IARJSET

ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 7.105 ⫘ Vol. 9, Issue 1, January 2022

DOI: 10.17148/IARJSET.2022.9115

Image Colorization using Convolutional Autoencoders and Generative Adversarial Networks

Arshiya Sarmai[1], Apratim Sadhu[2]

Third Year B.E., Department of CSE, Chandigarh University, Mohali, India[1,2]

Abstract- Image colorization is an emerging topic and a fascinating area of research in recent years. In this paper, we implemented deep learning algorithms to colorize black and white images. Convolutional autoencoder neural network and generative adversarial network (GAN) have been implemented on black and white images. The networks aim to convert the black and white images to their respective color format. The color format used for the implementation of the networks is RGB color space. The performance of the autoencoder model is evaluated using root means squared error, means squared error, means absolute error and colorization accuracy. The data used for the implementation of the networks consists of 7000 images in the RGB color space. The performance of the implemented autoencoder models and generative adversarial network is measured using mean absolute error. The colorization performance of the generative adversarial network is better that that of the autoencoder models.

Keywords: Autoencoders, Colorization, Generative adversarial Network, Generator, Discriminator, Mean squared error

I. INTRODUCTION

The coloring of black and white images has been a hot and fascinating topic in the research of image processing. The coloring of black and white images has a great impact on the analysis of images and information retrieval. A colored image contains more information than a grayscale image and in most of the scenarios color image is more useful to extract information from the image. Colour images are vivid and visually appealing to viewers. The most common application of image colorization is coloring of black-and-white historical images and improvement of surveillance feed.

Image colorization is the process of assigning colors to a grayscale image in order to make it more artistically appealing and perceptually meaningful. This is known to be a sophisticated task that often requires prior knowledge about the image content and manual adjustments to achieve artefact-free quality. Furthermore, since objects can be in different colors, there are multiple possible ways to assign colors to the pixels in an image, which means there is no distinctive solution to this problem.

The most common colored image contains three-dimensional information about the color of the image. A color image is defined by three primary colors: red(R), green(G) and blue(B) which is known as RGB color space. The grayscale is the common way of representing a black and white image that consists of only luminance hence it is 1 dimensional. Converting a color image to grayscale is an easy task. But, converting a grayscale to a color image of RGB color space is a much difficult part. This is because the color of an image can be restored by the correct combination of the three colors. It is difficult to decide which color corresponds to this one particular gray level which we are trying to convert to color. A reason for this is the absence of deterministic relation between the luminance of grayscale image and corresponding exact colors of the same image.

Traditional image colorization can take a lot of time. A manual approach to coloring historical images can take up to several days. These approaches are time-consuming, cumbersome and hectic processes. This is where deep learning techniques come into play. The application of deep learning models for the colorization of grayscale images is the aim of this research. We have used convolutional neural networks to color input grayscale images. The pixels size of the image is kept at 160×160. Two types of autoencoders have been designed and implemented. The first type of autoencoder contains Upsampling layers in the decoder part and the second type of autoencoder contains transpose convolutional layers in the decoder. The network architecture is evaluated using root means squared error. Other metrics used for the evaluation of the network are means squared error, mean absolute error and the colorization accuracy of the network. The second type of approach is implemented using generative adversarial networks(GANs). The performances of the networks are evaluated in this paper.

This paper is divided into six sections. The second section consists of related works in the field of image colorization using machine learning and deep learning. The third section contains an introduction to autoencoders and generative adversarial networks. The implementation has been discussed in section 4. A discussion of the results of the implementation of the networks is presented in section 5. The paper is concluded with a brief note in section 6.

II. RELATED WORKS

Richard Zhang et al.[1] in their paper have proposed a deep learning approach for image colorization. The CNN network designed by them consists of two variants, global hints network and local hints network. The main branch of their proposed network uses a U-Net architecture. The network is formed using 10 convolution blocks. The models are evaluated using the peak signal-to-noise ratio(PSNR). The local hints network predicts color distribution as a side task. The global hints network does not have any spatial information. The inputs in the global hints network are processed through 4 conv-relu layers with kernel size 1×1 and 512 channels. The authors have concluded that their network predicts user-intended actions based on learned semantic similarities. However, They have also mentioned that the network can also be over-optimistic and produce undesired non-local effects

Madhab Raj Joshi et al.[2] in their research have used deep convolutional neural networks in combination with Inception-ResnetV2 to convert grayscale to coloured images. The colour space used by them is CIE Lab. The objective functions used by them for quality assessment of the coloured image are mean squared error(MSE) and peak signal-to-noise ratio(PSNR). They have trained their CNN model on 1.2K historical images The authors have concluded that MSE, PSNR and accuracy of the models are 6.08%, 34.65dB and 75.23% respectively.

P Ajay Kalyan et al.[3] in their paper have used deep convolutional neural networks using Inception-ResnetV2 pre-trained model for high-level feature extraction. They have implemented the network on a decreased subset of ImageNet data. The colour space used by the authors for the conversion of BW images is CIE Lab. The authors in their paper have not mentioned any metrics used to evaluate the performance of the CNN model.

Abhishek Pandey et al.[4] in their research have used a deep CNN autoencoder with pre-trained layers of Inception-ResnetV2 as embedding to the autoencoder. The images used by them have been rescaled to 256×256 pixels. The authors have used mean squared error(MSE) as an evaluation metric for the model. Their encoder is divided into 4 parts. The encoder component produces mid-level features and the feature extraction component produces high-level features. They have merged these into the fusion layer of the autoencoder.

Federico Baldassarre et al.[5] in their research have used a similar approach to that of the previously mentioned works. They have carried out their work to colour historical images. They have used approximately 60,000 images with 10% data as validation data and a batch size of 100. The authors have used CIE Lab colour space to recolour the grayscale images. They have used the trained model on several test cases. The authors have concluded that the model can colorize some images with up to 80% accuracy.

Sindhuja Kotala et al.[6] in their paper have used a similar approach to Zhang et al. for automatic image coloration. They have used images from the ImageNet archive with each image rescaled to 224×224 pixels. The CNN architecture used by them is a VGG-style network with multiple convolutional blocks as proposed by Zhang et al. They have used multinomial log loss to compute the output color and used it as colour rebalancing.

Tung Nguyen et al.[7] in their paper have proposed a novel approach of using deep learning techniques for colorization of ukiyo-e—a genre of Japanese paintings. They have used a pre-trained convolutional neural network, originally designed for image classification to separate style from the content of images. The authors have then proposed a method to colourize the grayscale images and combine their content with the separated style to get a colourized output.

Guillaume Charpiat et al.[8] in their research have used a machine learning approach for automatic image colourization. They have used support vector machines(SVM) for this purpose. The colour space used by the authors to convert grayscale to colour images is CIE Lab. Their approach retains the multi-modality in the images until the prediction step. Their framework provides a principled way of learning local color predictors along with spatial coherence criteria as opposed to the previous methods which chose the spatial coherence criteria manually.

Saeed Anwar et al.[9] in their article have presented a comprehensive survey of recent state-of-the-art colorization using deep learning algorithms, describing their fundamental block architectures in terms of skip connections, input,etc. as well as optimizers, loss functions, training protocols, training data,etc. They have categorized the present colourization techniques into seven classes. The authors have concluded an observation on the trend in image colourization. The first trend that they have mentioned is that GAN-based methods deliver diverse colorization visually compared to CNN-based methods, the existing models generally deliver a sub-optimal result for scenes that are complex and have a large number of small objects in them, deep models with higher complexity have little improvement in terms of numbers, the diversity of networks in image colorization as compared to other image restoration is significant, unsupervised learning techniques will be mostly used for image colorization and the last one is attention mechanisms can improve the performance of image colourization.

Jeff Hwang et al.[10] in their paper have designed a CNN that accepts black-and-white images and outputs generated color images. Their work is inspired in part by Ryan Dahl's CNN-based system for automatic image colorizing[11]. The input pixel dimensions of the grayscale images are 224×224 pixels. The images are converted to CIELUV color space. They have initialized a part of their network using the VGG16 instance. The dataset used by the authors consists of several thousand images of urban and natural scenes. They have used two types of methods, the first one is the baseline regression method and the other one is the classification method. The accuracy of the classification and regression systems is 34.64% and 12.92% respectively for the U channel of CIELUV. The accuracy of the V channel is 24.17% for classification systems and 19.02% for the regression system. The authors have concluded that the main challenge their model faced is inconsistency in colors within individual objects.

Stephen Koo[12] in his paper attempted implantation using DCGANs(deep convolutional generative adversarial networks) for automatic image colorization. They have implemented their model on the CIFAR-10 dataset. The dataset consists of 60,000 images each of 32×32 pixels. Their baseline approach directly learns a mapping from the grayscale image space to the color image space. The model was trained on batches of 128 images up to 125 epochs. The learning rate decay used by the optimizer is 1×10-7 Their system achieved a training loss of 0.0071and a validation loss of 0.0073.

III. LITERATURE REVIEW

A. Color Space

Color spaces are the different modes of color that are used in image processing. color spaces, such as RGB, CMYK, YUV, HSV. But the most widely used color space is RGB. RGB stands for red, green and blue. An image having RGB color model includes three independent planes of images red, green and blue.
Primarily, every image that is colored is made up of three different images. An image that is in grayscale format is made up of only one matrix, while a colored image is made up of three different matrices.
The RGB color space is any additive color space based on the RGB model. In the RGB color model, red, green and blue colors which are also known as the primary colors are combined together in different methods such as to produce a various array of colors such as cyan, magenta and yellow. This color model is also known as the additive color model. Each color in the RGB color channel ranges from 0(least saturated) to 255(most saturated). The combination of all these values results in 16,777,216 different colors in the RGB color space. The RGB color space is depicted in figure 1.

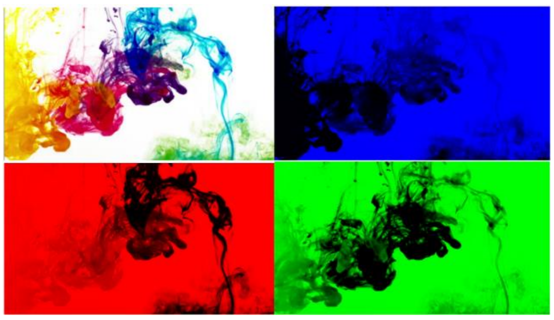

Fig. 1 RGB color space

The CIELAB color space also referred to as L*a*b* expresses the color as a combination of three values. L* is for perceptual lightness and a* and b* are for the four unique color spaces of human vision i.e. red, green, blue and yellow. The CIELAB space is three-dimensional color space which covers the entire human color perception range. The lightness value L* defines black at value 0 and white at value 100. The a* axis is relative to the green-red opponent colors, where negative values are towards green and positive values are towards red. The b* axis represents the blue-yellow opponent colors, where negative values are towards blue and positive values are towards yellow. The CIELAB color space is depicted in figure 2.

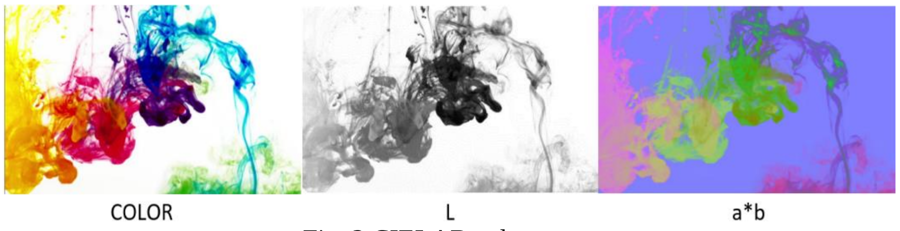

Fig. 2 CIELAB color space

B. Convolutional Auto-encoders

Autoencoders are unsupervised learning technique that uses feed-forward neural networks for representational learning. The basic idea of an autoencoder is to have an output with the same dimensionality as the input. An autoencoder replicates the data from the input to the output and is sometimes known as a replicator neural network.

The basic idea of an autoencoder is to compress the input data into low-dimensional code. This bottleneck part of the network is known as code. The part before the code is known as the encoder part and the part of the encoder after the code is known as the decoder.

An autoencoder has 3 components:

- Encoder: The encoder consists of layers that helps in compressing the input data and then it produces the code called the bottleneck.
- Bottleneck: It contains the information that is being compressed by the encoder and it is the most essential part of the network. It helps to choose which information is to be passed on to the decoder. It helps to prevent the overfitting of data due to its representation in the compressed form.
- Decoder: The decoder helps in reconstructing the compressed data into the desired output. It helps in building back the image from its features.

Important hyperparameters that are being used while training an autoencoder:

- Bottleneck size: The size of the code/bottleneck is one of the most important hyperparameters which is used in tuning the autoencoder. It helps in deciding how much of the data is being compressed.
- Number of layers: One of the most important hyperparameters is the number of layers in the encoder and the decoder.
- Number of nodes in a particular layer: In a stacked autoencoder, the number of layers per node decreases gradually and then it increases back in the decoder to get the same dimension as the input.
- Loss: The loss function that is used in the autoencoder is majorly dependent on the input served and the desired output. If the values lie between the range [0,1] use of binary cross-entropy is preferred and for spatial data such as image data, MSE(Mean squared error) is used.

The convolution autoencoder has a similar principle. It reconstructs images after passing them through the compression phase. The main difference between a traditional autoencoder and a convolutional autoencoder is the latter is focused using spatial relationship between points in order to extract features that have a visual interpretation[13]. Just like the encoder part of a convolutional autoencoder uses convolutional operation for the compression operation, The decompression operation uses a deconvolution operation. Deconvolution is also referred to as transposed convolutions. Similarly, the pooling operation in the encoder part is followed by a corresponding unpooling operation. Interestingly, the decoder in a convolutional autoencoder performs a similar operation to the backpropagation of gradient-based visualization.

The function of the convolutional autoencoder for coloring images is characterized by the number of filters in the output layer of the decoder part of the network. For converting a grayscale image to an image in the RGB color space, the number of filters in the output layer of the decoder is 3 for each color of the RGB color space. However, for LAB color space, the number of filters generally used is 2. The basic function of a convolutional autoencoder for image coloration is represented in fig 3.

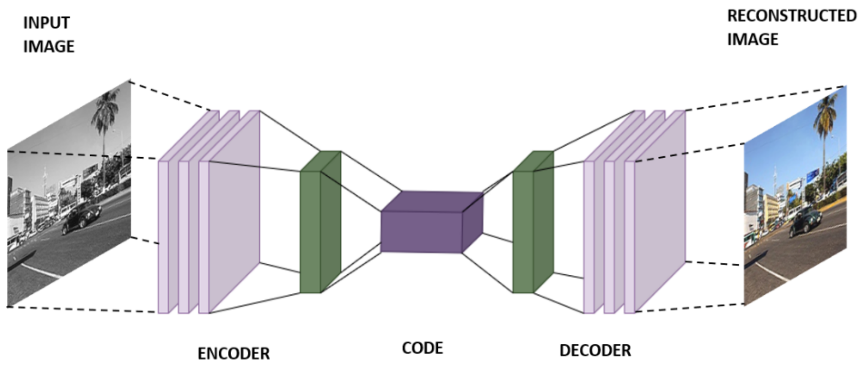

Fig. 3 A representation of convolution autoencoder for image coloration

C. GANs(Generative Adversarial Networks)

Generative Adversarial Networks(GANs) was first introduced in 2014 by Goodfellow et al[14]. GANs belongs to the set of generative models. They can produce or generate the new content. They are based on the concept of generative modelling. Generative modelling is an unsupervised technique which has the ability to recognise and learn the patterns in the input data in such a way that it can be used to generate or produce similar kind of data which could be the part of the original dataset.

GANs can be further divided into two sub-models:

Generator model: This model is trained so as to generate new data which could possibly be a part of the dataset.

Discriminator model: This model is trained to classify the original (original images from the dataset) and the generated images (obtained using the generative model).

The concept of GANs is presented in fig 4.

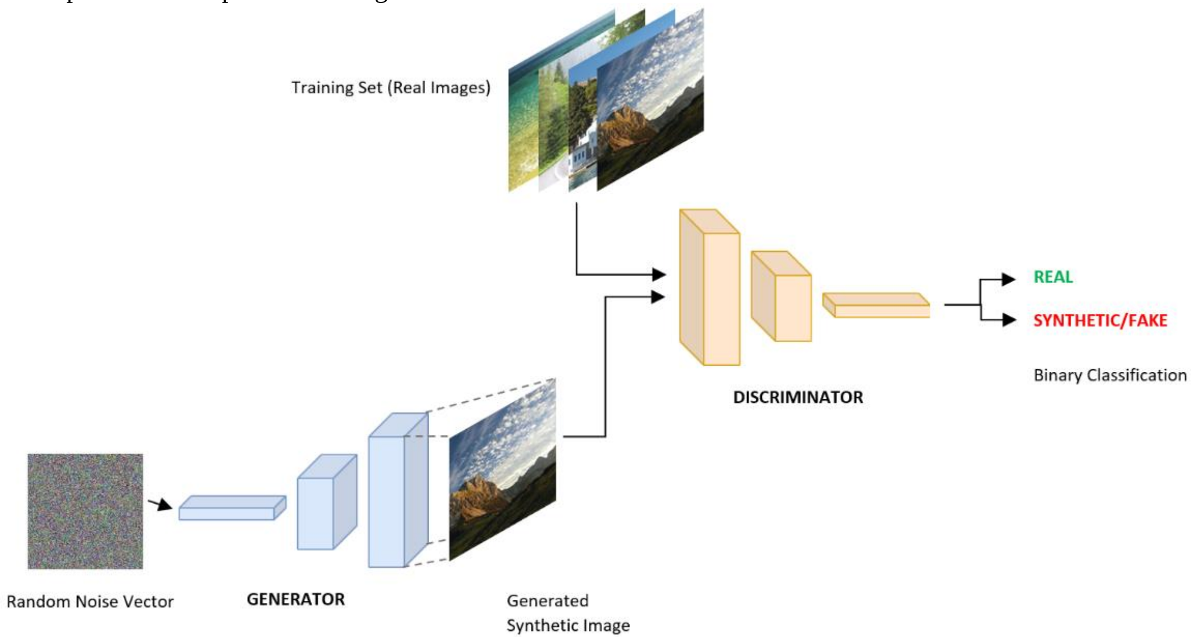

Fig. 4 Concept of Generative Adversarial Networks

The training of GANs is done by alternatively updating the parameters of generator and discriminator networks. The discriminator network takes d-dimensional inputs and a single output in (0,1) indicating that the basic function of a discriminator network is binary classification between real and fake samples. The (0,1) is a probability of the d-dimensional input being real. A value of 1 indicates that the samples are real and a value of 0 is an indication of synthetic samples. The output of the discriminator model is given by $D(\overline{X})$ when the input is $\overline{X}$. The goal of the discriminator is to correctly classify real images as 1 and synthetically generated images as 0. Hence the objective function for the discriminator denoted by J_D is denoted in equation 1 as,

$$\text{Maximise}_D\, J_D = \sum_{\bar{X} \in R_m} \log [D(\bar{X})] + \sum_{\bar{X} \in S_m} \log [1 - D(\bar{X})] \qquad (1)$$

The generator input is taken as the noise samples from the p-dimensional probability distribution and it is used to generate d-dimensional examples. Let R_m be the randomly sampled data taken from the real data, and S_m be the synthetic data being produced by the generator. Discriminator error is used to train the generator network to create synthetic samples. Firstly, a set of p dimensional data N_m is created for the noise samples $\{Z_1, \ldots Z_m\}$, and then generator is applied to the noise samples for creating the synthetic samples $\{G(Z_1), \ldots . G(Z_m)\}$[13].
This objective function will be maximised when the real examples will be labelled as 1 and the synthetic examples are labelled as 0.
The generator will only focus on the m generated synthetic examples, S_m as they are required to be labelled as real by the discriminator. The generator objective function J_G minimises the likelihood that discriminator will label the generated images as synthetic. Hence, the generator objective function is minimised when the discriminator will label the synthetic images as 1, i.e. real. The generator objective function is denoted in equation 2.

$$\text{Minimise}_G\, J_G = \sum_{\bar{X} \in S_m} \log [1 - D(\bar{X})] = \sum_{\bar{Z} \in N_m} \log [1 - D(G(\bar{Z}))] \qquad (2)$$

m samples of synthetic

examples

Alternatively, the objective function of the generator can be maximise log $[D(\bar{X})]$ for each $\bar{X}$ that belongs to S_m rather than to minimise log[$1 - D(\bar{X})$] and in the early iterations this objective function sometimes work more effectively.
Hence, the overall optimisation problem for both the discriminator and generator can be formulated as minimax game over J_D as presented in equation 3:

$$\text{Minimise}_G \text{Maximise}_D\, J_D \qquad (3)$$

For such an optimisation problem, the result is the saddle point of the problem.
For learning the parameters of the generator, stochastic gradient descent is used while for learning the parameters of the discriminator, stochastic gradient ascent is used. Gradient steps are updated between discriminator and generator alternatively. The discriminator uses k steps for one step of generator. Hence the gradient update steps are:

- K times repeated: Take a mini batch with a size of 2.m and take same amount of real and synthetic data. We can generate the synthetic examples by inputting the noise sample data to the generator while the real examples are taken from the real dataset. For maximising the likelihood that discriminator correctly label both the type of data stochastic gradient ascent is used for learning the parameters of the model. Backpropagation is performed for every update step for the mini batch of 2.m examples of real and synthetic data[13].
- Perform once: For this, the discriminator is hooked up with the generator at the end. For the parameters of the generator, stochastic gradient descent is performed to minimise the likelihood that the discriminator is able to classify correctly for the data. The gradient updates during the backpropagation are done by considering the parameters of the generative network. During the backpropagation, the gradients will be computed for both discriminative and generative networks but the parameters will be updated only for the generative network.

Generally, the value of k is taken to be small (less than 5). We can repeat this iterative process until it reaches the stage of Nash equilibrium. At this stage, the discriminator will not be able to classify correctly between both the synthetic and real data.
Factors to be taken care of while training:

- If we leave the updation of discriminator and only train generator for a longer period, then the generator will start producing samples which will be similar to each other. There will be very less difference between the generated samples. Hence to avoid this, the updation of both the discriminator and generator is done simultaneously.
- In the early iterations, it is possible that generator might give poor samples, resulting which $D(\bar{X})$ will be nearly 0. Hence loss function will be nearly equal to 0. This could result in very slow training of the parameters of the generator. To avoid this, we can maximise log $[D(\bar{X})]$ rather than minimising log[$1 - D(\bar{X})$] in the early iterations.

These two models are trained together so as to fool the discriminator model at least half the times, which means the generative model is producing possibly the similar images as the original dataset. If the discriminator is able to classify between the real and fake images, then no changes are made in the discriminator model parameters instead a sufficient amount of changes are made in the parameters of generator. Similarly, if the generator successfully fools the discriminator, then changes are made in the parameters of discriminator model.

IV. METHODOLOGY

A. The Dataset

The Landscape dataset[15] is used for this project. This dataset contains images of different landscapes such as buildings, mountains, streets, glaciers, trees, etc. It consists of two directories, one directory having colored images with the pixel values in the range [0,255] and the other directory with their corresponding images in grayscale format. Both the directories consist of 7129 images each. A sample of the images in both formats is presented in figure 5.

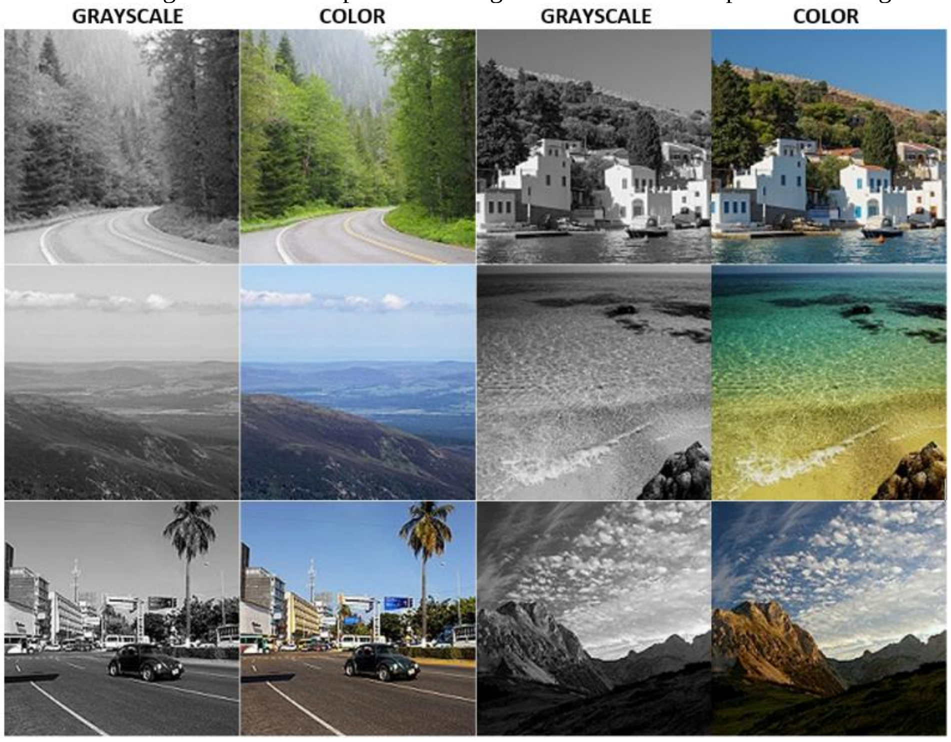

Fig. 5 Sample images from the dataset

B. Convolutional Auto-encoders

Three autoencoder models have been implemented on the dataset. The first implemented autoencoder model is named AE-RGB-1. The second autoencoder model is named AE-RGB-2 and the third model is named AE-LAB. The model architecture and implementation details have been discussed in the following sub-sections.

1) AE-RGB-1:

The first model implemented on the dataset is a 16 layered architecture. The images used for the implementation of the model have been resized to 160×160 pixels and scaled to a range of [-1,1]. The encoder part of the model consists of 8 convolution layers. AE-RGB-1 is implemented using Keras API[16]. The number of layers in the decoder of the model is 7. Upsampling layers of the constant size of 2×2 are used in the decoder of the model. The model architecture is summarized in table I.

TABLE I. AE-RGB-1 NETWORK ARCHITECTURE

Encoder			**Decoder**		
Layers	**Filters (Kernels)**	**Strides**	**Layers**	**Filters (Kernels)**	**Strides**
Conv	64 (3×3)	1×1	TransConv	256 (3×3)	1×1
Conv	64 (3×3)	1×1	TransConv	256 (3×3)	1×1
Conv	128 (3×3)	1×1	TransConv	128 (3×3)	1×1
Conv	128 (3×3)	1×1	TransConv	128 (3×3)	1×1
Conv	256 (3×3)	1×1	TransConv	64 (3×3)	1×1
Conv	256 (5×5)	1×1	TransConv	64 (3×3)	1×1

The hidden convolution layers of the model used the ReLU activation function. 'Same' padding is used in all the layers of AE-RGB-1. The use of 'same' padding with strides of 1 will retain the spatial dimensions of the images in the

subsequent layers. The activation function used in the output layer is sigmoid. The number of kernels in the output convolution layer is 3 which is used for predicting the three color channels of the RGB image. A constant kernel and stride is used in the decoder layers. The kernel and stride size in the encoder part of the model is variable in nature with two types of filter size. The total number of parameters of the AE-RGB-1 model is 2,953,955.
The number of images used for training AE-RGB-1 is 6000. The training set consists of 5000 images and 1000 are used to validate the training of the model. The objective loss function used for training the model is root means squared error (RMSE). The performance of the model is also evaluated on the based of colorization accuracy, mean absolute error(MAE) and mean squared error(MSE).

2) AE-RGB-2:
The second autoencoder model implemented on the dataset is a 16 layered architecture. The images used for the implementation of the model have been resized to 160×160 pixels and scaled to a range of [-1,1]. The encoder part of the model consists of 8 convolution layers. AE-RGB-2 is implemented using Keras API[16]. The number of layers in the decoder of the model is 7. Upsampling layers of constant size of 2×2 are used in the decoder of the model. The model architecture is summarized in table II.
TABLE II. AE-RGB-2 NETWORK ARCHITECTURE

Encoder			**Decoder**		
Layers	**Filters (Kernels)**	**Strides**	**Layers**	**Filters (Kernels)**	**Strides**
Conv	64 (3×3)	1×1	Upsamp		
Conv	64 (3×3)	2×2	Conv	32 (3×3)	1×1
Conv	128 (3×3)	1×1	Upsamp		
Conv	128 (3×3)	2×2	Conv	32 (3×3)	1×1
Conv	256 (3×3)	1×1	Upsamp		
Conv	256 (5×5)	1×1	Conv	16 (3×3)	1×1
Conv	256 (5×5)	2×2	Conv	3 (3×3)	1×1
Conv	64 (3×3)	1×1	Upsamp		

The hidden convolution layers of the model used ReLU activation function. 'Same' padding is used in all the layers of AE-RGB-2. The use of 'same' padding with strides of 1 will retain the spatial dimensions of the images in the subsequent layers. The activation function used in the output layer is tanh. Transpose convolution layer is used in the output layer with a kernel size of 3 for predicting the three color channels of the RGB image. A constant kernel and stride is used in the decoder layers and encoder layers alike. The kernel and stride size in the encoder part of the model is variable in nature with two types of filter size. The total number of parameters of AE-RGB-2 model is 2,879,491.
The number of images used for training AE-RGB-2 is 6000. The training set consists of 5000 images and 1000 are used to validate training of the model. The objective loss function used for training the model is root means squared error (RMSE). The performance of the model is also evaluated on the based of colorization accuracy, mean absolute error(MAE) and mean squared error(MSE).

3) AE-LAB:
The third model implemented on the dataset is a 16 layered architecture. The images used for the implementation of the model has been resized to 160×160 pixels and scaled to a range of [-1,1]. The encoder part of the model consists of 8 convolution layers. AE-LAB is implemented using [16]. Number of layers in the decoder of the model is 7. Upsampling layers of constant size of 2×2 is used in the decoder of the model. The model architecture is summarized in table III.
TABLE III. AE-LAB NETWORK ARCHITECTURE

Encoder			**Decoder**		
Layers	**Filters (Kernels)**	**Strides**	**Layers**	**Filters (Kernels)**	**Strides**
Conv	64 (3×3)	1×1	Upsamp		
Conv	64 (3×3)	2×2	Conv	32 (3×3)	1×1
Conv	128 (3×3)	1×1	Upsamp		
Conv	128 (3×3)	2×2	Conv	32 (3×3)	1×1
Conv	256 (3×3)	1×1	Upsamp		
Conv	256 (5×5)	1×1	Conv	16 (3×3)	1×1
Conv	128 (3×3)	2×2	Conv	2 (3×3)	1×1
Conv	64 (3×3)	1×1	Upsamp		

DOI: 10.17148/IARJSET.2022.9115

The hidden convolution layers of the model used the ReLU activation function. ‘Same’ padding is used in all the layers of AE-LAB. The use of ‘same’ padding with strides of 1 will retain the spatial dimensions of the images in the subsequent layers. The activation function used in the output layer is tanh. The number of kernels in the output convolution layer is 3 which is used for predicting the three color channels of the RGB image. A constant kernel and stride is used in the decoder layers. The kernel and stride size in the encoder part of the model is variable in nature with two types of filter size. The total number of parameters of AE-LAB model is 2,585,042. AE-LAB is used to convert greyscale images to color images in CIELAB color space. Therefore the output convolutional layer of the model consists of only two filters. The use of a 2 filter is used for prediction of a and b color channels of the images. The primary difference between AE-LAB and the previous two models is that the latter is used for predicting color in CIELAB color space and the first two models are used for predicting color in RGB color space.

C. ColorGAN

The generative adversarial network used for colouring grayscale images to RGB color space is implemented using Keras API. For the generator and discriminator models, we followed Deep Convolutional GANs (DCGAN) [17] guidelines and employed convolutional networks in both generator and discriminator architectures.

Generator Network: The generator network of the implemented ColorGAN uses U-Net architecture[18]. The generator network consists of 16 layers. The first part of the network is the contraction part and consists of convolutional layers with a different number of filters and a constant filter size of 4×4 along with a constant stride of 2×2. The part of the network is also called the encoder part. Apart from the first layer, batch normalization is applied to each layer of the encoder. Leaky ReLU activation function is applied after each convolutional layer.

The second part of the generator network is symmetric expanding part also known as the decoder. This part of the network is used for the reconstruction of the image in the specified color spaces. Transposed convolution layers are implemented in the part of the network with a varied number of filters in each layer and a constant filter size of 4×4 and a constant stride of 2×2. 50% dropout of weights is applied in the first three layers of the decoder part[19]. ReLU activation function is applied after each transposed convolution layer[20]. The number of layers and the number of filters in each layer is the same as the number of filters and the number of layers of the corresponding encoder part. This is the general property of U-Net architecture as proposed by[18].

Discriminator Network: The discriminator network uses simple convolutional layers. The number of convolutional used in the discriminator network is three with three different filters, i.e. 64,128 and 256 and a constant filter size of 4×4. ZeroPadding layer is implemented before another convolution layer with 512 filters and 4×4 filter size. Before the last convolution layer, another zeroPadding, batch normalization and Leaky ReLU[20] activation function is used[21]. The function of the discriminator is to classify real and synthetic images correctly. So the number of neurons in the output layer is 1. The U-Net architecture implemented in the generator network is presented in fig. 6.

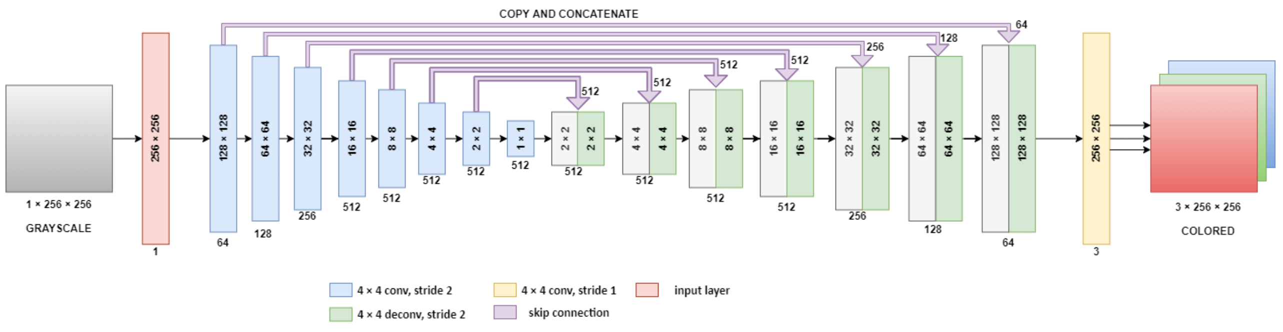

Fig. 6 The U-Net Architecture (256×256 input) of generator network

Strided convolutions are used instead of spatial pooling functions. This effectively allows the model to learn its own downsampling/upsampling rather than relying on a fixed downsampling/upsampling method[22]. This idea was proposed in [23]. Mean absolute error is used as a loss function for the generator network and binary cross entropy is used for the discriminator network[14]. The choice of binary cross entropy loss function is due to the fact that the function of the discriminator network is classification of real and synthetic images. Adam optimizer[24][25] with a learning rate of 0.0001 is used for training of both generator and discriminator network of the implemented COLORGAN.

ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 7.105 ≋ Vol. 9, Issue 1, January 2022

DOI: 10.17148/IARJSET.2022.9115

V. RESULTS AND DISCUSSIONS

A. AE-RGB-1

The number of images used for training AE-RGB-1 is 7000. The training set consists of 6000 images and 1000 are used for validation training of the model. The objective loss function used for training the model is root means squared error (RMSE). The performance of the convolutional auto encoder model is also evaluated on the basis of colorization accuracy, mean absolute error(MAE) and mean squared error(MSE). The first convolutional autoencoder model is trained till 35 epochs. The model achieved a training accuracy of 62.97% and a validation accuracy of 57.82%. The root mean squared error of the model on training and test set is 0.0854 and 0.0944 respectively. All the evaluation metrics of the model are summarized in table IV. The training history of RMSE, MSE, MAE and colorization accuracy of AE-RGB-1 is shown in figure 7. Fig 7. Shows that the model does not overfit during 35 epochs of training. The training and validation metrics of the model are converging at 35th epoch. The generated test images by the model in RGB color space have been presented in fig 7.

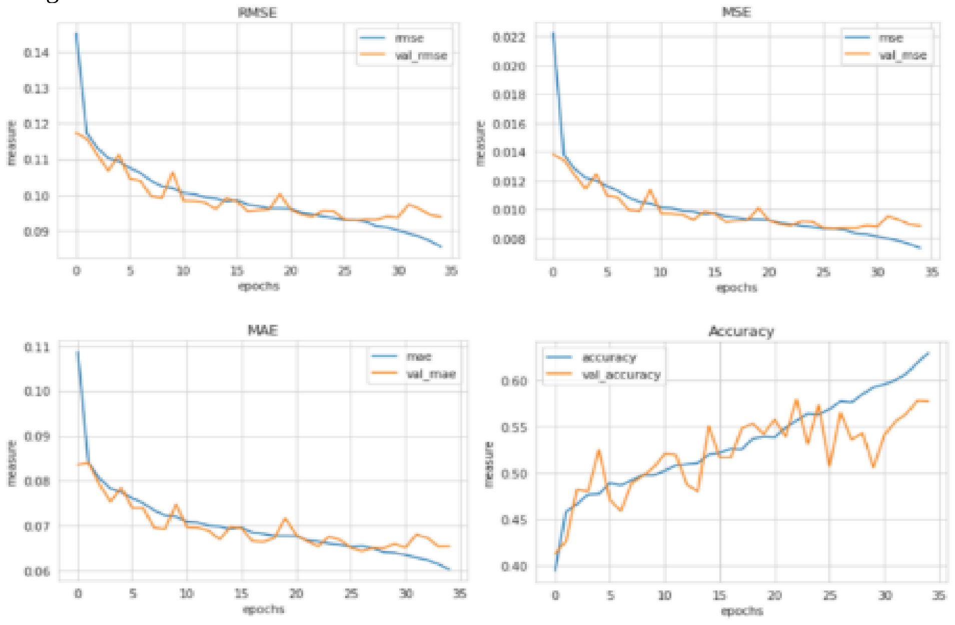

Fig. 7 History of RMSE, MSE, MAE and colorization accuracy of AE-RGB-1 for 35 epochs

B. AE-RGB-2

The number of images used for training AE-RGB-2 is 6000. The training set consists of 5000 images and 1000 are used for validation training of the model. The objective loss function used for training the model is root means squared error (RMSE). The performance of the convolutional auto encoder model is also evaluated on the basis of colorization accuracy, mean absolute error (MAE) and mean squared error (MSE). The first convolutional autoencoder model is trained till 35 epochs. The model achieved a training accuracy of 52.48% and a validation accuracy of 52.20%. The root mean squared error of the model on training and test set is 0.0725 and 0.0817 respectively. All the evaluation metrics of the model are summarized in table IV. The training history of RMSE, MSE, MAE and colorization accuracy of AE-RGB-1 is shown in figure 8. Fig 8. Shows that the model does not overfit during 35 epochs of training. The training and validation metrics of the model are converging at 35th epoch. The generated test images by the model in RGB color space have been presented in fig 8.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 7.105 ⁙ Vol. 9, Issue 1, January 2022

DOI: 10.17148/IARJSET.2022.9115

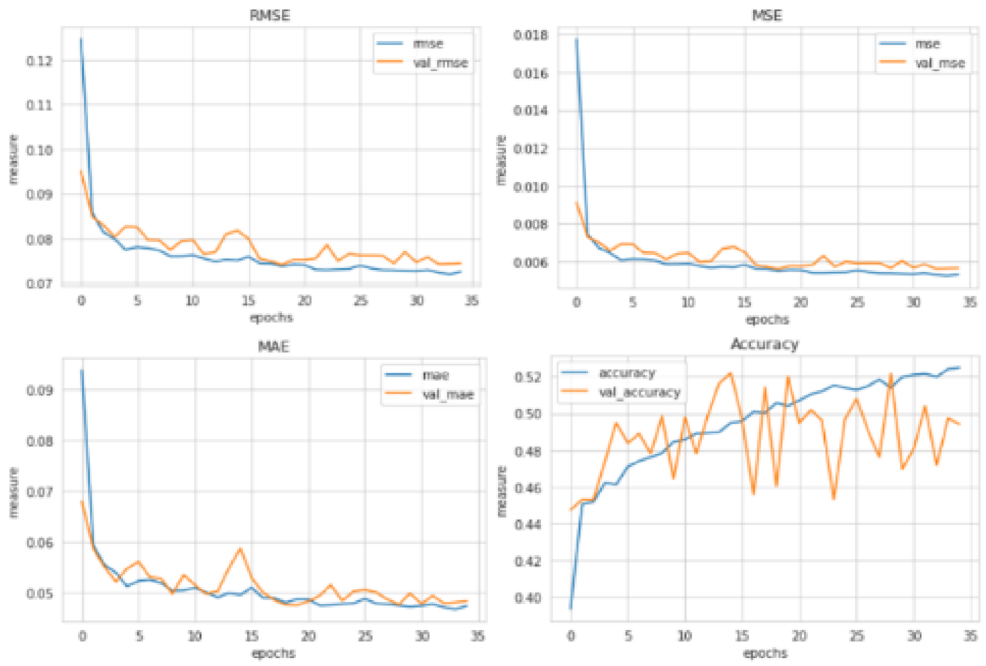

Fig. 8 History of RMSE, MSE, MAE and colorization accuracy of AE-RGB-2 for 35 epochs

C. AE-LAB

The number of images used for training AE-LAB is 7000. The training set consists of 6000 images and 1000 are used for validation training of the model. The objective loss function used for training the model is root means squared error (RMSE). The performance of the convolutional auto encoder model is also evaluated on the basis of colorization accuracy, mean absolute error (MAE) and mean squared error (MSE). The first convolutional autoencoder model is trained till 20 epochs. The model achieved a training accuracy of 68.00% and a validation accuracy of 68.57%. The root mean squared error of the model on training and test set is 0.0715 and 0.0753 respectively. All the evaluation metrics of the model are summarized in table IV. The training history of RMSE, MSE, MAE and colorization accuracy of AE-LAB is shown in figure 9. Fig 9. Shows that the model does not overfit during 20 epochs of training. The training and validation metrics of the model are converging at 20th epoch. The generated test images by the model in CIELAB color space have been presented in fig 9.

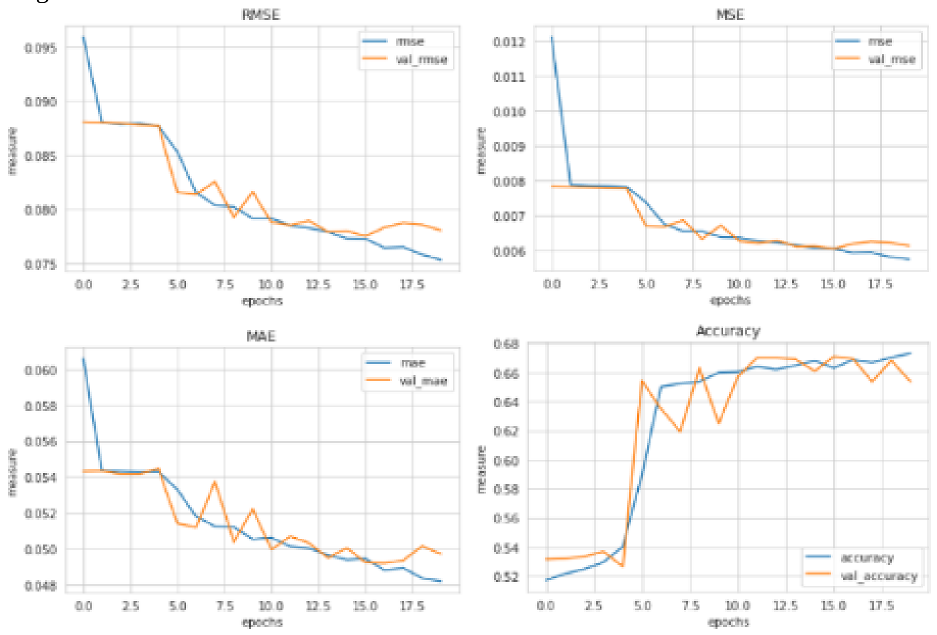

Fig. 9 History of RMSE, MSE, MAE and colorization accuracy of AE-LAB for 20 epochs

TABLE IV. COMPARISON OF RMSE, MSE, MAE AND COLORIZATION ACCURACY FOR THE AUTOENCODER MODELS

	RMSE	MSE	MAE	Colorization Accuracy (%)
AE-RGB-1	0.0944	0.0090	0.0654	57.82
AE-RGB-2	0.0817	0.0068	0.0587	52.20
AE-LAB	0.0753	0.0058	0.0477	68.57

D. ColorGAN

The number of images used for training the ColorGAN is 2,200. The number of images in the training set is 2,000 and the test set is 200. The images used for the implementation of the model has been resized to 256×256 pixels. The ColorGAN is trained till 100 epochs with a batch size of 64 for the training set and a batch size of 8 for the test set. Mean absolute error (MAE) is for training the generator network and binary cross entropy is used for training the discriminator network. The generator loss (MAE) after 100 epochs of training is 0.006477. The binary cross entropy loss or discriminator loss after 100 epochs of training is 0.003642. The generator and discriminator loss for 100 epochs is visualized in figure 10.

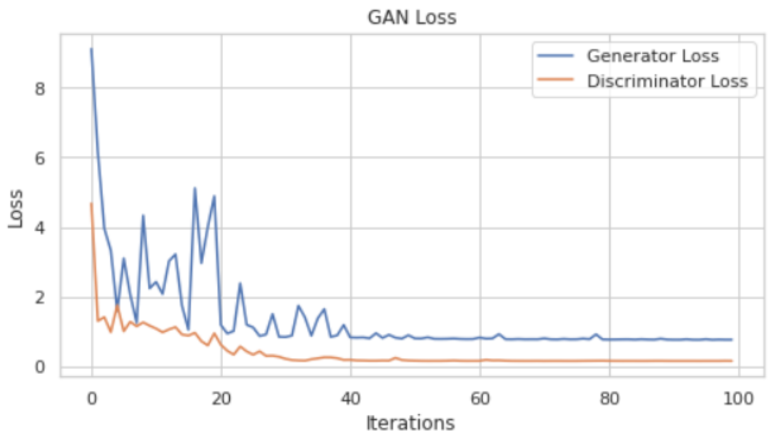

Fig. 10 History of generator and discriminator loss of ColorGAN for 100 epochs

Five random coloured images of the dataset by the autoencoder models and ColorGAN discussed above is visualized in figure 11. The coloured images of each network is compared with the ground truth image. The reconstructed images by AE-RGB-1 have lost its sharpness and the output coloured images are blurry in nature. The reconstruction accuracy of colors in the images by AE-RGB-2 is less than the other two autoencoders. This can be attributed by the fact that the coloured images in fig. 11 by AE-RGB-2 is not as good as the images by the other two autoencoders. The coloured images by AE-LAB is better as compared to the other two encoders. The pixels of the images are retained preventing the images to be blurry. The higher colorization accuracy of AE-LAB is the proof of better results in fig 11. The images coloured by ColorGAN has the best results among all the models. The colors of the images produced by ColorGAN is almost identical to that of the ground truth images. This can be attributed by the generator and discriminator loss of the implemented ColorGAN. The pixels of the reconstructed images by ColorGAN are almost as sharp as that of the ground truth images. This comparison among different models confirms that the result of ColorGAN is the best.

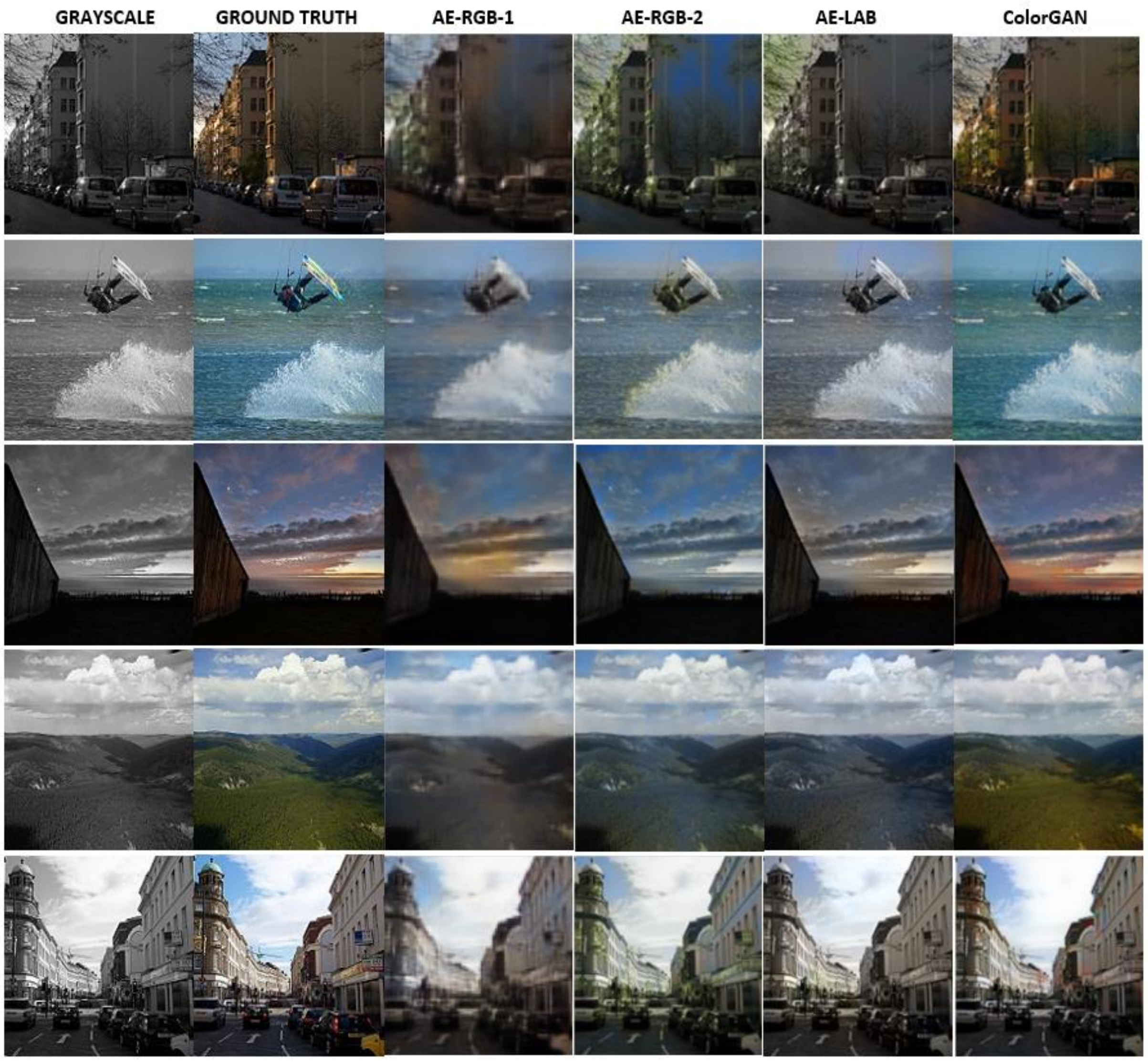

Fig. 11 Comparison of colorization performance of AE-RGB-1, AE-RGB-2, AE-LAB and ColorGAN models with ground truth image

VI. CONCLUSION

Three autoencoders models and one ColorGAN model on a landscape image dataset to colorize black-and-white images to color images are implemented. The two color space chosen for colorization is RGB and CIELAB. The AE-LAB autoencoder model performed better than AE-RGB-1 and AE-RGB-2 autoencoder models in terms of RMSE, MSE and MAE with values 0.0753, 0.0058, 0.0477 respectively. The performance of other two autoencoder models is at par. The colorization accuracy of AE-LAB is better than AE-RGB-1 and AE-RGB-2 with an accuracy of 68.57%. The performance of implemented ColorGAN model is best among all the other models implemented. The colorized images presented above attribute to the better performance of ColorGAN model than autoencoder models. The MAE of ColorGAN model is also better than those of the autoencoder models with a value of 0.006477. The results prove that the implemented ColorGAN's performance to colorize the grayscale images in better than those of the implemented autoencoder models.
The use of deep learning to colorize black-and-white images has a huge potential. This can be attributed by the results shown is this paper. The findings provided in this this paper will help to future researchers to develop deep learning models for image coloration. The autoencoder models as well as ColorGAN model implemented in this project will help to colorize any landscape image with a superior accuracy.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 7.105 ⋇ Vol. 9, Issue 1, January 2022

DOI: 10.17148/IARJSET.2022.9115

REFERENCES

[1] Zhang, Richard, et al. "Real-time user-guided image colorization with learned deep priors." arXiv preprint arXiv:1705.02999 (2017).
[2] Joshi, Madhab & Nkenyereye, Lewis & Joshi, Gyanendra Prasad & Islam, S. M. Riazul & Abdullah-Al-Wadud, Mohammad & Shrestha, Surendra. (2020). Auto-Colorization of Historical Images Using Deep Convolutional Neural Networks. Mathematics. 8. 2258. 10.3390/math8122258.
[3] p, Ajay kalyan and R, Puviarasi and Ramalaingam, Mritha, Image Colorization Using Convolutional Neural Networks (August 23, 2019). Proceedings of International Conference on Recent Trends in Computing, Communication & Networking Technologies (ICRTCCNT) 2019
[4] Pandey A., Sahay R., Jayavarthini C. (2020). Automatic Image Colorization using Deep Learning. International Journal of Recent Technology and Engineering, Volume-8 Issue-6, 1592-1595.
[5] Baldassarre, F., Morín, D.G., & Rodés-Guirao, L. (2017). Deep Koalarization: Image Colorization using CNNs and Inception-ResNet-v2. ArXiv, abs/1712.03400.
[6] Kotala S., Tirumalasetti S., Nemitha V., Munigala S. (2019). Automatic Colorization of Black and White Images using Deep Learning. IJCSN Journal Volume 8, Issue 2, 125-131.
[7] Nguyen, Tung & Mori, Kazuki & Thawonmas, Ruck. (2016). Image Colorization Using a Deep Convolutional Neural Network
[8] Charpiat, Guillaume & Bezrukov, Ilja & Hofmann, Matthias & Altun, Yasemin & Schölkopf, Bernhard & Lukac, R.. (2011). Machine Learning Methods for Automatic Image Colorization. Computational Photography: Methods and Applications, 395-418 (2010).
[9] Anwar, S., Tahir, M., Li, C., Mian, A.S., Khan, F.S., & Muzaffar, A.W. (2020). Image Colorization: A Survey and Dataset. ArXiv, abs/2008.10774.
[10] Hwang, J. (2016). Image Colorization with Deep Convolutional Neural Networks.
[11] R. Dahl. Automatic colorization. http://tinyclouds.org/colorize/, 2016.
[12] Koo, S. (2016). Automatic Colorization with Deep Convolutional Generative Adversarial Networks.
[13] Aggarwal, C. C. (2018). Neural networks and deep learning. Springer, 10, 978-3.
[14] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., & Bengio, Y. (2014). Generative Adversarial Nets. In Advances in neural information processing systems (pp. 2672–2680).
[15] https://www.kaggle.com/theblackmamba31/landscape-image-colorization
[16] https://keras.io/api/models/model/
[17] Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. CoRR, abs/1511.06434.
[18] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. MICCAI.
[19] Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res., 15, 1929-1958.
[20] Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical Evaluation of Rectified Activations in Convolutional Network. ArXiv, abs/1505.00853.
[21] Isola, P., Zhu, J., Zhou, T., & Efros, A.A. (2017). Image-to-Image Translation with Conditional Adversarial Networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 5967-5976.
[22] Nazeri, K., Ng, E., & Ebrahimi, M. (2018). Image Colorization Using Generative Adversarial Networks. AMDO.
[23] Springenberg, J.T., Dosovitskiy, A., Brox, T., & Riedmiller, M.A. (2015). Striving for Simplicity: The All Convolutional Net. CoRR, abs/1412.6806.
[24] Kingma, D.P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. CoRR, abs/1412.6980.
[25] https://keras.io/api/optimizers/adam/