Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IMPACT FACTOR: 7.056

IJCSMC, Vol. 10, Issue. 7, July 2021, pg.10 – 19

AUTOMATIC COLORIZATION USING CONVOLUTIONAL NEURAL NETWORKS

[1]N. Lakshmi Prasanna; [2]Sk. Sohal Rehman; [3]V. Naga Phani; [4]S. Koteswara Rao; [5]T. Ram Santosh

[1]Associate Professor, Dept of CSE, Vasireddy Venkatadri Institute of Technology, Nambur, Guntur (Dist.), Andhra Pradesh, INDIA

[2,3,4,5]B-Tech, Dept of CSE, Vasireddy Venkatadri Institute of Technology, Nambur, Guntur (Dist.), Andhra Pradesh, INDIA

DOI: 10.47760/ijcsmc.2021.v10i07.002

Abstract— Automatic Colorization helps to hallucinate what an input gray scale image would look like when colorized. Automatic coloring makes it look and feel better than Grayscale. One of the most important technologies used in Machine learning is Deep Learning. Deep learning is nothing but to train the computer with certain algorithms which imitates the working of the human brain. Some of the areas in which it is used are medical, Industrial Automation, Electronics etc. The main objective of this project is coloring Grayscale images. We have umbrellaed the concepts of convolutional neural networks along with the use of the Opencv library in Python to construct our desired model. A user interface has also been fabricated to get personalized inputs using PIL. 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.

Keywords— machine learning, Python Imaging Library(PIL), CNN (convolutional neural networks), opencv.

I. INTRODUCTION

Consider the grayscale photographs in Figure 1.1. 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 paper, 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.

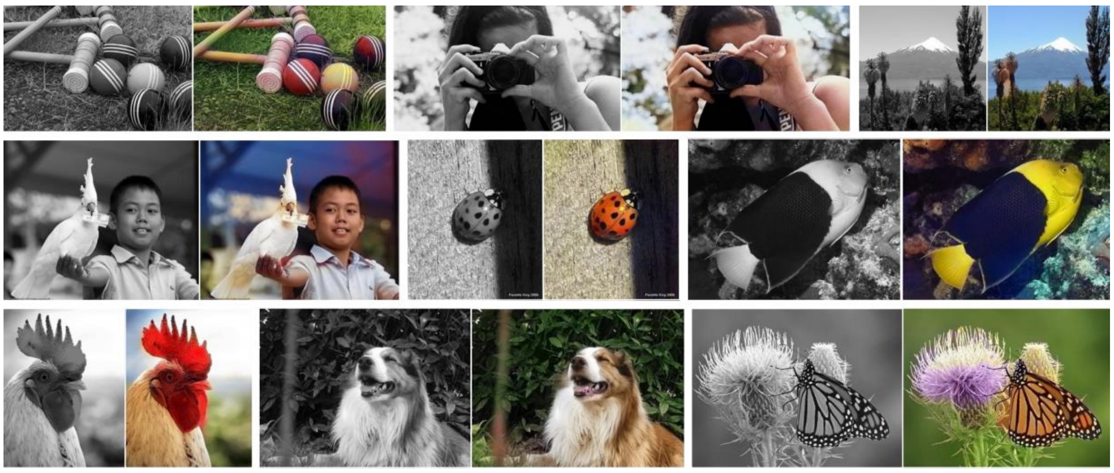

Figure 1.1. Example input grayscale photos and output colorizations from our algorithm.

Colorization of old black and white photos provides an eye-opening window to visualizing and understanding the past. People, at least from my generation, tend to imagine any scene from the years between the invention of the photograph and the invention of the color photograph as black and white, when of course humans have always perceived color from the dawn of civilization. It stretches the imagination to mentally add color to those old timey photographs but being able to picture 19th century London as anything other than a distant black and white caricature can enhance one's connection with, and appreciation for, history. So,we have developed a convolutional-neural based system that colorizes gray scale 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 the colorization task, the model needs to find characteristics to map grayscale images with colored ones.

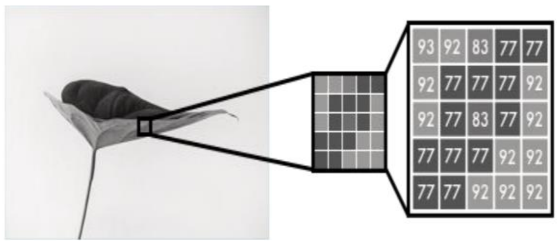

Figure.1.2. Representation of black and white image in grids of pixels

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:

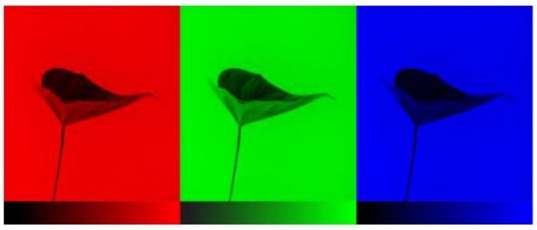

Figure.1.3. Three color channels

In this paper, we have trained a Convolutional Neural Network (CNN) to link a grayscale image input to a colorful output. In this we are using a pre-trained model that can be accessed by using opencv in python. Our idea is to use a fully automatic approach which produces decent and realistic colorizations.

II. RELATED WORK

2.1 MACHINE LEARNING

Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks. For simple tasks assigned to computers, it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand; on the computer's part, no learning is needed. For more advanced tasks, it can be challenging for a human to manually create the needed algorithms. In practice, it can turn out to be more effective to help the machine develop its own algorithm, rather than having human programmers specify every needed step.

The mathematical optimization study gives methods, theory and application domains to the field of machine learning. Data mining is a field of study related to ml, focusing on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning can be called as predictive analytics.

The discipline of machine learning employs various approaches to teach computers to accomplish tasks where no fully satisfactory algorithm is available. In cases where vast numbers of potential answers exist, one approach is to label some of the correct answers as valid. This can then be used as training data for the computer to improve the algorithm(s) it uses to determine correct answers. For example, to train a system for the task of digital character recognition, the MNIST dataset of handwritten digits has often been used.

A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers; but not all machine learning is statistical learning. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. Its application across business problems, machine learning is also referred to as predictive analytics.

Early days classifications for machine learning approaches divided them into 3 categories, depending on the nature of the signal or feedback available to the learning system. These were:

Supervised learning: The computer is provided with the sample data called "training data", by accessing and learning from it the computer can generate the output

Unsupervised learning: No labels are given to the learning algorithm (no training data), leaving it on its own to find structure in its input. Unlike supervised learning Unsupervised learning can be a goal in itself or a means towards an end (feature learning).

Reinforcement learning: A machine program (lines of code) interacts with a dynamic environment and there it must perform a certain task/goal as it navigates its problem space, the program is provided feedback that's analogous to rewards/outputs, which it tries to maximize.

Other approaches or processes developed but don't fit neatly into this three-fold categorization, and sometimes more than one approach is used by the same machine learning system. As of 2020, deep learning had become the best approach for much ongoing work in the field of machine learning.

Deep Learning is a subset of machine learning concerned with algorithms evolved by the structure and function of the brain called artificial neural networks (ANN). In addition to scalability, another usage of deep learning models is the ability to perform automatic feature extraction from raw data, also called feature learning.

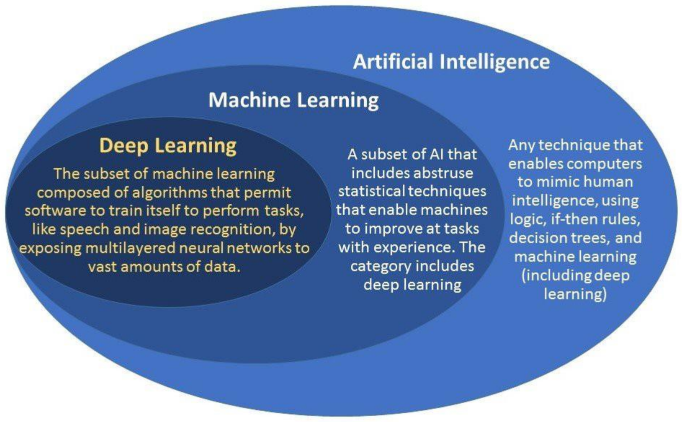
Artificial Intelligence
Machine Learning
Deep Learning
The subset of machine learning composed of algorithms that permit software to train itself to perform tasks, like speech and image recognition, by exposing multilayered neural networks to vast amounts of data.
A subset of AI that includes abstruse statistical techniques that enable machines to improve at tasks with experience. The category includes deep learning
Any technique that enables computers to mimic human intelligence, using logic, if-then rules, decision trees, and machine learning (including deep learning)

Figure.2.1. Overview of Machine Learning

2.2 CONVULUTION NEURAL NETWORKS

A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically, this includes a layer that performs a dot product of the convolution kernel with the layer's input matrix. This product is usually the Frobenius inner product, and its activation function is commonly ReLU. As the convolution kernel slides along the input matrix for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, fully connected layers, and normalization layers.

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.

CNNs are often used in image recognition systems. In 2012 an error rate of 0.23% on the MNIST database was reported. Another paper on using CNN for image classification reported that the learning process was "surprisingly fast"; in the same paper, the best published results as of 2011 were achieved in the MNIST database and the NORB database. Subsequently, a similar CNN called Alex Net won the ImageNet Large Scale Visual Recognition Challenge 2012.

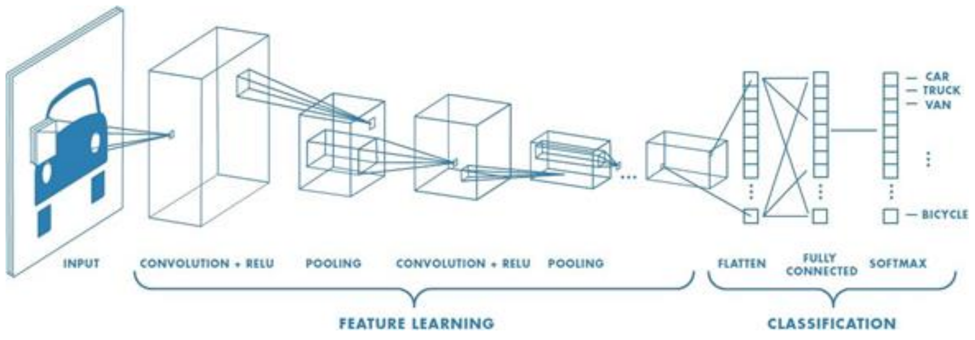

Figure.2.2. Convolutional Neural Network Model

2.3 LIBRARIES

Opencv:

Python bindings of the widely used computer vision library Opencv utilize Numpy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image. The Numpy array as a universal data structure in Opencv for images, extracted feature points, filter kernels and many more vastly simplifies the programming workflow and debugging.

Opencv 2 includes major changes to the C++ interface, aiming at easier, more type-safe patterns, new functions, and better implementations for existing ones in terms of performance (especially on multi-core systems). Official releases now occur every six months and development is now done by an independent Russian team supported by commercial corporations.

The first alpha version of Opencv was released to the public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and five betas were released between 2001 and 2005. The first 1.0 version was released in 2006. A version 1.1 "pre-release" was released in October 2008.

The second major release of the Opencv was in October 2009. Opencv 2 includes major changes to the C++ interface, aiming at easier, more type-safe patterns, new functions, and better implementations for existing ones in terms of performance (especially on multi-core systems). Official releases now occur every six months and development is now done by an independent Russian team supported by commercial corporations.

Tkinter:

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and Mac OS X installs of Python. The name Tkinter comes from the Tk interface**.**

As with most other modern Tk bindings, Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. Tkinter calls are translated into Tcl commands, which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application.

Numpy:

Numpy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. Numpy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays; using these requires rewriting some code, mostly inner loops, using Numpy.

Such arrays can also be viewed into memory buffers allocated by C/C++, python, and Fortran extensions to the CPython interpreter without the need to copy data around, giving a degree of compatibility with existing numerical libraries. This functionality is exploited by the SciPy package, which wraps a number of such libraries (notably BLAS and LAPACK). Numpy has built-in support for memory-mapped ndarrays.

Python Imaging Library(PIL):

Python Imaging Library is a free and open-source additional library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux. Some of the file formats supported are PPM, PNG, JPEG, GIF, TIFF, and BMP. It is also possible to create new file decoders to expand the library of file formats accessible. Pillow offers several standard procedures for image manipulation.

These include: per-pixel manipulations, masking and transparency handling, image filtering, such as blurring, contouring, smoothing, or edge finding, image enhancing, such as sharpening, adjusting brightness, contrast or color, adding text to images and much more.

III. REVIEW OF LITERATURE

Richard Zhang, JUN-YAN ZHU [1] published a paper. i.e Deep network to predict the color of an image, given the grayscale version and user inputs. Describe the objective of the network, then describe the two variants of our system (i) the Local Hints Network, which uses sparse user points, and (ii) the Global Hints Network, which uses global statistics and define our network architecture. For this dataset used was of kalboard 360 and applied it on weka to analyze the data mining techniques. In this paper, WEKA (Waikato Environment for Knowledge Analysis) is used for the analysis of data and to build a model to get predictive outcome. This system is currently trained on points; we find that in this regime random sampling covers the low-dimensional workspace surprisingly well. However, a future step is to better simulate the user, and to effectively incorporate stroke-based inputs that traditional methods utilize. Integration between the local user points and global statistics inputs would be an interesting next step. Our interface code and models are publicly available at https://richzhang.github.io/ideepcolor , along with all.

Sudha Narang [2] has done CNN model along with the VGG16 model- based architecture to color a grayscale image, taken as input from user via a GUI. In this study, 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. 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 International Journal of Engineering Science and Computing, May 2019 21876 http://ijesc.org/ 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.

Richard Zhang [3], used Colorization Turing Test, Loss Function, Self Supervision algorithm and two parametric functions**.** Here 240 images as a dataset related to six categories like Kitchen, Beach, Bedroom, Castle, Living Room, Outdoor. These images are trained using LEARCH algorithm through L channel. 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.

R. Dahl [4], has done Deep Convolutional Generative Adversarial Networks. Adversarial modeling framework provides an approach to training a neural network model that estimates the generative distribution. proposed. We use the YUV color space instead of the RGB color space, since the YUV color space minimizes the per-pixel correlation between the color channels. The Y channel encodes the luminance component of the image, and can be interpreted as the grayscale version of the image, while the U and V color, or chrominance, channels encode the colors of the corresponding pixels. Our colorization systems thus take as input the Y component and outputs a prediction for the UV components. He trained our adversarial models using Algorithm 2 with minibatch size of 128 images, a learning rate decay of $1 \times 10-7$ for every iteration, and a step decay of 1 2 for every 25 epochs. The discriminator is heavily regularized with a dropout probability of 0.8 in the convolutional layers and 0.5 in the penultimate fully-connected layer.

IV. ALGORITHM AND FLOW MODEL

a. CAFFE MODEL

Caffe (Convolutional Architecture for Fast Feature Embedding) is a deep learning framework that allows users to create image classification and image segmentation models. Initially, users create and save their models as plain text. prototxt files. After a user trains and refines their model using Caffe, the program saves the user's trained model as a CAFFEMODEL file. A CAFFEMODEL file is a machine learning model created by Caffe.

It contains an image classification or image segmentation model that has been trained using Caffe. CAFFEMODEL files are created from. PROTOTXT files. CAFFEMODEL files are binary protocol buffer files. As such, you cannot open, examine, and edit them in a source code editor, as you would PROTOTXT files. CAFFEMODEL files are meant to be integrated into applications that can utilize trained image classification and image.

b. PROTXT FILE

A PROTOTXT file is a prototype machine learning model created for use with Caffe. It contains an image classification or image segmentation model that is intended to be trained in Caffe. PROTOTXT files are used to create. CAFFEMODEL files. PROTOTXT files are intended to be used as prototype Caffe (cross-platform) machine learning models. However, because PROTOTXT files are plain text files, you can open and edit them in any text editor if needed.

PROTOTXT files are serialized using Google's Protocol Buffers serialization library, and they contain:

1. A list of the neural network layers a model contains.

2. Each layer's parameters, including its name, type, input dimensions, and output dimensions

3. Specifications for connections between layers

4.3 PROCESS

Previous approaches to black and white image colorization relied on manual human annotation and often produced de-saturated results that were not “believable” as true colorizations. we decided to attack the problem of image colorization by using Convolutional Neural Networks to “hallucinate” what an input grayscale image would look like when colorized.

To train the network started with the ImageNet dataset and converted all images from the RGB color space to the Lab color space.

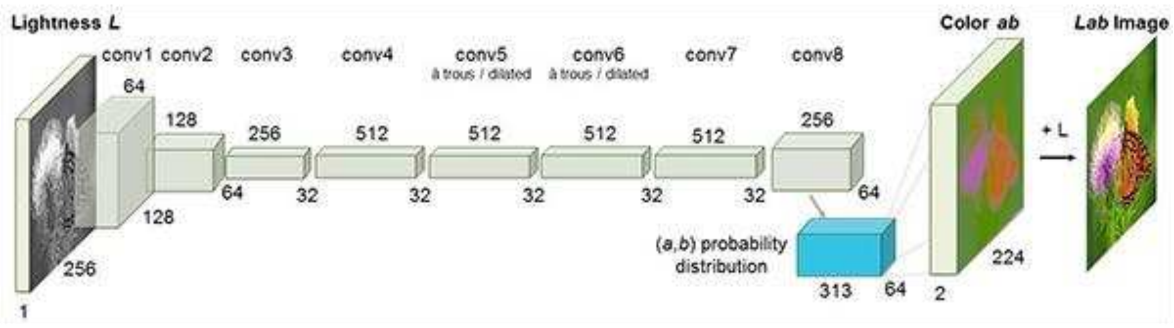

Figure 4.1: Architecture for colorization of black and white images with deep learning

Similar to the RGB color space, the Lab color space has *three channels*. But *unlike* the RGB color space, Lab encodes color information differently:

- The ***L* channel** encodes lightness intensity only.
- The ***a* channel** encodes green-red.
- And the ***b* channel** encodes blue-yellow.

Since the *L* channel encodes only the intensity, **we can use the *L* channel as our grayscale input to the network.**

From there the network must **learn to predict the *a* and *b* channels.** Given the **input *L* channel** and the **predicted *ab* channels** we can then form our **final output image**.

The entire (simplified) process can be summarized as:

- Convert all training images from the RGB color space to the Lab color space.
- Use the ***L* channel** as the input to the network and train the network to predict the ***ab* channels.**
- Combine the input ***L* channel** with the predicted ***ab* channels.**
- Convert the Lab image back to RGB

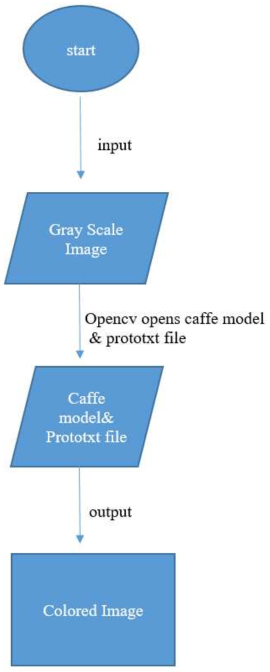
start
input
Gray Scale Image
Opencv opens caffe model & prototxt file
Caffe model& Prototxt file
output
Colored Image

Figure 4.2 Work Flow of Automatic Colorization

Given a grayscale photograph as input, this paper attacks the problem of hallucinating a plausible color version of the photograph. This problem is clearly under constrained, so previous approaches have either relied on significant user interaction or resulted in de-saturated colorizations. We propose a fully automatic approach that produces vibrant and realistic colorizations.

V. RESULTS

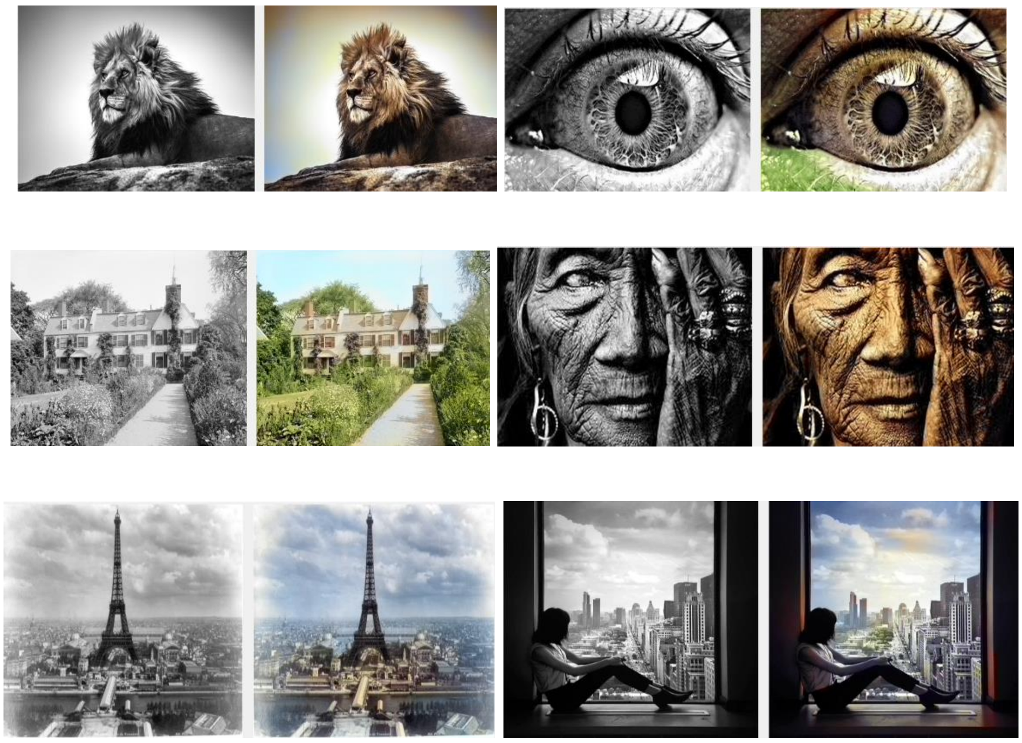

Figure 5.1. Example input grayscale photos and output colorizations from our algorithm.

In this the left column is grayscale version, given as input and the right side is the output obtained from our model. our model produced outstanding results compared to other models with respect to some of the properties like color, quality, accuracy etc.

In many situations we can take advantage of the model. Some of the cases are historical pictures, old motion pictures etc.to our code is the image data set.

VI. CONCLUSION

We have discussed above the Caffe model and Prototxt file to colorize a given gray image using a color transfer mechanism. It uses texture features to recognize various objects in the user provided reference image and also describes a color transfer method. To achieve better colorization, we generate micro scribbles at high confidence areas and the complete colorization is performed using CNN algorithm.

The combination of CNN with Opencv gives highly optimized colorization of images. We presented a method of fully automatic colorization of unique grayscale cartoon images combining state-of-the-art CNN techniques. Using the Opencv and color representation, we have shown that the method is capable of producing a plausible and vibrant colorization of certain parts of individual images even when applied to a moderately sized dataset that has properties which make it harder to colorize than natural images, but does not perform as well when applied to video sequences.

Future Scope

In order to be applicable for video, the method would currently require further refinement, performed manually by an artist. If trained on a larger dataset, the predictive power of the model would increase and is likely to produce more consistent colorization. For future work, it would be interesting to compare colorization produced by ResNet models with significantly more depth (which require more computational resources to train) and models based on conditional generative adversarial networks. Additionally, the CNN model could be adjusted to generate scribbles to use in conjunction with the algorithms, instead of full colorization. This could lead to results that more closely match the currently used colorization methods that apply color to cartoon movies.

REFERENCES

[1]. Richard Zhang, Phillip Isola, Alexei A. Efros: Colorful Image Colorization. arXiv print arXiv:1603.08511, 2016.

[2]. Cheng, Z., Yang, Q., Sheng, B.: Deep colorization. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 415–423.

[3]. Dahl, R.: Automatic colorization. In: http:// tiny clouds.org/colorize/. (2016)

[4]. Jeff Hwang and You Zhou.Image Colorization with Deep Convolutional Neural Networks. http://cs 231n.stanford.edu/ reports/2016/pdfs/219_Report.pdf

[5]. Charpiat, G., Hofmann, M., Sch¨olkopf, B.: Automatic image colorization via multimodal predictions. In: Computer Vision–ECCV 2008. Springer (2008) 126–139.

[6]. Ramanarayanan, G., Ferwerda, J., Walter, B., Bala, K.: Visual equivalence: towards a new standard for image fidelity. ACM Transactions on Graphics (TOG) 26(3) (2007).

[7]. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014).

[8]. A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages.

[9]. Colorizing B&W Photos with Neural Networks https://blog.floydhub.com/colorizing-b-w-photos-with-neural-networks/

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

[11].https://learning.oreilly.com/library/view/machine-learning with/9781785889936/

[12]. Dataset reference: https://www.kaggle.com/

[13].K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv: 15 12. 03385, 2015.

[14].Z. Cheng, Q. Yang, and B. Sheng. Deep colorization. In Proceedings of the IEEE International Conference on Computer Vision, pages 415–423, 2015.

[15].R. Dahl. Automatic colorization, Jan 2016. http:// tinyclouds.org/colorize/.

[16].I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Advances in Neural Information Processing Systems, pages 2672–2680, 2014.

[17].K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. CoRR, abs/1502.01852, 2015. [7] 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. In Proceedings of the 13th Annual ACM International Conference on Multimedia, MULTIMEDIA ’05, pages 351–354, New York, NY, USA, 2005. ACM.

[18].Y. Qu, T.-T. Wong, and P.-A. Heng. Manga colorization. In ACM SIGGRAPH 2006 Papers, SIGGRAPH ’06, pages 1214–1220, New York, NY, USA, 2006. ACM.

[19].A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. CoRR, abs/1511.06434, 2015.

[20].N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):1929–1958, 2014.