**KATHMANDU UNIVERSITY**

**SCHOOL OF ENGINEERING**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**Dhulikhel, Kavre**



**A Mini-Project Report**

**On**

**“Lyrics Generator ”**

**Submitted By:**

Aayush Malla (29)

**Submitted To:**

Santosh khanal

Assistant Professor

Department of Computer Science and Engineering

**Date of Submission**: 11th March, 2020

**Table of Contents**

1. **Introduction……………………………………………………………………………....1**

**1.1 Problem Definition…………………………………………………………………...1**

**1.2 Motivations for Doing the Project…………………………………………………..2**

**1.3.Objectives of the Project work……………………………………………………....2**

1. **Related works……………………………………………………………………………3**
2. **Datasets…………………………………………………………………………………...4**
3. **Methods and Algorithms Used………………………………………………………....5**
4. **Experiments……………………………………………………………...........................9**
5. **Evaluation of Results…………………………………………………………………..11**
6. **Discussion on Results…………………………………………………………………..12**
7. **Contributions of each group member………………………………………………....13**
8. **Code……………………...……………………………………………………………....13**
9. **Conclusion and Future Extensions to the Project……………………………………14**

**References……………………………………………………………………………….15**

# **Introduction**

Colorization of grayscale images is a simple task for the human imagination. A human need only recall that sky is blue and grass is green; for many objects, the mind is free to hallucinate several plausible colors. The high-level comprehension required for this process is precisely why the development of fully automatic colorization algorithms remains a challenge. Colorization is thus intriguing beyond its immediate practical utility in graphics applications. Automatic colorization serves as a proxy measure for visual understanding.

**1.1 Problem Definition**

Given a grayscale photograph as shown in figure 1 (left) as input, this project deals with the problem of giving each pixel an appropriate color to create a plausible colored version of the photograph as shown in figure 1 (right). In a grayscale image, all the pixel colors are only determined by one channel as opposed to an actual colored image whose pixel colors are determined by three channels namely Lightness (L), Red-Green Spectrum (a) and Blue-Yellow Spectrum (b). With the decrease in color channels in grayscale images, a lot of valuable information for analyzing the image is lost. We intend to regain the lost information in a grayscale image by the process of image colorization. The goal of image colorization is to add colors to a gray image such that the colorized image is perceptually meaningful and visually appealing.



**Fig 1.1: Image Colorization**

**1.2 Motivations for Doing the Project**

The pictures and video tapes from the past were created in grayscale before the first colored image was taken in 1861. It was 1960 when colored photographs actually started to dominate the world, and it has been that way ever since. Some people might think that black and white photos are antiquated and have no use in the modern world. However, there is a reason why balck and white photography is still popular.

The photographs and videotapes created in the past have great importance to us. They resemble our history, how we human developed from the industrial age to the modern era.

Like the world famous actor Charlie Chaplin’s movies were taped in Black and White. Everyone around the globe enjoys watching the legend’s beautiful act. Who would not like the humor that Charlie Chaplin has. The photos captured from the world war and many historic events are black and white. So to colorize these photos and videos would bring them back to life in a sense. Restoring the color to grayscale provides a research area in many fields. Like when a researcher looks down a microscope all he/she can see is a grayscale image of the cellular level organisms. What if we can colorize them and see them as colorful beings that would certainly be interesting. What if surveillance cameras capturing grayscale footage at night could be somehow colorized that would create much clearer footage much more recognizable. Due to these facts we came up with an idea why not contribute some more in this work.

**1.3 Objectives**

The objectives of our project are:

* To design a model that can accurately predict the color of a pixel based on its Lightness value
* To colorize black and white photos taken back decades ago and bring then to colorful reality

# **Related Works**

**i. Scribble-based Colorization**

Levin et al. proposed an effective approach that requires the user to provide colorful scribbles on the grayscale target image. The color information on the scribbles are then propagated to the rest of the target image using least-square optimization. Huang et al. developed an adaptive edge detection algorithm to reduce the color bleeding artifact around the region boundaries. Yatziv et al. colorized the pixels using a weighted combination of user scribbles. Qu et al. and Luan et al. utilized the texture feature to reduce the amount of required scribbles.

**ii. Example-based Colorization**

Unlike scribble-based colorization methods, the example-based methods transfer the color information from a reference image to the target grayscale image. The example-based colorization methods can be further separated into two categories according to the source of reference images:

(1) Colorization using user-supplied example(s). This type of method requires the user to provide a suitable reference image. Inspired by image analogies and the color transfer technology, Welsh et al. employ the pixel intensity and neighborhood statistics to find a similar pixel in the reference image and then transfer the color of the matched pixel to the target pixel. It is later improved by taking into account the texture feature. Charpiat et al. proposed a global optimization algorithm to colorize a pixel. Gupta et al. developed an colorization method based on superpixel to improve the spatial coherence. These methods share the limitation that the colorization quality relies heavily on example image(s) provided by the user. However, there is not a standard criteria on the example image(s) and thus finding a suitable reference image is a difficult task.

(2) Colorization using web-supplied example(s). To release the users’ burden of finding a suitable image, Liu et al. and Chia et al. utilize the massive image data on the Internet. Liu et al. compute an intrinsic image using a set of similar reference images collected from the Internet. This method is robust to illumination difference between the target and reference images, but it requires the images to contain identical object(s)/scene(s) for precise per-pixel registration between the reference images and the target grayscale image. It is unable to colorize the dynamic factors (e.g. person, car) among the reference and target images, since these factors are excluded during the computation of the intrinsic image. As a result, it is limited to static scenes and the objects/scenes with a rigid shape (e.g. famous landmarks). Chia et al. proposed an image filter framework to distill suitable reference images from the collected Internet images. It requires the user to provide a semantic text label to search for suitable reference image on the Internet and human-segmentation cues for the foreground objects. In contrast to the previous colorization methods, the proposed method is fully automatic by utilizing a large set of reference images from different categories (e.g., animal, outdoor, indoor) with various objects (e.g., tree, person, panda, car etc.).

1. **Datasets**

The current version of the software has been implemented on a dataset consisting of images of faces of people of different ages taken in different background settings. Each of the images is of 256 X 256pixels. For training, we have used a dataset of 840 images.

From the dataset, we have split the data into training part, testing part and validation part with the ratio 70:20:10. We have taken about 70% of the images from the dataset to train the model and after training, 10% has been used for validation. For testing purposes, 20% of images which are all in black and white format have been taken. Each of the testing data is also of 256 pixels by 256 pixels and consists of faces of people. After splitting the data into two parts, each of the training data is passed into the image generator function to enrich our dataset. The image generator takes the given images and performs different actions on it such as zooming to about 20%, rotating the image to a maximum of 20 degrees on either side and flipping the image which is called Image Augmentation. By performing these actions, multiple versions of the same image are created which will help enhance the learning process.



**Fig 3.1: Pie Chart Showing No. of Images Used for Specific Purpose**

# **4. Methods and Algorithms Used**

**4.1 Data Preprocessing**

After increasing our dataset with the given modifications, we convert the color scheme of the images to a CIE format.The original RGB format is not that suitable for training the model. The training is done based on the weights of the grayscale values from the original image. If we take the RGB values then, the weights for a red pixel, greenpixel and blue pixel will appear the same and recoloring to the exact color will be only correct in 1 out of 3 pixels. But, in CIE format the colors are distinguished in 3values i.e L for Lightness, a for Red-Green spectrum and b for Blue-Yellow spectrum.

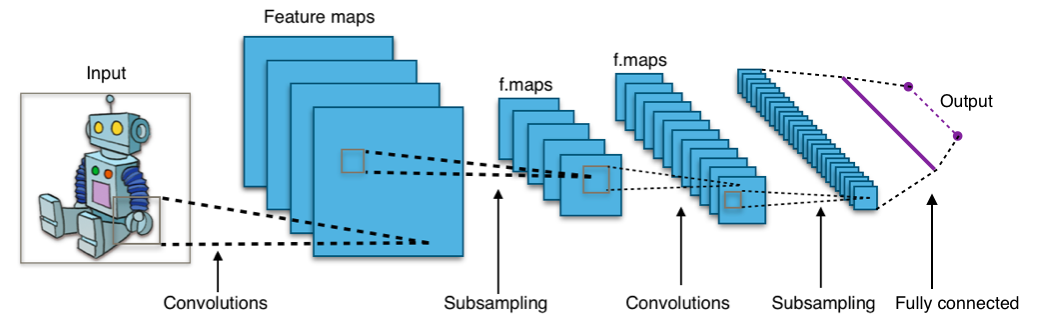
**4.1.1 Tools Used**

i. Skimage.color

Skimage.color is a package that helps in changing the color scheme of an image. It contains functions such as rgb2lab which converts rgb color scheme to lab color scheme, lab2rgb which converts lab color scheme to rgb color scheme, rgb2gray which converts a colored image to grayscale image, etc.

**4.2 CNN Algorithm**

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, natural language processing, and financial time series.



**Fig 4.1: A Typical CNN Architecture**

We have designed the model using a convolution neural network or a CNN. In the current state of our software, we have designed the neural network with an input layer, 5 hidden layers and an output layer.

A 2-D Convolution layer is taken as the input layer. A 2-D convolutional layer applies sliding convolutional filters to the input. The layer convolves the input by moving the filters along the input vertically and horizontally and computing the dot product of the weights and the input, and then adding a bias term. This layer takes the input in the shape 256 by 256 by 1 and uses Rectified Linear Unit as activation function. The hidden layers consist of 3 2-D convolution layers and two UpSampling2D layers. An UpSampling2D layer simply doubles the dimensions of the input and does not perform any learning. As the 2-D convolution layer decreases the dimensions of the input while learning, an UpSampling2D layer is required to regain the dimensions of the original input. The hidden layers still need to be further tested in order to either increase or decrease the number of layers for better accuracy. And lastly, the output layer simply provides the output prediction.

The last phase of the learning process is testing the trained model in order to determine its authenticity. In this phase, we took the previously mentioned test data consisting of 22 images and processed them in the same way the training data was processed. The lightness value of the test data is passed into the model and the model predicts the a and b values. With the lightness value and the predicted a and b values, we created the colorized image in the CIE color scheme. The image is then converted into the RGB format.

**4.2.1 Tools Used**

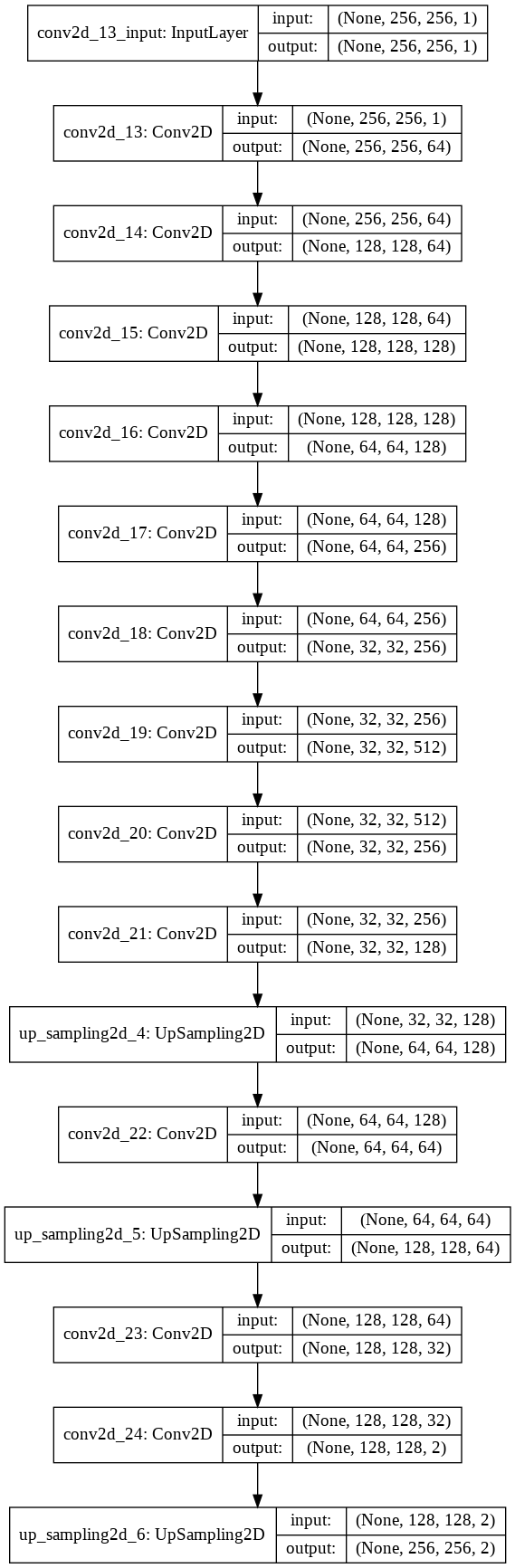
**i. Keras**

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK or Theano. It has many sub-packages that help in building the model. Some of the packages are:

**Keras.models:** It helps define the model

**Keras.callbacks:** It is used for creating checkpoint per epoch so that the highest accuracy is stored

**Keras.layers:** It contains different functions such as Activation, Conv2D, Conv2DTranspose, Dense, Dropout, Flatten, InputLayer, UpSampling2D, etc. which help define the layers on the CNN model.



**Fig 4.2: CNN Model used**

**5. Experiments**

To determine the best possible accuracy, we trained the model with different optimizers such as sgd, rmsprop, etc. and different loss function such as Mean Squared Error. The results obtained are shown below:

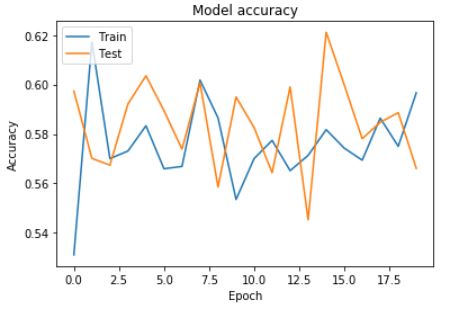
Parameters:

### **Experiment 1**

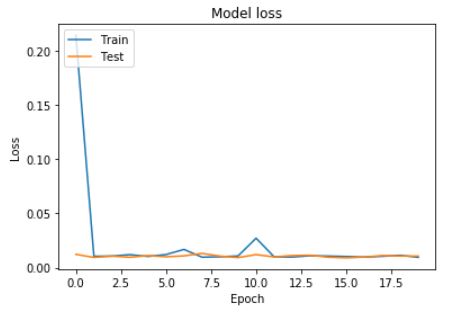
**Loss Function**: Mean Squared Error(mse)

**Optimizer:** rmsprop

Epochs: 20



**Fig: Model Accuracy**



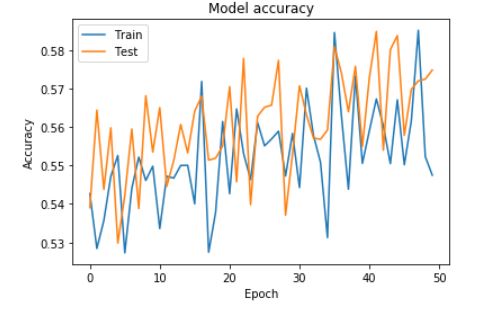
**Fig: Model Loss**

1. **Experiment 2**

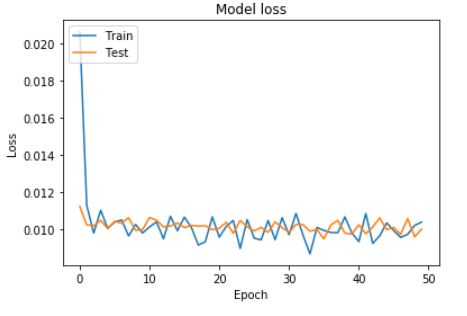
**Loss Function**: Mean Squared Error(mse)

**Optimizer:** sgd

Epochs: 50



**Fig: Model Accuracy**



**Fig: Model Loss**

**6. Evaluation of Results**

**i) Mean Squared Error**

Mean Squared Error (MSE) of an estimator measures the average squares of the errors i.e. the average squared difference between the estimated values and the actual value. The MSE is a measure of the quality of an estimator- it is always non-negative and values closer to zero are better.

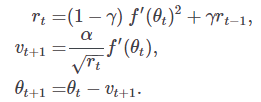
The MSE of the estimator is calculated as:

MSE = ∑(𝑦−𝑦𝑝)2 𝑛 = 1.29

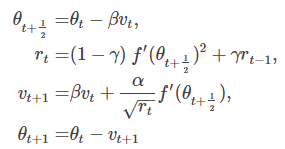
**ii) RMSprop**

RmsProp is an optimizer that utilizes the magnitude of recent gradients to normalize the gradients. We always keep a moving average over the root mean squared (hence Rms) gradients, by which we divide the current gradient. Let f′(θt) be the derivative of the loss with respect to the parameters at time step t. In its basic form, given a step rate α and a decay term

γ we perform the following updates:



In some cases, adding a momentum term β is beneficial. Here, Nesterov momentum is used:



RmsProp has several advantages; for one, it is a very robust optimizer which has pseudo curvature information. Additionally, it can deal with stochastic objectives very nicely, making it applicable to mini batch learning. With RmsProp as the optimizer and Mean Squared Error as the loss function, the accuracy obtained was 62% whereas the loss was near about 0.

**iii) Sgd**

Stochastic gradient descent (often abbreviated SGD) is an [iterative method](https://en.wikipedia.org/wiki/Iterative_method) for [optimizing](https://en.wikipedia.org/wiki/Mathematical_optimization) an [objective function](https://en.wikipedia.org/wiki/Objective_function) with suitable [smoothness](https://en.wikipedia.org/wiki/Smoothness) properties (e.g. [differentiable](https://en.wikipedia.org/wiki/Differentiable_function) or [subdifferentiable](https://en.wikipedia.org/wiki/Subgradient_method)). It can be regarded as a [stochastic approximation](https://en.wikipedia.org/wiki/Stochastic_approximation) of [gradient descent](https://en.wikipedia.org/wiki/Gradient_descent) optimization, since it replaces the actual gradient (calculated from the entire [data set](https://en.wikipedia.org/wiki/Data_set)) by an estimate thereof (calculated from a randomly selected subset of the data).[[1]](https://en.wikipedia.org/wiki/Stochastic_gradient_descent#cite_note-Taddy2019-1) Especially in [big data](https://en.wikipedia.org/wiki/Big_data) applications this reduces the [computational burden](https://en.wikipedia.org/wiki/Computational_complexity), achieving faster iterations in trade for a slightly lower convergence rate. With sgd as the optimizer, the accuracy obtained was about 58% with loss 0.01.

**7. Discussion on Results**

Out of the experiments performed, the model that used the RmsProp optimizer had both training accuracy and validation accuracy more that the one that used the stochastic gradient descent optimizer. Moreover, the loss value is also less than that of stochastic gradient descent. Hence, the model with RmsProp as optimizer and Mean Squared Error as loss function has better results.

# 

# 

# 

# **8. Contributions of each group member**

|  |  |
| --- | --- |
| **Members** | **Contributions** |
| Bipin Bohara | * Data preprocessing * Algorithm testing and optimizing |
| Aayush Malla | * Model Testing * Tested model on real life examples |
| Pratik Rajbhandari | * Researched on Related Works * Dataset Augmentation * Experimented with Model Layers |
| Bishal Sarangkoti | * Experimented with different loss functions, optimizers and learning rates. |

# 

# **9. Code**

Github Link: https://github.com/Rpratik13/Image-Colorization

# **10. Conclusion and Future Extensions to the Project**

In this work we design and develop a methodology, strongly based on Convolution Neural Network to colorize gray level images. Our report shows a series of steps to ﬁnally obtain a trained neural network which predicts the color of a gray level pixel. The methodology joins two main ideas from previous works: color reduction using vector quantization and using a group of neighboring pixels to predict the color of a single pixel. We implement the methodology and test it with an open set of images. To a human eye the images obtained show good results and have colors close to the intuition. As the prediction highly depends on the set of training images, the images more similar to the training set show better results than those more different. This project is a work in progress and we are convinced that with more extensive tuning and precise conﬁguration the results will improve. Nevertheless, we believe that the methodology proposed is an interesting ﬁrst step towards colorizing images without the need of heavy image processing algorithms.

**Future Enhancements**

* Using the colorization process in each frame of a black and white video to obtain a colored version of the video
* Taking a larger dataset while training the model to obtain better prediction accuracy
* Taking dataset containing more than just faces of people to colorize all kinds of sceneries
* Predict distribution of possible colors for each pixel and then re weight at training time to emphasize rare color

**References**

1. A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization.
2. Y.-C. Huang, Y.-S. Tung, J.-C. Chen, S.-W. Wang, and J.-L. Wu. An adaptive edge detection based colorization algorithm and its applications.
3. L. Yatziv and G. Sapiro. Fast image and video colorization using chrominance blending
4. Y. Qu, T.-T. Wong, and P.-A. Heng. Manga colorization
5. Q. Luan, F. Wen, D. Cohen-Or, L. Liang, Y.-Q. Xu, and H.-Y. Shum. Natural image colorization.
6. T. Welsh, M. Ashikhmin, and K. Mueller. Transferring color to greyscale images.
7. G. Charpiat, M. Hofmann, and B. Scholkopf. Automatic image colorization via multimodal predictions.
8. R. K. Gupta, A. Y.-S. Chia, D. Rajan, E. S. Ng, and H. Zhiyong. Image colorization using similar images
9. X. Liu, L. Wan, Y. Qu, T.-T. Wong, S. Lin, C.-S. Leung, and P.-A. Heng. Intrinsic colorization.
10. A. Y.-S. Chia, S. Zhuo, R. K. Gupta, Y.-W. Tai, S.-Y. Cho, P. Tan, and S. Lin. Semantic colorization with internet images.