EE 610 (Image Processing) - Report for Course Project

Image Colorization

Aman Butoliya
203070001
Electrical Engineering
203070001@iitb.ac.in

Hiranmay Mondal
2130790017
Electrical Engineering
213070017@iitb.ac.in

***Abstract*—Colorization of image is very challenging problem as the image conditions keep on changing such as size, format sentiments etc. Though one can anticipate colors from semantics, for example grass is always green, sky is always blue etc. However, such semantics aren't useful when objects are non-semantic such as humans, buildings etc. In this project, I review some of the recent papers that deals with image colorization problem. I tried implementing a paper that uses combination of a deep Convolutional Neural Network trained from scratch with high-level features extracted from the InceptionResNet-v2 pre-trained model[1]. Thanks to its fully convolutional architecture, this encoder-decoder model can process images of any size and aspect ratio.**

Key Words: Deep Lerning, colorization, CNN, transfer learning, pytorch.

I. Introduction

Coloring gray-scale images can have a big impact in a wide variety of domains, for instance, re-master of historical images and improvement of surveillance feeds. Information content of gray scale image in rather limited and adding colors to it can provide more details about its semantic. Deep learning model such as Inception, ResNet or VGG are trained on color images, hence previous colorization steps may improve the result. Designing and implementation of such automatic systems is challenging and the main is to fool human eye into believing that the result of this systems are accurate. In this context, I tried to solve above mentioned problem using combination a deep Convolutional Neural Network architecture and the pretained Inception model, namely Inception-ResNet-v2. CNN is trained from scratch, Inception-ResNet-v2 is used as a high-level feature extractor which provides information about the image contents that can help their colorization. Due to time and computational power constraints, image data used is small and because of that results are not up to the mark.

II. Background

The problem Image colorization has been the focus of significant research efforts over the last two decades while most early image colorization methods were primarily influenced by conventional machine learning approaches. Recently, new approaches based on transformer and attention shown to achieve outstanding result compared previously existing models[**3**]. The aim of colorization is to convert a grayscale image to color image, typically captured decades ago, when technological advancements were limited. Hence this process is more a form of image enhancement than image restoration.

Early colorization architectures were plain networks. A network is classified as plain if it possesses a simple, straightforward architecture with stacked convolutional layers, i.e., no, or naive skip connections. This implemented model is based on encoder-decoder architecture however authors had introduced a pre-trained network into the equation.

We consider images of size H × W in the CIE L*a*b* color space. Starting from the luminance component $X_L \epsilon R^{H*W*1}$, the purpose of this model is to estimate the remaining components to generate a fully colored version $\tilde{X} \epsilon R^{H*W*3}$. In short, lets assume that there is a mapping $\mathcal{F}$ such that

$$\mathcal{F} : X_L \rightarrow (\tilde{X_a}, \tilde{X_b})$$

where $\tilde{X_a}$, $\tilde{X_b}$ are the a*, b* components of the reconstructed image, which combined with the input give the estimated colored image $\tilde{X} = (X_L, \tilde{X_a}, \tilde{X_b})$.

In order to be independent of the input size, architecture is fully based on CNNs, a model that has been extensively studied during literature. In brief, a convolutional layer is a set of small learnable filters that fit specific local patterns in the input image. Layers close to the input look for simple patterns such as contours, while the ones closer to the output extract more complex features. Authors have choosen the CIE L*a*b* color space to represent the input images, since it separates the color characteristics from the luminance that contains the main image features. Combining the luminance with the predicted color components ensures a high level of detail on the final reconstructed image.

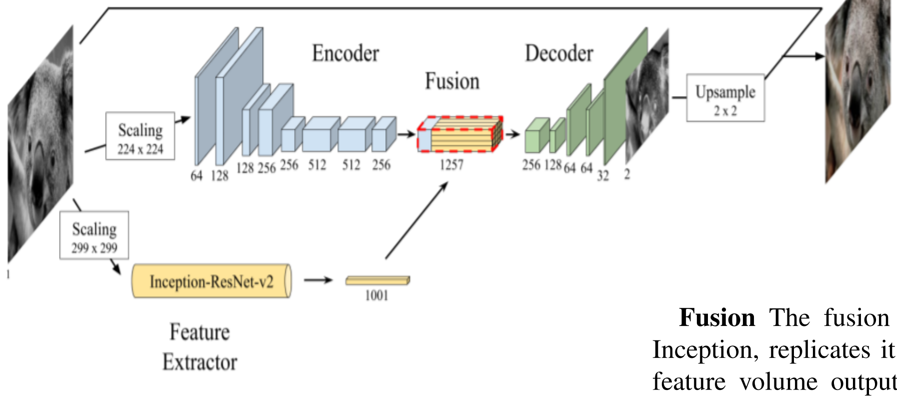

Fig. 1: stacking luminance component two times

III. Data preprocessing

The dataset used contains about 7k color images i.e. RGB images. As explained earlier RGB format doesn't prove to be significant in image colorization. Instead LAB color space particular provides better insight as it can separete color information of an image from its luminance.

First, this dataset is converted into LAB color space. Luminance component of this color space is used as input to both encoder and inception model. For encoder, grayscale image is resized into (224*224) component and for extractor model it is resized into (299*299).

Encoder The Encoder processes H × W gray-scale images and outputs a H/8 × W/8 × 512 feature representation. To this end, it uses 8 convolutional layers with 3 × 3 kernels. Padding is used to preserve the layer's input size. Furthermore, the first, third and fifth layers apply a stride of 2, consequentially halving the dimension of their output and hence reducing the number of computations required.

Feature Extractor High-level features, e.g. "underwater" or "indoor scene", convey image information that can be used in the colorization process. To extract an image embedding we used a pre-trained Inception model. First, we scale the input image to 299×299. Next, we stack the image with itself to obtain a three channel image (as shown in Fig. 2) in order to satisfy Inception's dimension requirements. Next, we feed the resulting image to the network and extract the output of the last layer before the softmax function. This results in a 1001 × 1 × 1 embedding.

Fusion The fusion layer takes the feature vector from Inception, replicates it $HW/8^2$ times and attaches it to the feature volume outputted by the encoder along the depth axis. This method was introduced by [16] and is illustrated in Fig. 3. This approach obtains a single volume with the encoded image and the mid-level features of shape H/8 × H/8 × 1257. By mirroring the feature vector and concatenating it several times we ensure that the semantic information conveyed by the feature vector is uniformly distributed among all spatial regions of the image. Moreover, this solution is also robust to arbitrary input image sizes, increasing the model flexibility. Finally, we apply 256 convolutional kernels of size 1×1, ultimately generating a feature volume of dimension H/8 × W/8 × 256.

Decoder Finally, the decoder takes this H/8×W/8×256 volume and applies a series of convolutional and up-sampling layers in order to obtain a final layer with dimension H × W × 2. Up-sampling is performed using basic nearest neighbor approach so that the output's height and width are twice the input's.

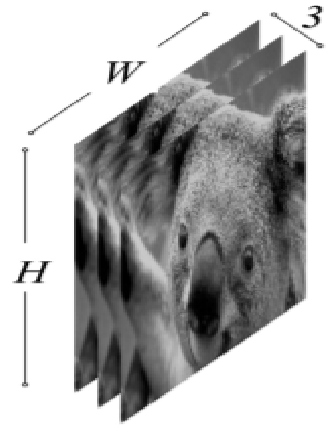

Fig. 2: An overview of model architecture

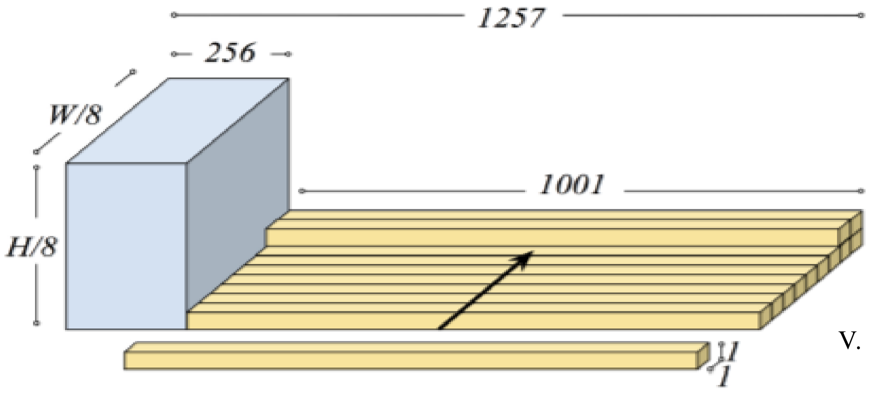

Fig. 3: Fusing the Inception embedding with the output of the convolutional layers of the encoder

IV. Experiments

The optimal model parameters are found by minimizing an objective function defined over the estimated output and the target output. In order to quantify the model loss, we employ the Mean Square Error between the estimated pixel colors in a*b* space and their real value. For a picture X, the MSE is given by

$$C(X,\theta) = \frac{1}{2HW}\sum_{k\epsilon a,b}\sum_{i=1}^{H}\sum_{j=1}^{W}(X_{k_{i,j}} - \tilde{X}_{k_{i,j}})$$

where θ represents all model parameters, $X_{k_{i,j}}$ and $\tilde{X}_{k_{i,j}}$ denote the ij:th pixel value of the k:th component of the target and reconstructed image, respectively. This can easily be extended to a batch B by averaging the cost among all images in the batch, i.e. 1/|$\mathcal{B}$| $\sum_{X\epsilon\mathcal{B}} C(X,\theta)$.

While training, this loss is back propagated to update the model parameters θ using Adam Optimizer [28] with an initial learning rate η = 0.001. During training, we impose a fixed input image size to allow for batch processing. Choice of dataset is what networks accuracy depends on, more the variety of images more semantics the model can capture. Originally, authors have trained this network on 60k images while I trained it on 6k images. pictures are heterogeneous in shape, therefore all images in the training set are rescaled to 224 × 224 for the encoding branch input and to 299 × 299 for Inception. Each image gets stretched or shrunk as needed, but its aspect ratio is preserved by adding a white padding if needed.

V. Result and Conclusion

Once trained, we fed our network with some images. The results turned out to be not that good for some of the images. Due to the small size of training set results aren't much better. However, even if the training size is smaller, if training data consists of same kind of semantics then model will work better for that kind of images. As the model is simple and data size is small, model prediction for converting grayscale images to color images is not up to the mark. The result of the trained model can be seen below.

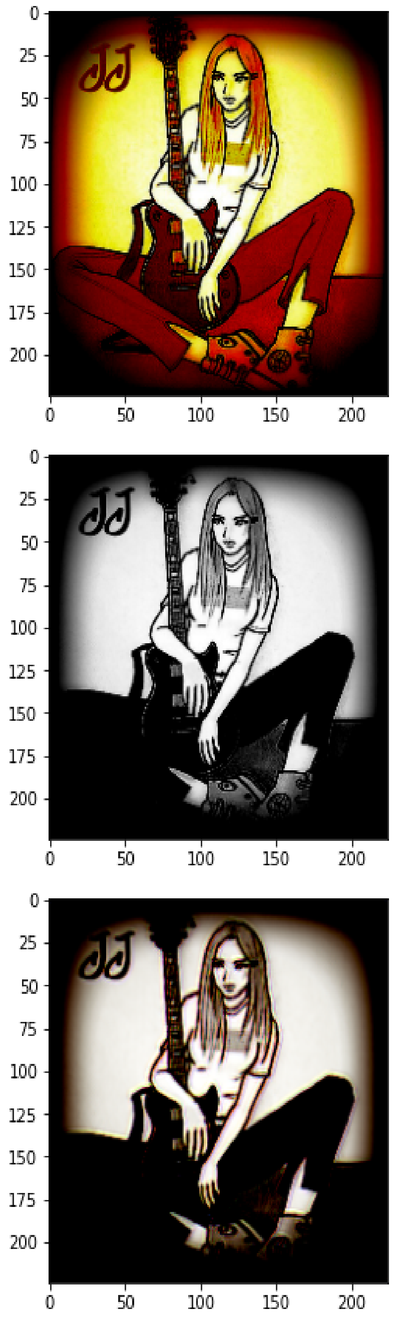

Fig. 4: Results 1, a(Color Images), b(Grayscale Image), c(Predicted Image)

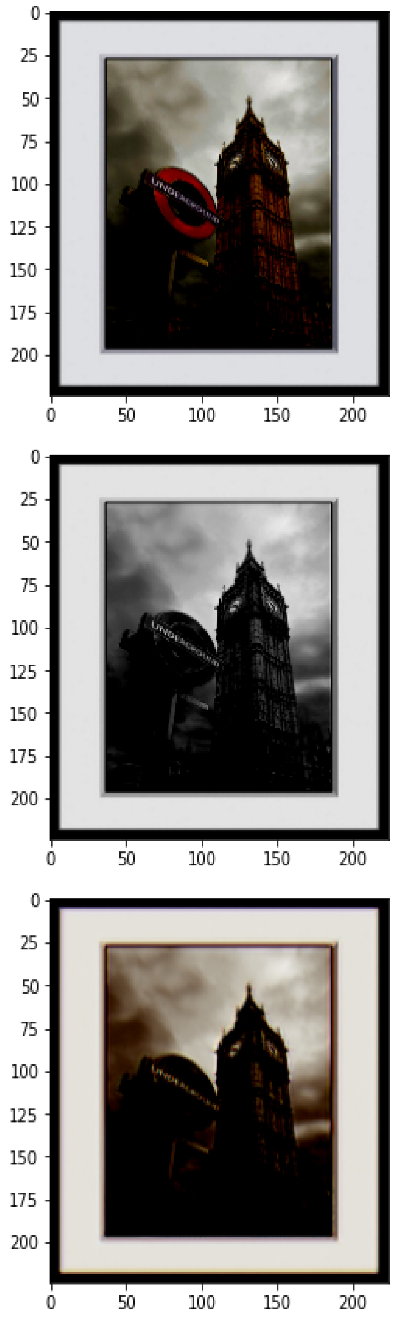

Fig. 5: Result 2, a(Color Images), b(Grayscale Image), c(Predicted Image)

VI. Conclusion and Future Work

This project validates that an end-to-end deep learning architecture could be suitable for some image colorization tasks. In particular, this approach is able to successfully color high-level image components such as the sky, the sea or forests. Nevertheless, the performance in coloring small details is still to be improved. As dataset in too small, only a small portion of the spectrum of possible subjects is represented, therefore, the performance on unseen images highly depends on their specific contents. To overcome this issue, our network should be trained over a larger training dataset.

A better mapping between luminance and a*b* components could be achieved by an approach similar to variational autoencoders, which could also allow for image generation by sampling from a probability distribution. Recent, state-of-the-art model for the same has been introduced this year. This model is based on transformer and attention mechanism however its architecture is encoder-decoder based. Needless to say, in implemented model variational encoder can improve accuracy.

Finally, this colorization technique can be applied to videos as well, it will be good to see behaviour of such models in historic videos.

Acknowledgment

We would like to express his gratitude towards Professor Amit Sethi who provided the necessary help and guidance required to summarize this report. They would also like to thank him for providing clarifications wherever required.

References

[1] Federico Baldassarre, Diego Gonz´alez Mor´ın, and Lucas Rod´es-Guirao, "Deep Koalarization: Image Colorization using CNNs and Inception-Resnet-v2", December 2017

[2] Saeed Anwar, Muhammad Tahir, Chongyi Li, Ajmal Mian, Fahad Shahbaz Khan, “Image Colorization: A Survey and Dataset", November 2020

[3] Manoj Kumar, Dirk Weissenborn and Nal Kalchbrenner, "Colorization Transformer", March 2021