model perceptual distance, a natural objective function, as used in [1, 2], is the Euclidean loss L2(·,·)L2(·,·) between predicted and ground truth colors:

$$\mathrm{L}_2(\widehat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{2}\sum_{h,w} \| \mathbf{Y}_{h,w} - \widehat{\mathbf{Y}}_{h,w}{}_2^2$$

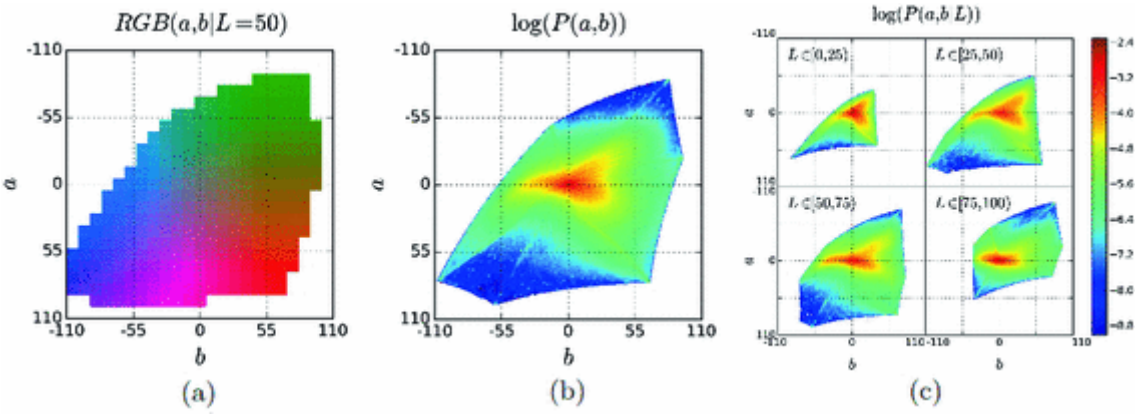
However, this loss is not robust to the inherent ambiguity and multimodal nature of the colorization problem. If an object can take on a set of distinct *ab* values, the optimal solution to the Euclidean loss will be the mean of the set. In color prediction, this averaging effect favorsgrayish, desaturated results. Additionally, if the set of plausible colorizations is non-convex, the solution will in fact be out of the set, giving implausible results.

Instead, we treat the problem as multinomial classification. We quantize the *ab* output space into bins with grid size 10 and keep the Q=313Q=313 values which are in-gamut, For a given input XX, we learn a mapping Zˆ=G(X)Z^=G(X) to a probability distribution over possible colors Zˆ∈[0,1]HxWxQZ^∈[0,1]HxWxQ, where *Q* is the number of quantized *ab* values.

$$L_{cl}(\hat{\mathbf{Z}}, \mathbf{Z}) = -\sum_{h,w} v(\mathbf{Z}_{h,w}) \sum_{q} \mathbf{Z}_{h,w,q} \log(\hat{\mathbf{Z}}_{h,w,q})$$

To compare predicted ZˆZ^ against ground truth, we define function Z=H−1gt(Y)Z=Hgt−1(Y), which converts ground truth color YY to vector ZZ. We then use multinomial cross entropy loss Lcl(·,·)Lcl(·,·), defined as:

where v(·)v(·) is a weighting term that can be used to rebalance the loss based on color-class rarity. Finally, we map probability distribution ZˆZ^ to color values YˆY^ with function Yˆ=H(Zˆ)Y^=H(Z^),



2.2 Class Rebalancing

The distribution of *ab* values in natural images is strongly biased towards values with low *ab* values, due to the appearance of backgrounds such as clouds, pavement, dirt, and walls. gathered from 1.3M training images in ImageNet Observe that the number of pixels in natural images at desaturated values are orders of magnitude higher than for saturated values. Without accounting for this, the loss function is dominated by desaturated *ab* values. We account for the class-imbalance problem by reweighting the loss of each pixel at train time based on the pixel color rarity. This is asymptotically equivalent to the typical approach of resampling the training space .Each pixel is weighed by factor wϵRQwϵRQ, based on its closest *ab* bin.

$$v(\mathbf{Z}_{h,w}) = \mathbf{w}_{q^*}, \text{ where } q^* = \arg\max_q \mathbf{Z}_{h,w,q}$$

$$\mathbf{w} \propto \left((1-\lambda)\widetilde{\mathbf{p}} + \frac{\lambda}{Q}\right)^{-1}, \quad \mathbb{E}[\mathbf{w}] = \sum_q \widetilde{\mathbf{p}}_q \mathbf{w}_q = 1$$

To obtain smoothed empirical distribution pˆ˜ϵΔQp~ϵΔQ, we estimate the empirical probability of colors in the quantized *ab* space pϵΔQpϵΔQ from the full ImageNet training set and smooth the distribution with a Gaussian kernel GσGσ. We then mix the distribution with a uniform distribution with weight λϵ[0,1]λϵ[0,1], take the reciprocal, and normalize so the weighting factor is 1 on expectation. We found that values of λ=12λ=12 and σ=5σ=5 worked well.

3.System Analysis

3.1 Proposed System

Our proposed method is a fully automated process. To implement it, we propose and compare two distinct convolutional neural network architectures trained under various loss functions. We aim to compare each variant based on results obtained as individual images and videos. we investigate colorization on a specific subset of images extracted from a cartoon movie. It is common for cartoons to be drawn without colors, but the presence of color greatly increases the visual appeal of the movie. Traditionally, the colors are added later on in the creative process by scribble based methods, which require considerable effort on the part of the user.

3.2 Existing System

we can classify the existing system as reference image, Scribble-based method, and Convolutional Neural Network.

Based on Reference Image

Figure 3.1 Based on Reference Image

4. Software Requirements

4.1 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding; make it very attractive for Rapid Application Development. Python is simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distribute. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program does not catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

4.2 FEATURES IN PYTHON

Easytocode: Pythonisahigh-levelprogramminglanguage.Pythonisveryeasytolearnthelanguageas compared to other languages like C, C#, JavaScript, Java, etc. It is very easy to code in python language and any body can learn python basics in a few hours or days. It is also a developer-friendly language

Free and Open Source: Python language is freely available at the official website and you can download. Since it is open-source, this means that source code is also available to the public. So you can download it as, use it as well as share it.

Object-Oriented Language: One of the key features of python is Object- Oriented programming. Python supports object-oriented language and concepts of classes, objects encapsulation, etc.

GUI Programming Support: Graphical User interfaces can be made using a module such as PyQt5, PyQt4, Python, or Tk in python. PyQt5 is the most popular option for creating graphical apps with Python.

High-Level Language: Python is a high-level language. When we write programs in python, we do not need member the system architecture ,nor down need to manage the memory.

Extensible feature: Python is an **Extensible** language. We can write us some Python code into C or C++ language and also we can compile that code in C/C++language.

Python is Portable language: Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

Python is Integrated language : Python is also an Integrated language because we can easily integrated python with other languages like C, C++,etc.

Interpreted Language: Python is an Interpreted Language because Python code is executed line by line at a time .like other languages C , C++,Java, etc. the reins need to compile python code this make site a debug our code. The source code of python is converted into an immediate form called **bytecode**.

Large Standard Library: Python has a large standard library, which provides a rich set of module and functions so you do not have to write your own code for everything. There are many libraries present in python for such as regular expressions, unit-testing, web browsers, etc.

Dynamically Typed Language: Python is a dynamically typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of variable.

4.3 FRAMEWORK

Flask is a web application framework written in Python. Armin Ronacher , who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are projects. Web Application Frame work or simply Web Framework represents a collection of libraries and modules that enables a web application developer to write applications without having to bother about low-level details such as protocols, thread management etc.

i. **WSGI:** Web Server Gateway Interface (WSGI) has been adopted as a standard for Python web application development. WSGI is a specification for a universal interface between the webserver and the web applications

ii. **Werkzeug :** It is a WSGI toolkit, which implements requests, response objects, and other utility functions. This enables building a web framework on top of it. The Flask framework uses Werkzeug as one of its bases.

iii. **Jinja2:**Jinja2 is a popular templating engine for Python. A web templating system combines a template with a certain data source to render dynamic web pages.

iv. **Render template:** render template is a Flask function from the flask. templating package. render_template is used to generate output from a template file based on the Jinja2 **engine** that is found in the application's templates folder. Note that render_template is typically imported directly from the flask package instead of from flask. Flask is often referred to as a micro framework. It aims to keep the core of an application simple text extensible. Flask does not have built-in abstraction layer for database handling, no does it have form a validation support.

Open CV

OPENCV was started at Intel in 1999 by **Gary Bradsky**. OPENCV now supports a multitude of algorithms related to Computer Vision and Machine Learning and is expanding day by day. OPENCV supports a wide variety of programming languages such as C++, Python, Java, etc., and is available on different platforms including Windows, Linux, OS X, Android, and iOS. Interfaces for high-speed GPU operations based on CUDA and OPENCV are also under active development. OPENCV-Python is the Python API for OPENCV, combining the best qualities of the OPENCV C++ API and the Python language. OPENCV-Python is a library of Python bindings designed to solve computer vision problems. Compared to languages like C/C++,Python is slower. That said, Python can be easily extended with C/C++, which allows us to write computationally intensive code in C/C++ and create Python wrappers that can be used as Python modules. This gives us two advantages: first, the code is as fast as the original C/C++ code (since it is the actual C++ code working in background) and second, it easier to code in Python than C/C++. OPENCV- Python is a Python wrapper for the original OPENCV C++implementation.

OPENCV-Python makes use of Numpy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OPENCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries

that use Numpy such as SciPy and Matplotlib.

3.4 Important Python Libraries

1.Numpy: NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

2.skimage: Scikit-image, or skimage, is an open source Python package designed for image preprocessing. ***scikit-image*** is an image processing Python package that works with NumPy arrays which is a collection of algorithms for image processing.

5.System Requirements

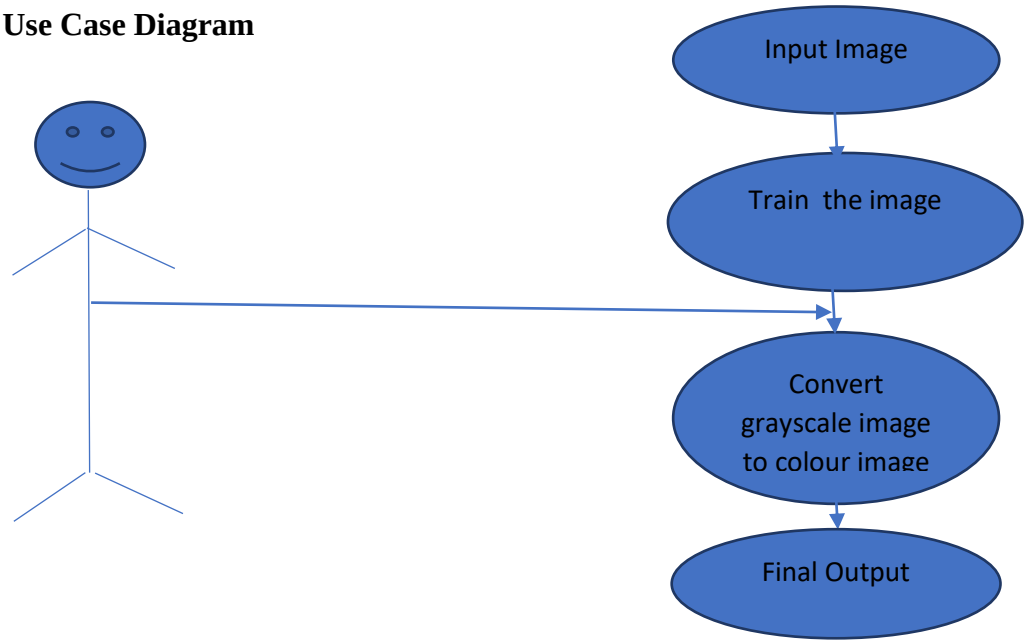
Software	Hardware
Operating System:Windows10	Speed : 2 GHz and above
Script: Python	Ram:8 GB
Interpreter: Python 3.10.0	Processor : AMD Ryzen 3 3250U
IDE: PyCharm	
Framework: Flask	

6.Project Description

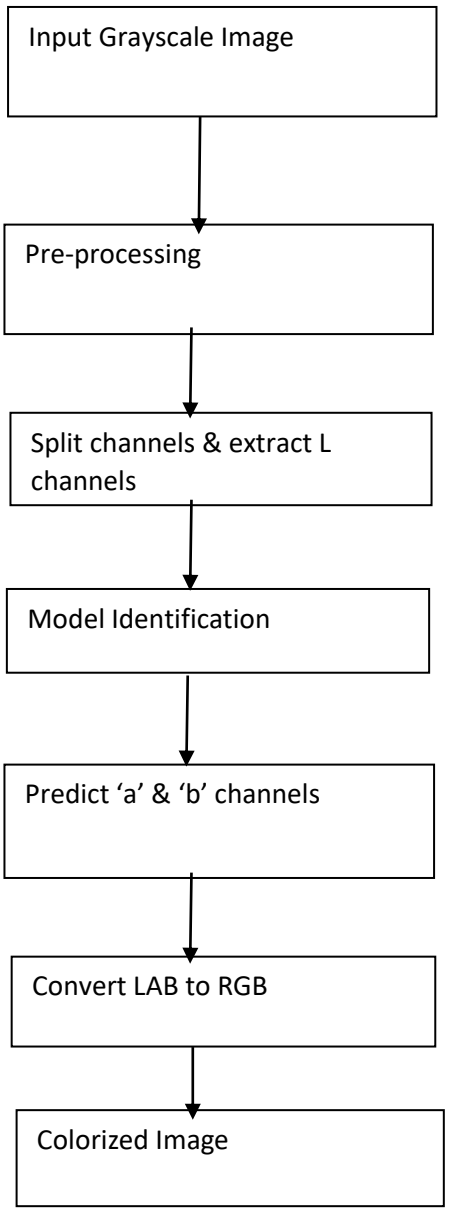
Colorization has been studied by researchers in the field of computer graphics and image processing since Wilson Markle colorized pictures from the Apollo space program in 1970. Adding color to photographs by hand is a tedious process, which requires that the artist segment the image and then assign colors to each segment. The aim of our project is to design an algorithm and interactive system that automatically colorizes a monochrome image with human guidance. The algorithm takes a grayscale image and some color scribbles drawn by a human and produces a fully colorized image that is both consistent with the scribble and the image semantics.

While colorization itself has numerous applications such as image enhancement and film restoration, the unsupervised learning techniques that allowed us to add color to images can also enable automatically learning information such as depth from an RGB image for applications such as autonomous driving. Colorization is a useful research task for improving our understanding of the visual world. We are using images from the ImageNet data set. The ImageNet data can be download from the ImageNet official website. It is a diverse dataset that includes images from dogs and trucks to animated characters. To evaluate our algorithm, we will convert color images to grayscale, add scribbles to the images, and compute the peak signal-to-noise ratio between the predicted image and the ground truth image. This is a common metric used to evaluate the performance of colorization algorithms

6.1 Use Case Diagram



6.1 Process Flow Diagram



7 ALGORITHM

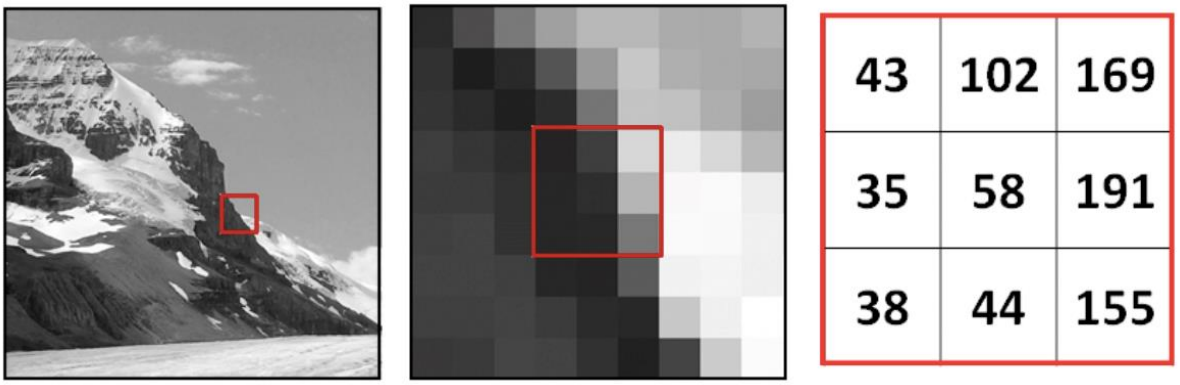
7.1 Convolutional Neural network

A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data.

- A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.
- The pre-processing required in a ConvNet is much lower as compared to other classification algorithms.
- While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.
- The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex.
- Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field.
- A collection of such fields overlap to cover the entire visual area.
- A CNN consists of multiple layers of small computational units that only process portions of the input image in a feed-forward fashion. Each layer is the result of applying various image filters, each of which extracts a certain feature of the input image, to the previous layer. Thus, each layer may contain useful information about the input image at different levels of abstraction.

C O L O R R E P R E S E N T A T I O N

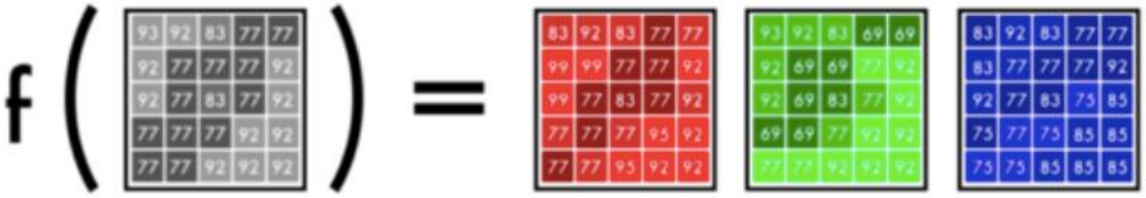
So, how do we render an image, the basics of digital colors, and the main logic for our neural network. We can say that grayscale images can be represented in grids of pixels.



- Each pixel has a value that corresponds to its brightness. The values span from 0–255, from black to white. While, a color image consist of three layers: Red, Green, Blue (RGB) layer.

Let's imagine splitting a green leaf on a white background into three channels. As we know that the color of the leaf is only consist of the green layer. But, the leaf actually present in all three layers. The layes not only determine color, but also brightness.

Just like grayscale images, each layer in a color image has value from 0-255. The value 0 means that it has no color in that layer. If the value is 0 for all color channels, then the image pixel is black. A neural network creates a relationship between an input value and output value. In this project the network needs to find the traits that link grayscale images with colored ones. So, we should search for the features that link a grid od grayscale values to the three color grids.



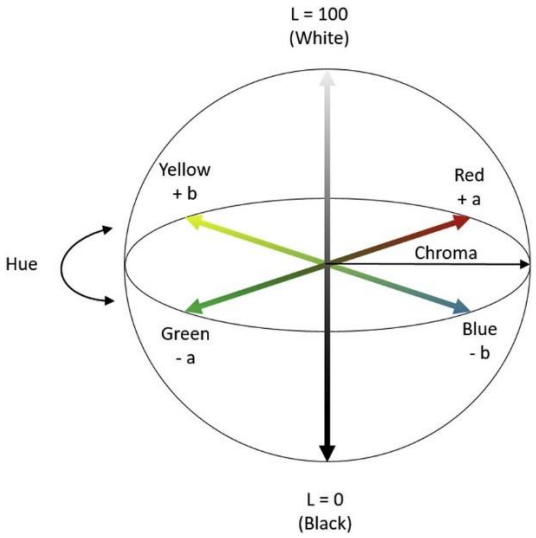
f() is the neural network, [B&W] is our input, and [R],[G],[B] is our output.

DEFINING THE COLORIZATION PROBLEM

Our final output is a colored image. We have a grayscale image for the input and we want to predict two color layers, the ab in Lab. To create the final color image we'll include the L/grayscale image we used for the input. The result will be creating a Lab image.

Convert Image from RGB to LAB color

This part is important because we working on color images anyway or we working on RGB image, meaning every image is very important and every channel is very important and we need to predict the value in every channel. So instead of doing that, for this project the easy way is by converting the RGB to Lab. Before we jump into the code, we should know about the CIELAB color space into this diagram.



The CIELAB, or CIE L* a* b, color system represents quantitative relationship of colors on three axes: L value indicates lightness, and a* and b* are chromaticity coordinates. On the color space diagram, L* is represented on a vertical axis with values from 0 (black) to 100 (white). The a* value indicates red-green component of a color, where +a* (positive) and -a* (negative) indicate red and green values, respectively. The yellow and blue components are represented on the b* axis as +b* (positive) and -b* (negative) values, respectively. At the center of the plane is neutral or achromatic. The distance from the central axis represents the chroma (C), or saturation of the color. The angle on the chromaticity axes represents the hue (ho). The L, a, and b values can be transcribed to dermatological parameters. The L* value correlates with the level of pigmentation of the skin. The a* value correlates with erythema. The b* value correlates with pigmentation and tanning.
Now, let's first define the colorization problem in terms of the CIE Lab color space. Like the RGB color space, it is a 3-channel color space, but unlike the RGB color space, color information is encoded only in the a (green-red component) and b (blue-yellow component) channels. The L (lightness) channel encodes intensity information only.
By iterating on each image, we convert the RGB to Lab. Think of Lab image as a grey image in L channel and all color info stored in A and B channels. The input to the network will be the L channel, so we assign L channel to X vector. And assign A and B to Y.
To change the RGB into Lab image we using `rgb2lab()` function from `skimage` library. After converting the color space using the function `rgb2lab()` we select the grayscale layer with: `[ : , : , 0]`. This is our input for the neural network. `[ : , : , 1: ]` selects the two color layers, green–red and blue–yellow.
The Lab color space has a different range in comparison to RGB. The color spectrum ab in Lab ranges from -128 to 128. By dividing all values in the output layer by 128, we bound the range between -1 and 1. We match it with our neural network, which also returns values between -1 and 1.

7.2 CNN Architecture for Colorization

The architecture proposed by Zhang et al is a VGG-style network with multiple convolutional blocks. Each block has two or three convolutional layers followed by a Rectified Linear Unit (ReLU) and terminating in a Batch Normalization layer. Unlike the VGG net, there are no pooling or fully connected layers.

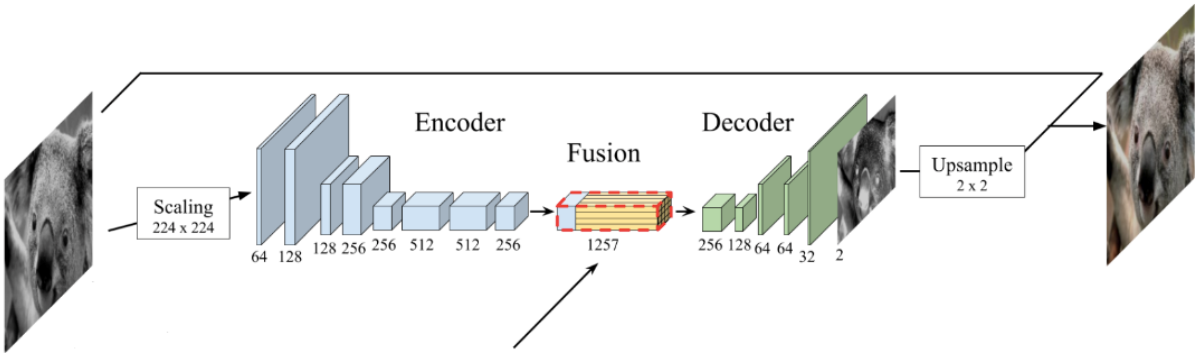


Image above is about CNN architecture for colorization.

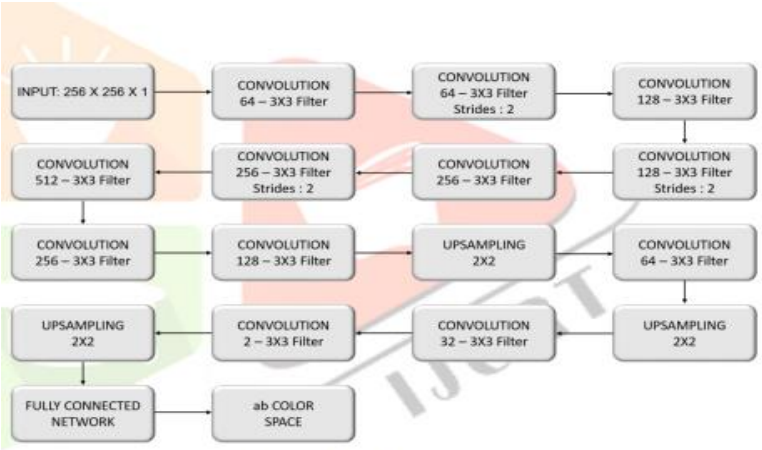


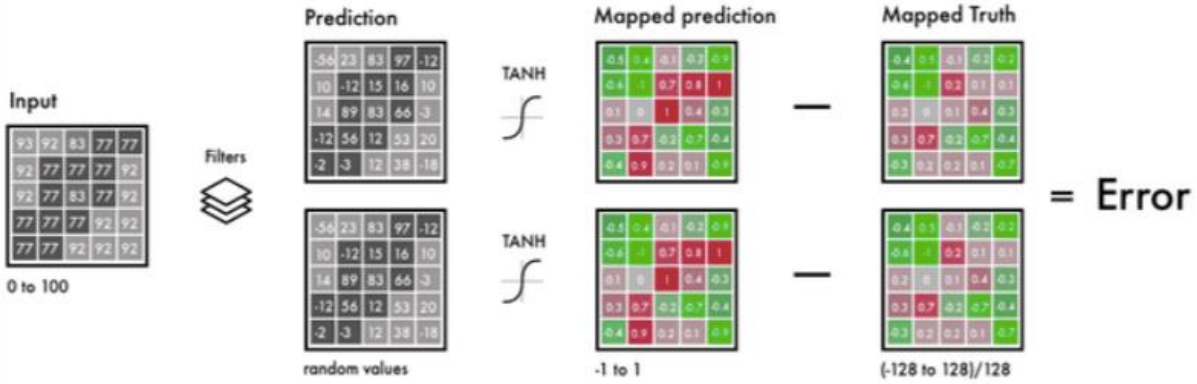
Figure 5.1 Architecture

Encoder

As we can see from the image above, the input image is rescaled to 224x224. The input represented by H x W x 1 (L component) a grayscale images. While the output H/8 x W/8 x 512 feature representation. It uses 8 Convolutional layers with 3x3 kernels that alternate stride 1 and padding to preserve input size, stride 2 to halve the input size. The encoder network, each convolutional layer uses a ReLu activation function.

Decoder

To create convolutional layers to up-sampling. The final output H x W x 2 (ab component) that applies a series of convolutional layers. For the last layer we use tanh instead of Relu. This is because we are colorizing the image in this layer using 2 filters, A and B. A and B values range between -1 and 1 so tanh (or hyperbolic tangent) is used as it also has the range between -1 and 1. Other functions go from 0 to 1.

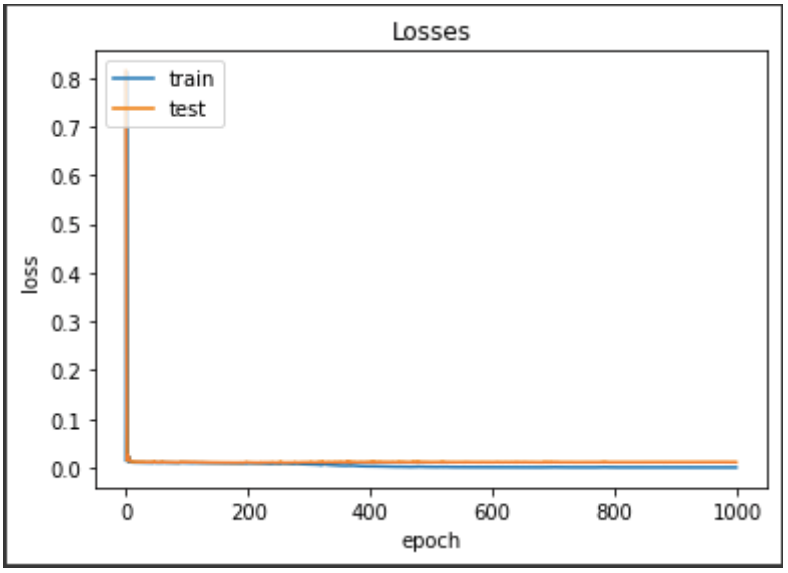


From the left side we have the grayscale input, our filters, and the prediction from our neural network. We map the predicted values and the real values within the same interval. This way, we can compare the values. The interval ranges from -1 to 1. To map the predicted values, we use a tanh activation function. For any value you give the tanh function, it will return -1 to 1. The true color values range between -128 and 128. This is the default interval in the Lab color space. By dividing them by 128, they too fall within the -1 to 1 interval. This "normalization" enables us to compare the error from our prediction. After calculating the final error, the network updates the filters to reduce the total error. The network continues in this loop until the error is as low as possible.

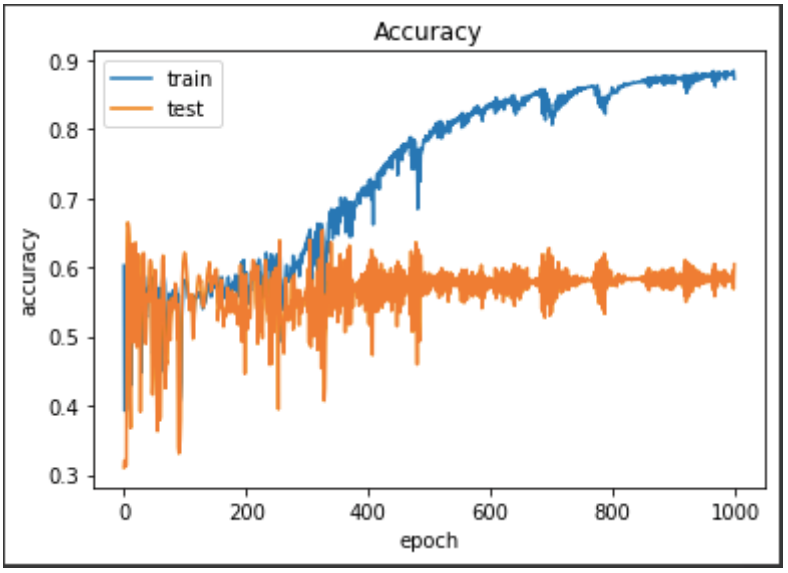
M O D E L F I T T I N G

After we build the model, we can fit the model into the data. We will let the data train with more epochs since we have small numbers of data. We can see the history of our model through this chart.

History of Losses

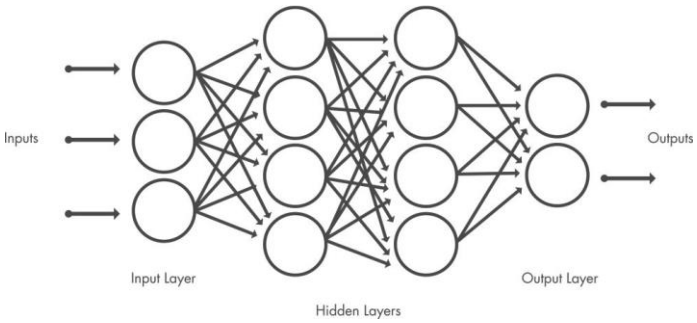


History of Accuracy



Feature Learning, Layers, and Classification

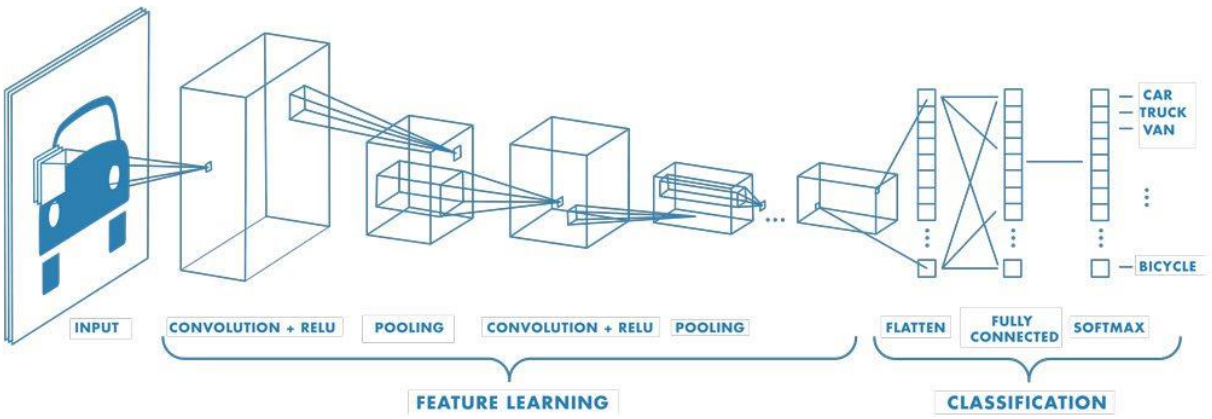
Like other neural networks, a CNN is composed of an input layer, an output layer, and many hidden layers in between.



These layers perform operations that alter the data with the intent of learning features specific to the data. Three of the most common layers are: convolution, activation or ReLU, and pooling.

- **Convolution** puts the input images through a set of convolutional filters, each of which activates certain features from the images.
- **Rectified linear unit (ReLU)** allows for faster and more effective training by mapping negative values to zero and maintaining positive values. This is sometimes referred to as *activation*, because only the activated features are carried forward into the next layer.
- **Pooling** simplifies the output by performing nonlinear downsampling, reducing the number of parameters that the network needs to learn.

These operations are repeated over tens or hundreds of layers, with each layer learning to identify different features.



Example of a network with many convolutional layers. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer.

8.METHODOLOGY

Agile method proposes incremental and iterative approach to software design. The agile process is broken into individual models that designers work. The customer has early and frequent opportunities to look at the product and make decision and changes to the project.

i. Agile model is considered unstructured compared to the waterfall model.
ii. Small projects can be implemented very quickly .For large projects ,it is difficult to estimate the development time.
iii. Error can be fixed in the middle of the project. Documentation attends less priority than software development.
iv. Every iteration has its own testing phase .It allows implementing regression testing. Every time new functions or logical released.
v. Inagiletestingwhenaniterationend,shippablefeaturesoftheproductisdeliveredto the customer. New features are usable right ager shipment. It is useful when you have good contact with customers.
vi. Testers and developers work together.
vii. At the end of every sprint, user acceptance is performed.
viii. It requires close communication with developers and together analyse requirements and planning.
ix. Agileisnotonlyaboutapplyingthesetpracticesindevelopingasoftware.Italso bringsinachangein the team's mind-set, which drives them towards building a better software, working together and eventually landing them a happy customer

8.1 ROLES

SCRUM

SCRUM is an agile development method which concentrates specifically on how to manage tasks within a team-based development environment. Basically, scrum is derived from activity that occurs during a rugby match. Scrum believes in empowering the development team and advocates working in small teams. The outcome is a version of scrum that is unique and specific, in order to have a process that works for us. Scrum is part of the agile movement. Agile is a response to the failure of the dominant software development project management paradigms and borrows many principles from learn manufacturing. The agile manifesto placed a new emphasis on communication and collaboration, functioning software, team self-organization, and the flexibility to adapt to emerging business realities. Scrum's early advocates were inspired by empirical inspect and adapt feedback loops to cope with complexity and risk. Scrum emphasizes decision making from real world results rather than speculation. Time is divided into short work cadences, known as sprints, typically one week or two weeks long. The product is kept in

a potentially shippable state at all times. Scrum is a simple set of roles, responsibilities and meetings that never change. It consists of three roles, and their responsibilities are explained as follows:

SCRUM MASTER

i. Master is responsible for setting up the team , sprint meeting and removes obstacles to progress.
ii. Helps everyone involved understand and embrace the scrum values, principles, and practices.
iii. As a facilitator, scrum master helps the team resolve issues and make improvements its use of scrum.
He is responsible for protecting the team from outside interference.
iv. He takes a leadership role in removing impediments that inhibit team Productivity.
Acts as a coach, providing development process leadership.

PRODUCT OWNER

i. Role played by ASWATHY DEVI M L

ii. The product owner creates product backlog, prioritizes the backlog and is responsible for the delivery of the functionality at each iteration.
iii. He maintains and communicates to all other participants a clear vision of what the scrum team is trying to achieve.
iv. He is the only authority responsible for what will be developed and in what order.
v. One of the most important responsibilities of product owner is to manage product backlog.

SCRUM TEAM

Team manages its own work and organizes the work to complete the sprint or cycle. The team is a self- organizing and cross-functional group of people who do the hands-on work of developing and testing the product. Since the team is responsible for producing the product, it must also have the authority to make decisions about how to perform the work. The team is therefore self- organizing: team members decide how to break work into tasks, and how to allocate tasks to individuals, through out the Sprint.

The team size should be kept in the range from five to nine people, if possible (a large number make communication difficult, while a smaller number leads to low productivity and fragility) Note: A very similar term, “scrum Team,” refers to the team plus the scrum master and product owner.

SPRINT

The sprint backlog is a list of tasks identified by the scrum team to be completed during the scrum sprint. During the sprint planning meeting, the team selects some number of product backlog items, usually in the form of user stories, and identifiers the tasks necessary to complete each user story. Most teams also estimate how many hours each task will take someone on the team to complete. It’s critical that the team selects the items and

size of the sprint backlog. Because they are the people committing to completing the tasks, they must be the people to choose what they are committing to during the scrum sprint. The sprint backlog is commonly maintained as a spreadsheet, but it is also possible to use your defect tracking system or any of a number of software products designed specifically for scrum or agile. The sprint backlog makes visible all the work that the Development Team identifies as necessary to meet the sprint goal. To ensure continuous improvement, includes at least one high priority process improvement identified in the previous Retrospective meeting. The sprint backlog is a plan with enough detail that changes in progress can be understood in the daily scrum. The Development Team modifies the sprint backlog throughout the sprint.

Only the Development Team can change its sprint backlog during a sprint. The sprint backlog is a highly visible, real time picture of the work that the Development Team.

9 PRODUCT BACKLOG

USER STORY ID	USER STORIES	PRIORITY	COMMENTS FROM SCRUM MASTER IF ANY	COMMENTS FROM PRODUCT OWNER IF ANY
1	Problem Identification The problem of colorizing grayscale images	Very high		To Store Grayscale Images
2	Data Collection	Very high	Several Grayscale Images	• PEXELS • Dreams time
3	Creating GUI	Very high	GUI had created within the specified time	Creating an interactive GUI for the proposed system
4	Requirements • Hardware • Software	High		• Hardware ❖ Speed : 2 GHz and ❖ RAM : 8.00 GB ❖ Processor:AMD Ryzen 3 3250U • Software Windows
5	Objectives	Very high		• The main objective ofthis project is to reduce the manualwork. • To convert grayscale images to Color Image
6	Training	Very high	Number of days extended for obtaining accurate model	Convolutional Neural Network
7	Testing	Very high	Completed with in	

			specified time	
8	Technology/Tools Used	Very high		• Flask • OPENCV • HTML • CSS • JavaScript
9	Documentation	High	Completed within allotted time	Documenting the works done

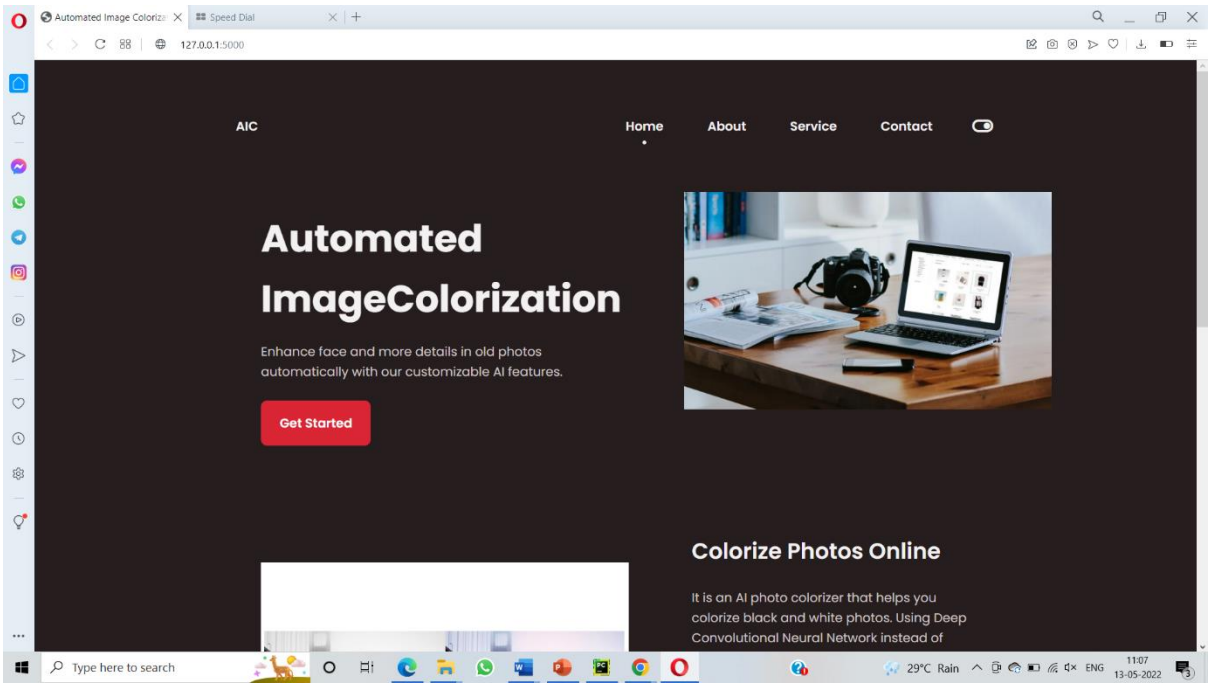
10 Sprint Backlog

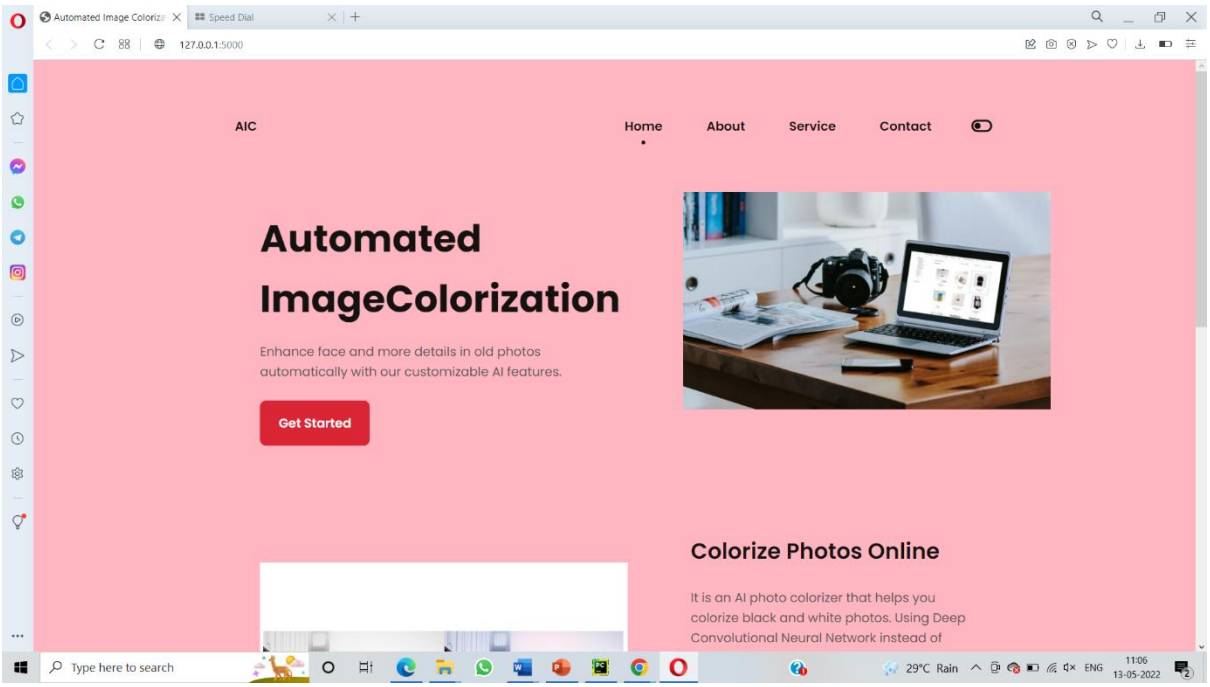
SL.NO	TASK/SPRINT	NOT STARTED	IN PROGRESS	COMPLETED
1	PROBLEM IDENTIFICATION			DONE
2	DATA COLLECTION			DONE
3	GUI			DONE
4	REQUIREMENTS			DONE
5	OBJECTIVE			DONE
6	TRAINING			DONE
7	TESTING			DONE
8	TECHNOLOGY/TOOLS USED			DONE
9	DOCUMENTATION			DONE

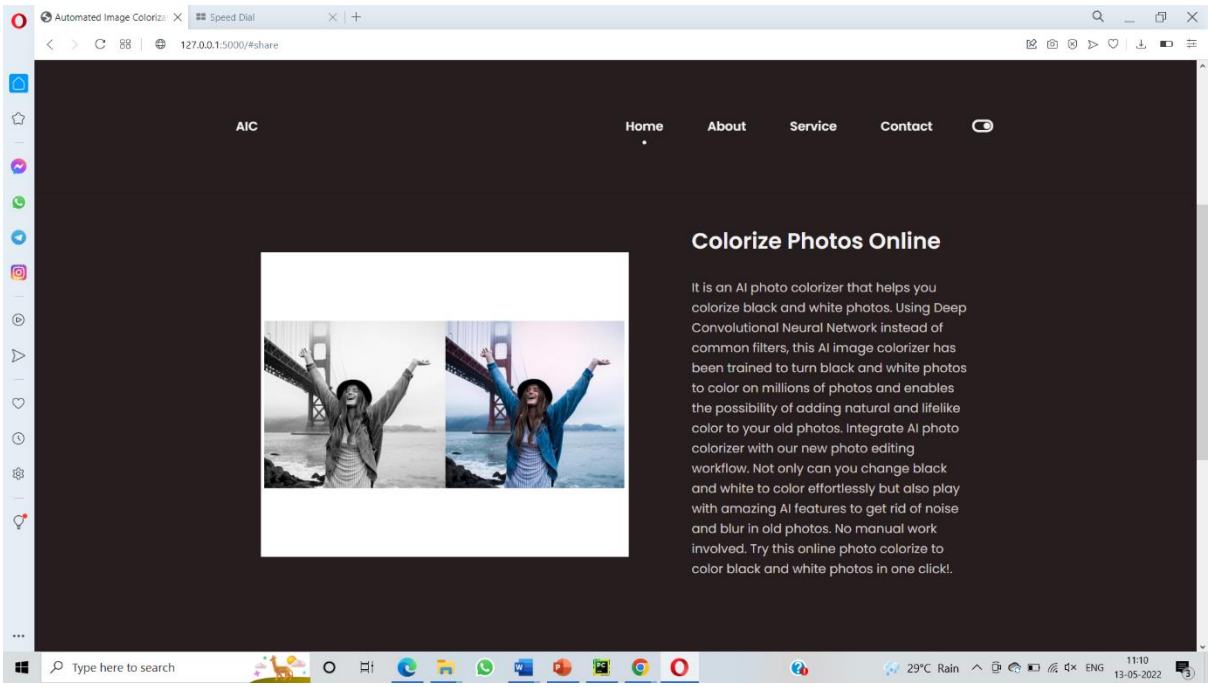
11.Scrum Board

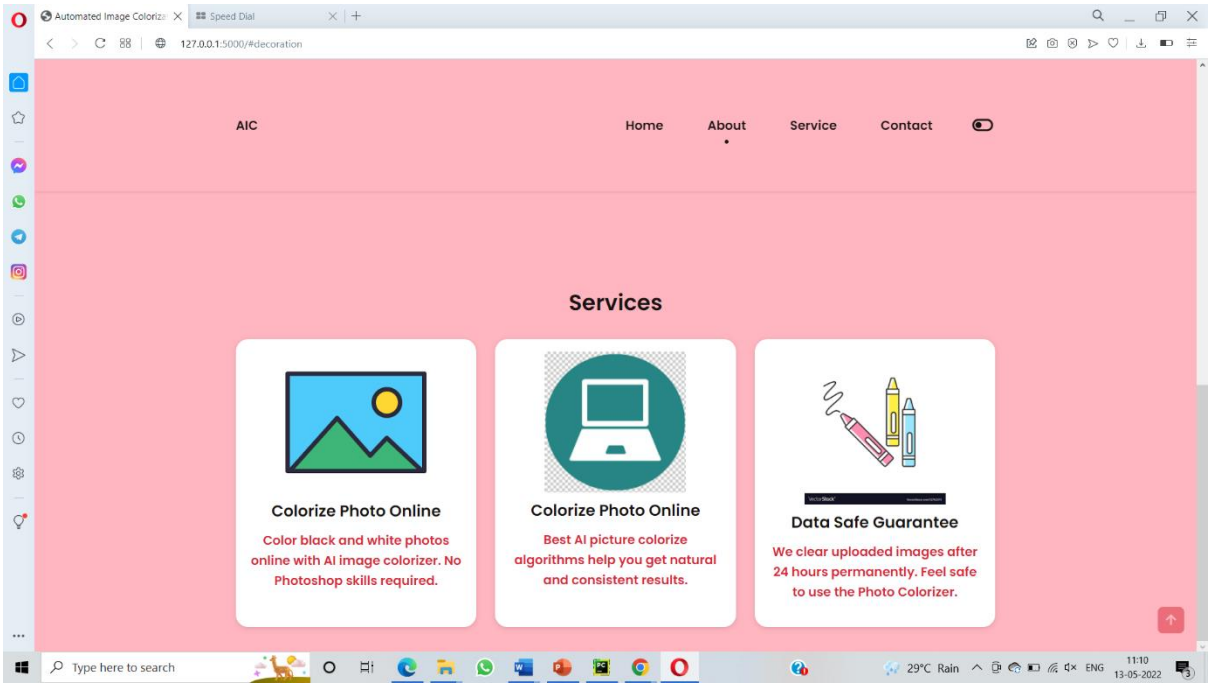
SL. NO	TASK/SPRINT	START	FINISH	DURATION	STATUS
1.	SPRINT-1 Problem identification	08/02/2022	16/02/2022	8 Days	Done
2.	SPRINT-2 Data Collection	21/02/2022	20/03/2022	28Days	Done
3	SPRINT-3 GUI	26/03/2022	04/04/2022	10Days	Done
4	SPRINT-4 Training	27/02/2022	25/03/2022	27 Days	Done
5	SPRINT-5 Testing	26/03/2022	8/04/2022	14 Days	Done
6	SPRINT-6 Documentation	5/05/2022	13/05/2022	3 Days	Done

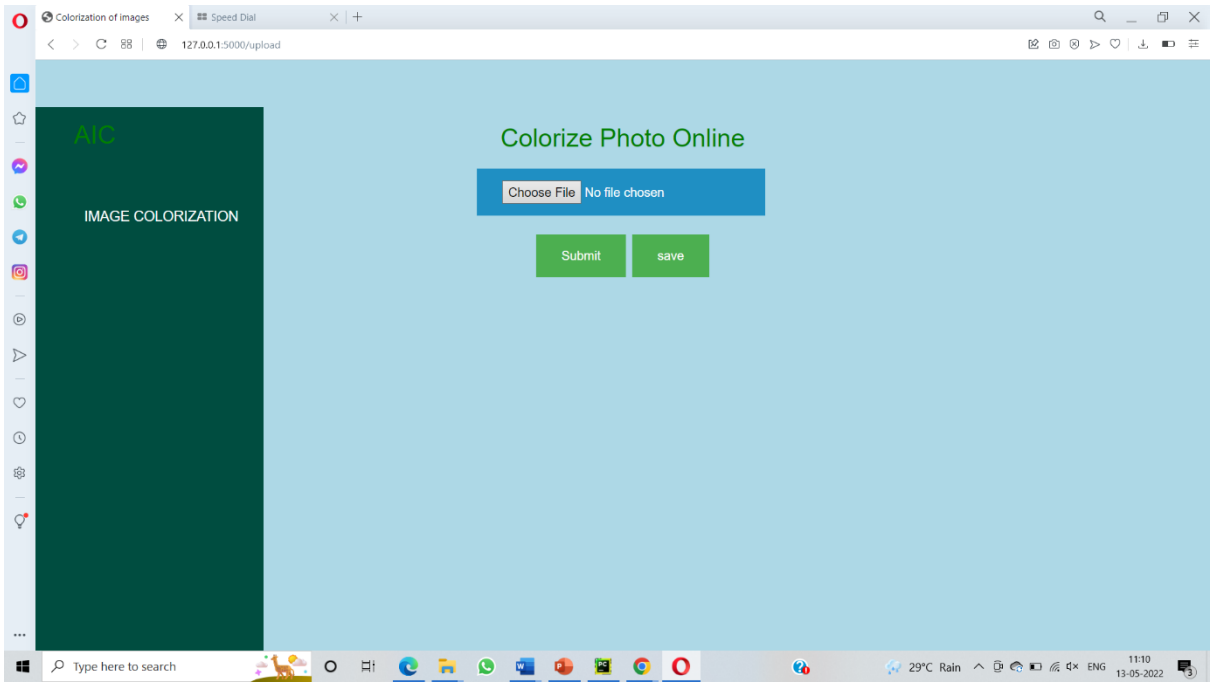
12.Results/Screenshots

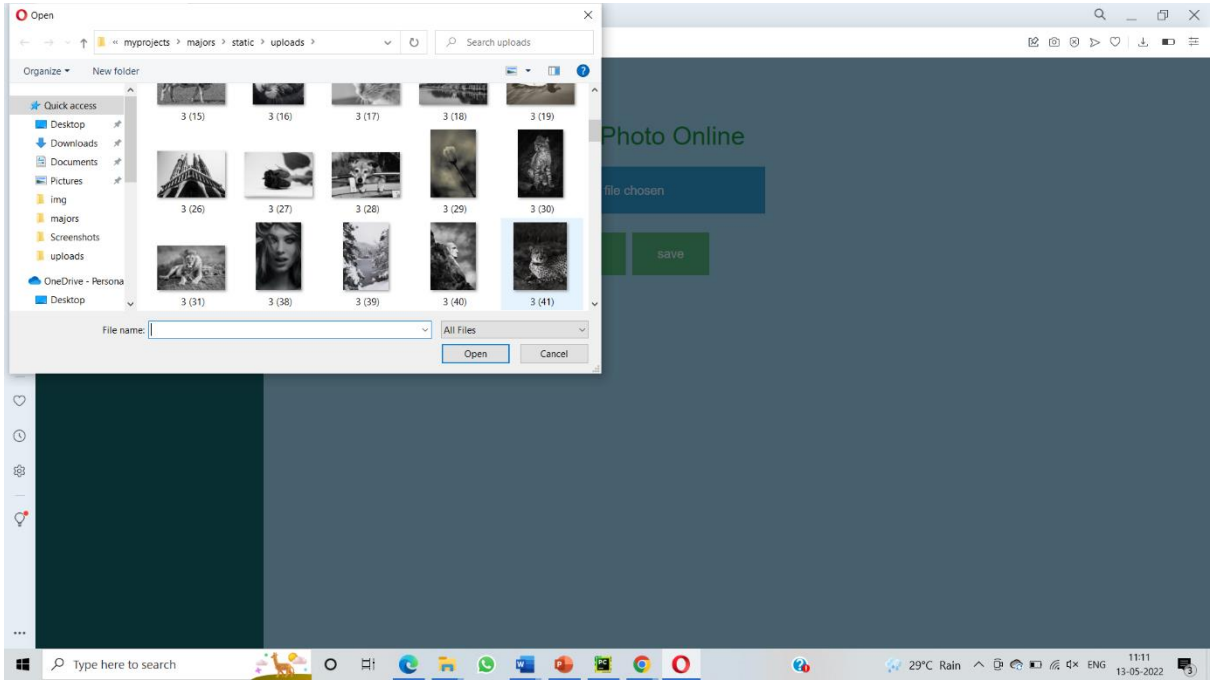


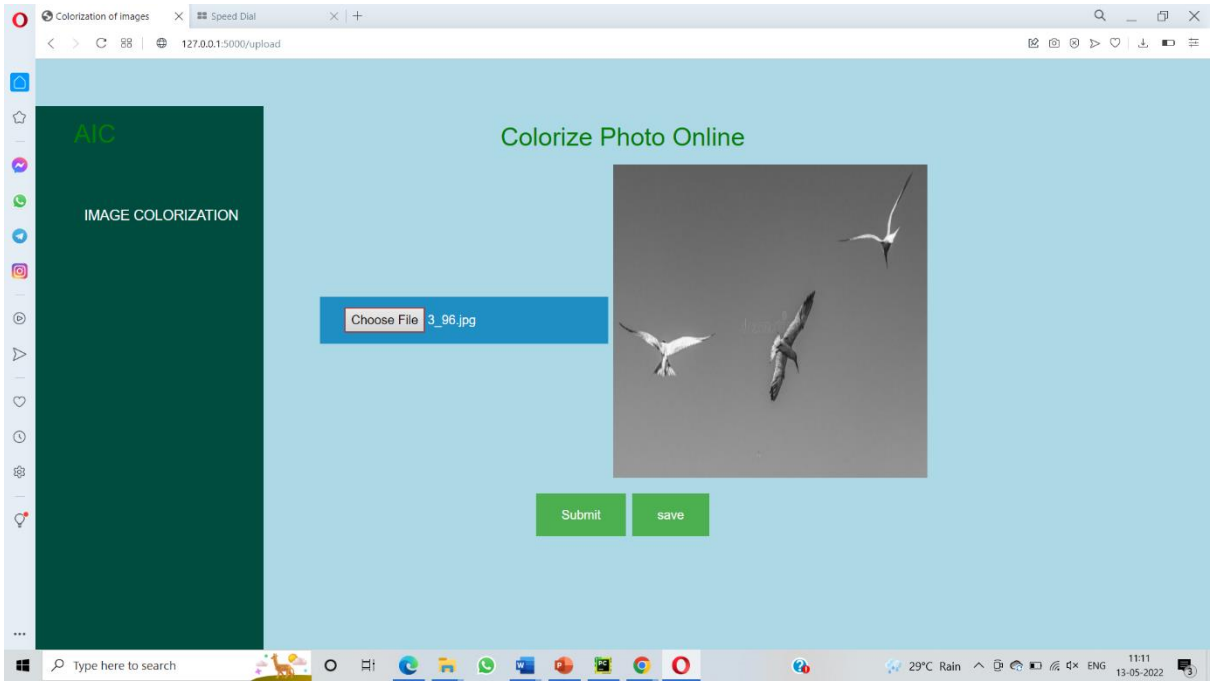


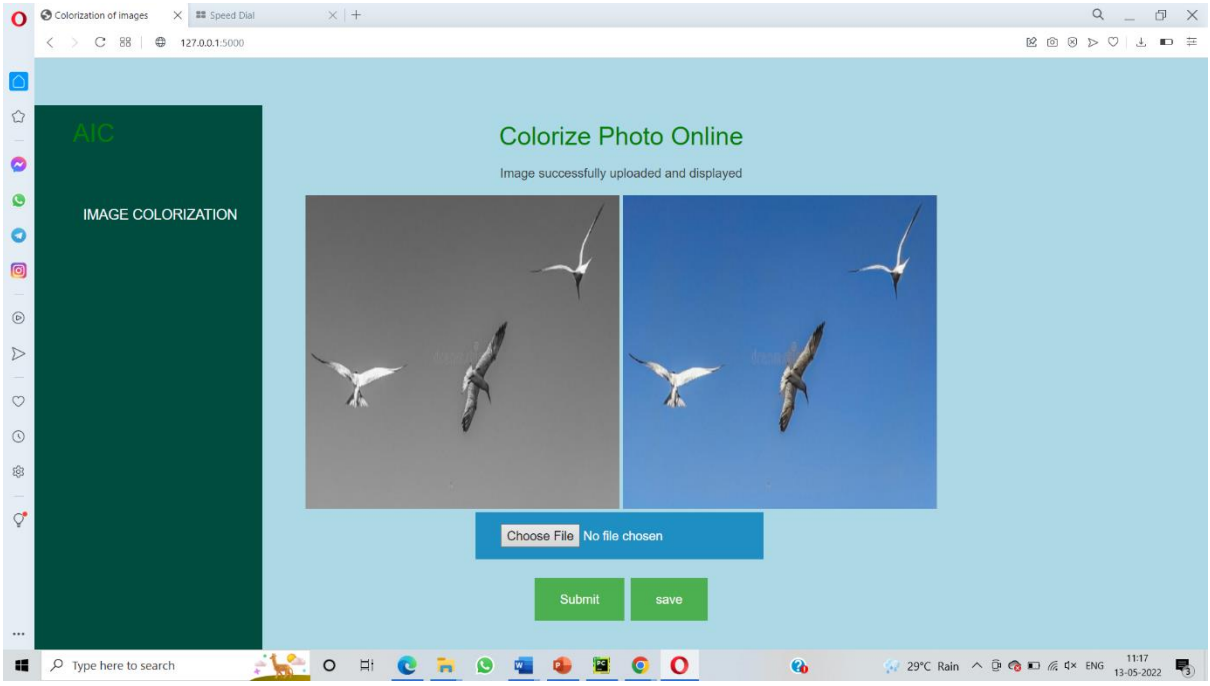

AIC
Home
About
Service
Contact
Colorize Photos Online
It is an AI photo colorizer that helps you colorize black and white photos. Using Deep Convolutional Neural Network instead of common filters, this AI image colorizer has been trained to turn black and white photos to color on millions of photos and enables the possibility of adding natural and lifelike color to your old photos. Integrate AI photo colorizer with our new photo editing workflow. Not only can you change black and white to color effortlessly but also play with amazing AI features to get rid of noise and blur in old photos. No manual work involved. Try this online photo colorize to color black and white photos in one click!.


AIC
Home
About
Service
Contact
Services
Colorize Photo Online
Color black and white photos online with AI image colorizer. No Photoshop skills required.
Colorize Photo Online
Best AI picture colorize algorithms help you get natural and consistent results.
Data Safe Guarantee
We clear uploaded images after 24 hours permanently. Feel safe to use the Photo Colorizer.

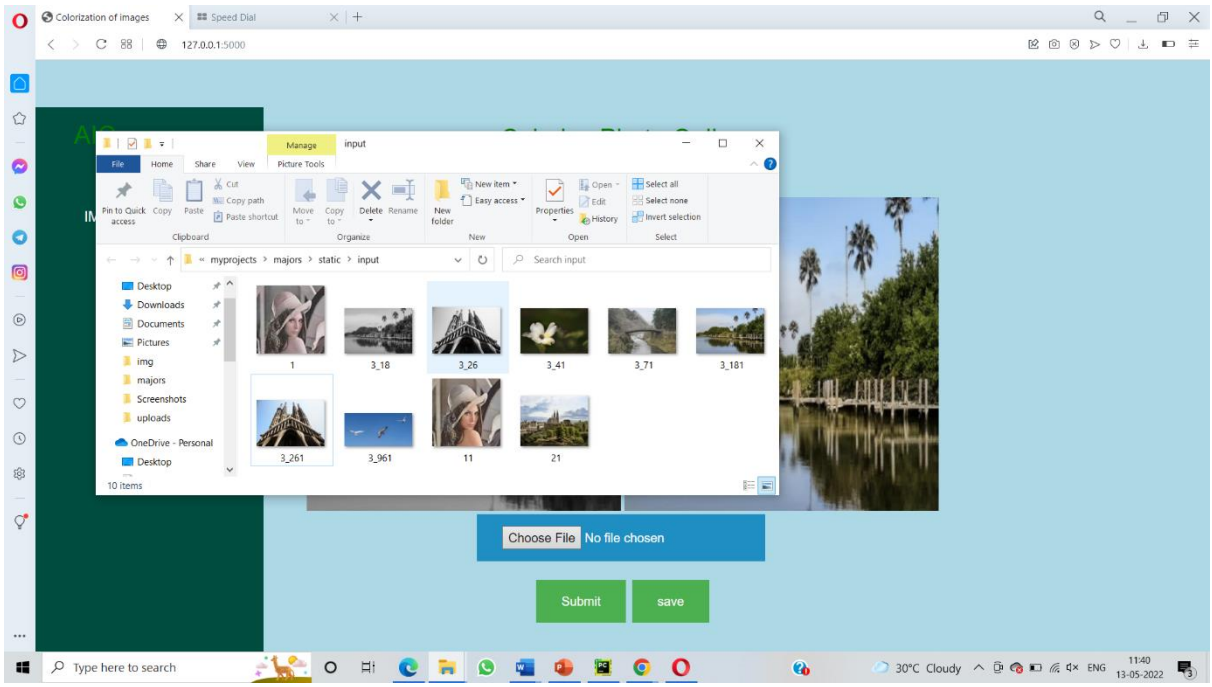

Colorization of images
Speed Dial
127.0.0.1:5000/upload
AIC
IMAGE COLORIZATION
Colorize Photo Online
Choose File
No file chosen
Submit
save
Type here to search
29°C Rain
ENG
11:10
13-05-2022


Open
myprojects
majors
static
uploads
Search uploads
Organize
New folder
Quick access
Desktop
Downloads
Documents
Pictures
img
majors
Screenshots
uploads
OneDrive - Persona
Desktop
3 (15)
3 (16)
3 (17)
3 (18)
3 (19)
3 (26)
3 (27)
3 (28)
3 (29)
3 (30)
3 (31)
3 (38)
3 (39)
3 (40)
3 (41)
File name:
All Files
Open
Cancel
Photo Online
file chosen
save
Type here to search
29°C Rain
ENG
11:11
13-05-2022


Colorization of images
Speed Dial
127.0.0.1:5000/upload
AIC
IMAGE COLORIZATION
Colorize Photo Online
Choose File
3_96.jpg
Submit
save
Type here to search
29°C Rain
ENG
11:11
13-05-2022



Output Folder



13.Future Enhancement

Through our experiments, we have demonstrated the efficacy and potential of using deep convolutional neural networks to colorize black and white images. In particular, we have empirically shown that formulating the task as a classification problem can yield colorized images that are arguably much more aesthetically-pleasing than those generated by a baseline regression-based model, and thus shows much promise for further development. Our work therefore lays a solid foundation for future work. Moving forward, we have identified several avenues for improving our current system. To address the issue of color inconsistency, we can consider incorporating segmentation to enforce uniformity in color within segments. We can also utilize post-processing schemes such as total variation minimization and conditional random fields to achieve a similar end. Finally, redesigning the system around an adversarial network may yield improved results, since instead of focusing on minimizing the cross-entropy loss on a per pixel basis, the system would learn to generate pictures that compare well with real-world images. Based on the quality of results we have produced, the network we have designed and built would be a prime candidate for being the generator in such an adversarial network.

14.Conclusion

While image colorization is a boutique computer graphics task, it is also an instance of a difficult pixel prediction problem in computer vision. Here we have shown that colorization with a deep CNN and a well-chosen objective function can come closer to producing results indistinguishable from real color photos. Our method not only provides a useful graphics output, but can also be viewed as a pretext task for representation learning. Although only trained to color, our network learns a representation that is surprisingly useful for object classification, detection, and segmentation, performing strongly compared to other self-supervised pre-training methods.

15.References

1. Cheng, Z., Yang, Q., Sheng, B.: Deep colorization. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 415–423 (2015)
2. Dahl, R.: Automatic colorization (2016). http://tinyclouds.org/colorize/.
3. Charpiat, G., Hofmann, M., Schölkopf, B.: Automatic image colorization via multimodal predictions. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part III. LNCS, vol. 5304, pp. 126–139. Springer, Heidelberg (2008)
4. Lasagne. https://github.com/Lasagne, 2015.
5. R. Dahl. Automatic colorization. http://tinyclouds.org/colorize/, 2016.
6. X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In International conference on artificial intelligence and statistics, pages 249–256, 2010.
7. B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik. Hyper- ´ columns for object segmentation and fine-grained localization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 447–456, 2015.
8. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015.
9. S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.
10. D. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.
11. A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing .