# Deep Koalarization: Image Colorization using CNNs and Inception-Resnet-v2

E4040.2021Spring.YLEE.report
Yena Lee yl4315
*Columbia University*

## Abstract

*The original paper suggests a model that combines its own deep convolutional neural network and features extracted from the Inception-ResNet-v2 pretrained model. The model successfully performs on colorizing high-level images such as the sea, forests, or the sky particularly; on image types that require smaller details; however, the model shows lower accuracy. The authors also conducted a user study to measure "public acceptance." They ran their model with a reduced subset of ImageNet dataset to reduce running time; therefore, the network is expected to be improved when training over a larger dataset.*

## 1. Introduction

Since a gray-scale image contained limited information compared to colored images, adding color components to a gray-scale image can provide more useful information of the image. In order to train gray-scale images using features from pre-trained models such as Inception[1], ResNet[2], or VGG[3], prior colorization step is required to improve the results. In this paper, the authors build a model that automates the auto-colorization process to a certain extent with feature extractor from a pre-trained model.

2002, Welshet. al. presented an approach to colorize images by transferring related colors of the images. The approach takes low-level features and suggests multi-modality of color values in pixels. The other research suggests a graffiti-based approach that requires users to determine color of some image areas before training. This method inspired animators and cartoon engineers; however, it is highly dependent on user's skills and judgements. Over the years, improvements on reducing error rates using Coulvolutional Neural Network for object recognition has brough tremendous attention to deep learning in the computer vision community. Cheng (2016) suggests a coloring method that use mid-level features and encoding to colorize images. In the original paper, the model architecture is written from scratch and also suggests a validation; however, the architecture also combines with the high-level features of a pre-trained model.

The authors used a subset of ImageNet dataset for training due to time constraints, which feeds restricted varieties of images to the model when training. Nevertheless, they compare other colorizing models with different approaches conducted by other researchers in the results section.

## 2. Summary of the Original Paper

### 2.1 Methodology of the Original Paper

Generally color images are represented in RGB color space, meaning the images consist of three color layers; a red, a green, and a blue layers. Each pixel value in a corresponding layer represents its brightness in the range of values from 0 to 255.

Since the constructed model in the paper is to estimate full color components after feeding images in the gray scales, color images with RGB scale need to be converted in CIE L*a*b* color space for the input images. CIE L*a*b* color space, not like RGB color space, separates the luminance components layer and the color components layer.

The network is mainly divided four parts: encoder, feature extractor, fusion layer, and decoder parts. First, all of the pixel values in each image layer need to be rescaled and centered in the range of [-1, 1] in preprocessing part. Input images for encoder part is scaled in 244*244; and input images for feature extractor embedding part is scaled in 299*299.

The encoder part requires to construct a CNN layers from scratch to process gray scale images. 8 convolutional layers are written with 3*3 kernels. To keep the input size, padding is used. In the first, third, and fifth layers stride of 2 is used to reduce the dimension of the outputs. This part gives an output of H/8 * W/8 * 512 feature representation.

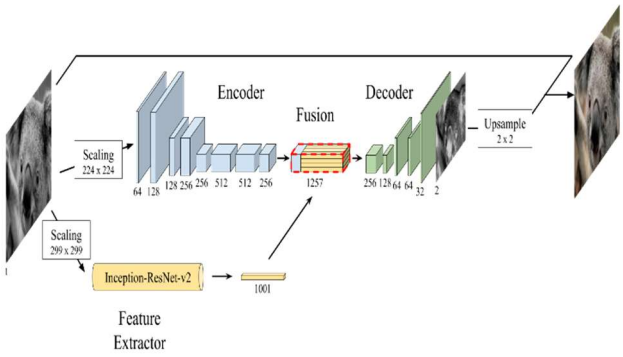
Feature Extractor part is to extract high level features of pre-trained model Inception-ResNet-v2 model and create an embedding with our input images. After preprocessing input images with a scale 299*299, the luminance images need to be stacked three times to satisfy pre-trained model's dimension requirements. After feeding the images to the network, features can be extracted from the last layer before the softmax activation function.

In the fusion layer part, the embedding created from last part needs to be replicated in HW/8^2 times then concatenated to the features created in the convolutional layers of the encoder. This process creates a single volume of shape H/8 * H/8 * 1257. 256 convolutional kernels of size 1*1 is applied, and it generates H/8 * W/8 * 256 feature dimensions.

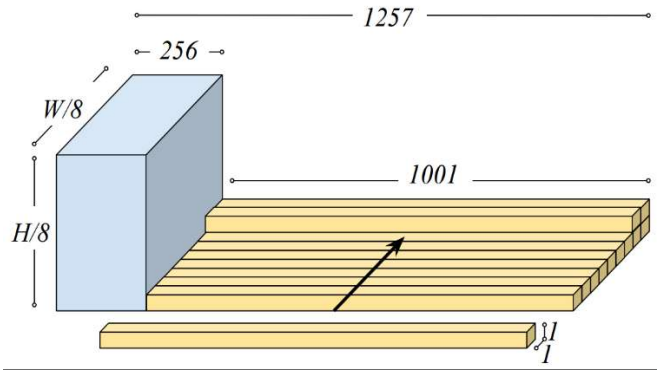
The last part of the network is decoding part. The decoder takes the volume from previous part and applies convolutional layer and up-sampling method to achieve

final output layer volume with H * W * 2, which contains color components (a*b*) in CIE L*a*b* color space.

In the training part, Mean Squared Error function is selected as the object function of the model to quantify the model loss between the estimated values in the color components and the real values of the training images. While training, stochastic gradient descent with Adam optimizer is used with learning rate of 0.001. With a fixed input image, batch processing is allowed during training.




| Layer | Kernels | Stride | Layer | Kernels | Stride | Layer | Kernels | Stride |
|---|---|---|---|---|---|---|---|---|
| conv | 64 x (3 x 3) | 2 x 2 | fusion | - | - | conv | 128 x (3 x 3) | 1 x 1 |
| conv | 128 x (3 x 3) | 1 x 1 | conv | 256 x (1 x 1) | 1 x 1 | upsamp | - | - |
| conv | 128 x (3 x 3) | 2 x 2 | | | | conv | 64 x (3 x 3) | 1 x 1 |
| conv | 256 x (3 x 3) | 1 x 1 | | | | conv | 64 x (3 x 3) | 1 x 1 |
| conv | 256 x (3 x 3) | 2 x 2 | | | | upsamp | - | - |
| conv | 512 x (3 x 3) | 1 x 1 | | | | conv | 32 x (3 x 3) | 1 x 1 |
| conv | 512 x (3 x 3) | 1 x 1 | | | | conv | 2 x (3 x 3) | 1 x 1 |
| conv | 256 x (3 x 3) | 1 x 1 | | | | upsamp | - | - |




## 2.2 Key Results of the Original Paper

After training, the paper reproduced some images with the network and the results for some images were satisfying and realistic. However, since not all the subset of the original data is used in the training, the authors point out the network performance will improve better when they feed the network with the whole ImageNet dataset. The model performance was better with green vegetation images among the image results.

The authors also compared the results obtained from the model with other colorization approaches introduced in three different papers as well. In the presented results, motorcycle images overperform the other methods, creating more realistic colors.

They also conducted a user study to evaluate their performance. Nine artificially re-colorized images and three original images are selected for the evaluation. 41 different users participated in the poll to pick images which they believe are artificially re-colorized ones.

Overall, 45.87% of users miss-classified the faked images as original ones. Other than the evaluation process, they tested the model on historical pictures as well. Since the original images are stored in gray scales, the results have to be evaluated with personal judgements..

## 3. Methodology (of the Students' Project)

In this paper I decided reconstruct the model with same structure used in the original paper. I used a classified image dataset to train my model. The dataset includes 34,745 images classified with 1,000 categories in the training and 3,923 images, 1,000 categories in the validation data randomly selected from the ImageNet dataset. The original images are stored in color scales. Before training the model, preprocessing the color images needs to be done to make all of the scales in the each images same for feeding encoder part and feature extractor part respectively.

For the pre-trained Inception-ResNet-v2 embedding, the volume of scales is rescaled in 299*299 pixels; for the encoder input images, it is rescaled in 244*244 pixels. Also, the colored input images are converted to a CIE*L*a*b color space in this process manually before feeding the model as we want to reproduce grey scale images in colored versions.

In the encoding part, I chose the exact same architecture that are used in the original paper. The encoder layers accept input size of H*W*1 which is luminance components in an image structure, and the output size of this part is H/8 * W/8 * 512. The encoder includes 8 convolutional layers in its architecture.

In the feature extraction part, I downloaded the weights of the Inception-ResNet-v2 pre-trained model in h5 file and used its high-level features to create an embedding. The whole layers were loaded in the part except for its last softmax activation layer. The embedding, then, is piled three times to fit the model's dimension requirements.

In the fusion layer part, I combined the embedding features with the encoding features.
For the last part of the network, the decoding part is constructed with 8 convolutional layers applied with up-sampling and stochastic gradient descent with Adam optimizer with learning rate of 0.001.

## 3.1. Objectives and Technical Challenges

Since the original ImageNet dataset is tremendously huge in its size (more than14,000,000 images), in the original dataset, the authors used 10% of the original ImageNet dataset in the training which is approximately 60,000 images; however, I found using 10% of the original dataset still too large for myself to train the model due to time and memory constraints. Therefore, I downloaded mini version of the original ImageNet dataset from Kaggle. The dataset is 3.95GB containing 34,745 images classified with 1,000 categories in the training and 3,923 images, 1,000 categories in the validation data from original ImageNet dataset. In the original paper, the image dataset is converted in a continuous TFRecord since storing each image and embedding in separately will impact performance during training stage, but I just saved the image dataset and embedding respectively.

## 3.2.Problem Formulation and Design Description

Since the network built in the original paper was not extensive to reconstruct, I used exact same structures used in the original data when building my model.

## 4. Implementation

## 4.1. Deep Learning Network

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_13 (InputLayer) | [(None, 256, 256, 1) | 0 | |
| conv2d_255 (Conv2D) | (None, 128, 128, 64) | 640 | input_13[0][0] |
| conv2d_256 (Conv2D) | (None, 128, 128, 128 | 73856 | conv2d_255[0][0] |
| conv2d_257 (Conv2D) | (None, 64, 64, 128) | 147584 | conv2d_256[0][0] |
| conv2d_258 (Conv2D) | (None, 64, 64, 256) | 295168 | conv2d_257[0][0] |
| conv2d_259 (Conv2D) | (None, 32, 32, 256) | 590080 | conv2d_258[0][0] |
| conv2d_260 (Conv2D) | (None, 32, 32, 512) | 1180160 | conv2d_259[0][0] |
| input_14 (InputLayer) | [(None, 1000)] | 0 | |
| conv2d_261 (Conv2D) | (None, 32, 32, 512) | 2359808 | conv2d_260[0][0] |
| repeat_vector_5 (RepeatVector) | (None, 1024, 1000) | 0 | input_14[0][0] |
| conv2d_262 (Conv2D) | (None, 32, 32, 256) | 1179904 | conv2d_261[0][0] |
| reshape_5 (Reshape) | (None, 32, 32, 1000) | 0 | repeat_vector_5[0][0] |
| concatenate_5 (Concatenate) | (None, 32, 32, 1256) | 0 | conv2d_262[0][0]<br>reshape_5[0][0] |
| conv2d_263 (Conv2D) | (None, 32, 32, 256) | 321792 | concatenate_5[0][0] |
| conv2d_264 (Conv2D) | (None, 32, 32, 128) | 295040 | conv2d_263[0][0] |
| up_sampling2d_6 (UpSampling2D) | (None, 64, 64, 128) | 0 | conv2d_264[0][0] |
| conv2d_265 (Conv2D) | (None, 64, 64, 64) | 73792 | up_sampling2d_6[0][0] |
| up_sampling2d_7 (UpSampling2D) | (None, 128, 128, 64) | 0 | conv2d_265[0][0] |
| conv2d_266 (Conv2D) | (None, 128, 128, 32) | 18464 | up_sampling2d_7[0][0] |
| conv2d_267 (Conv2D) | (None, 128, 128, 16) | 4624 | conv2d_266[0][0] |
| conv2d_268 (Conv2D) | (None, 128, 128, 2) | 290 | conv2d_267[0][0] |
| up_sampling2d_8 (UpSampling2D) | (None, 256, 256, 2) | 0 | conv2d_268[0][0] |

Total params: 6,541,202
Trainable params: 6,541,202
Non-trainable params: 0

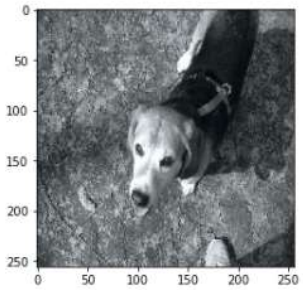
## 4.2. Software Design

The high-level feature of pre-trained model of Inception-ResNet-v2 was implemented first in the jupyter notebook for creating embedding features. Then I constructed encoder, fusion, and decoder layers. After rescaling the input images, the images were fed in the network to train.

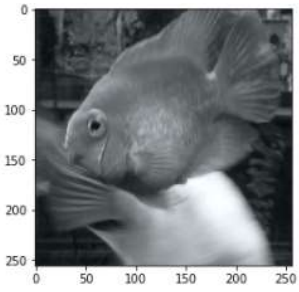
## 5. Results

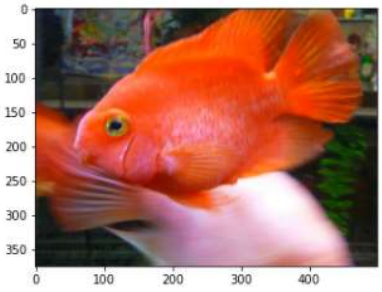
## 5.1. Project Results

Above images of a dog and fish are generated images after training the network. The result was quite different with my expectation and disappointing.







## 5.2. Comparison of the Results Between the Original Paper and Students' Project

The results in the original paper were successful, but I used small dataset to train my network and used small batch and epoch sizes, the generated images for test images were hard to say it is colorized. Due to memory constraints, I had to train the network with only 2 epoch; however, the loss in the training stage was decreasing. If I had enough memories and time, I think the model I constructed would still have room for improvements.

## 5.3. Discussion of Insights Gained

## 6. Conclusion

Training was not successful at all. Due to memory and constraints, the model didn't seem to learn. I didn't use the dataset I stated in the above section since kernel kept on dying when feeding the images even when the dataset I was going to use was not that huge. The constructed network was still running, however, so if I could resolve the memory problem, the results are considered to be improved at some levels.


## 6. Acknowledgement

Community codes on GitHub
[1]https://github.com/ajaychaudhary7/Image-Colorization
[2]https://github.com/humblefool01/Image-Colourization
[3]https://github.com/ahemaesh/Deep-Image-Colorization


## 7. References


[1]https://github.com/ecbme4040/e4040-2021spring-project-ylee-yl4315
[2]Baldassarre F, Morín DG, Rodés-Guirao L. Deep Koalarization: Image Colorization using CNNs and Inception-ResNet-v2 2017:1– 12.
[3]https://www.kaggle.com/ifigotin/imagenetmini-1000


## 8. Appendix

### 8.1 Individual Student Contributions in Fractions

| | Yl4315 | | |
|---|---|---|---|
| Last Name | Lee | | |
| Fraction of (useful) total contribution | 1 | | |
| What I did 1 | . | | |
| What I did 2 | . | | |
| What I did 3 | . | | |

8.2 If/as needed: additional diagrams, source code listing, circuit schematics, relevant datasheets etc.