Colorization of Grayscale Images using deep CNN and ResNet feature extractor

Anonymous
UMass Amherst
anonymous@umass.edu

Anonymous
UMass Amherst
anonymous@umass.edu

Abstract

Colorization of grayscale images is a multi-modal problem which has numerous use cases in the fields of art, science and medicine. This project presents an ensemble approach of deep convolutional neural network and ResNet high level feature extractor to colorize black-and-white images of fruits, landscapes and animals without human assistance. The final output image is compared with the original colored image and scored according to the mean squared error formula to determine the efficacy of our approach. Our model achieves close to 10 times lower MSE compared to a baseline model consisting of an 8-layer CNN with no high level feature extraction.

1. Introduction

There is a significant loss of information when a colored image is transformed into a grayscale image. Getting the colored image back from the grayscale version is a challenging and interesting task since there could be many possible solutions to it. Using deep learning techniques to achieve this is not only interesting from an aesthetic perspective, but also has wide ranging applications which include coloring old black and white movies, filling colors in sketches & cartoons, coloring medical images to assist easy and early disease detection, etc.

CNNs have emerged to be one of the most popular techniques when it comes to image related computer vision tasks. The sheer versatility of CNNs along with feature extraction capabilities of widely trained networks such as ResNet, makes it a strong contender for this colorization problem [4, 9, 10, 13].

The objective of this project is to understand how deep convolutional neural networks can help achieve colorization of grayscale images. Our colorization model uses Inception-ResNet v2 [14] as a feature extractor, and is trained on three types of data: Fruits [2], Animals [1] and Landscapes [3]. Using multiple datasets makes sure that the model has a good variety of contrasting colors for robust training. Using ResNet as a feature extractor is beneficial because it has connections which are parallel to conventional CNN layers. Since these connections are constantly active, gradients can backpropagate through them efficiently and speed up the training process.

Choosing a good evaluation metric for the colorization problem is a tricky task, mainly because the problem can be treated as both classification or regression with multiple solutions for each input. The model will be evaluated using the following two techniques:

- Mean Squared Error (MSE) as the loss function between the input image and the output image.
- Colorization Turing Test: Show participants the colorized and the actual colored images and ask them to point out the original ones.

Simultaneously, we try to answer questions such as why certain transformations like conversion of RGB images to Lab color map need to be made, how the colorization can depend on the contrast amongst the images in the training data, how to avoid getting brown colored images from the model predictions and at the same time improve generalization error, strategies to upsample the image sizes since neural networks tend to shrink them with depth.

This report is organized as follows: Section 2 talks about the related work done in the field of image colorization using CNNs, Section 3 presents a detailed description of our approach to this problem. Section 4 contains the results of our implementation along with the baseline model results, and Section 5 concludes the report and includes future work.

2. Related work

Historically, image colorization techniques were restricted to human-assisted methods, with archaeologists and movie connoisseurs colorizing old art using the manual scribble method - which is time-taking and dependent on the person's skills. In recent times, with the development

Table 1. A systematic review of image colorization techniques using CNNs.

Paper	**Dataset used**	**Methodology**	**Advantages**	**Disadvantages**
[15]	1.3M images from ImageNet.	All resolution changes are achieved through spatial downsampling or upsampling between convolutional layer blocks.	This method fooled humans 32% of the time in a colourization Turing test.	The model doesn't use any pooling layers so the the parameter space is large.
[12]	10^3 high quality images of outdoor scenes.	Based on vectorized CNN. It consists of a low-level feature network, color fusion and refinement.	Patch-level evaluation establishes more exact color mapping to achieve visual satisfactory result and runtime performance.	Use of a small dataset restricts the generalization capabilities.
[6]	ILSVRC 2012 classification dataset.	VGG-16 model to extract features. Uses HSV color space and Euclidean distance as the loss function.	Introduced the idea of "hypercolumns" in a CNN which is a vector of all the activations above that pixel.	Most of the results look black and white or sepia toned.
[5]	SIFT Flow, LEARCH-120, SUN-1519, Places-1000	Extract global descriptors of the reference images, group these images into different clusters adaptively and compute the semantic histogram of each cluster.	The mixture learning technique achieves high color-semantics consistency.	Expensive computation algorithms as multiple networks are required for the ensemble.
[7]	6 scene categories of the SUN dataset.	Used LEARCH to train a second degree objective function in the color-intensity maps, comparable to a random Gaussian field. Bag-of-features retrieval using SIFT features is computed on the grayscale image.	Scene-independent as well as scene-specific training is performed using LEARCH image regressor, which results in a practical colorization model.	This method performs well only when scene information is available.
[11]	ImageNet, Sun database,	The base network is VGG-16 with the FC8 classification layer is discarded, and CONV1_1 filter layer operating on a single intensity channel instead of subtracting the mean from RGB.	The color histogram prediction prvents jarring artifacts and also handles uncertainty and ambiguity in images.	Each pixel is processed separately which is very costly. Extracting dense hypercolumns is memory intensive.

of deep neural networks, this problem has been taken up for automation.

2.1. Colorization using CNNs

CNNs view the colorization problem as a regression or a classification problem, and use parametric methods to solve them. [8] approached colorization as a regression problem, and computed local priors on small patches of the image, as well as global priors on the entire image. The low-level, mid-level and global-level features were passed through a fusing layer, the output layer consisted of a score for each pixel determining its color. [11] approached colorization as a classification problem and used the pre-trained VGG-Net model for feature extraction. It used spatially localized hypercolumns to predict the color distribution for each pixel. 1 shows few of the recent works on image colorization using CNNs along with their pros and cons.

2.2. Colorization using GANs

GANs can take image colorization to a whole different level because along with learning the input-output mapping, they can learn the loss as well. [16] used cycle GANs for image to image translation and could achieve outputs like changing the seasons in a landscape image, applying style transfer on a horse image to make it look like a zebra, etc. Along with colorization, they can be used to reconstruct images.

In our approach we explore the parametrization of the colorization problem and use CNNs to solve them.

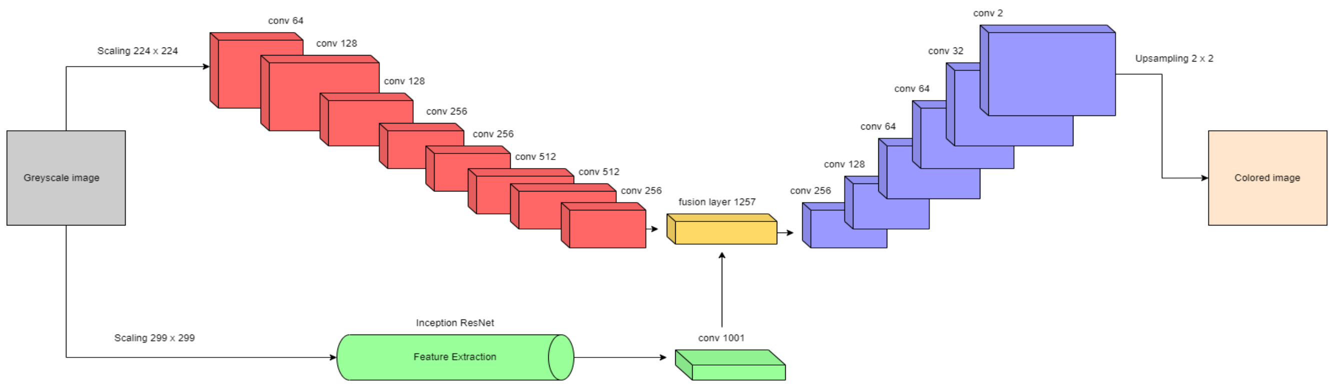

Figure 1. Image colorization model architecture

3. Approach

We have treated colorization as a regression problem where we input gray-scale images and let the model predict the colors in the image.

3.1. Data Pre-processing

Most of the input images are represented using the RGB color space. Images in RGB color space cannot be used straight away as inputs to our model. The colorization problem requires the inputs to be gray-scale and target outputs to be color information in nature. In the RGB color space all the 3 channels contain the color information, so we need a way to separate the color information from the gray-scale image. We make use of the Lab color space to model this problem. The Lab color space is used to represent images using 3 channels (L, a and b) where L stands for the luminosity channel and is equivalent to a gray-scale image which can be fed to the neural network or machine learning model. The a and b channels contain all the color information which the deep learning model is going to predict. The intuition behind using L channel as the input is that it has a lot of semantic information in it, for instance grass is usually green, sky is usually blue, sunsets are a composition of orange, red and yellow. Color information is encoded into the a and b channels. In channel a, green and red colors are represented using the negative and positive values respectively. In channel b, blue and yellow colors are represented using negative and positive values respectively. 2 shows how the three channels (L, a & b) look like in Lab color space.

We input an image of size H * W * 1 (the gray-scale component which is the L channel) and the model predicts H * W * 2 values (predictions for a and b channels in the Lab color space). We then combine the input gray-scale image with the predicted H * W * 2 values to have a complete colored image in the Lab color space of shape H * W * 3. This can then be converted back into the RGB color space for visualization and interpretation.

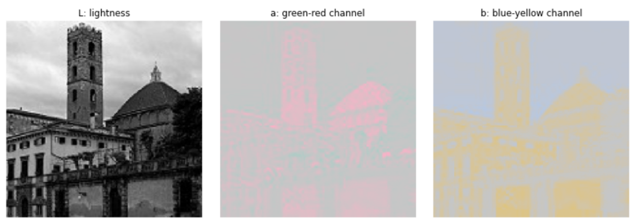

Figure 2. Lab representation of an image

3.2. Architecture

The model architecture consists of four components.

- Encoder: The input image is fed to the encoder part of the network to get mid-level feature representations from the gray-scale image.
- Feature Extractor: Instead of just passing the input image through the encoder, we also pass the input image through the pre-trained Inception-ResNet-v2 module to get feature representations. We chop off the output layer from the ResNet module and take the feature representations from the last but one layer often termed as the embeddings as input for the upcoming layers of our model.
- Fusion Layer: We merge the features extracted using the Encoder and the Inception module in this layer so that it can be fed as input to the fourth component in our neural network.
- Decoder Layer: This layer takes in the mid-level features as input and learns more complex feature representations to predict the color channels in the Lab color space.

We then combine our color channel output with the L channel which we passed as input to the network, to get the final colored image.

Table 2. Architecture overview.

Encoder Layer	Fusion Layer	Decoder Layer
Conv Layer with 64 filters of shape 3 * 3 and stride 2	Conv Layer with 256 filters of shape 1 * 1 and stride 1	Conv Layer with 128 filters of shape 3 * 3 and stride 1
Conv Layer with 128 filters of shape 3 * 3 and stride 1		UpSample
Conv Layer with 128 filters of shape 3 * 3 and stride 2		Conv Layer with 64 filters of shape 3 * 3 and stride 1
Conv Layer with 256 filters of shape 3 * 3 and stride 1		UpSample
Conv Layer with 256 filters of shape 3 * 3 and stride 2		Conv Layer with 32 filters of shape 3 * 3 and stride 1
Conv Layer with 512 filters of shape 3 * 3 and stride 1		Conv Layer with 16 filters of shape 3 * 3 and stride 1
Conv Layer with 512 filters of shape 3 * 3 and stride 1		Conv Layer with 2 filters of shape 3 * 3 and stride 1
Conv Layer with 256 filters of shape 3 * 3 and stride 1		UpSample

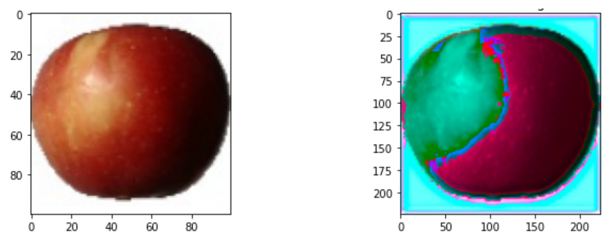
Figure 3. Results of baseline model on Fruits dataset

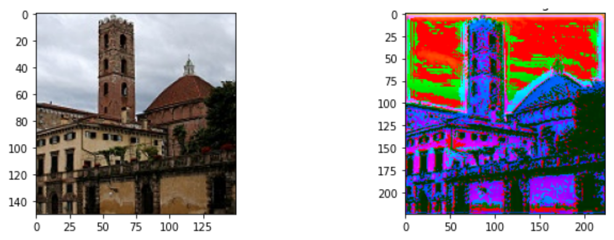
Figure 4. Results of baseline model on Landscape dataset

3.3. Loss function, training & evaluation metric

Colorization of images is a complex problem and defining a loss function isn't that easy. What makes this problem interesting is that, there is more than one possible correct answer to the problem. The input gray-scale image which is fed to the network can have many possible colorings. For our use case, we have leveraged the average mean squared loss which is calculated by summing up the squared differences of the values predicted for the a and b channels and their actual ground truth values, and then avergaing over all the samples. We use the Adam Optimizer and backpropagate the loss to update the model weights and biases.

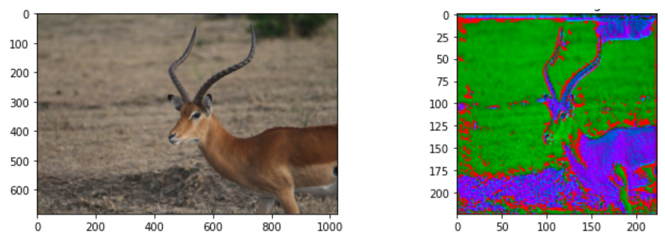
Figure 5. Results of baseline model on Animals dataset

Table 3. Baseline model MSE scores

Dataset	MSE
Fruits	4.5408e-04
Landscapes	0.0056
Animals	0.0105

4. Experiment

4.1. Baseline Model

Our baseline model involved training a deep CNN model with the following features:

- **2-D Convolution layers**: 8 convolutional layers with number of filters as {16, 32, 64, 128, 256, 128, 64, 64, 32, 2} respectively.
- **Activation function**: Leaky ReLU activation after every convolutional layer except the last one. TanH activation after the last convolutional layer.
- **Batch Normalization**: Normalization layer is added after every Leaky ReLU activation.

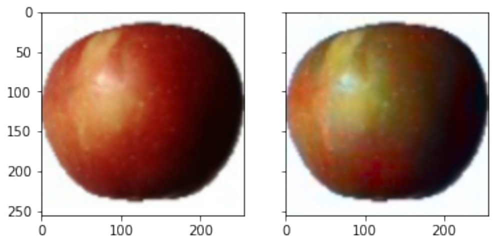

Figure 6. Results of final model on Fruits dataset

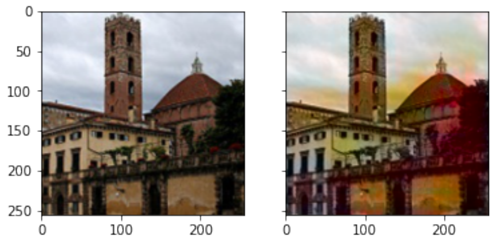

Figure 7. Results of final model on Landscape dataset

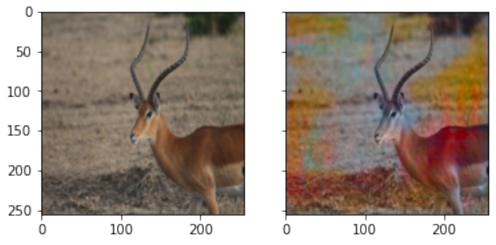

Figure 8. Results of final model on Animals dataset

- **Pooling**: A max-pooling layer is added after every alternate batch normalization layer.

We trained our baseline model on a subset of the Fruits, Animals and Landscapes dataset with 10,000 sample images from each dataset. Figures 3, 4, 5 show samples from the validation dataset and the output image produced by our model. These input images were scaled and converted to Lab format before feeding to the neural network. Table 3 shows the MSE scores achieved by the baseline model. The MSE score was calculated using Equation 1, where Y denotes predicted AB values, $\overline{Y}$ denotes ground truth AB values, and n is the total number of samples.

$$MSE = \frac{\sum_{i=0}^{n} (Y_i^2 - \bar{Y}_i^2)}{n} \tag{1}$$

4.2. Final CNN + ResNet Feature Extractor Model

The final model consisting of CNN and ResNet feature extractor was trained on the entire datasets of Fruits (90k), Animals (19k) and Landscapes (10k). These input images were scaled and converted to Lab format before feeding to the neural network. Table 4 shows the MSE scores achieved by the final model.

Table 4. Final model MSE scores

Dataset	**MSE**
Fruits	5.6623e-05
Landscapes	0.00824
Animals	0.0105

Figures 6, 7, 8 show samples from the validation dataset and the output image produced by our model. Following are our observations:

- The model performs comparatively well on the fruits dataset because this dataset contains images of isolated fruits with very little object occlusion or noise. The landscape and animal datasets are more varied with various kinds of buildings, and animals in different poses, making it difficult for the model to find a pattern.
- The model is good at colorizing the sky and grass in the landscapes dataset since these features are consistent among the input images. On the other hand, the model sometimes fails to colorize the buildings correctly because there is a variety of contrast colored buildings in the dataset.
- The model does a fair job in colorizing particular areas in the animals dataset but is unable to colorize the entire image i.e. some areas still remain in grayscale.

4.3. Turing Test

We performed a turing test for real-life evaluation of our colorization model. This experiment was conducted with four UMass Amherst students, in which they were shown original RGB images along with colorized images generated by our model, and asked to differentiate between them. The turing fraction was calculated using Equation 2 on the baseline model outputs as well as final model for comparison purposes. Lower the turing fraction score, the closer the colorized image is to the real image, so the model is performing better. Table 5 shows the results, where we can see that our model is better at fooling humans than the baseline model.

$$turingfraction = \frac{\#\, of\ correctly\ differentiated\ images}{\#\, of\ total\ images} \tag{2}$$

5. Conclusion

Convolutional neural networks are being used increasingly in computer vision applications. It is interesting to

Table 5. Colorization Turing Test Results

Dataset	**Baseline Model**	**Final Model**
Fruits	0.9881	0.72
Landscapes	1	0.9762
Animals	1	0.9930

see how AI algorithms are able to model complex real life problems. With the advancements in computational speed and low cost of storage, these types of neural networks are well suited for computer vision tasks. In this project, we have explored the colorization problem using CNNs and a pre-trained Inception-ResNet-v2 module. We use grayscale images, feed it to our network and predict the color information in the image. After having carried out the experiments, we noticed that a single CNN model with many layers isn't able to perform as well as a CNN which is fed with features extracted from the Inception-ResNet-v2 module. The ResNet model was pre-trained on images from the ImageNet dataset. Using the outputs of the last but one layer from the resnet and feeding them as input to our decoder part of the CNN helped us achieve much better results in colorization. The model did much better on fruits and buildings datasets and fairly well on the animals dataset.

5.1. Future Work

Colorization models have a lot of potential in fields like medicine and art. The feature extractor branch can be modified by using other popular architectures like AlexNet, GoogLeNet, etc to see which works best for the concerned use case. Instead of using a CNN, locally connected artificial neural networks can also be used. The intuition behind using them lies in the fact that pixels which are close to each other are more likely to have similar color information. Using a locally connected network might help in achieving better colorization results accompanied with its speed up as opposed to a standard CNN architecture.

References

[1] Animals dataset. In *https://www.kaggle.com/ navneet-surana/animaldataset*. 1

[2] Fruits 360 dataset: A dataset of images containing fruits and vegetables. In *https://www.kaggle.com/ moltean/fruits*. 1

[3] Landscapes dataset. In *https://www.kaggle.com/ huseynguliyev/landscape-classification*. 1

[4] Mahesh Agrawal and Kartik Sawhney. Exploring convolutional neural networks for automatic image colorization. In *http://cs231n.stanford.edu/reports/2017/pdfs/409.pdf*. 1

[5] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Colorization using neural network ensemble. *IEEE Transactions on Image Processing*, 26(11):5491–5505, 2017. 2

[6] Ryan Dahl. Automatic colorization, 2016. 2

[7] Aditya Deshpande, Jason Rock, and David Forsyth. Learning large-scale automatic image colorization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 567–575, 2015. 2

[8] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi 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):1–11, 2016. 2

[9] You Zhou Jeff Hwang. Image colorization with deep convolutional neural networks. In *http://cs231n.stanford.edu/reports/2016/pdfs/219*$_{R}$*eport.pdf*.1

[10] Bhuvneshwar Kushaagra G, Yash Malviya. Autocolorization of monochrome images. In *http://cs231n.stanford.edu/reports/2017/pdfs/418.pdf*. 1

[11] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *European conference on computer vision*, pages 577–593. Springer, 2016. 2

[12] Xiangguo Liang, Zhuo Su, Yiqi Xiao, Jiaming Guo, and Xiaonnan Luo. Deep patch-wise colorization model for grayscale images. In *SIGGRAPH ASIA 2016 Technical Briefs*, pages 1–4. 2016. 2

[13] Ajay Kalyan P, Puviarasi R, and Mritha Ramalaingam. Image colorization using convolutional neural networks. In *Proceedings of International Conference on Recent Trends in Computing, Communication Networking Technologies (ICRTCCNT) 2019*. 1

[14] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017. 1

[15] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016. 2

[16] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 2