Dissertation on

"Colorization of Black and White Videos"

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

UE18CS390A – Capstone Project Phase - 1

Submitted by:

Sahith Kurapati	**PES1201800032**
Kalp Aghada	**PES1201800067**
Saksham Gupta	**PES1201800140**
Om Shreenidhi	**PES1201800176**

Under the guidance of

Dr. Jayashree R.
Professor
PES University

January - May 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

‘Colorization of Black and White Videos’

is a bonafide work carried out by

Sahith Kurapati	**PES1201800032**
Kalp Aghada	**PES1201800067**
Saksham Gupta	**PES1201800140**
Om Shreenidhi	**PES1201800176**

in partial fulfilment for the completion of seventh semester Capstone Project Phase - 1 (UE18CS390A) in the Program of Study - Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period Jan. 2021 – May. 2021. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 6th semester academic requirements in respect of project work.

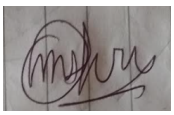
Signature Dr. Jayashree R Professor	Signature Dr. Shylaja S S Chairperson	Signature Dr. B K Keshavan Dean of Faculty

External Viva

Name of the Examiners	**Signature with Date**
1. _____	_____
2. _____	_____

DECLARATION

We hereby declare that the Capstone Project Phase - 1 entitled **“Colorization of Black and White Videos”** has been carried out by us under the guidance of Dr. Jayashree R., Professor and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester January – May 2021. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

PES1201800032	**Sahith Kurapati**	
PES1201800067	**Kalp Aghada**	
PES1201800140	**Saksham Gupta**	
PES1201800176	**Om Shreenidhi**	

ACKNOWLEDGEMENT

I would like to express my gratitude to Prof. Jayashree R., Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE18CS390A - Capstone Project Phase – 1.

I am grateful to the project coordinators for organizing, managing, and helping with the entire process.

I take this opportunity to thank Dr. Shylaja S S, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department. I would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT

This project aimed to tackle the problem of colorizing videos effectively. Previous projects that had been done weren't able to do this efficiently as they always encountered two major problems.

The first problem faced was the inability to produce good clarity images with accurate colors. The colors predicted were very dull and fuzzy. Moreover the pixel values predicted by the models weren't consistent and hence the images were never up to the standard expected.

The second problem faced was while reassembling the frames to make it back into a video the color varied between frames, this led to a flickering effect. This was not ideal as videos are supposed to be seamless and hence this was another problem that other projects were unable to handle.

This project aims to overcome these problems by using an ensemble model and a damping factor. An ensemble model is created to predict a frame more accurately by using many different modeling algorithms. The ensemble model then aggregates the prediction of each model and results in one final prediction. While it may reduce the clarity of one frame, overall it improves the predicted accuracy.

The damping factor is used to solve the flickering effect that had been discussed. Once all the frames have been predicted, while assembling them the damping factor is applied. This ensures that all the frames are balanced accordingly. All of the predicted frames will match each other and hence the pixels in consecutive frames will be a similar value.

Using these methods the project will be able to solve the previous pre-existing problems. The video produced will be of good clarity. It will be clear and the transitions between frames will be seamless to make it truly feel like a video.

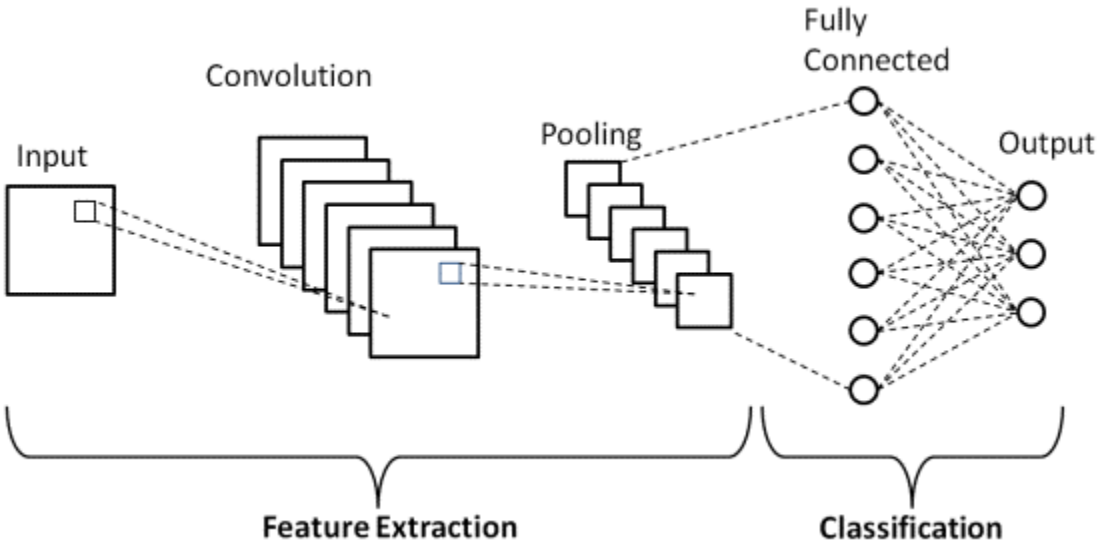
TABLE OF CONTENT

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Although the idea revolving around colorizing black and white or grayscale photos and videos has been around for quite some time, it has only recently been shed some light upon again with the revolutionizing concept of convolutional and pooling layers in neural networks. These kinds of layers are what enable images to be processed at a pixel level in an efficient and smooth manner, only to be delegated off at the end of processing, to artificial neural networks in a way they seem to understand.

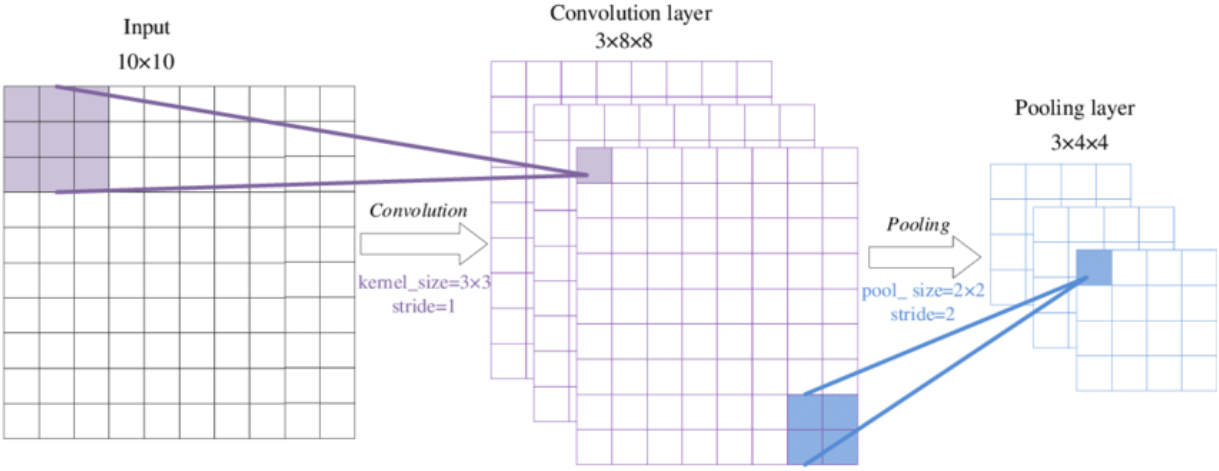


(Figure 1)

Along with the kind of data to be used as input to feed the layers of a neural network, there is one more important aspect of how the concept of color can be induced into the image which consists of only a single channel representing the intensity of grayscale value at each pixel on a scale from 0 - 255. This is the problem which was led to believe years ago that such a solution cannot be made for. The problem at hand had been thought of as simple but once it came into focus, it seemed to be harder than deemed.

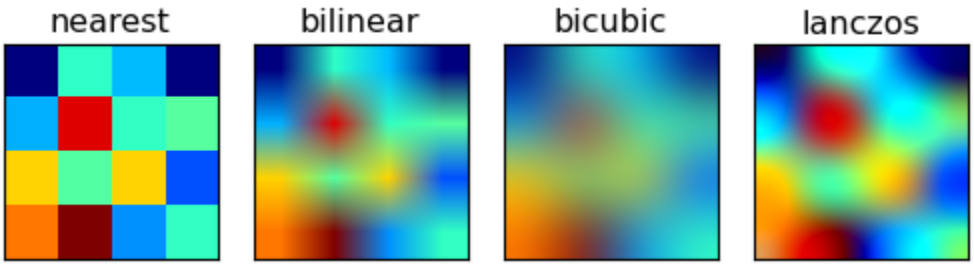
The way this problem can be solved is by using a concept called downscaling and upscaling with a color range induced at the data compression point in the convolutional neural network. The principal concept behind downscaling was to retain the most predominant characteristics in an image while compressing the digital attributes such as height, width, color channels, etc. This can be achieved in a variety of ways, but the most popular of which, use filters with predefined matrix values that are used to perform the action of downscaling only in a particular method of augmenting the image. Some

examples include, making the edges contrasting to the content within the edges, or even to keep similar looking shapes and sizes and discard the rest at a proportionate scale.



(Figure 2)

Once this image is downscaled to some required dimensions and color scale, the next thing to do is to induce color by primarily adding 3 different channels RGB, namely red, green, and blue. Depending on the context to be used and the application requirement, sometimes a 4th channel is also introduced called the alpha channel. This channel is to introduce a layer of opacity or transparency on top of the existing color channels to give a blended feel to modern day pictures. This induction of color has no standard approach, but comes with different formulas designed specifically for each application as required. Many researchers have discovered numerous formulae which work rather differently.



(Figure 3)

The final major component to colorizing images is the upscaling of the image. There are multiple approaches discovered which work in a reverse differential approach of downscaling. The simplest of which is the nearest neighbor interpolation method, where the pixel values are copied directly into more number of slots in the higher resolution image. Another popular method is the bi-linear interpolation, where the corner pixel values are copied directly, but the pixels in between are calculated based on the principle of even distribution.

CHAPTER 2

PROBLEM STATEMENT

Extensive study has been carried out related to the colorization of black and white, grayscale images using a compendium of approaches like regression, and convolutional neural networks.

(Figure 4)

This capstone project in essence, focuses on colorizing grayscale videos.

In particular, it focuses on using and if viable, improving upon existing image colorization approaches to colorize grayscale pictures and extrapolate it to colorizing grayscale videos by de-framing or de-stitching frames of a video, colorizing individual frames in accordance with a damping factor to ensure consistency among said frames, and stitching the frames back together to form a colorized video.

CHAPTER 3

LITERATURE REVIEW

This chapter covers the current literature survey done and required to help shape, inform, and reform our study of the subject on this project.

3.1 Colorful Image Colorization

Published by Richard Zhang, Phillip Isola, and Alexei A. Efros, from University of California, Berkeley, in 2016.

3.1.1 Plausibility Focused Colorizer Model

Given a grayscale photograph as input, this paper attacks the problem of a plausible color version of the photograph. This problem is clearly underconstrained, so previous approaches have either relied on significant user interaction or resulted in desaturated colorizations. This paper proposes a fully automatic approach that produces vibrant and realistic colorizations based on plausibility rather than the ground truth value.

(Figure 5)

3.1.2 Unique Factor

What makes this paper stand out is that this model has undergone a colorization turing test. The colorization turing test had an observer that was presented with 2 sets of images, one which was the output of the colorizer model and the other was the original colorful picture. The observer had to try and guess the real image versus the generated image.

The input data used for training the model has undergone color rebalancing for color diversity in the generated image. This means the input images had relatively similar amounts of each RGB color channel present. This is good for having variation in the model generated images.

3.1.3 Approach and Pipeline

The model works on the basis of spatial upsampling and downsampling. This was done to ensure a completely balanced amount of color during training. This results in a brighter and more vibrant output image from the model. The neural network architecture uses only convolution layers and no pooling layers.

This model got rid of pooling layers to retain more enriched information of colors within the image. This way, a lesser amount of information is lost during the forward propagation of the image data through the neural network.

The model generated output images are tested by using them in a pretrained VGG network based classification system. This was added to the pipeline to ensure a better testing strategy. The model was also tested on various datasets such as ImageNet, SUN, and legacy grayscale images to verify the generational capacity of the network architecture. The main drawback of the model is that it does not try to predict the ground truth value, rather it focuses on predicting a plausible or believable version of the colorful image to the human eye.

3.2 Image Colorization with Deep Convolutional Neural Networks

Published by Jeff Hwang and You Zhou, from Stanford University, in 2016.

3.2.1 Abstract

This paper, as its title suggests is a convolutional neural network approach to solving the problem of colorizing pictures. It claims to be better than other regression based models, in particular, the model created by Ryan Dahl.

There are two processes this system of colorization consists of.

The first process converts the training RGB image into a different color space, extracts the luminance to use it as an input to the model, and uses the remaining color channels as target values.
The second process accepts a black and white or a grayscale video, and generates previously mentioned color channels as target values.

3.2.2 How this Approach Stands Out

This approach aims to perform better than regression by using CNNs. It only learns from what it has seen in the past, that is, it's training data, and returns results without the need for any human intervention.

As extensive experimentation has proven, CNNs do particularly well at tasks pertaining to differentiating and classifying colors, patterns, objects, and shapes into classes. This makes it an apt choice to color images as these factors correlated with color choice in target channels.
It claims to improve upon Ryan Dahl's regression model that solves the same problem. The reason Ryan Dahl's model needs improvement is because the output albeit being fairly accurate, was generally sepia colored/toned.

3.2.3 Pipeline

The system takes a string of 224 x 224 pixel images in RGB space for its training data set. It converts these images into CIE's LUV color space. This color space separates the luminosity and color channels. Color space is a way of representing perceivable colors, for example, the RGB color space, and the CIEXYZ color space.

The black and white or grayscale luminance channel L is fed as input to the model, and the color channels U and V are target values.

For testing, the system reads a 224 x 224 x 1 grayscale image. It then proceeds to generate two arrays of the same dimension, corresponding to U and V channels of CIELUV color space. Then, L, U and V are concatenated to form a predicted image.

Some noteworthy details are that this system used ReLU (rectified linear unit), as its activation function, simply because it's much easier to compute than other alternatives of activation functions. It upscales images from each output layer, allowing global spatial features to be locally concentrated, by taking an element-wise sum with VGG16 layers. It also uses L2 as a loss function.

3.2.4 Results

About 270 images from the ImageNet dataset were used to calculate the performance of this model. The testing showed an accuracy of 29%, which is substantially better than the accuracy of Ryan Dahl's regression model which stood at 17%, giving credit to their initial goals and claims of this project.

(Figure 6)

3.2.5 Drawbacks

There are however, a few drawbacks to this approach.

The most significant of which has to do with the behaviour of ReLU function. In a neural network, the model parameters could happen to be updated in a way such that the function's active region is always in the zero-gradient section. Here, the back-propagated gradients will always be zero.

There is also a noticeable amount of noise in the colorized images, that result in blobs or irregular spheres of concentrated color regions, interspaced throughout the entire image.

(Figure 7)

There are regions where gradients don't appear as smoothly as they ought to, and appear choppy. The system also performs classification without considering the values of the surrounding pixels which might be the cause of these irregularities in colorization. There also exists an involuntary image dimming issue.

3.3 Automatic Colorization of Black and White Images using Deep Learning

This paper was published by Sindhuja Kotala, Srividya Tirumalasetti, Vudaru Nemitha and Swapna Munigala, from Osmania University- Stanley college of Engineering and Technology for Women.

3.3.1 Abstract

This paper aimed to give an overall idea of how to convert grayscale images into colorful images and then further extend this concept into coloring videos. To get a perfect colored image it requires a lot of manual effort which can be very strenuous, moreover the pictures chosen for reference must be hand-picked and chosen with great care so that a more accurate colored picture can be produced. The paper aimed to color images using an unmanned method while retaining the accuracy of the colored picture produced.

The paper used a deep learning model due to the significant advancements that had been made at the time. They employed deep learning approaches to color images and apply the same concepts to color videos. A special importance was given to everything being automated and not requiring any human efforts or interference.

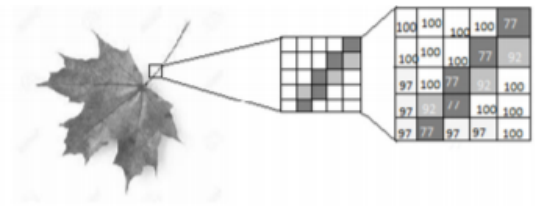
3.3.2 Unique Factors

3.3.2.1 The Colorization Problem

The colorization problem consists of two aspects, the first being the RGB color space and the other being the CIE color space.

In RGB color space, black and white images can be characterized as a grid of pixels, where the value of each pixel ranges from 0 to 255. Here 0 indicates black and 255 indicates white. RGB color images consist of a red layer, a green layer and a blue layer.

Although an image may seem like it may belong in only 1 of these 3 spectrums, it is present in all 3 spectrums as we can clearly see from the image of the leaf below.



(Figure 8, 9)

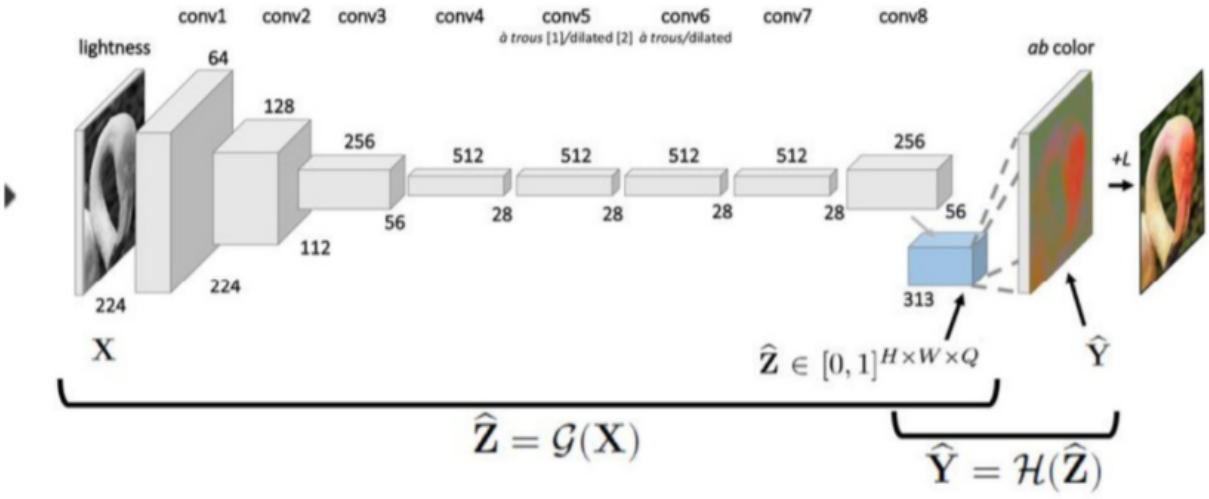
Therefore there becomes a need for a CNN that links the grayscale images to the colored images using these 3 color grids.

Now, coming to the CIE color space of the colorization problem. Like RGB, CIE too has 3 color space channels. However, out of these 3 channels only 2 of them are responsible for storing the encoded color information.

There is also an L (lightness) channel that stores the intensity encoding information. To colorize an image we therefore need 2 values, the first being a bin number which has a range between 0 and 312. The other being the L value which ranges between 0 and 255.

3.3.2.2 CNN Architecture for Colorization

The architecture features a VGG-style network with multiple convolutional blocks as proposed by Zhang. It makes use of Rectified Linear Unit (ReLU) as an activation function and also uses a Batch Normalization Layer. What's even more interesting is that this architecture has no pooling layers. The image passed is rescaled into a 224x224 sized image.



(Figure 10)

3.3.3 Results and Drawbacks

The images produced were dull in color. The dataset used for training was ImageNet which consisted mainly of animals such as cats and dogs. The model therefore was able to predict these images well. Outdoor pictures or sceneries were also accurately predicted by the model.

To make the images more vibrant a multinomial loss function along with color rebalancing was used. Similar results were seen while trying to extend this concept to colorize grayscale videos.

(Figure 11)

3.4 Automated colorization of grayscale images

This paper was published by Kaleem Ahmed Qureshi and Professor C Vanmathi at IRACST in April, 2012.

3.4.1 Goals of the Project

This research paper was released in 2012 and as a result does not use Deep learning methods at its core. The main purpose of this paper was to develop an unassisted method of colorizing images. Up until that point little research had been conducted in the area of automated colorization. This was usually completely assisted or semi-assisted.

This paper uses true image colorization with 256x256x256 image which contains all the colors. The paper focuses on the problem that different shades of grey could map to more than just one RGB color. So identifying different shades of grey would not work. The task would be to color the shades of grey accurately.

3.4.2 Proposed Method

The first step in the process of colorizing the image would be to identify a similar reference image from a database of images using the given input image. The technique used for this step is CBIR. This stands for Content based Image recognition. It retrieves the image from the database based on the color, texture, similarity and shape.

The second step in the process involves coloring the grayscale image using the image retrieved from the CBIR system. The idea is to find a pixel that matches the source color image to the target grayscale image in terms of chromaticity and replicate that onto the grayscale image.

If no exact match can be found in the second step then using the mean and standard deviation of the chromaticity of the closest matching pixels an RGB value is generated that is estimated to be the correct representation of the grayscale pixel.

3.4.3 True color image processing

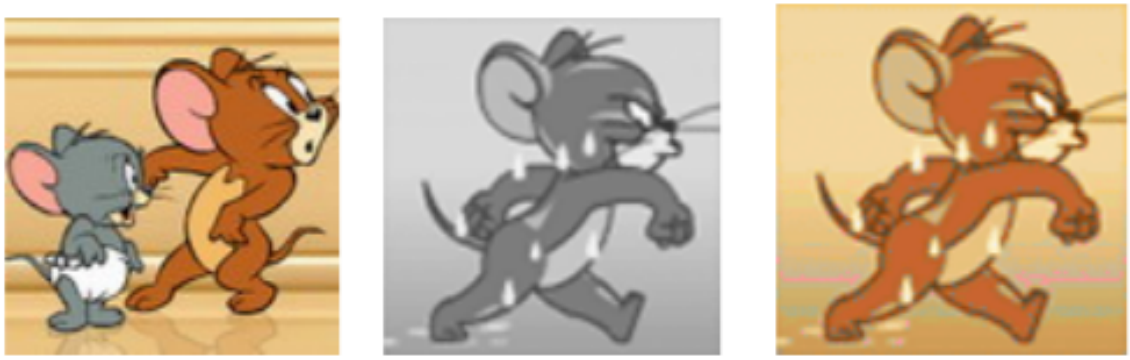
The pixel values that do not match the exact chromatic value need to be processed. The RGB values of the pixel are generated. The reference image is converted into the L$\alpha\beta$ color space. The true color image is searched for matching chromatic value of pixel using greedy search technique. When the pixel is found out the value of the pixel is set as the new color of the pixel.

The Manhattan distance between the chromaticity of the current pixel and the surrounding pixels are found. The pixel with the closest manhattan distance is selected. If the chromatic value is found the

αβ value is added to the Lαβ color space and then transformed back to the RGB color space and the image is colored using the current pixel value.

3.4.4 Conclusions and Drawbacks

In conclusion, the paper did provide an effective method to colorize grayscale images with a high accuracy.



(Figure 12)

However there are several drawbacks.

The drawback of this method was that the result ultimately depended on the algorithm that chose the reference
image. And in choosing the reference image extra delays would be added. Furthermore, the database had to be very extensive. It wouldn't be able to color an image which did not exist in its own database.

This is not very ideal for Video colorization as it is not possible to find a reference image for each frame of the
video. Not only would it be time consuming, it is also highly unlikely such a database would exist. frames in videos tend to have multiple objects in them as a result a database that contains individual images of objects would not be effective in this scenario.

CHAPTER 4

DATA

This chapter serves to describe the data under consideration. Understanding the way all the data work is vital in the process of creating a good solution to the problem at hand.

4.1 Overview

To produce a good output which would comprise a good accuracy of the predicted pixel value along with a clear colorized picture, it is very important to select a dataset which comprises various features that can be extracted and trained on by the neural network. Most videos comprise of outdoor scenery along with people and thus the dataset has been made accordingly.

4.2 Gathering the Data

The data was gathered by scraping Google Images. To ensure good accuracy, the scraped images had to be manually checked to remove any unwanted images. These included images with a very prominent watermark, images that were sketches, etc. Manual effort was also required to crop the images if any border was present around the image to ensure that the model would not take that as a feature.

4.3 Automatically Preprocessing the data

To automate the entire process of getting the data into the required format for the model to train on, a script was made that included the following functionality:-

1. **Resizing the image**

   The requirements for this are the path to the folder containing the dataset and also the path to the folder to which the user requires to put the resized images in. The function makes use of the PIL library which is present in python.

2. **Color to Grayscale images**

   The requirements for this are the path to the folder containing the resized image and also the path to the folder to which the user requires to put the desired grayscale images. This function too makes use of the PIL library which is present in python.


(Figure 13, 14)

The final images are fed into the model. The model splits these images into both training and testing datasets. Using these it tries to produce the output which is a colored image.

CHAPTER 5

SYSTEM REQUIREMENTS SPECIFICATION

5.1 Introduction

This document details the Software Requirements Specification for “Colorization of black and white videos”. The document provides a complete description of the software requirements for building the Colorization of black and white videos application.

The primary aim of this document is to describe in detail the whole system, workflow, and clearly list all its functionalities.

The document provides a detailed description of the functional requirements, non functional requirements as well as other factors necessary to provide a complete and comprehensive description of the requirements for the software.

5.1.1 Project Scope

The main purpose of this project is to be able to convert grayscale videos into a colorized version. One of the most important use cases of this project could be to deploy this system for CCTV footage which is generally recorded in black and white to save on bandwidth and storage.

The project aims to combine multiple colorizer models using different ensemble techniques to get the best precision and clarity of color. The main limitation of the system is that not every type of video can be colorized completely. The model can only learn based on the type of data fed into it.

5.2 Product Perspective

Colorizing grayscale images has been around for quite some time. This project intends to extend that application by colorizing videos, which are essentially a sequence of stitched images

.

5.2.1 Product Features

The main feature of the product is to colorize grayscale videos. The user must first input the grayscale video which he/she wishes to colorize. The user must then wait for the video to finish converting after which point they can either view the video or download it. This entire process will be made very simple and will feature a very simple and minimalistic GUI.

5.2.2 Operating Environment

The training of the machine learning models is tasking and therefore inefficient to run on normal desktops/laptops. For this purpose AWS SageMaker has been used. SageMaker is specifically built for training complex machine learning models with dedicated systems to hasten the process. We have used a t2.large instance which has 2 CPU cores and an 8GB RAM.

For the production ready website we would use either the t2.micro instance from AWS or a heroku web server. Since the model has already been built less computing power would be required.

5.2.3 General Constraints, Assumptions and Dependencies

Constraints:

- Hardware constraint: Since the model works on images and videos, it requires more computing resources.
- The application will be able to process video qualities only upto a certain extent.

Assumptions and Dependencies:

- There must be a sufficient amount of training and testing input data.
- Building the neural network architecture depends on using already existing deep learning frameworks such as Tensorflow, Keras, Caffe, PyTorch, etc.

5.2.4 Risks

The possible risks are, but not limited to -

1. Inadequate physical hardware resources
2. Logistic delays
3. Inadvertent loss of data

5.3 Functional Requirements

- **Sequence of operations**
    1. Scrape relevant images for the model
    2. Format the images as per requirement
    3. Build models to colorize images
    4. Test the models
    5. Fine Tune Parameters for each of the models
    6. Ensemble all the models
    7. Build GUI to allow users to colorize their own videos

- **Automated Data Preprocessing**

  One thousand images of scenery and other types of landscapes were scraped off the internet to be used as the training dataset. After acquiring the images, cleaning the dataset was necessary to maintain the quality of the inputs.

  A python script was used to convert all the images into their grayscale forms. The images were resized to ensure that the size of the input to the machine learning model would always be the same. To get the most optimal width and height for the image we plotted the distributions of height and width for all the images and used the ones of maximum frequency. All the grayscale images of size 256X256 represented the entire training dataset.

- **Training**

  The training phase consists of building multiple neural networks based on the principles used for convolution and pooling filters. The different models can be tuned with different hyperparameters to give a slightly different output quality of the image.

As different models can learn different parts of an image better, all of the outputs of these models must be aggregated with ensemble techniques. The various functions of ensemble models must be tweaked and played around with to get the most optimal final image quality.

- **Consequences of parameters**

  1. Epochs - The number of epochs affect the error rate of the models. An error rate curve measuring epochs and the error will be consulted to arrive at the appropriate number of epochs.
  2. Batch size - The batch size defines the number of samples to work with. The different models implemented here are likely to define their own appropriate batch size.
  3. Learning rate - The amount of change to the model during each step. An error rate curve will be drawn with respect to different learning rates, and the appropriate learning rate will be chosen.

- **Fine Tuning of the Parameters**

  1. Dampening Factor: To colorize a grayscale video we must first split it into its respective frames. We then colorize each of these individual frames. While reassembling the frames to form a video, a flickering effect will be generated as not all pixel values will be identical between the frames. To avoid this flickering effect we introduce a dampening factor. This must be tried and tested to see which value would be best suited to produce a seamless video.
  2. Grid Search Method: Grid search is a tuning technique that attempts to compute the optimum values of hyperparameters. It is an exhaustive search that is performed on the specific parameter values of a model. The model is also known as an estimator. Grid search exercise can save us time, effort and resources.

- **Relationship of outputs to inputs**

  The model is fed grayscale images and the outputs are compared with their corresponding RGB images. Based on the output of the model the correction of the weights are made by backpropagation. The models learn by

identifying textures and patterns in the images given. It understands the "mood" of the image and the corresponding nodes light up and colour the image appropriately.

- **GUI**

  The user interface is a flask based web application where a user can easily upload a grayscale video to be processed and get a colorized version as the output. This web application can simply be deployed on any python application supporting cloud servers. The deep learning ensemble can be made into an API running on top of cloud functions as a microservice.

5.4 External Interface Requirements

5.4.1 User Interfaces

- Required screen formats with GUI standards for styles.
- Screen layout and standard functions (e.g. help).
- Relative timing of inputs and outputs.
- Availability of some form of programmable function key.
- Error messages.

The UI will be a web interface that will follow minimalistic design principles, and will allow for an input of a grayscale video supported by an uploading mechanism and the retrieval of a colored output video supported by either a viewing or a downloading mechanism. An instruction page will be made to facilitate easy use of the tool. Appropriate error messages will be displayed in case instructions are not followed, along with a suggestion of what the problem could be.

5.4.2 Hardware Requirements

5.4.2.1 Developer Requirements

As training is a very intensive process in terms of resources any of the following are feasible to get a good, accurate model:

1. A very high spec CPU with a bare minimum of:
   - Intel Core i7 processor

- 16GB of RAM
- SSD with at least 256 GB of storage

2. If training on GPU using tensorflow-gpu then the minimum requirements for the graphics card are:
    - 8 GB of GPU
    - NVIDIA GPU driver version 410.x
    - CUDA 10 toolkit and driver

3. A cloud service with VM having at least:
    - 2 vCPUs
    - 8 GB of RAM
    - Moderate Network Performance

5.4.2.2. User Requirements

To convert video from grayscale to colored doesn't require much in terms of hardware resources from the user's side. Some of the basic requirements would be:

- A network connection to upload the video
- Space to store the colored video (optional)

5.4.3 Software Requirements

5.4.3.1. Machine Learning model

This is the most essential component of the project. This model contains the trained weights and other parameters to correctly colorize a black and white image.

- Version / Release Number
  Several models will be built during the course of this project and it is likely that tweaks and changes will be made continuously on the model. So several versions of the model will exist.

- Operating Systems

To train the model, an AWS SageMaker notebook instance will be used. It will run on an Ubuntu operating system with 8GB RAM.

- Tools and libraries.
  For building the model several libraries will be used in tandem. These include Tensorflow, Keras, Caffe, sklearn.

5.4.3.2. Web Application

The final product of the project will be demonstrated through a web application. The user will be allowed to upload their black and white video which they wish to colorize. The backend python script would then break the video into frames and pass each frame through the machine learning model built. The final output would then be uploaded back onto the website for the user to download and use.

- Version / Release Number

  The web application to be built will be a simple user friendly interface with minimal features. Therefore the modifications made to the web application will be minute and few.

- Operating Systems

  The web application will be hosted on either a free tier instance on AWS or on heroku. Since the machine learning model has already been trained, the computation power required would be lesser. The operating system used is Ubuntu

- Tools and libraries.

  The web server app is built using Flask.

5.5 Non-Functional Requirements

5.5.1 Performance Requirement

The tool shall support all grayscale videos of an appropriate input format specified in the instructions page. In addition, the tool shall be capable of delivering an output video with

a reasonable efficiency and delay. In case the video format is not supported, or an error from the user's end has occurred, an error message will be displayed mentioning the same. The tool will be available to use whenever the server running the backend and frontend is up. The tool shall also be available to use locally if a copy of the frontend and more importantly the backend is on the system in question.

5.5.2 Security Requirements

The security concern of this project lies in safekeeping the videos uploaded by users onto the web application. Care has to be taken to ensure that the video uploaded by the user is not used maliciously. Furthermore the videos must not be stored on the web-server system after the user has finished making use of the web application.

CHAPTER 6

SYSTEM DESIGN

Given the nature of the problem, in order to come up with the best possible solution our approach is elaborate in the design of the system. Special care has been taken to avoid any possible drawbacks that could cascade to the later phases of implementation. In designing the system the needs and abilities of both the user and developer have been taken into consideration. The user is presented with a user friendly UI for convenience while hiding the essential but complicated back-end that carries the functionality of the system.

The design consists of the following components:-

1. **Landing Page**

   The user is directed to the landing page on opening the website. For the convenience of the user the steps to colorize their input video is visible. The user is given the option to upload their video for colorization. A status bar is used to indicate the extent of completion of the upload. This is done to keep the user up to date about the status of their video.

2. **Back-end Receiving Video**

   Once the grayscale video is successfully uploaded by the user, a python file is executed. The python file preprocesses the grayscale video. This includes dividing the video into frames as well as resizing the frames into fixed sizes of 256x256 images. This is done so as to have a sense of familiarity in the input layers of the machine learning models. The frames are then individually passed onto the machine learning models.

3. **Machine Learning Models**

   An ensemble of machine learning models is used to ultimately colorize the image. Each model colorizes each frame individually. All four models being completely different will bring their own properties into each frame.

   The final image is produced by taking an aggregate of all the images produced by the four models. The pixels of each image are read and the resulting weighted average of the pixel at that position is considered the final pixel value.

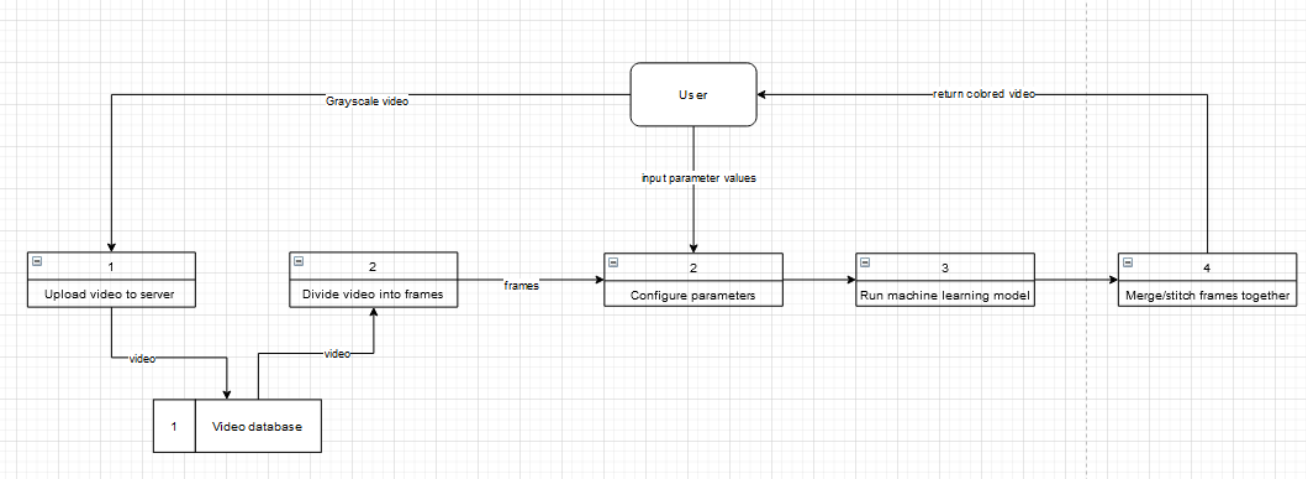
   Furthermore the resulting pixel value is compared with the predicted pixel value at the same location for the previous frame. An appropriate damping factor is applied on the current predicted pixel value to prevent flickering.

4. **Merging and Uploading**

The colorized frames are then merged together to produce the final colorized video. This video is then uploaded for the user to download and assess for themselves. For the convenience of the user, they are given the option to upload another video for colorization.

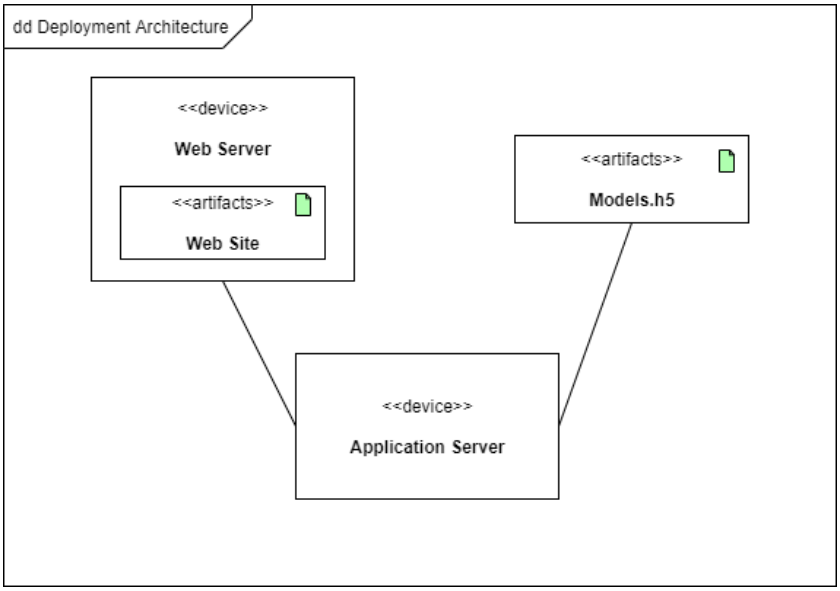
These four steps comprise the main design components of the system. Each step was carefully analyzed and only then acted on to ensure the best possible end product.

A High level diagram is shown below:-



(Figure 15)

The packaging and deployment architecture is as follows



(Figure 16)

In terms of design , non functional requirements also had to be taken into consideration. These were as follows:-

1. Performance Requirements
    a. Reasonable Efficiency
    b. Reasonable Delay
    c. User friendly UI
        i.
2. Security Requirements
    a. User data must not be stored
    b. System must only color the data

CHAPTER 7

IMPLEMENTATION AND PSEUDOCODE

```
#Encoder

Input(shape=(256, 256, 1,))
Conv2D(64, (3,3), activation='relu', padding='same', strides=2)
Conv2D(128, (3,3), activation='relu', padding='same')
Conv2D(128, (3,3), activation='relu', padding='same', strides=2)
Conv2D(256, (3,3), activation='relu', padding='same')
Conv2D(256, (3,3), activation='relu', padding='same', strides=2)
Conv2D(512, (3,3), activation='relu', padding='same')
Conv2D(512, (3,3), activation='relu', padding='same')
Conv2D(256, (3,3), activation='relu', padding='same')

#Fusion

RepeatVector(32 * 32)
Reshape(([32, 32, 1000]))
concatenate([encoder_output, fusion_output], axis=3)
Conv2D(256, (1, 1), activation='relu', padding='same')

#Decoder

Conv2D(128, (3,3), activation='relu', padding='same'
UpSampling2D((2, 2))
Conv2D(64, (3,3), activation='relu', padding='same')
UpSampling2D((2, 2))
Conv2D(32, (3,3), activation='relu', padding='same')
Conv2D(16, (3,3), activation='relu', padding='same')
Conv2D(2, (3, 3), activation='tanh', padding='same')
UpSampling2D((2, 2))
```

This is the basic blueprint of the models that will be built barring changes to the parameters.

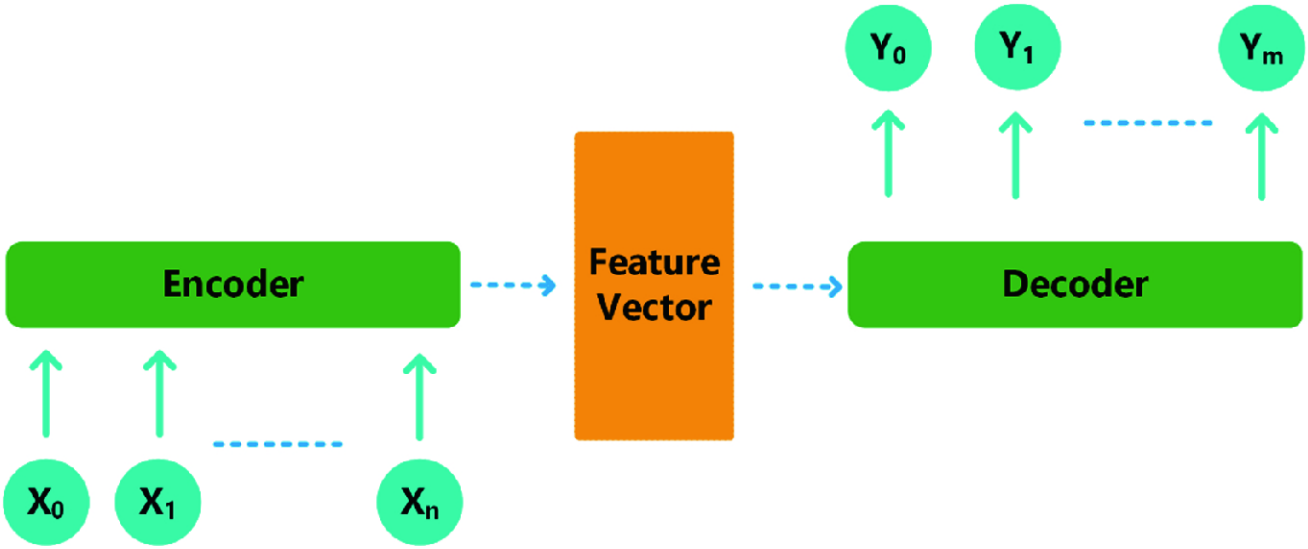
Firstly we use an Encoder, which is a Convolutional Neural network that takes the input, and outputs a feature map/vector/tensor. These feature vectors hold the information, the features, that represents the input.

The decoder is again a network (usually the same network structure as encoder but in opposite orientation) that takes the feature vector from the encoder, and gives the best closest match to the actual input or intended output.

The encoders and decoders are trained together. The loss function is based on how close the generated output is to the required output.

The entire architecture has a structure like that of a sand glass. The encoder encodes the image into a vector and the decoder decodes the vector to try and reconstruct the same image with color.

The above code is merely the first of many versions of several possible models. The model can be improved by changing the hyper parameters and toying with them until the desired output is reached.



(Figure 17)

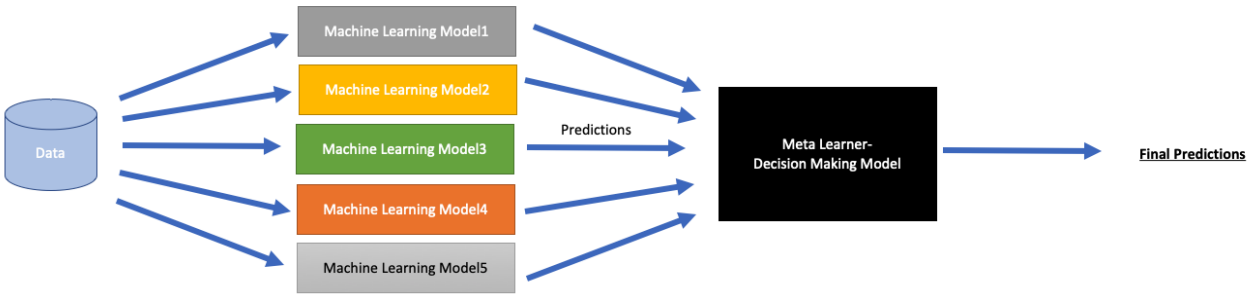
CHAPTER 8

CONCLUSION OF CAPSTONE PROJECT PHASE-1

In conclusion, phase 1 of this project completes the gathering of data, building each of the deep learning neural network models, and finally combining all of the models together as an ensemble so as to get the best possible colorization scheme out of them.

The data required for training and testing purposes are images that are scraped from google images pertaining to a particular search term keywords. This raw data collected is filtered to eliminate the images with little to no color, images with too much noise and images with watermarks covering a large enough portion of the image required for training.

This data is used as input data for training and testing based on either 80/20 split, 70/30 split, or 60/40 split depending on the model requirements. Multiple image colorizer models are architected and implemented with the use of external frameworks in python such as tensorflow or keras.



(Figure 18)

All of the individual image colorizer models are combined into a single ensembled meta learner model with tuned hyperparameters for a more accurate representation of the color channels on the images. The models are combined with an equal aggregation method or distributed aggregation method which favours the more performing models with a relatively higher influence rate.

CHAPTER 9

PLAN OF WORK FOR CAPSTONE PROJECT PHASE-2

The in-depth literature survey and experimentation carried out shows that building a convolutional neural network based solution to this problem is the most viable out of all available alternatives. As far as the plan of work is concerned, the timeline initially mentioned is going to be followed.

Phase-2 will be focused on creating an ensemble of the grayscale picture colorization models, by combining a number of models. The models will be selected for an ensemble based on their performance as measured alone on image colorization, based parameters to calculate accuracy and consistency.

The models would then be combined by deciding upon mixture parameters that determine the amount of influence or impact each model would have. These parameters would be ascertained by measuring accuracy of output through experimentation, and arriving at suitable values for the same.

The models would then be integrated along with a damping factor to retain memory of previous image frames in order to have a smooth, consistent, clear and clutter free output. This can arguably prove to be a challenging task, given the amount of research available to draw from.

A GUI would also be made in order to make this tool accessible to the end user, where a user can upload a video, decide upon configuration parameters, and receive a colored video as output that can be downloaded locally. The GUI would be web-based, minimalistic, and easy to use.

REFERENCE / BIBLIOGRAPHY

[1] Richard Zhang, Phillip Isola, Alexei A. Efros, "Colorful Image Colorization", University of California, Berkeley, 2016

[2] Jeff Hwang, You Zhou, "Image Colorization with Deep Convolutional Neural Networks"

[3] Sindhuja Kotala, Srividya Tirumalasetti, Vudaru Nemitha, Swapna Munigala, "Automatic Colorization of Black and White Images using Deep Learning", 2019

[4] Kaleem Ahmed Qureshi, Prof. C. Vanmathi, "Automated colorization of grayscale images", 2012

APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS

Batches	Number of samples to process before updating parameters.
CIBR	Content based image retrieval.
CIE	International Commission on Illumination.
CNN	Convolutional Neural Network.
Convolution	Mathematical combination of two functions to produce a third function.
Deep Learning	Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial intelligence.
Downscale	Reduce in size (of the original image).
Grayscale	Colored in shades of gray.
GUI	A GUI (graphical user interface) is a system of interactive visual components for computer software.
ImageNet	A database of images arranged in a word based tree hierarchy on the internet.
LUV color space	CIE's LUV color space, where L is the luminance channel, and U and V are color channels.
Multinomial	Consisting of several terms
Noise	In the context of images, a random variation in either color or brightness or luminance in an that stands out from the image.
Keras	Keras is a free and open-source software library for deep learning models.

Parameters	Various numerical values that determine the efficiency and accuracy of the model.
Rectified Linear Unit (ReLU)	Activation function that is defined as the positive part of the argument.
SUN Dataset	Scene Understanding Dataset.
TensorFlow	TensorFlow is a free and open-source software library for machine learning.
Upscale	Increase in size (of the original image).
VGG Net	Visual Geometry Group convolutional neural network architecture.
XYZ color space	CIE's XYZ color space, where Y is luminance, Z is quasi equivalent to blue (of the RGB color space) and X is a mixture of three non-negative RGB values.