Final Project Report
ENGR-E-533 Deep Learning
Image Colorization Project

Siddartha Rao, Sumeet Mishra, Varun Miranda

December 17, 2019

Abstract

Image colorization is a interesting project as color adds information to an image. Also, colored images are more nicer to look at than black and white images. Our motivation was to color historical images which doesn't have a ground truth. We tried to address this problem with convolution layers and then with Inception ResNetV2 which is pretrained on ImageNet dataset. We then combined the output of both models to color the grey scale images.

1 Introduction

The whole concept behind image colorization is to try to learn the high level and the low level features in the image and provide colors based on the color patterns detected in the training data. For instance, grass should be colored green, and the sea and sky should be blue. Based on the background the tinge of green or blue will vary. If there is a sunrise on the horizon, for instance, then the model should learn to color the sky differently based on the background image and provide orange streaks rather than day time when the sun is positioned differently in the image.

Of course based on the black or white shade the model can get a hint about whether the color should be dark or light. However, to predict the correct color will definitely be a challenge and there is always a chance, for example for the sea to be bluer than the ground truth.

2 Literature Survey

Our main architecture is based on Federico Baldassarre, Diego Gonzalez Morın, Lucas Rodes-Guirao's Image Koalarization paper [3]. Richard Zhang, Phillip Isola, Alexei A. Efros's Colorful Image Colorization paper [2] worked on a basic CNN architecture without the ResNet and did not always have a better result in comparison to the Image Koalarization paper, despite training with L2 Regression Loss and using a class rebalancing network.

Besides, we felt that adding a Inception ResNet v2 will improve our model accuracy as it learns the high level features. The challenge however, is that we are pre-training the ResNet with ImageNet weights while we are using the COCO dataset in our project.

S. Iizuka, E. Simo-Serra, and H. Ishikawa used two encoders [1] in parallel before combining it in a fusion layer and passed into a decoder. Apparently the paper didn't provide better results as Image Koalarization either hence we further concluded that adding an Inception ResNet layer is necessary besides the Encoder-Decoder network.

3 Dataset

We used the COCO dataset for our project. We have trained the data in three phases. First in order to evaluate the best model we used a total of 5000 images, the training set consists of 3500 images, validation 500 images and the test set 1000 images. The images in the dataset are all colored and we use this as the ground truth. In order to keep the format of all the images same, we performed various transformation on the data including flips,zoom and shears to prevent over-fitting.

Once we tune the best parameters we moved on to the next phase of our project and trained it on more images. Among a total of 15000 images we used 13000 for train and 2000 for validation and tested the data on 1000 new images to see if our results improve by training on more images. Turns out, it predicted better and we had more color in our images

We took our project one step further and trained it on 25000 and then 40000 images and our results improved considerably. We were almost able to reciprocate the ground truth or at the very least made a reasonable guess of the color.

Finally we used historical images to check if our model is giving decent enough colors in our images. As it does not have a ground truth, it will be interesting to find out which features are being learnt well.

4 Architecture

4.1 Input

The encoder decoder model takes LAB image of size 224×224 as input and the ResNetV2 model takes B/W image of size 299×299. The LAB image comes from the CIELAB color space, where L* stands for luminance and it ranges from black(0) to white(100), a* from green(-128) to red(+128) and b* from blue(-128) to yellow(+128).

The L part is fed as the input and the model predicts the A and B color schemes before we convert the LAB to RGB color scheme to obtain our reconstructed output. The B/W input will be fed into the Inception ResNet layer which improves our model.

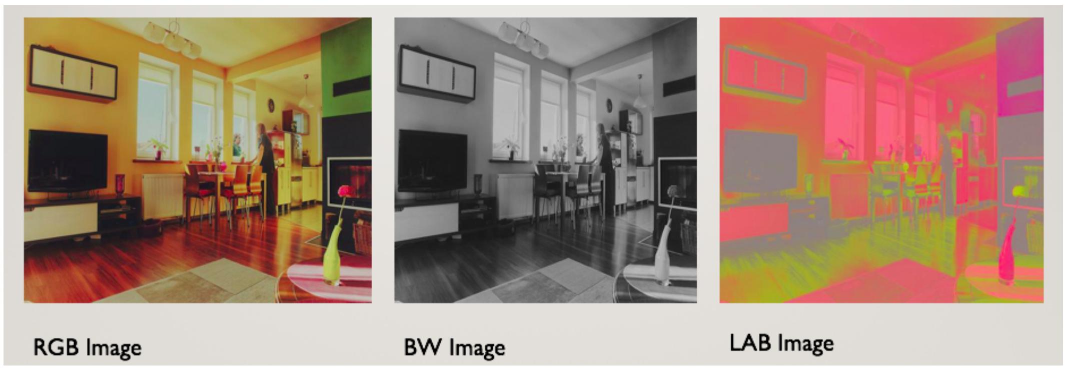

Figure 1: RGB Image, BW image and LAB image

4.2 Model

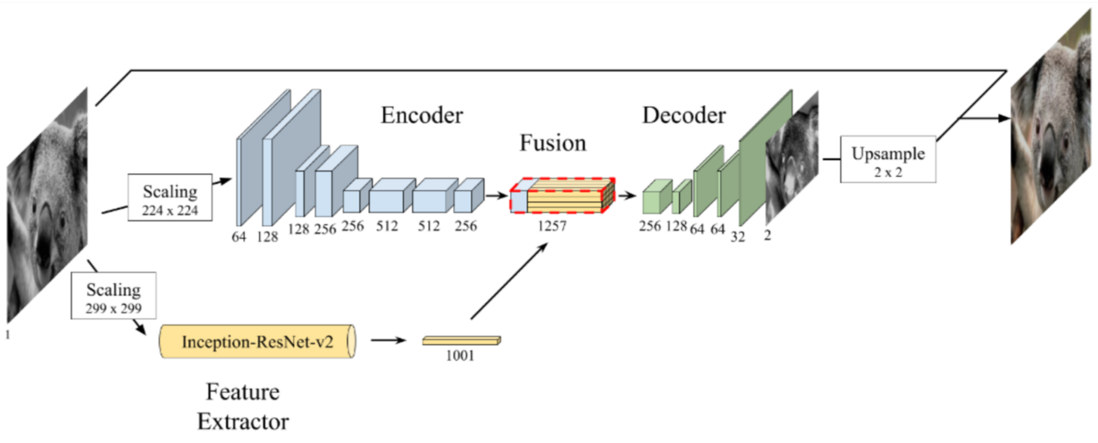

Figure 2: Architecture

The network is divided into four parts, Encoder,Inception-ResNet-v2, Fusion layer and Decoder. The Encoder obtains the mid and low level features and the ResNet obtains the high level features. These two are merged in the Fusion layer and is passed to the Decoder which uses these features to predict the output.

Encoder

The encoder takes the LAB image and scales it to size 224×224 as its input. It has 8 convolution layers with 3×3 kernels. The first,third and fifth layer have stride of 2, which helps in reducing the dimensionality of their output by half and downsizes the image. The output of the encoder is a vector of size $28 \times 28 \times 256$.

Inception-ResNet-v2

This model has been pre-trained on imagenet weights and it accepts a B/W image and scales it to size 299×299 it will try to extract the high level features and output a image embedding vector of 1001 length.

Fusion layer

The image embedding vector from the ResNet should be of the same tensor shape as the output from the encoder so the vector is repeated 28 times so that a 28×28 shape is achieved. Then the concatenation happens and the fusion layer is created. The output of the fusion layer passes as an input into the decoder.

Decoder

The output of the fusion layer is fed to the decoder. The decoder has 5 convolution layers with 3×3 kernels. Up-sampling layers are included in every alternate layer to resize the image so as to obtain the final output of size $224 \times 224 \times 2$. As mentioned earlier, the 2 filters of size 224×224 depict the output color schemes A and B.

4.3 Loss Function

We have used mean squared error as our loss function. We measure the distance between the target pixel value and our predicted pixel value for each of the color layers

Mean Square Error $=\sum(\hat{y} - y)^2/n$

4.4 Optimizer

Adam optimizer is an adaptive learning rate technique that combines RMSProp and Stochastic Gradient Descent to attempt to locate the global minimal loss.

It uses squared gradients to improve the learning rate like RMSProp and it takes the advantage of momentum from Stochastic Gradient Descent.

5 Results

5.1 COCO Test Images

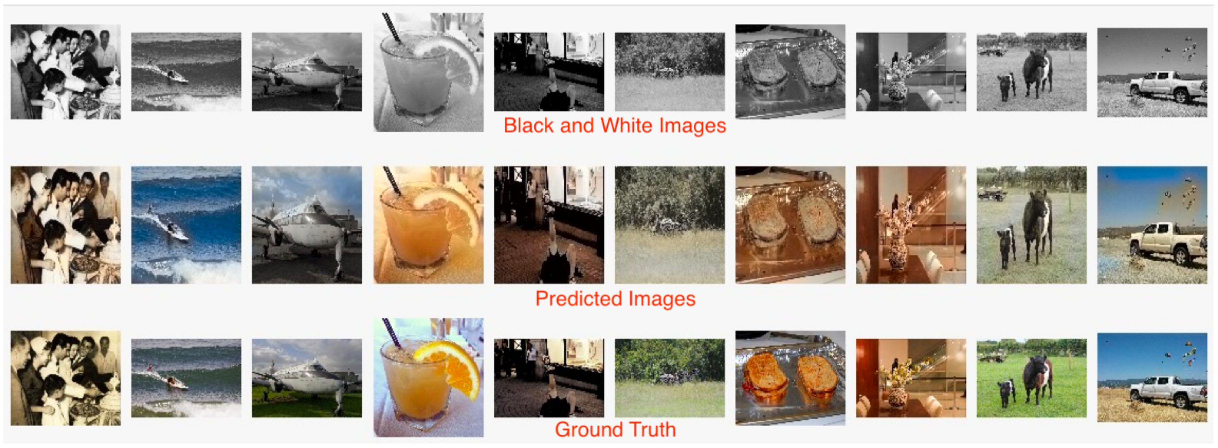

Figure 3: Results after training with 4000 images

These images are predicted as a result of training on 4000 images. It is able to generalize pretty well on these images as we can observe in the second image from the left. It colored the sea too blue than expected. This is because the model didn't "see" enough training data. A similar observation on the sky can be made on the fourth image from the left with respect to the sky. This can be made better when we train on more data.

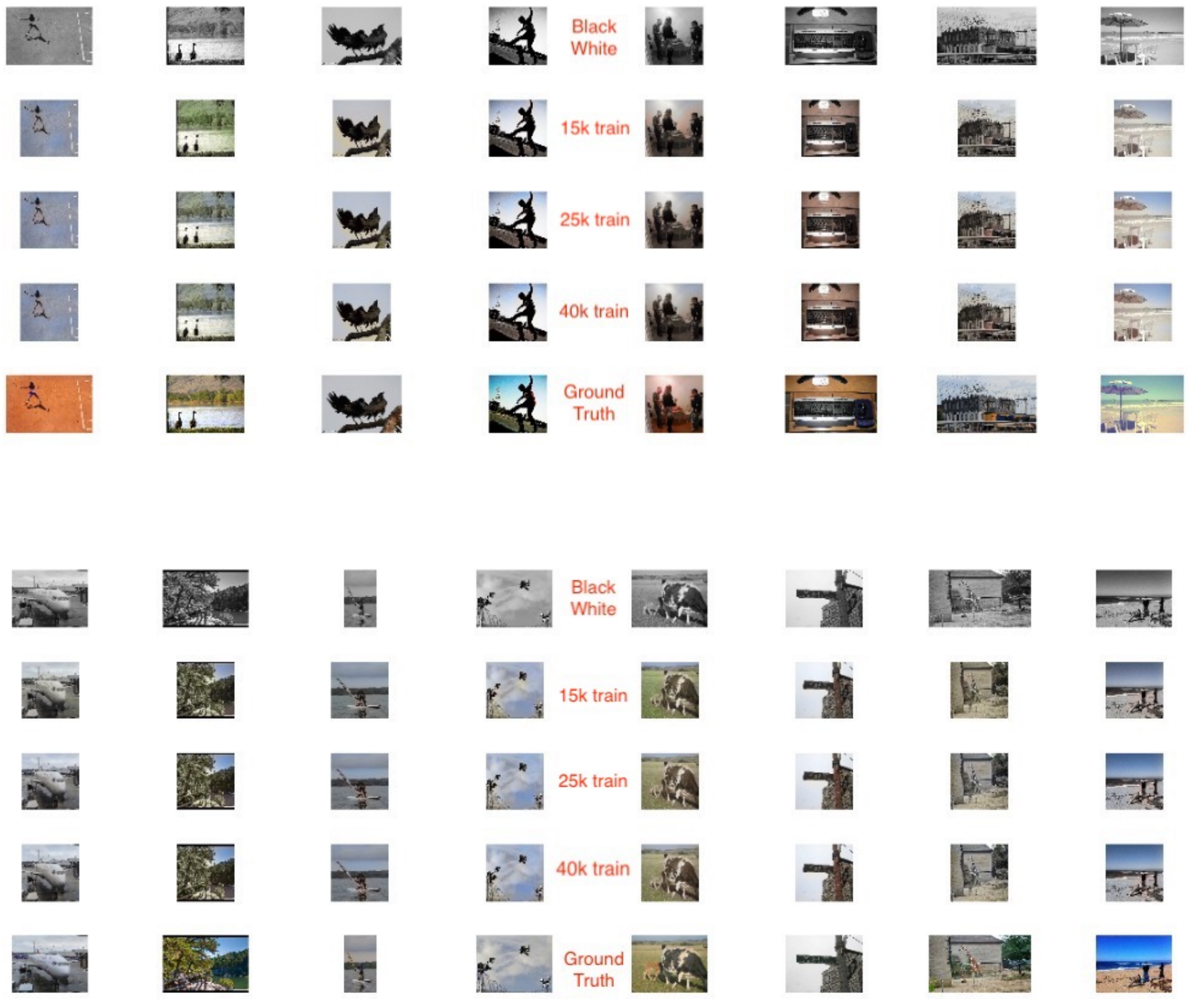

Figure 4: Comparing the results across models trained with different data train length

As our length of the training data increases, our model generalizes better. It is interesting to note that in the Tennis court color prediction in the first image it predicts a hard surface rather than a clay surface which can be explained as a hard court is a more common tennis surface than the clay surface.

In some of the images there is no clear distinction between the images with 15k, 25k and 40k train data respectively, this can be explained as when the image already learnt all the features of the images and is saturated, i.e. the validation loss is not reducing further. The MSE is calculate pixel wise and we found the **train error** as **0.006** and the **validation error** as **0.011**.

5.2 Historical Images

It is interesting and we were curious to test our model on historical images as they don't have a ground truth. Probably to understand which features it will learn well. Sometimes the historical images are not exactly black and white so it is converted into black and white and then run through the model. The results are depicted below:

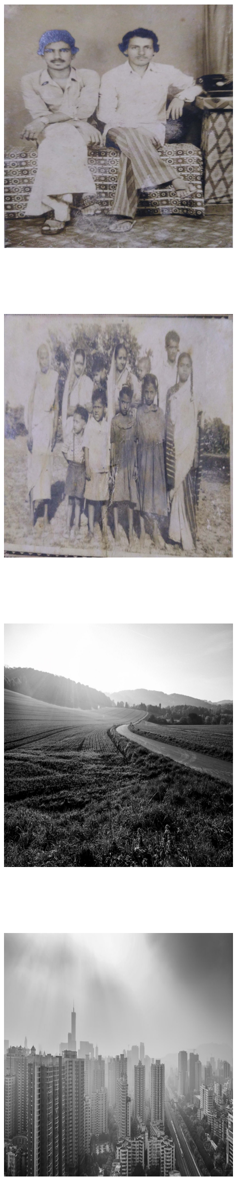
Figure 5: Historical image

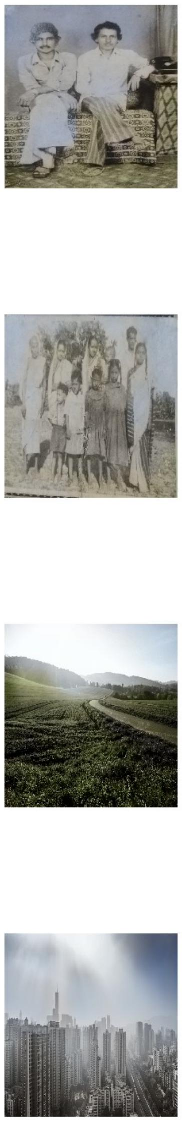
Figure 6: Predicted color image

The first two images are historical images we procured offline and the last two images were taken online. We are not aware of the ground truth for both sets of images.

We have displayed some of the best results here, the other historical images as well as their predicted color images can be found in our source code (Jupyter Notebook)

6 Conclusion

The image color prediction is not always straightforward and it definitely cannot be made perfect. As mentioned earlier, sometimes it predicts the ocean to be bluer and the grass to be greener. The reason we used more images to train is so that it generalizes better based on the images it learns from.

Also, the inception layer is pretrained on imageNet weights. In actuality, had we used the imageNet dataset there was a chance of getting better results. It was interesting to study the results of the pretrained imageNet weights on the COCO dataset.

If there were more training images for the clay court example, there is always a chance that it will predict an orange color court instead of the commonly occuring blue color court, this further proves that if more training data is supplemented the model will always learn better.

References

[1] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. ACM Transactions on Graphics (TOG), 35(4), 2016.

[2] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. ECCV, 2016.

[3] Federico Baldassarre, Diego Gonzalez Morın, Lucas Rodes-Guirao. Deep Koalarization: Image Colorization using CNNs and Inception-Resnet-v2

[4] Dataset: http://cocodataset.org/download

[5] Keras: https://keras.io/

[6] https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c