Image Colorization

A PROJECT REPORT

Submitted by

Jainish Pandya (19CP007)
Dip Patel (19CP008)
Arshil Vahora (19CP014)

In partial fulfillment for the award of the degree of
B. TECH. in COMPUTER ENGINEERING

4CP31: PROJECT-I

BIRLA VISHVAKARMA MAHAVIDYALAYA
(ENGINEERING COLLEGE)
(An Autonomous Institution)
VALLABH VIDYANAGAR

Affiliated to

GUJARAT TECHNOLOGICAL UNIVERSITY, AHMEDABAD

Academic Year: 2022 – 2023

BVM ENGINEERING COLLEGE, VALLABH VIDYANAGAR-388120

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our project guides Prof. M. I. Hasan and Dr. N. M. Patel for their guidance and support throughout the project. Project ideas tips from both the guide that what we can do further to the project had provided us great motivation. Their expertise and invaluable advice have been instrumental in the successful completion of this project. We are motivated to improve our presentation skills by their advices. Their insights and suggestions were greatly appreciated and helped to improve the quality of the project and this report. We are also motivated to write a research paper on the project which we will definitely do in the future by our mentors.

We would also like to thank our department for providing and encouraging us to use the GPU due to which we had saved plenty of time to run the Generative Models. Without it the completion of the project was nearly impossible.

Finally, we would like to thank our friends for their encouragement and support during the course of this project.

Abstract

Image colorization is a technique used to add color to black and white images. In this project we present a novel approach to image colorization using generative adversarial networks(GANs). Our model consists of two networks a generator network and a discriminator network, trained to collaboratively colorize the grayscale images. The generator network takes in grayscale images and outputs the corresponding colorized images, while the discriminator network aims to distinguish between real and generated colorized images. By training a GAN model on a dataset of grayscale images and their corresponding colored versions, we are able to generate realistic colorizations for grayscale images. Our results show that the generated colors are not only visually appealing, but also semantically consistent with the original grayscale images. We further demonstrate the potential of our approach by applying it to the variety of image types like eye retina, facial images, landscapes. We also have try to colorized the video by taking the images from every frame. It can be used to colorized the old black and white movies or sports matches etc. An android application which converts black and white retina image to color is developed. It can be useful to the medical field.

1. Introduction

The image colorization project is a technique used to add color to black and white images. This can be done manually by an artist, but there area also algorithms and techniques that can automatically colorize images. The goal of the project is to add color to old, black and white images, bringing them to life and giving them a more modern and vibrant appearance. This project is chosen by us to implement and better, deep understand the state of art GANs models and other deep learning models like neural style transfer, Autoencoders etc.

GANs are a type of machine learning model that are designed to be able to generate new data that is similar to a given training set. In the context of image colorization, GANs are trained on a large dataset of color images, and then can be used to add color to black and white images in a way that is consistent with the colors that appear in the training set. This report will cover the background and motivation for using GANs for image colorization, as well as a detailed description of the technical approach used in the project. It will also include results and analysis of the effectiveness of the colorization method, as well as potential applications and future work in this area.

There are many different GAN models that have been developed for the task of image colorization. The specific model used in a given project may depend on the specific goals and requirement of the project as well as the availability of the training data and computational resources.

Some examples of these models include:

- Deoldify: a GAN model that uses a U-NET architecture and incorporates a perceptual loss function to improve the quality of the generated colors
- HDGAN: a GAN model that uses a hierarchical structure to generate high-resolution color images
- Pix2Pix: a GAN model that uses a U-Net architecture and is trained to generate color images from input images in a supervised manner
- ColorGAN: a Gan model that uses a combination of adversarial and reconstruction loss to colorize images

An android application which can convert the given black and white image to the colored image also has been developed mainly for the eye retina colorization which further can be extended to the any type of the image.

We have also tried a concept known as neural style transfer which can transfer the style of one image to another image and whether it can transfer the color as style or not.

2. Related Work and Background

2.1 Requirement Analysis

Image colorization is useful for variety of application like old or damaged photographs or videos, adding color to medical images for improved diagnosis.

Images and Videos colorization is useful for enhancing the visual appeal of your pictures and make them more engaging and informative. Adding colors to videos make it more accessible and appealing to modern audience, and can preserves important culture and historical effects. Another benefit of image and video colorization is that they can improve the contrast and clarity of the media. By adding color, it can be easier to distinguish between different objects and surfaces in the images and videos, which can aid in the interpretation and analysis of the media.

Image colorization can be a useful tool in the medical field, particularly in the analysis and diagnosis of medical images. In many cases, medical images are of black and white modalities, such as X-rays or CT scans. These can be difficult to interpret, especially when it comes to identifying different tissue types or structures within the body.

By adding color to these images, it can be easier to differentiate between different structures, and to identify abnormal behavior using different color. This can help to make more accurate and precise diagnoses, and can ultimately lead to better patient care.

Additionally, colorized medical images can be more visually appealing and engaging, which can help to improve patient communication and education. For example, colorized images can be used to show patients what is happening inside their bodies, or to help explain medical procedures or treatments in a more accessible and understandable way. Overall, the use of image colorization in the medical field can be a valuable tool for improving the accuracy and effectiveness of medical diagnosis and treatment.

Application Requirements:

- Ability to upload gray-scale image and get colorized image with user friendly interface.
- To download colorized image.

2.2 Literature Survey:

One of the first image colorization techniques was to manually add colors to gray scale image. Though it gives very accurate output it is very time consuming. Another approach to image colorization is the use of machine learning algorithms. These algorithms can be trained on large datasets of color and grayscale images, and can then be used to automatically colorize new

grayscale images. This method has the potential to produce more accurate and detailed results than colorization software, but it requires a large amount of data and computational resources.

Colorization of a image is challenging due to varying conditions of imaging that need to be dealt via a single algorithm. The problem is also severely ill-posed as two out of the three image dimensions are missing; although the scene semantics may be helpful in many cases, for example, grass is usually green, clouds are usually white, and the sky is blue. However, such semantic priors are highly uncommon for many man-made and natural objects e.g. shirts, cars, flowers etc. Moreover, the colorization problem also inherits the typical challenges of image enhancement, such as changes in illumination, variations in viewpoints, and occlusions.

As we know that One of the key challenges in image colorization is the high degree of ambiguity in determining the correct colours for an image. This is because an image may have multiple valid colorizations, and there is no objective way to determine which one is "correct". To address this challenge, many automatic colorization methods have been developed that use machine learning algorithms to learn patterns in colorized images and apply these patterns to grayscale images.

Domonkos Varga[1] proposed the idea of automatic coloring of cartoon images, since they are very different from natural images, they pose a difficulty as their colors depend on artist to artist. So, the data-set was specifically trained for cartoon images, about 100000 images, 70% of which were used in training and rest for validation. But unfortunately, the color uncertainty in cartoons is much higher than in natural images and evaluation is subjective and slow.

Shweta Salve [2] proposed another similar approach, employing the use of Google's image classifier, Inception ResNet V2. The system model is divided into 4 parts, Encoder, Feature extractor, Fusion layer and Decoder. The system is able to produce acceptable outputs, given enough resources, CPU, Memory, and large data-set. This is mainly proof of concept implementation.

Yu Chen [3] proposed a approach to mainly address the problem of coloring Chinese films from past time. They used existing data-set with their data-set of Chinese images, fine- tuning the overall model. The network makes use of multi- scale convolution kernels, combining low and middle features extracted from VGG-16.

V.K. Putri [4] proposed a method to convert plain sketches into colorful images. It uses sketch inversion model and color prediction in CIELab color space. This approach is able to handle hand drawn sketches including various geometric transformations. The limitation found was that, data-set is very limited but it works well for uncontrolled conditions.

Richard Zhang [5] has proposed a optimized solution by using huge data-set and single feed-forward pass in CNN. Their main focus. They used human subjects to test the results and were able to fool 32% of them. can have various number of neurons. The various attempts used various

architectures. In some papers, generally number of neurons is same as the dimension of the feature descriptor extracted from each pixel coordinates in a gray-scale image.

2.3 Technologies:

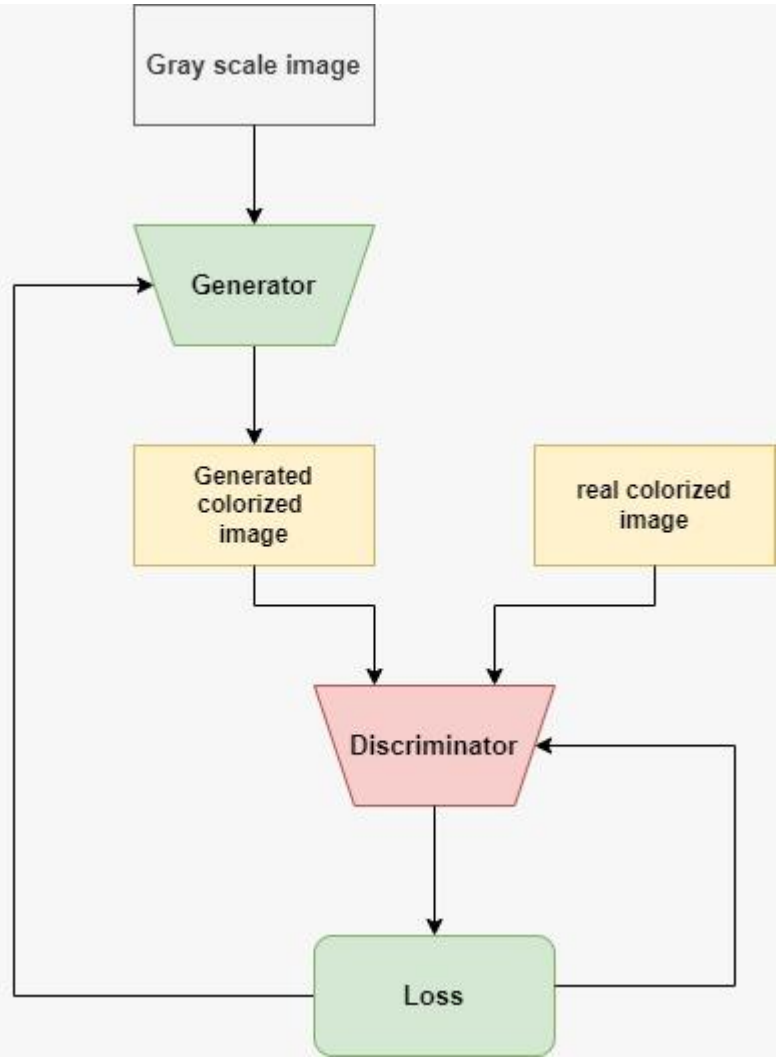
1) **PyTorch:** It is open-source machine learning framework. It is widely used for deep learning and other application. PyTorch is built on the concept of tensor, which are dimensional arrays that can be used to represent data in variety of formats. This allows PyTorch to handle a wide range of data types and to support operations such as numerical computation, machine learning, computer vision.

2) **Android Development**: It is used to develop android application for mobile devices. Through this application we can use SQLite data for managing relational database and also used for communication with server.

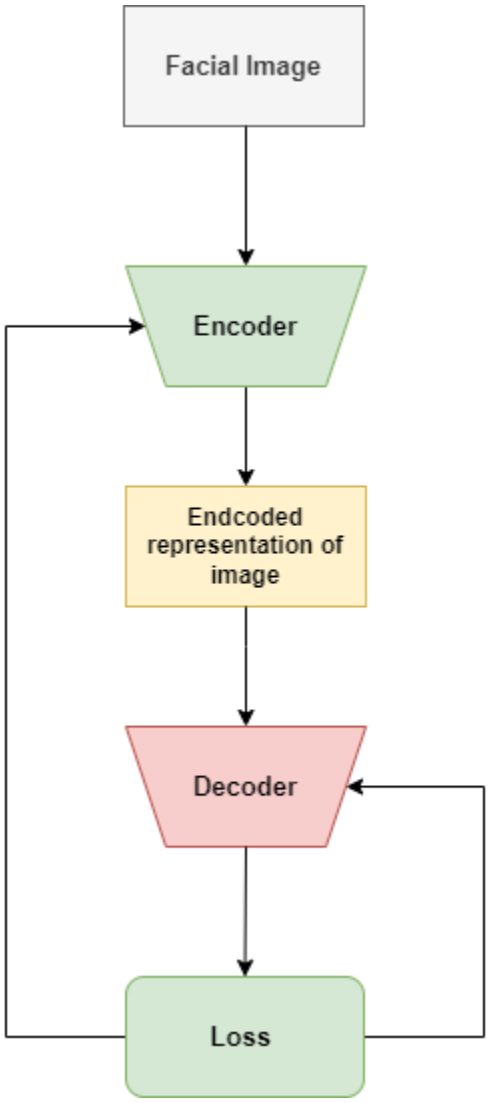
3) **Flask**: Flask can be used as a web server to host and run web applications. When used in this way, Flask provides the tools and features necessary to handle requests and responses from web browsers, and to serve web pages, JSON data, and other content to users. Flask can be run on a local development server for testing and debugging, or it can be deployed to a production server for use by users.

3. Modeling and Design

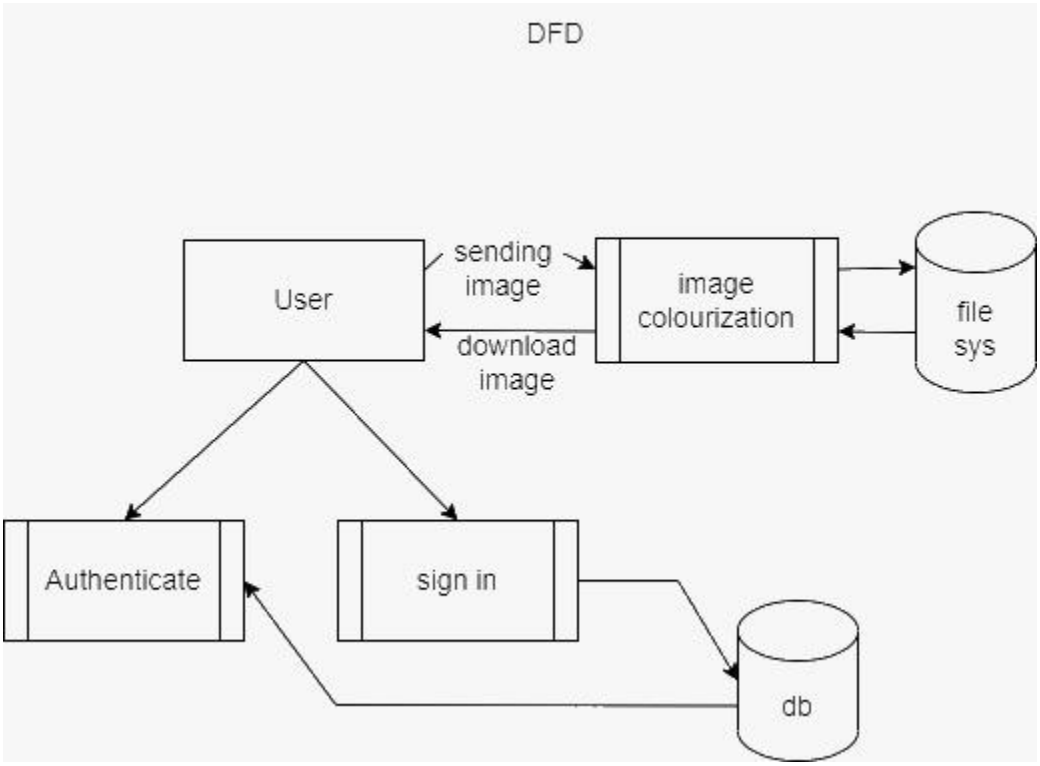
3.1. Flowchart of GAN model



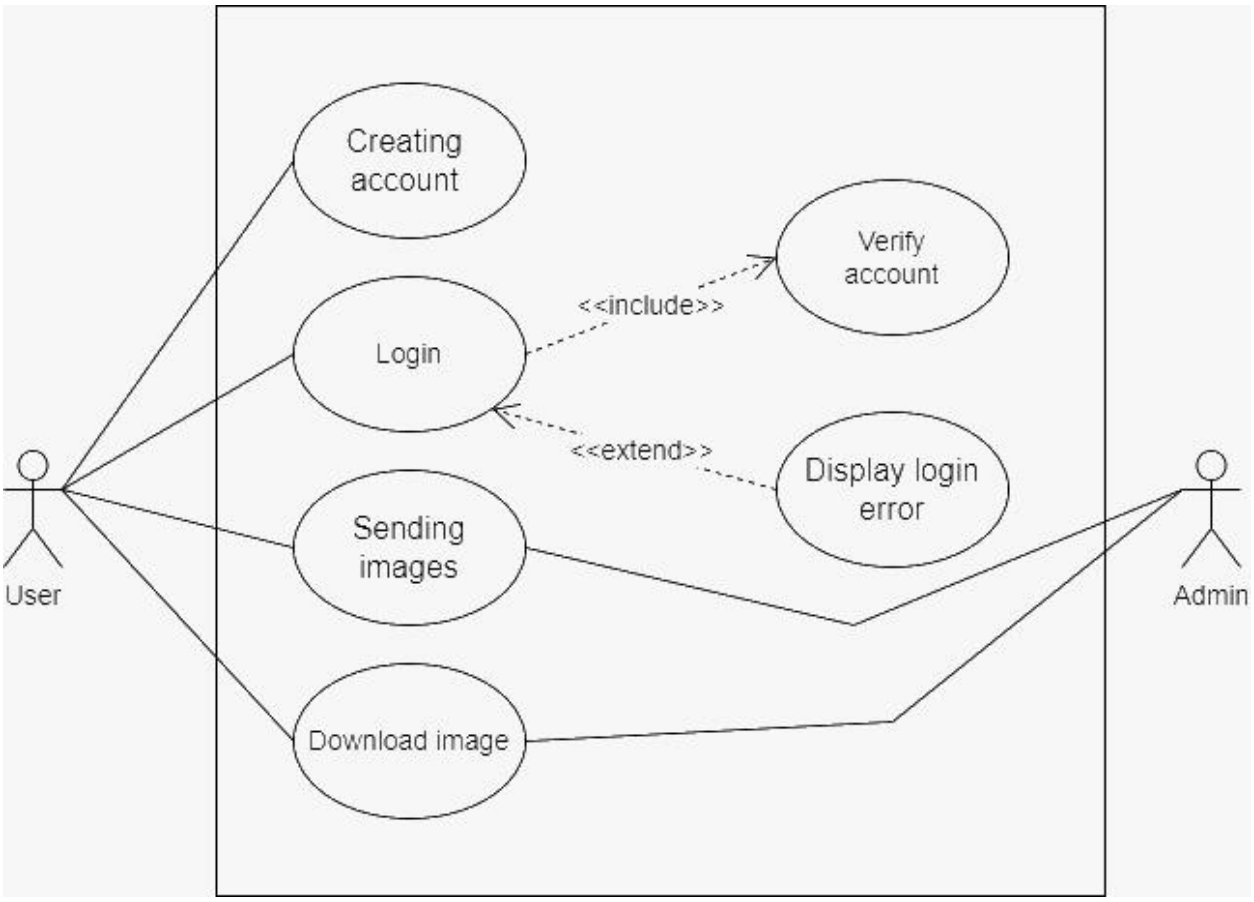
3.2. Flowchart for the Auto Encoder



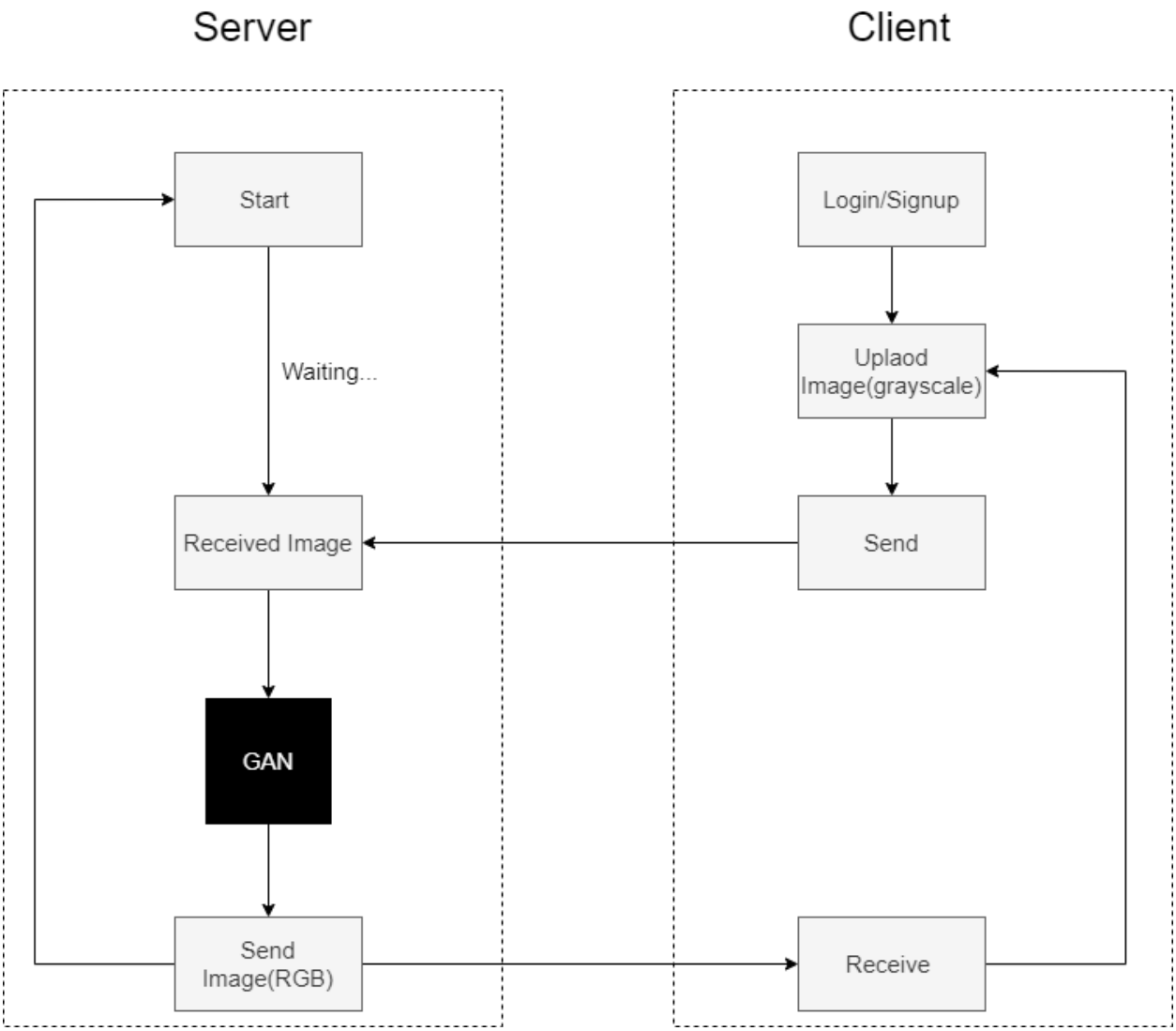
3.3. DFD diagram of our android application



3.4. Usecase diagram of our android application



3.5. Flowchart for android application



4. Implementation

4.1 Generative Adversarial Network

We selected project of image colorization using GAN(generative adversarial neural-network). As we all know that GAN has two component 1st is generator and 2nd is discriminator. Generator try to generate instances from given data distribution so in simple word it tries to mimic the data distribution of given dataset as show in (fig 1), now here data set can be of any type like: stack of images, collection of music or spread sheet ,however generation of data through spread sheet doesn't make any sense thus in most case it can be either images, video or mp4s. On the other hand, discriminator tries to discriminate real images and fake image (one that are generated by generator).Output of discriminator is between 0 and 1 and we interpret output of discriminator as probability of given image is real. In simple word it means probability near 1 means more real while near 0 means fake.

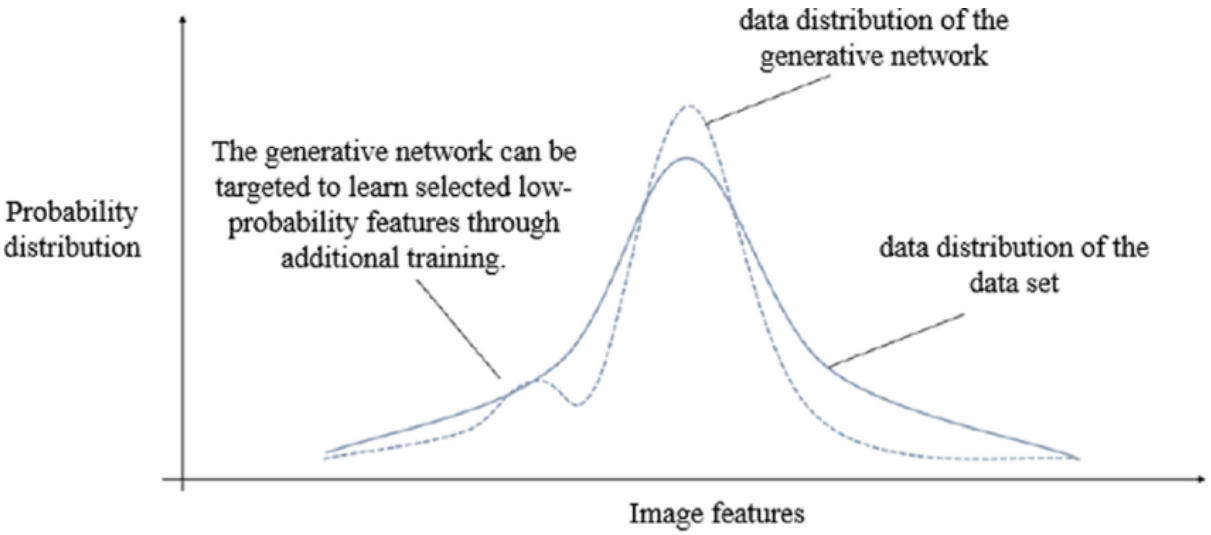


Figure 4.1

Here in our case, we have to generate colored images from grey scale images. For that we also here require generator and discriminator. In that generator will take grey image as input and produce colored image as output.

To generate images from grey scale image we need to have both generator and discriminator model and once we have both the model then we can start learning and after learning we can throw away discriminator model and store only generator model because that what we need as end product.

We decided to uses UNET as generator model in our project because of (Phillip Isola, 2018). But directly jump into generation of colored images is required much experience and we haven't worked on GAN yet. For each member of our group, it is new that we haven't played with so we thought it would be better to start with basics and slowly make progress toward our project. As we all know in programming if we want to learn new language the first program that we going to write is 'hello-world' in same manner if we want to learn GAN, we should start by generating 0-10 using MNIST dataset.

Without having any knowledge about architecture of generator we made simple generator and try to generate images. Once learning was completed, we saw result and we were completely socked. Because it is nothing but random noise and at that time, we realize that its not going to be easy. After two or three iteration of model correction we still didn't get the result, we want so we thought now its time that we should start read some of the research paper on GAN and get some idea about how its actually worked and some implementation stuff. During searching and reading we came across paper on DC-GAN. DC-GAN uses unique architecture for generator and also listed some of the implementation stuff. So we followed it doing rectification for two or three time and got the result that we want.

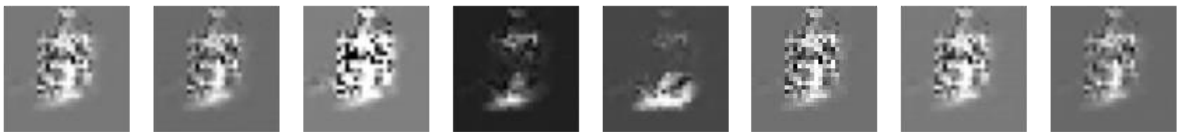


Figure 4.1 Result of initial iteration

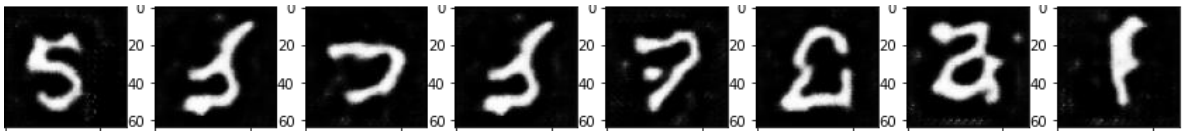


Figure 4.2 Result after improving initial model

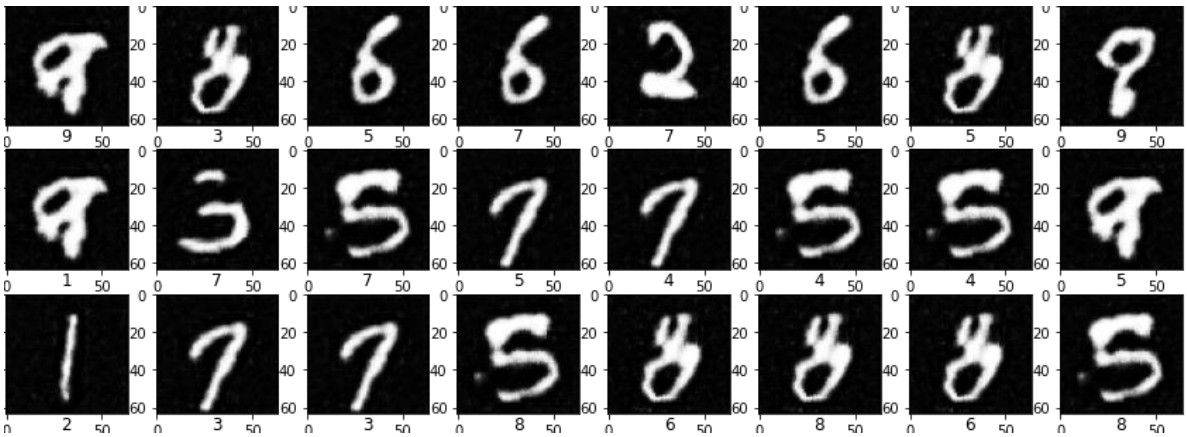


Figure 4.2 Result after improving initial model

4.2 Image Colorization

Now we are ready to move forward but one hurdle still in our way that was working with images that have more than one channel. Till now we our working with grey scale MNIST image but now we have to breck this barrier and apart from that we still know very little about pytorch framework. I think we barely scratch the surface of it so we spent some of our time to understand frame work better by reading documentation as well as some of the YouTube videos. During that time, we got to know about various stuff like how to create our custom dataset, how to load images from directory and also some nice implementation details for custom models(That we will use when we have to generate our own model for our generator).

Now after getting knowledge of pytorch, we embark our journey by creating custom UNET model. There is nothing sacrosanct about UNET model, we can use other model as well but research shows that we can get better result by using UNET so that is the reason we opt UNET as our generator model.

I think now it is better time to make deeper dive into some the details of generator and our data set.

First let me explain about our data set.

Name of dataset : MIRFLICKR25k

Basically, this data contains 25k images, grey scale and corresponding colored one. But catch over here is that instead of straight away giving us RGB image they(author of dataset) gave us LAB representation of image. So at that time we come to know about existence of such representation. After knowing the fact that they gave us LAB representation now we all were curious to know why they gave us such representation. And after doing research of 2 to 3 hours we got to know that in LAB representation we only need to produce 2 channels while in case of RGB we need to produce 3 channels so by using LAB we can greatly reduce the work need to be carried out. In below image we show difference LAB and RGB.

Figure 4.4 RGB vs LAB

We clearly see from picture that it is quite tedious to product all three channels of RGB while in case of LAB it is quite easy because first channel is grey that we already have and apart from this we still need to produce two which are comparatively easy to produce. So that is the reason why author of dataset gave us LAB representation of dataset rather than giving RGB straight away. This thing made our task much easier and now we are very close to create our own model.

So, after our dataset is ready now it was time to create Generator model. As we describe earlier, we use UNET architecture for our model. Skip-connection that are present in UNET model will hold

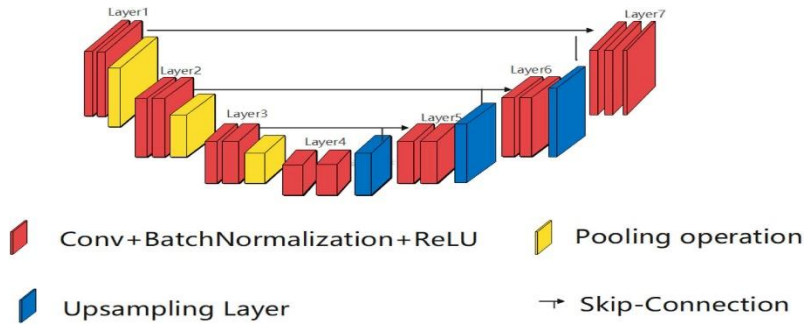


Figure 4.5 UNET

4.2.1 Information about structure of image.

Now let's talk about discriminator. Discriminator task in relatively easy it has to assign probability to given image while discrimination has to generate image so it is relatively harder than discriminator's task. Now according to paper (Phillip Isola, 2018) they suggested in that patch GAN will brings good result in compare with simple discriminator. So, lets understand first what we mean by patch here.

In simple discriminator what we will do is that it will simply output real number between 0 to 1 that represent truthiness of given image. But in Patch GAN, discriminator instead of giving single real number between 0 to 1 it will give matrix of size n*n, in which each item contains real number between 0 to 1 that shows truthiness probability of particular patch cover by that item. According to (Phillip Isola, 2018) this will bring good result.

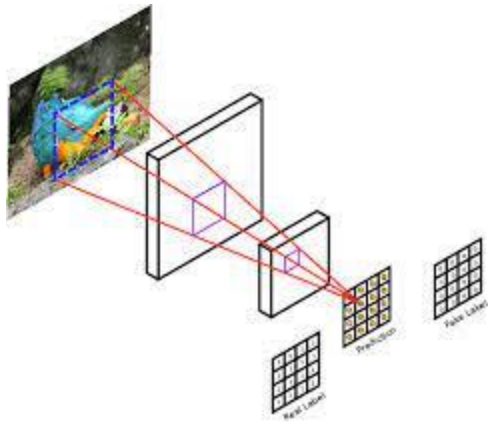


Figure 4.6 Patch GAN

Once discriminator and generator is ready now it is time to train our model. Here for training, we use GPU for faster processing because we have to process 25K images. But instead of training our model directly on 25k images we rather choose to train it on fraction of images 10k reason behind this choice is simple, if there is any error (logical) in our model than it could be easily detected in less time. So in this way we rectify our model two to three time and then after train our model on whole image dataset.

On big hurdle that we face while training our model was time. To train our model on 25k image will take tremendous amount of time more than 45min to complete one epoch. And in case of GAN, we never know minimum number of epochs will require to get descent output. So, for that reason we often train our model on 5 epochs and then save the weights of our models onto

disk.[Fact: size of generator weights is **around 1.6GB** while size of discriminator weights is **around 1.6MB** so you can see clearly difference how hard generator to be trained]

So next time if we want another five epochs then we load previous weights from disk and then after start training and once training was completed, we again store weight of our model on disk. And in this way, we managed to train our model.

Once the training was completed, we generate image from generator for given grey scale input and then we plot that image using matplotlib on window. Output of generated image was show below.

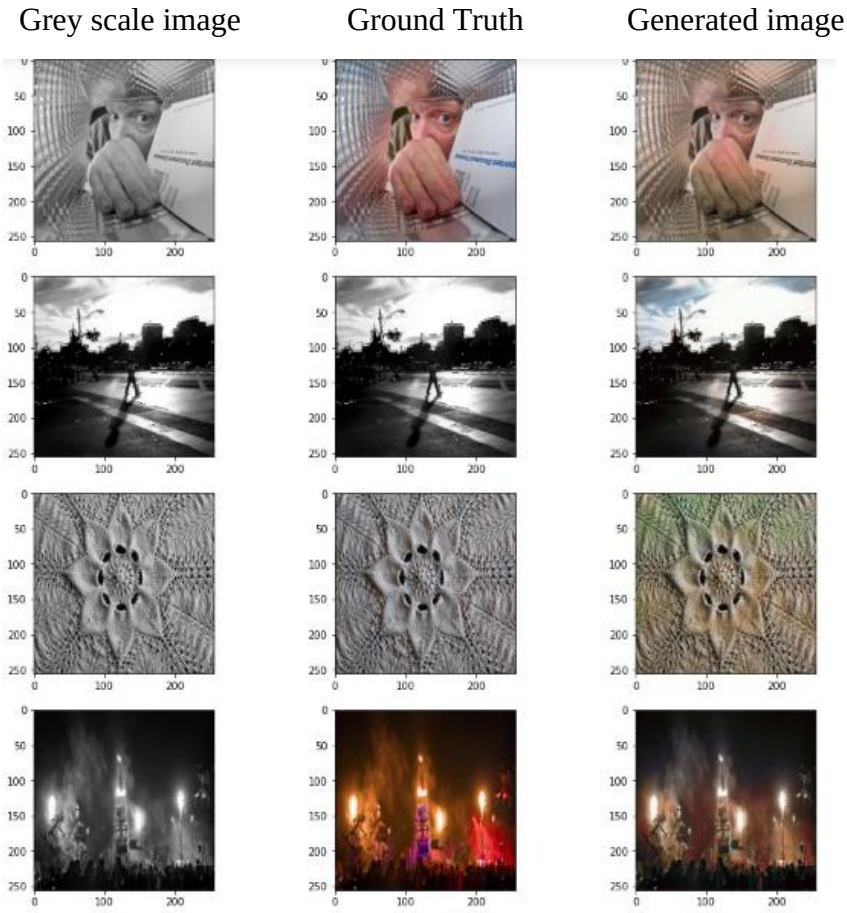


Figure 4.6 Results

So, after getting result we plot graph of Probability vs D(x) & D(G(z)). From graph we can clearly see that for some batches probability of image generated by generator will reach to 0.5 it means that it close to real because discriminator can't even able to distinguished.

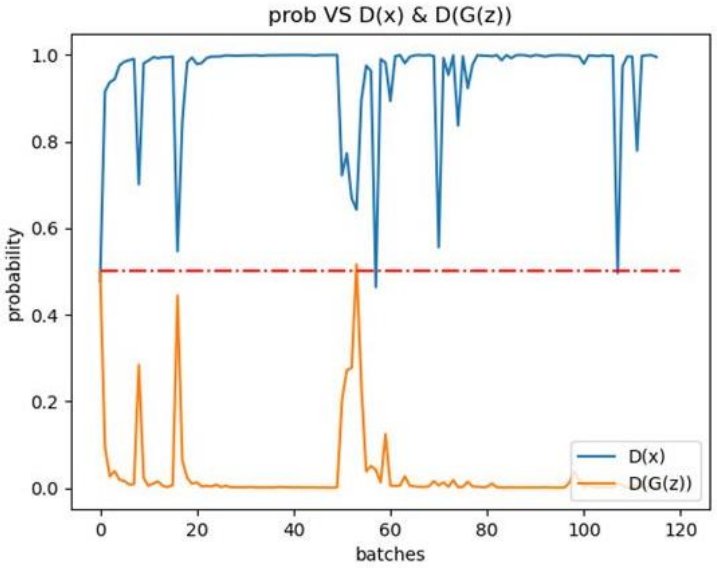


Figure 4.7 Probability vs D(x) & D(G(z))

Now hence our model is working quite good on images but still there is some issues like in some image we can clearly see patches of random colour. And still result doesn't look quite promising so for that reason we decided to train our model on particular dataset like colourise human face or colourise grey scale image of retina. Both of the task were relatively easy compare with natural images because in natural image there is more diversity while in images of retina and images of human faces have less diversity. So with that we train our model on both the dataset. And In that we got relatively good result. But that result comes after doing some modification in our loss function.

Gray scale Ground truth Generated image

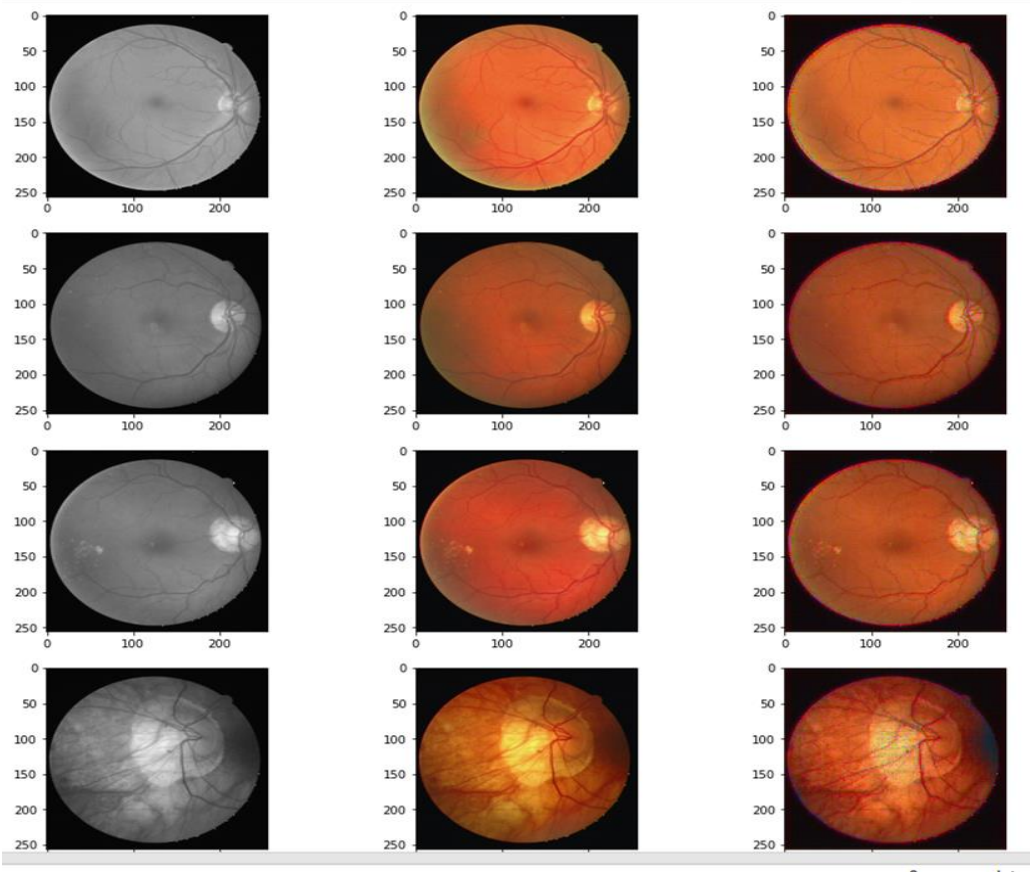
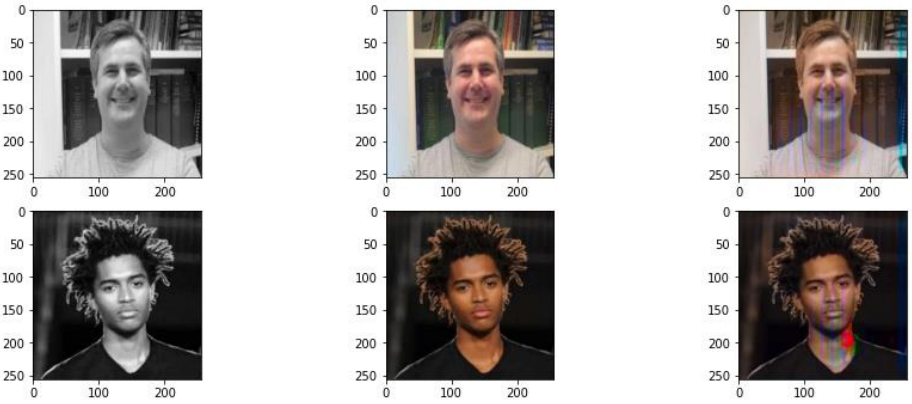


Figure 4.8 Retina Image Colorization 1



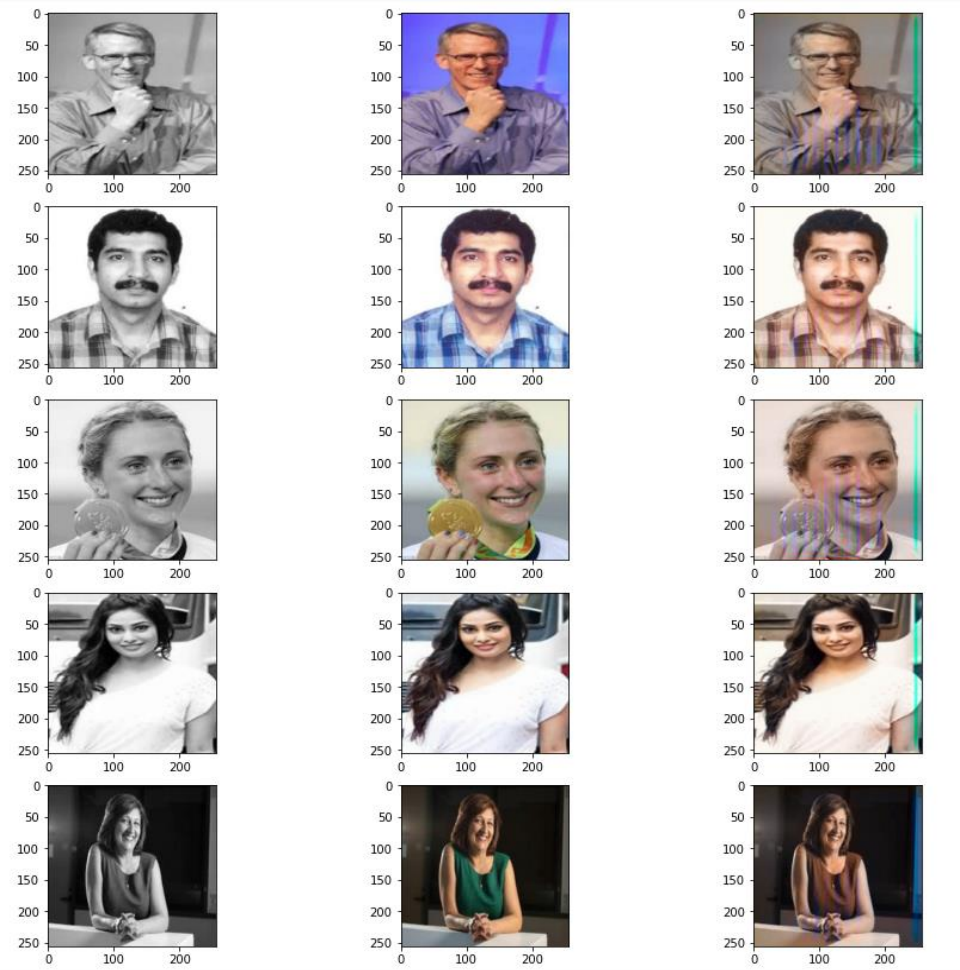


Figure 4.9 Human Face Colorization 1

As we can see quality of image being generated is quite good then our prior once. And in training of this model we use technique call transfer learning. So in that we use weights of model on previously learned dataset and in this way we can save quite good amount of time on training.

Some images is not look like quite colorized the reason behind this is that when we are in training phase at that time for that image discriminator can easily tell that it is fake one and this behaviour is not useful in training of generator because if discriminator outputs near zero output for images then for that images there is small gradient and face problem of vanishing gradient. So that's why its look like more of a greyed one.

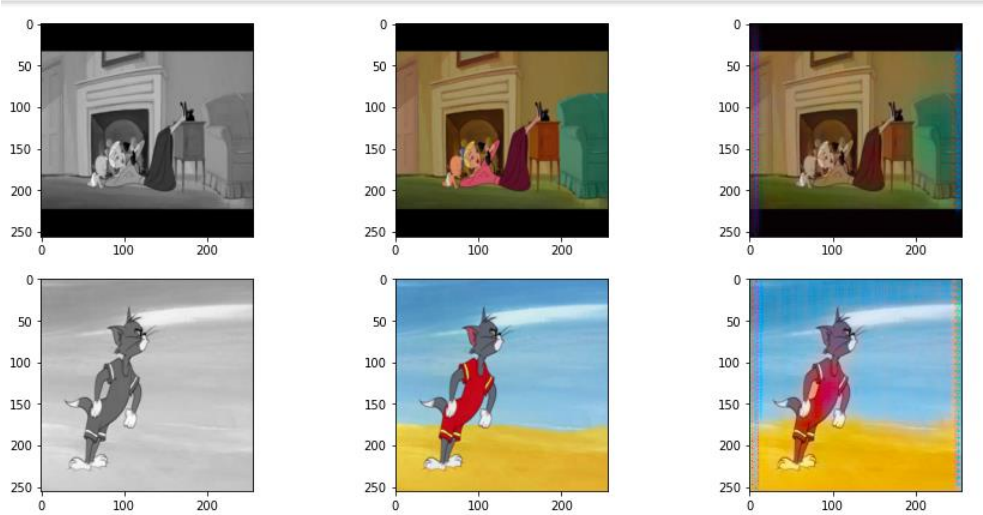
4.3 Video Colorization

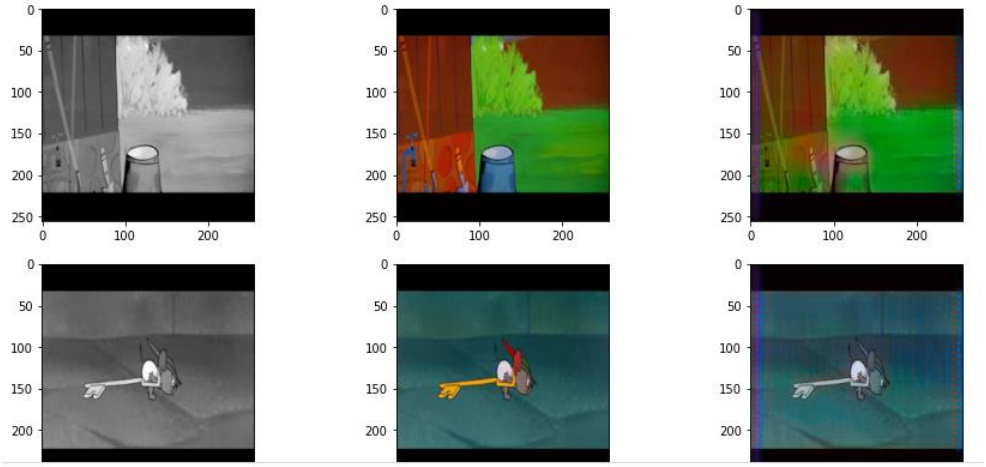
After applying our model on different dataset we thought that it would we now easy to apply it on videos. So for that we pick one episode of tom & jerry. And After that we train our model on that episode. But before that we have to do certain tasks like convert our video into picture. For that we use open cv library. It provide all the necessary feature that we want. Now after splitting video of 35min we have about 35k image. So each second have roughly 100 frames. After carefully watching all images we came to know that most of images in single second were same. So for that reason we randomly take 10 images from each-second. In this way we greatly reduce size of our dataset.

Once the dataset is ready, now it is time to train our model on that dataset. For this also we use previously trained model weights so in this way we save significant amount of time. After 5 epochs we get got result on images so we thought this much training was enough.

Once our model is train, now it is time to take grey scale image and turn that into coloured one using generator. For that we took the same video, using open-cv we split that video into corresponding images. Once we have images then we could feed those images into generator. What our generator will do is to generate colored one out of it and once we have coloured image then we can put back into queue. And after, all images convert into its corresponding coloured one then we can merge those images into video with the help of open-cv. Below we show some of the batches result it look quite real for some images.

However quality of colored video is not so great. On factor is hence there are 100 images per frame so if there are only 50% in one second colored properly then other 50% uncolored one disrupt the whole result and this is what happens with us.





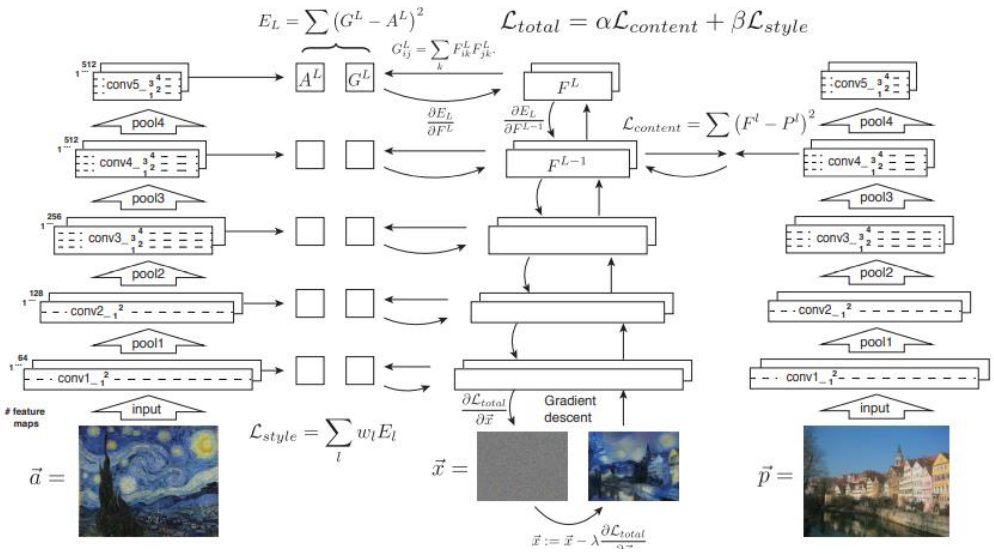
Figured 4.10 Video Result

4.4 Neural Style Transfer

As our guide Prof. M. I. Hasan suggested that try to implement some application of the project we have tried a different approach known as the Deep Art. We have taken the idea from one of the video lecture of IIT-Madras deep learning course where Prof Mitesh Khapara mentioned a neural network which famously known as the deep-art.

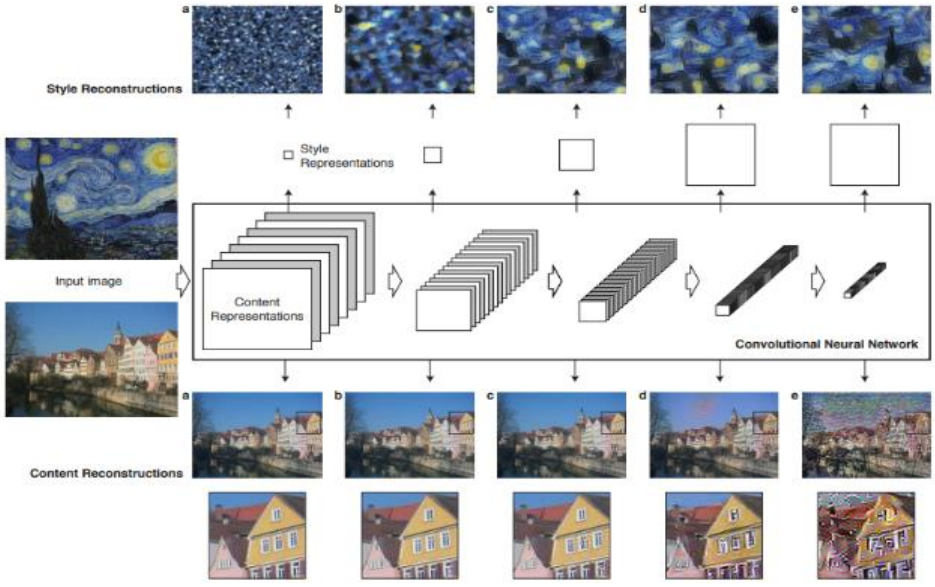
Neural Style Transfer is machine learning technique that allows you apply the style of the one image to the content of another image proposed by firstly [i]. In the paper [i] author has proposed the neural network which takes two image. The result is a new image that combines the content of the first image with the style of the second. For example Taj hotel image can be used as the first image and the painting made by M. F. Husain present at reception of the Taj hotel can be used as the style image and newly generated image will look like the Taj hotel but will have an artistic touch of the M. F. Husain's painting.

The process of neural style transfer involves using a convolutional neural network (CNN) to extract the content and stule of the two input images, and then combing them in a way that pererves the content of the first image while applying the style of the second. This is typically done by the training the CNN on large dataset of images and then using it to generate the new image. We have used the pre-trained model VGG-19 which contains the 19 layer convolutional network.



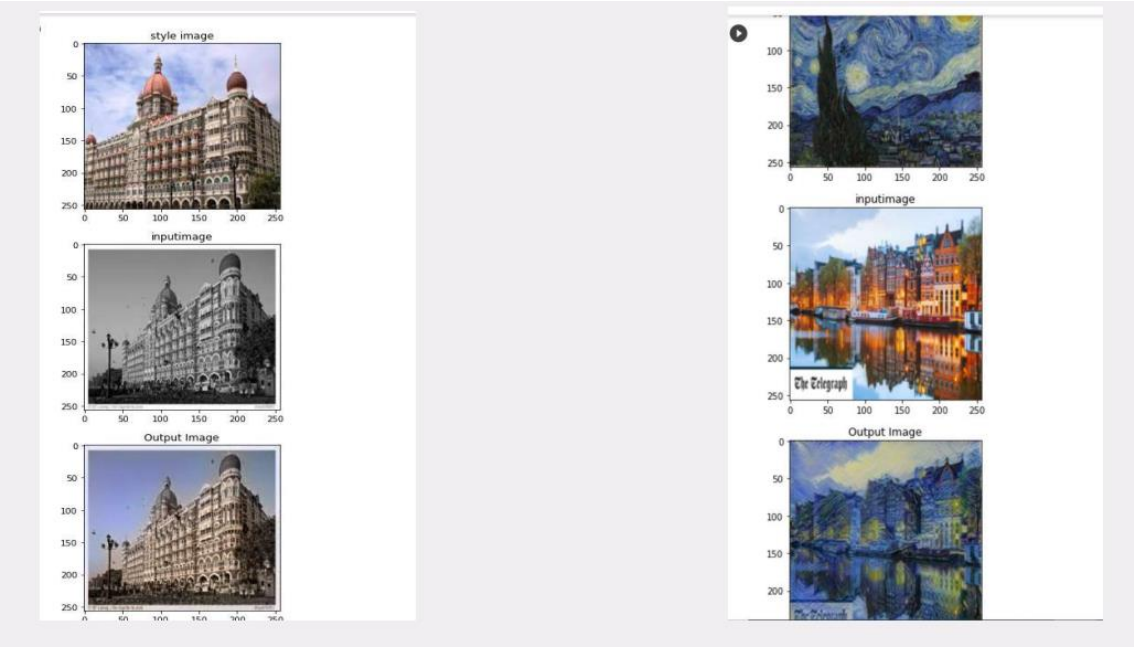
Figured 4.11 Neural Style Transfer

Two separate loss fucntions for both images has been taken and the main loss function is the sum of the two losses. The convolution layer tries to extract the main features of the image and reduce the size of the image. If more convolution layers has been used on the image it will lose it originality and content only the features from which it can be classified to the particular class. Only 5 convolution layers have been used to retain the originality of the content image. Same number of the convolution layer used in the style loss as it clears from the image that deeper the convolution more abstract the style of the image. Gram-matrix is used in the style loss function.



Figured 4.12 Convolutional Operation Result

After that gradient decent is runned for the 50 epochs. A question had arrised in our mind that can we transfer the color as a style to another image. As you can see in the image that Van goah's painting's color is extracted in the 5th convolutional layer. So can it work with black and white image as content image(first-image) and similar colored image of that place. Promising result can be seen in the image. Conclusion could be made from the result that color of the image can be seen as the style of that image and if similar objects are appear in the both image it would transfer the color as style to the black and white image. In image a street of Amsterdum can be scene with the artistic touch of the Van Gouh's famous painting The Starry Nights. Nevertheless, we find it truly fascinating that a neural system, which is trained to perform one of the core computational tasks of biological vision, automatically learns image representations that allow – at least to some extent – the separation of image content from style.



Figured 4.13 Style Transfer Result

4.5 Application :-

4.5.1 Login / Signup module

This module provides login and signup functionality. User have to register himself before using our app. It helps us to know better who is using our application and will help us to provide more personalized content with the help of user's data.

We have divided this module into two parts:

1. Frontend
2. Backend

4.5.1.1 Front-End

We have created front end of the app in android using it's built-in widget like EditText, Button, Progress Dialog etc.

To register user in Our system we are taking

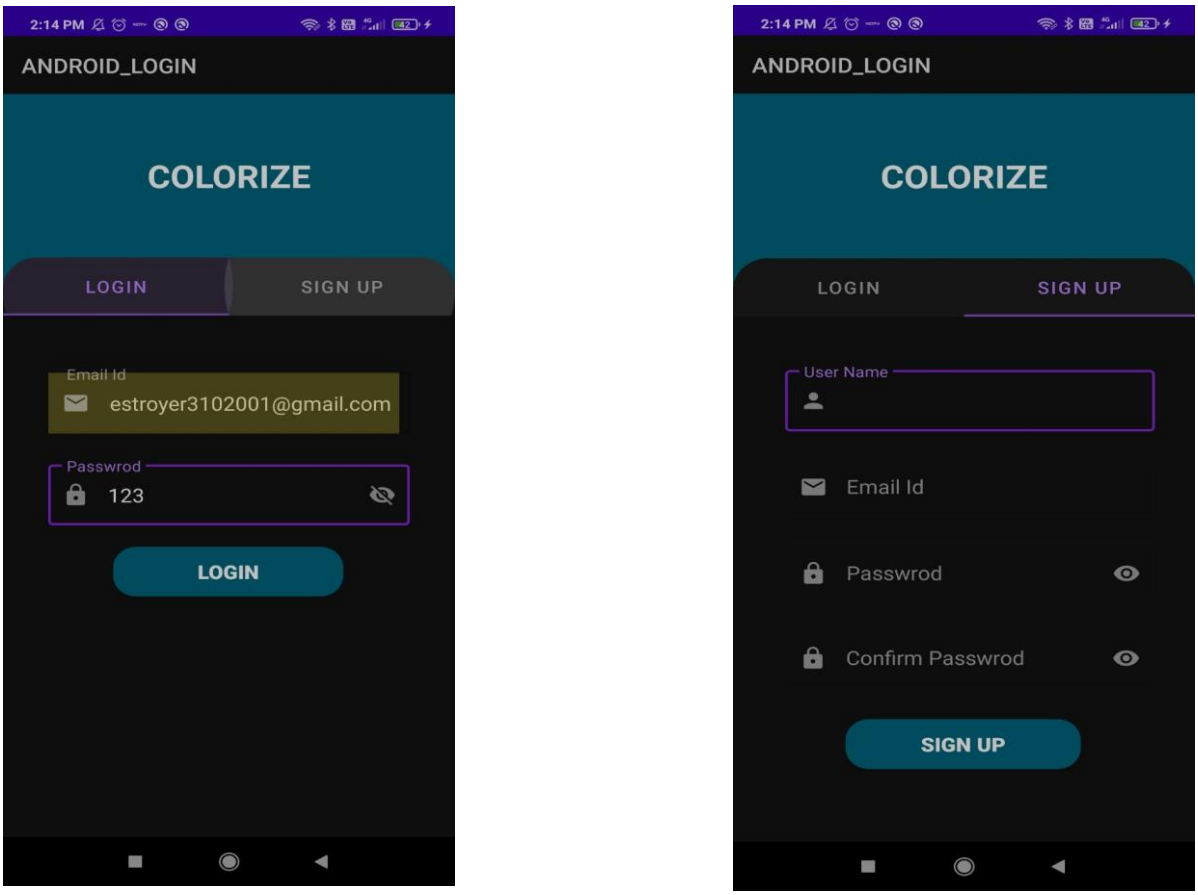
I. Name
II. Email
III. Password
IV. Confirm Password

fields from the user.

The validation we have checked are:

I. Fields should not be empty.
II. Password and confirm password should be same.
III. User with same email should not be already present in our system.

If any of the above condition is violated respected error message will be shown to the user using android toast



Figured 4.14 Login/Signup Page

4.5.1.2 Back-End

In the Back-End at the sign-up time the User-name, Email-id and password has been stored to SQLite database. At the time of the Login if user-email and password has been matched to that stored in the database user is allowed to access the application.

4.5.2 Main-Module

The main module of the application can be divided in two modules

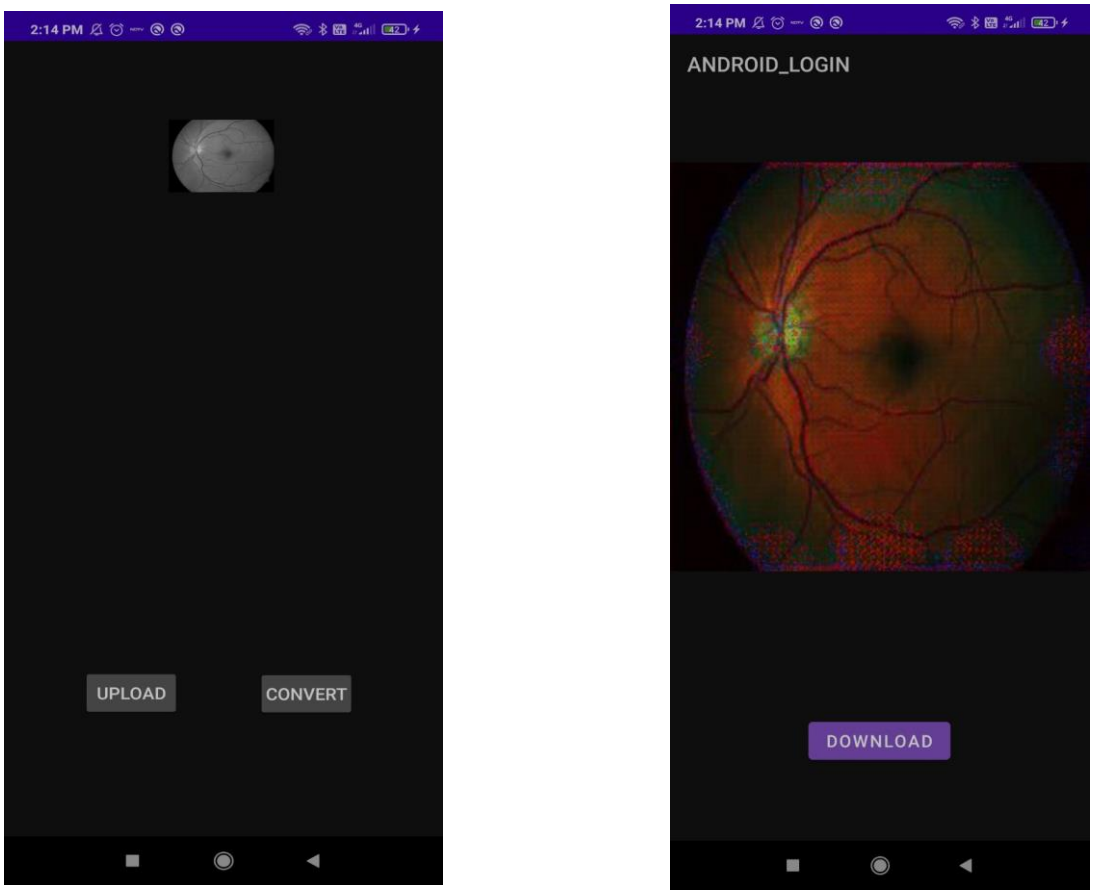
4.5.2.1 Frontend

Before uploading the image application shows two buttons.

1. Upload button is used to select the image from the media of mobile

2. Convert image will send request to the server for converting the gray-scale image to the colorized one

After sending the request UI will show the converted image with the help from the ImageView instance. and download option will be provided to user to store the image to the mobile application.



Figured 4.15 Result Interface

4.5.2.2 Backend side

The gpu where our model is stored is made the server for our application. Flask is used as the backend technology. We have not host our application on the server. So to run our application the server and the client must be on the same network. We then make request to the server by clicking on the convert button. Image is sent with http request.

Following are the steps that take place at the server side.

1. Our GAN model has been loaded with the help of the pickle.
2. The function has been define which uses the model to make predictions. It takes the image as an input and returns the resulted image.
3. Image which is given as an input may not always be in same size required to the model. So we also need to perform some operation to convert the image to make it suitable as input.
4. The resulted image has been sent back to client in response.

4.5.2.3 Http Request handling from client

Http request handling from the client side has been done using the OKHttp. To send the request following steps should be follow.

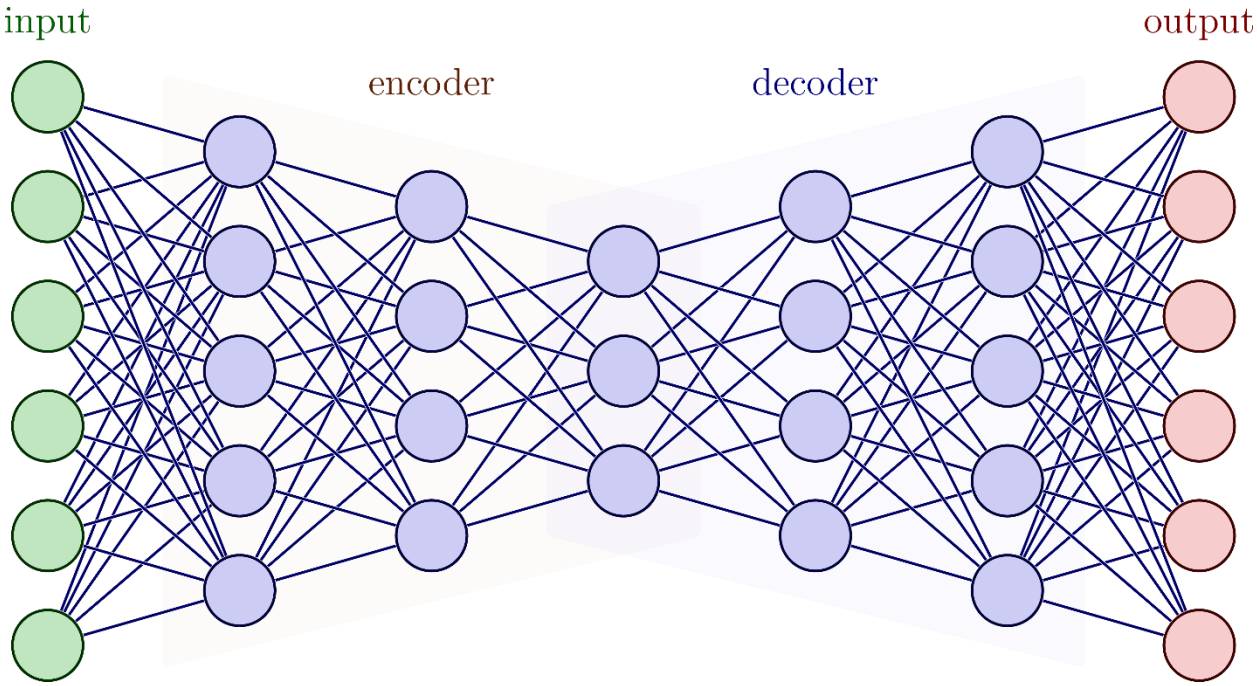
1. Add the OkHttp dependency to your app's build.gradle file.
2. Create an instance of the OkHttp Client class.
3. Create the request using the Request.Builder class
4. Send the request by calling the newCall method of the OkHttpClient instance and pass in the request object.
5. In the onResponse callback, you can access the response body using the response.body() method, which returns a ResponseBody object. You can then read the response data as a string by calling the string() method on the ResponseBody object.

4.6 Face Recognization

4.6.1 Autoencoders :-

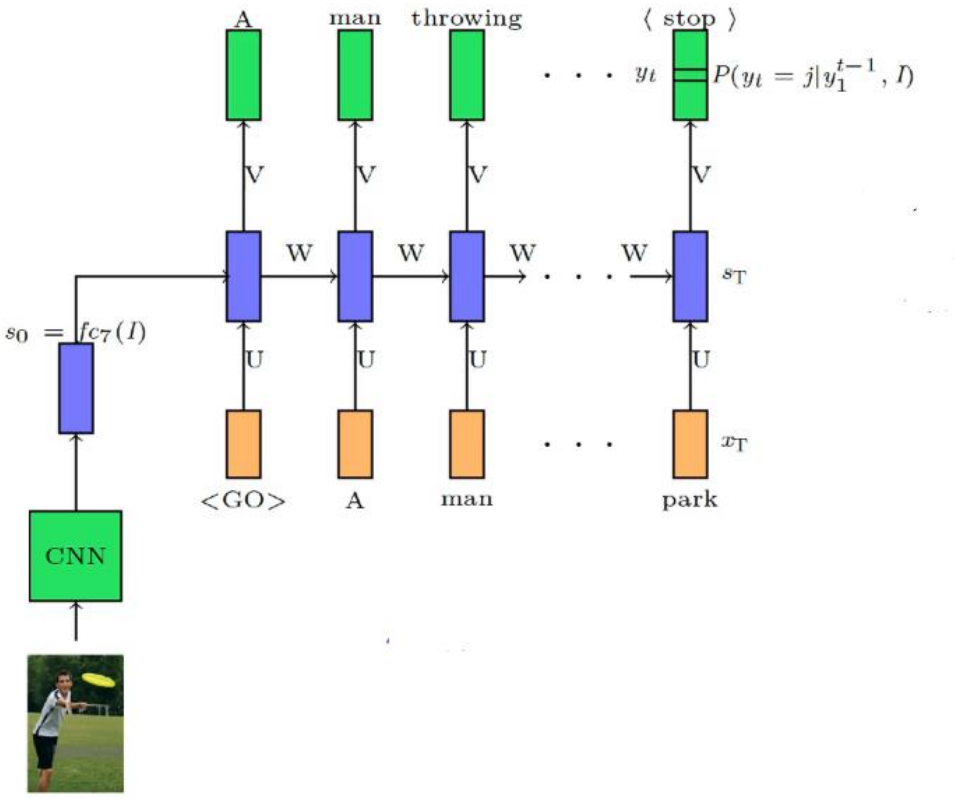
We have made an application for colorize the eye ratina. Model to colorize the facial image also has been developed. We have planned to make a separate application to colorize the facial image in which people can upload black and white image of their beloved ones and get their colorized image. To maintain the identity database of the people facial recognition can also be applied. We have implemented the Autoendcoder to recognize the facial images.

An autoencoder is artificial neural network used for unsupervised learning of efficient codings. The aim of an autoencoder is to learn a representation(encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore signal "noise". Along with the reduction side, a reconstructing side is learnt, where the auto encoder tries to learnt, where the autoencoder tries to generate from the reduced encoding a representation as close as possible to its original input, hence its name. It has mainly two part encoder and decoder. The encoder part of the network takes in inpt data and compresses it into a lower dimensional repsresntation, called latent representation or bottleneck. The decoder part of the network then takes the latent representation and decodes it back into the orginal input data. Autoencoders can be used for a variety of tasks, such as dimensinality reduction, anomaly detection, and image denoising. They are particularly useful for unsupervised learning, as they do not require labeled data for training.



Figured 4.16 Encoder and Decoder

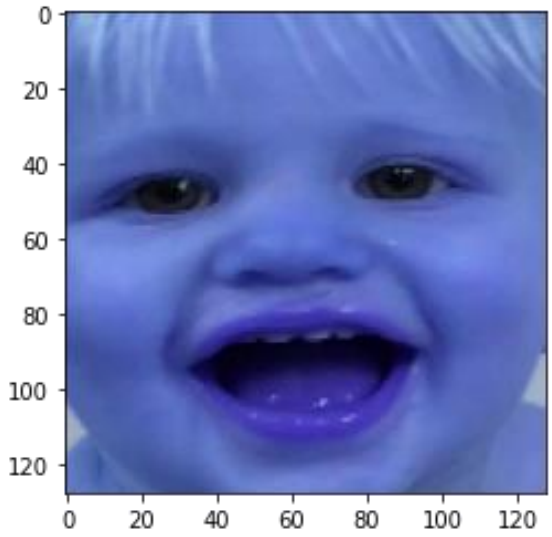
The autoencoder can also be used in the image-captioning task where captioning can be done using the RNN or LSTM but the image represenation to those networks must be given as compress format as image usally can take 1024 or more neurons to represent it.



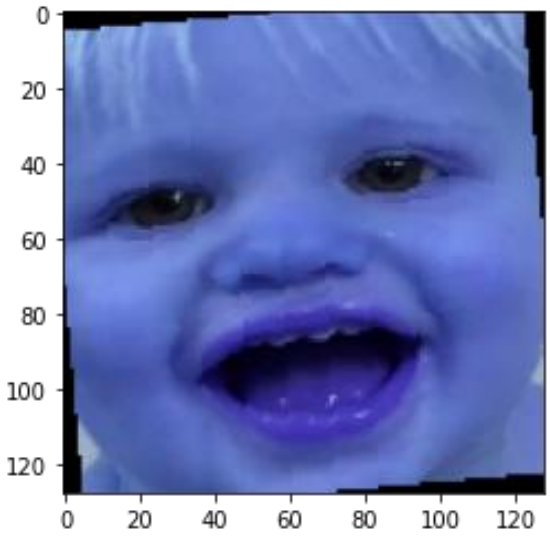
Figured 4.17 Image Captioning

Autoencoders can be used for the face recognition. Autoencoder encodes the images to the representation from which Decoder can decodes it. To recognize the given image it also has been encoded to the middle representation. This representation then matched with all the images in the batch. Image with the least loss is recognized as the given image's identical face. We have trained our model for the images in the batch of 64. The images should be recognized evenif they have been made rotation. Our model can get the correct result with the rotation of 15 degree.

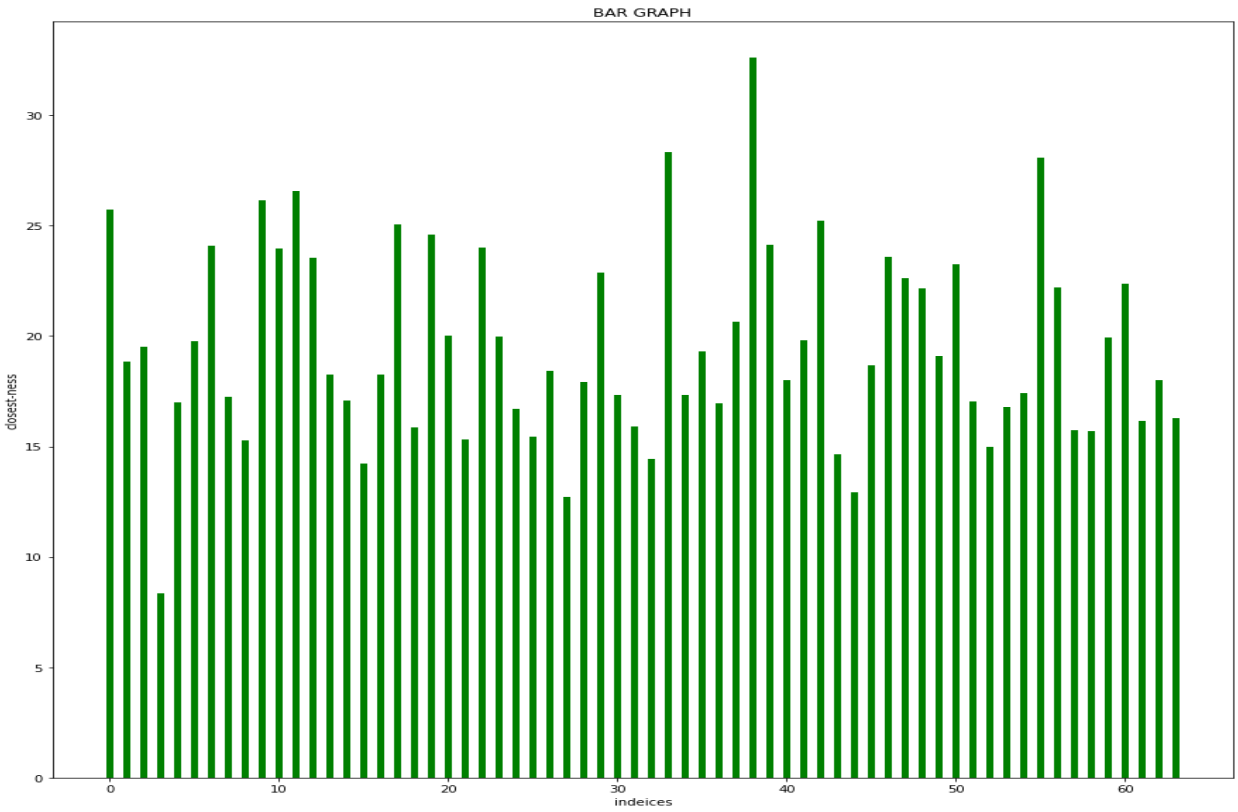
Here the first image is in the training dataset. When the 2nd image has been given to the Autoencoder here is the loss graph of that image with respect to every image in the batch. The lowest loss can be seen with the first image which its match. The results aren't that promsing. If we rotate the image more than 15 degree it won't give the accurate result each time. We will try to improve our model on that.



Figured 4.18 Original Image



Figured 4.19 Rotated Image



Figured 4.20 Result

5. Conclusions and Future Scope

5.1 Conclusion

In conclusion we apply GAN to colorized grey scale image. First, we use our model on natural images in that we got good result but still it was not sufficient so for that reason we thought that it would be beneficial if we restrict our dataset to particular type. So, for that reason we apply our model on grey scale retina image and human face. In those dataset diversity is quite low so for that reason it would easy for our model to capture particular pattern. Our model did great on those two datasets as we shown earlier the results of that. After our model doing great on those two datasets, we thought, why shouldn't we apply our model on video.

For video we took one episode of tom & jerry cartoon and try to colorize it by first decomposing our video into set of images and then train our model on that set of images and letter we use our model to generate corresponding colorized image for given grey scale image. Once we have colorized images then we can re-compose our video. In video also our model work quite good but frankly speaking there are some issues like there are some random colour patches in image and also for some part of video it colorized quite good but in some part it did the worst. So our future task is also centric towards this.

By far we are the only one who can colourised grey-scale image because we have stored our model locally on our device. So to make it available for all we decided to make an android application through which android user can colourised their image. At there we use client-server architecture in which user phone is client and server is our local machine. We use Flask as framework for implementation because we use pytorch for deep learning so it make sense that we go for same language framework to avoid issues related to joining.

After working with different datasets and particularly with human face we thought it would be nice to spend our time on human face detection. So for that reason we use auto-encoder model wherein encoder model take image as input and generate latent-space. Now, we can use this latent-space to find out close-ness between two images.

5.2 Future Scope

1.) We face several issues in video colorization so in future we want to rectify this. Apart from that there are several different technique to learn GAN within which we want to train our model using WGAN-GP(Wasserstein GAN with gradient penalty). Studies shows that this technique will create more diverse result but with more time on training.

2.) auto-encoder working great but still need some improvement so we are going to work on that to improve out model as well as make it more reliable for human face detection.

3.) Want to use attention model on image colorization to see just how it works on that.

References

[1]. C. A. S. Domonkos Varga and T. Szirfffdfffdnyi, “Automatic cartoon colorization based on convolutional neural network”. Available : https://core.ac.uk/download/pdf/94310076.pdf, 2017
a.

[2]. S. Salve, T. Shah, V. Ranjane, and S. Sadhukhan, “Automatization of coloring grayscale images using convolutional neural network”, Apr. 2018

[3]. Yu Chen, “Automatic colorization of images from Chinese black and white films based on CNN”, 2018.

[4]. V. K. Putri and M. I. Fanany, “Sketch plus colorization deep convolutional neural networks for photos generation from sketches ”, 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), Sep. 2017,

[5]. R. Zhang, P. Isola, and A. A. Efros, “Colorful image colorization,” in ECCV, 2016

[6].