Still Image Colorization using CNN
Sudha Narang¹, Srishti Bansal², Deepali³
Assistant Professor¹, Student²,³
Department of Computer Science and Engineering
Maharaja Agrasen Institute of Technology, New Delhi, India

Abstract:
Image colorization is a process of automatically putting colors in a gray scale image where there is no manual work required and this task is illusively tricky. The underlying issue is that the resultant image needs to be perceptually meaningful, and visually appealing. To solve this issue, we have designed a CNN model along with VGG16 architecture that recognizes a grayscale image and returns a colored image output. We have umbrellaed the concepts of image recognition, color prediction and convolutional neural networks along with the use of Keras library in Python to construct our desired model. A user interface has also been fabricated to get personalized inputs.

Keywords: Color prediction, Convolutional Neural Network, Image recognition, VGG16, tkinter.

I. INTRODUCTION
Grayscale image is the one which contains only grey shades, ranging from pure white to pure black. It has no color information and gives details only about the brightness. So, to get the details in the image, colors are important. Grayscale images can be quite misleading due to lack of colors. In earlier times, colored images were not in use and so, pictures from that time do not provide us the accurate details of the ancient times. Therefore, adding colors to those images is important to get precise information from them. So, we have developed a convolutional-neural based system that colorizes grayscale images automatically. Formerly, this was done manually, which meant literally painting the image. Software did help, but still manual work was involved. The user had to give details about boundaries, what colors to put, etc. Colorization requires considerable user intervention and remains a tedious, time-consuming, and expensive task. So, in this paper we try to build a model to colorize the grayscale images automatically by using some modern deep learning techniques. In colorization task, the model needs to find characteristics to map grayscale images with colored ones.

Figure.1. Representation of black and white image in grids of pixels
As we can see in the above picture, every black and white image can be represented in grids of pixels. Each pixel represents a value which is actually the brightness value of the image. This value ranges from 0-255, from black to white. Color image has three layers: red, green and blue, which means that the image is present in all the three channels. This color will represent brightness. Three color channels of the image of the leaf can be seen below:

Figure.2. Three color channels

In this paper, we have trained a Convolutional Neural Network (CNN) to link a grayscale image input to a colorful output. We have trained our model on over a thousand images. Our idea is to use a fully automatic approach which produces decent and realistic colorizations.

A. IMAGE RECOGNITION
Image recognition is the process which is used to recognize objects in images like places, writing and actions. Computers use very advanced vision technologies along with a camera and artificial intelligence software for recognizing images. Image recognition has a huge number of visual machine-based applications, such as self-driving cars, accident avoidance systems, guiding autonomous robots, performing image content search, etc. While human brains recognize objects easily, computers have difficulty in recognizing images. Deep machine learning is required by image recognition software. Performance of the software is finest at convolutional neural net processors because otherwise, the task requires very large amount of power due to its compute-intensive nature. Comparative 3D models are used as algorithms for image recognition from different angles using edge detection or by other components. These algorithms are generally trained on millions of pre-labeled images using the benefits of computer learning which helps to recognize images more accurately.

B. COLOR PREDICTION
Color prediction is done prior to the actual colorization of an image. It actually provides initial colorization to the image before user intervention and is considered to be one of the most important tasks in colorization. In this step, the input image is compared with the dataset images to match with a similar image
from the trained dataset. By doing this, we get some idea about the actual color of the image. Local predictors cause some ambiguities in the prediction, so global methods need to be developed which are not dependent only on the limited neighborhood texture-based classification.

II. LITERATURE REVIEW

Different methods and approaches have been proposed for colorizing a gray scale image automatically. Some of the possible ideas included the use of Support Vector Machines (SVM), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Generative Adversarial Networks (GAN) and other classification algorithms. In this paper, we will train a model using Convolutional Neural Network to map from a grayscale image input to a distribution over quantized color value output, i.e. take a black and white photo and produce a colored version of it. Depending on what is in the picture, it is possible to tell what the color should be. This is a classification problem in which class-rebalancing is used at training time which increases the diversity of colors in the resulting image. The model will only learn how to color images, allowing it to focus at what matters.

A. CONVOLUTIONAL NEURAL NETWORK (CNN)

In CNN classification models, more details can be extracted and not just the final classification result. Various studies have shown how to visualize the intermediate layers of a CNN and it seems that objects like car wheels and people start becoming recognizable by layer three itself. The intermediate layers in such classification models can give essential details about the colors in an image.

B. LOSS FUNCTION

The loss function that has been used in our model is the mean squared error (MSE) function. It is easily understandable and generally works quite well. Mathematically, it is represented as:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (w^T x(i) - y(i))^2$$

It is the average of the squared error i.e. it is the sum of the square of the difference between our target variable and the predicted values, over all the data points. This function effectively disciplines larger errors severely because of its formulation.

III. METHODOLOGY

In our paper, we have used CNN model along with the VGG16 model-based architecture to color a grayscale image, taken as input from user via a GUI. The entire model and the GUI have been coded using Python language. The dataset used in the model comprises of a thousand colored images. All the images are of 256 x 256 resolution. This dataset is used to train the model by importing it and applying CNN model on it.

A. BASIC APPROACH

The image set used for the model is divided into the training image set and the testing image set. The train image set is given to the machine in raw input vector representation. On this input set, deep neural network (convolution operations) is applied. On the test image set, image patch segmentation and effective patch extraction is done. Effective image patch extraction gives us the set on which deep neural network is applied. Deep neural network is applied in the form of layers: input layer, hidden layer, output layer, soft max classification. From the CNN applied train image set, network parameter is passed to the CNN applied test image set. Using the network parameter, both the sets are compared and majority voting is done and we finally get the classification results.

B. MODEL ARCHITECTURE

VGG16 is a pre-trained image classification model that is used to extract features for colorization. This model has been used as it has a simple architecture but is still competitive with other pre-trained models like UNet, ResNet, etc.
We have constructed our model based on the VGG16 architecture using the Keras library in Python. To implement the model, firstly, an image has been feed forwarded through the VGG16 architecture and from it, a few layers have been extracted. These layers have been then upscaled to the original image size (256x256). This gives a lot of information about what is in that image and we are then able to color our image using this information. Instead of re-developing the whole RGB color image, we train our models to give two color channels which are then merged with the grayscale input image to give a YUV image. In this, the Y channel makes sure that the output intensity will always be same as the input intensity. Also, it is simple to convert a YUV image to and from RGB. After passing through VGG16, we use 1x1 convolution to get two UV channels which are concatenated with grayscale channel and produce a YUV image (256 x 256 x 3). We used Rectified Linear Units (ReLU) as the activation function in our entire model, leaving only at the last output to the UV channels. In the last output, we used sigmoid function to flatten values between 0 and 1. Behind every convolution, we have used batch norm (BN). After passing the grayscale image through VGG16, the model infers some color details using the highest layers and then upscales it and adds details obtained from the next highest layer. The model repeats this and works down to the bottom of the VGG16 till we get a 256x256x3 tensor. This way we used residual connections in our model to add details while working to the bottom of the VGG16. The model built occupies quite less memory as compared to the other models (like, where after VGG16 we would get 963 channels and then concatenate all of them together to give a hypercolumn tensor; these hyper columns consume a lot of memory).

C. USER INTERFACE
A user interface has been created where the user can select a grayscale image of 256 x 256 resolution to get a colored output for the same. We used tkinter to develop our Python GUI (graphical user interface). Tkinter is the most commonly used standard interface to the Tk GUI toolkit which is shipped with Python.

In this, we first import our module and then create main window where we can add any number of widgets and apply the event Trigger on them.

IV. RESULT ANALYSIS
After training our model for 25-30 epochs (one epoch means that the entire dataset has been passed through the neural network, both forward and backward, exactly once), we got an accuracy of around 60% and loss of approx. 1%, as depicted in the graphs below:

![Model accuracy graph for 25 epochs](image1.png)

![Model loss graph for 25 epochs](image2.png)

At this point, the grayscale images in the testing dataset were converted into vintage kind (sepia-toned) images with very little coloring effect. The outcome obtained were something like this:

![Image results at 60% accuracy (25 epochs)](image3.png)
We can see here that in the grayscale images, the grey shades have been converted into the warm vintage-kind tones. The sepia tones show that the model started to colorize the images when it is trained for only 25 epochs. Using the python GUI created, the user can select any grayscale image of his choice in the following way:

Figure.10. Selecting grayscale image for colorization using GUI created
When we trained our model completely (for 500 epochs), the accuracy obtained was approximately 72% and the model loss was about 0.25%.

Figure.11. Model accuracy graph for 500 epochs
Figure.12. Model loss graph for 500 epochs

After training the model for 500 epochs, we got the auto-colorized images from the model using the interface created. The accuracy, though, obtained from the model is not very large. It is approximately 72% which means that our model works well in seventy percent times. It is able to recognize the images properly and then color them according to the trained dataset. The results obtained are varying.

Some of the results are shown as below:

Figure.13. Colored image output (sample 1)
Figure.14. Colored image output (sample 2)
Figure.15. Colored image output (sample 3)

We notice here that the first two images have been colored by the model quite decently while in the third image, there are some patches which have been left uncolored. From the colored images obtained, we can conclude that our model could colorize the warm tones more efficiently and accurately as compared to the lighter ones.

V. CONCLUSION

Image colorization is a task wherein color prediction and pixel prediction play a major role. In this paper, titled ‘Still Image Colorization Using CNN’, we have seen that by using deep CNN and a well-built VGG16 based architecture, we get quite appropriate results to colorize a grayscale image that are nearly indistinguishable from the real color photos. From the colored
image results obtained, we can conclude that our model could colorize the warm tones more efficiently and accurately as compared to the lighter ones. Although the model developed here has been trained for colorization purpose, it can be well-suited for applications with object or image classification, detection and segmentation.

VI. FUTURE SCOPE

The model developed here can be improved in various ways to give us better outcomes. The VGG16 architecture applied here can be replaced with more modern models like ResNet where more layer pooling and more training would give better results. The model currently uses only 256x256 images which restricts its usage. It can be modelled for full size images and a more diverse and wide-ranged dataset can be used to give better results as the model will be trained more efficiently. The GUI created can be made more interactive and converted into an online source for public use. Also, adversarial networks like Generative Adversarial Network (GAN) can be used on the bigger datasets to make the model more complex and provide better classification results.

VII. REFERENCES


[10]. Introduction to Loss Functions https:// blog.algorithmia.com/introduction-to-loss-functions/

