भारतीय प्रौद्योगिकी संस्थान
काशी हिन्दू विश्वविद्यालय
IIT
INDIAN INSTITUTE OF TECHNOLOGY
BANARAS HINDU UNIVERSITY

Exploratory Project

Image Colourisation via Raspberry Pi

Exploratory Project Submitted by

1. Raunak Pandey
   21095092
   raunak.pandey.ece21@itbhu.ac.in

2. Anjali Tiwari
   21095016
   anjali.tiwari.ece21@itbhu.ac.in

3. Yalamanchi Giridhar Madhav
   21095130
   ygiridhar.madhav.ece21@itbhu.ac.in

4. Jarpula Ajay Nayal
   21095054
   jarpulaajay.nayak.ece21@itbhu.ac.in

Under the guidance of Mr Atul Kumar.

Abstract

Manual colourization of black and white images is an uphill task and inefficient. It has been attempted using Photoshop editing, but it is difficult as it requires extensive research, and a picture can take up to one month to colorise. A pragmatic approach to the task is to implement sophisticated image colourization techniques. Colourisation of grayscale images has become a more researched area in recent years, the most recent development is using GANs and U-net. We attempt to apply this concept to the colourisation of Victorian images, which have a tinted colour scheme and are not precisely very brightly coloured. Previous similar research focused mainly on the colourization of natural images, while we have trained on Victorian images. We propose to train the model, get inferences, and deploy it on a Raspberry Pi. We are using CNN and pre-trained Resnet to train our model due to the limited computer resources available to us.

Introduction - with the description of the Problem statement

When photography was just starting in old times, most images were in black and white (B&W). Hence, efforts to colourize old B&W images started to give the image a different perspective and a beautiful insight into the captured moments. Colourisation efforts began to appear in the early 1900s, using paints and brushes. This painstaking process took days, sometimes weeks, to generate a rough recreation of reality. The trend has shifted to the use of computer software such as Adobe Photoshop, GIMP etc, but the procedure remains the same. The present-day techniques for manual colourization of these B&W images are: cleaning the image, adjusting the image tones and

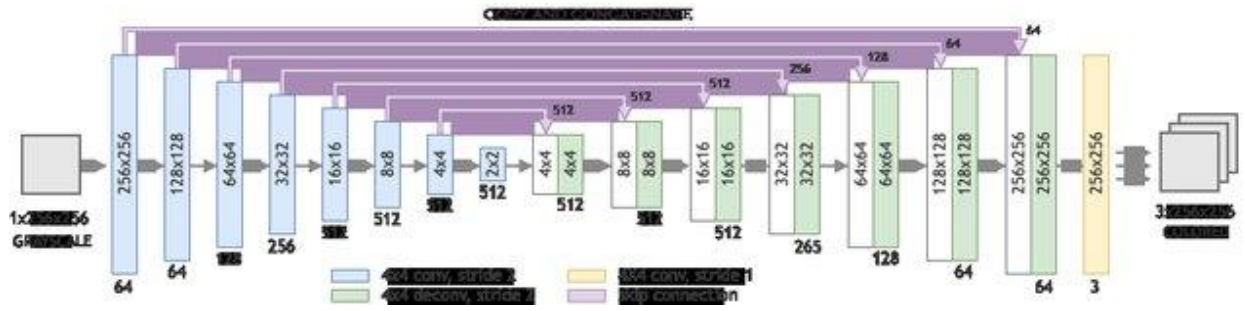
contrast, converting the image to CMYK and finally, adding solid colour to specific entities in the picture. But even digital colourisation using software like Photoshop, which helped in colourisation, pixel by pixel, along with improved handling of colour bleeding and colour continuity, and reduced manual work to some extent, now seems tedious. Thanks to colour cameras, a massive gallery of photographs is available. It offers great information to determine the colour schemes of familiar objects, scenes, lighting conditions, and colour intensities. With advancements in technology and extensive research in deep learning, models can be trained to learn knowledge and then apply it to colourise old photographs.

Using such techniques to colorize the photographs can modernise and automate how things are done and relieve the pressure on the colourizing artists to some extent. We implemented different CNN and GAN models used for image colourizing and provided their quantitative and qualitative

comparisons. We also show that incorporating pre-trained models can significantly improve performance, making the training and hyperparameter tuning processes less cumbersome. We created a custom dataset of high-resolution images; categorised them according to scenery, background, and artistic theme since the colours involved in such photos are generic and not complex.

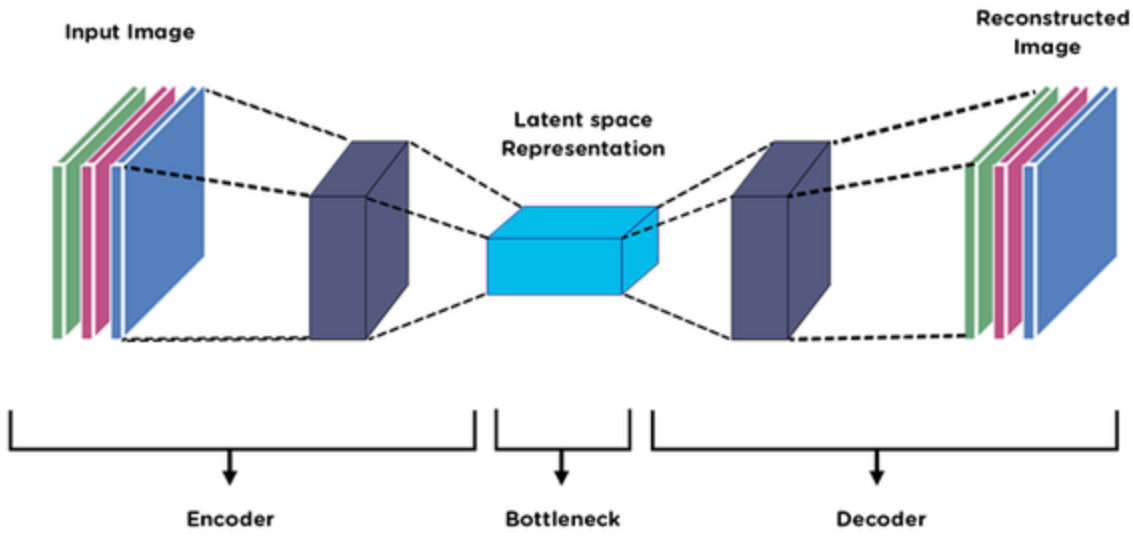
Detailed literature survey of existing techniques

1. Neural Networks: Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs) are popular choices for colourization tasks. CNNs can learn to map grayscale input images to their corresponding coloured versions. GANs consist of a generator network that produces coloured images and a discriminator network that distinguishes between real and generated images. The generator and discriminator networks are trained simultaneously to improve the quality of generated colour images.



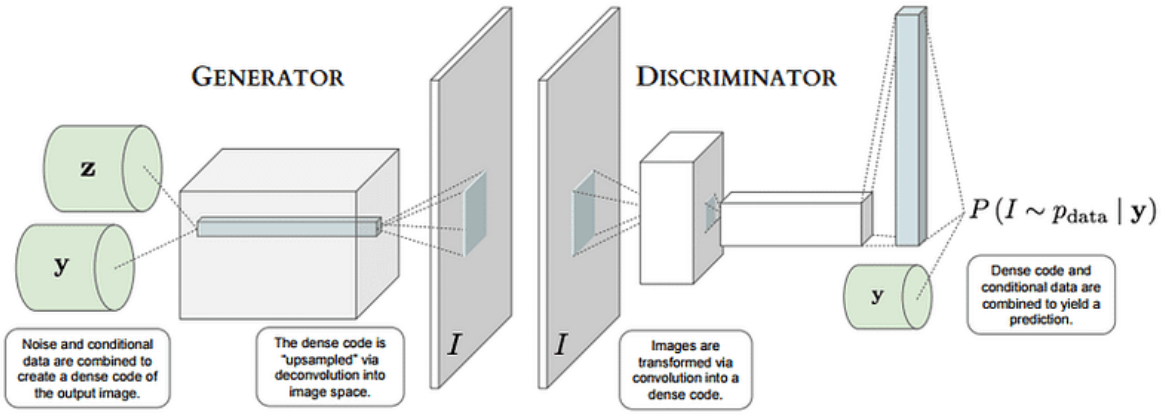
U-net structure used in GANs training for the colourisation of black and white images

2. Autoencoders: Autoencoders are neural networks that learn to encode input images into a lower-dimensional representation and then decode them back to their original form. By training an autoencoder on a large dataset of coloured images, it can learn to capture the patterns and relationships between grayscale and colored versions. Once introduced, the autoencoder can be used to colorize new black-and-white images.



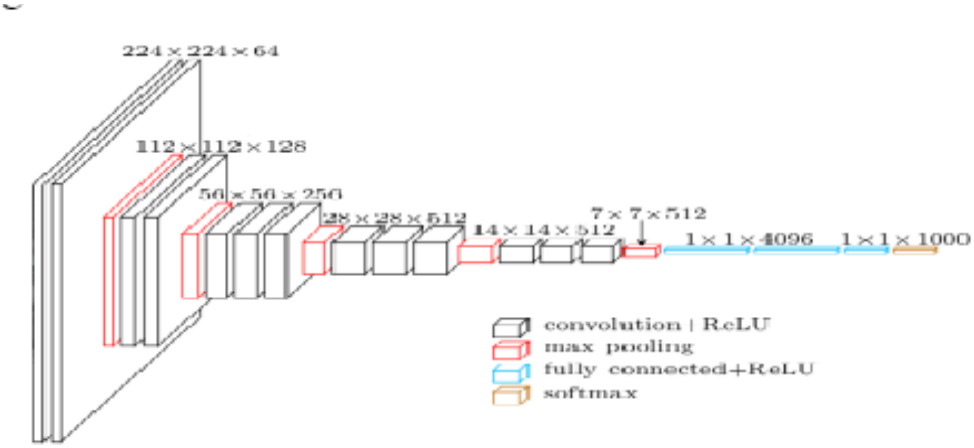
Structure of an Autocoder aimed at colourisation

3.Conditional Image Generation: Another approach is to use conditional image generation models such as Conditional Variational Autoencoders (CVAEs) or Conditional GANs (cGANs). These models can take a grayscale image as input along with a conditioning variable, such as a colour hint or a reference image, and generate a corresponding coloured output. The conditioning variable provides additional information to guide the colourization process.



Conditional GAN

4.Transfer Learning: Pretrained models such as VGGNet or ResNet, trained on large color image datasets like ImageNet, can be fine-tuned for colorization tasks. By removing the last layer(s) and replacing them with layers specific to the colorization task, these models can be adapted to generate color information from grayscale images. This makes the model less general but improves their accuracy on the aforesaid task.



Neural network architecture of VGG19

Some famous and notable research papers for the same task are:

1. "*Colorful Image Colorization*" by Richard Zhang, Phillip Isola, and Alexei A. Efros (2016): This paper introduces a deep learning approach to automatic colorization using a CNN architecture. They proposed a loss function that combined classification and regression components to encourage globally consistent and plausible colorings.

2. "*Deep Koalarization: Image Colorization using CNNs and Inception-ResNet-v2*" by Gustav Larsson, Michael Maire, and Gregory Shakhnarovich (2017): The authors presented a method that utilised a deep CNN architecture inspired by the Inception-ResNet-v2 network, for image colourization. They also introduced a novel loss function that encouraged natural colorization.

3. "*Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification*" by Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa (2016): This paper proposes a deep learning approach that simultaneously performs colourization and classification. They introduced a global and local image prior network that incorporated local and global information to generate colour predictions.

4. "*Colorization as a Proxy Task for Visual Understanding*" by Richard Zhang, Phillip Isola, and Alexei A. Efros (2017): This work explored using colorization as a proxy task for visual understanding. They demonstrated that by training a colorization model on a large dataset, they could extract features useful for various computer vision tasks such as object recognition and scene classification.

5. "*Real-Time User-Guided Image Colorization with Learned Deep Priors*" by Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S. Lin, Tianhe Yu, and Alexei A. Efros (2017): The authors presented a user-guided

image colourization approach that allowed users to interactively guide the colourization process. They used a deep CNN trained on a large dataset to generate colour suggestions that the user could refine.

Image showing results of the paper.

These papers contribute to machine learning-based black-and-white image colourization. They provide insights into various techniques and architectures used to tackle the problem and can serve as references for further exploration.

Methodology:

The methodology of building an ML model for colourizing black-and-white images and deploying it on a Raspberry Pi involves several steps. Here's an overview of the process:

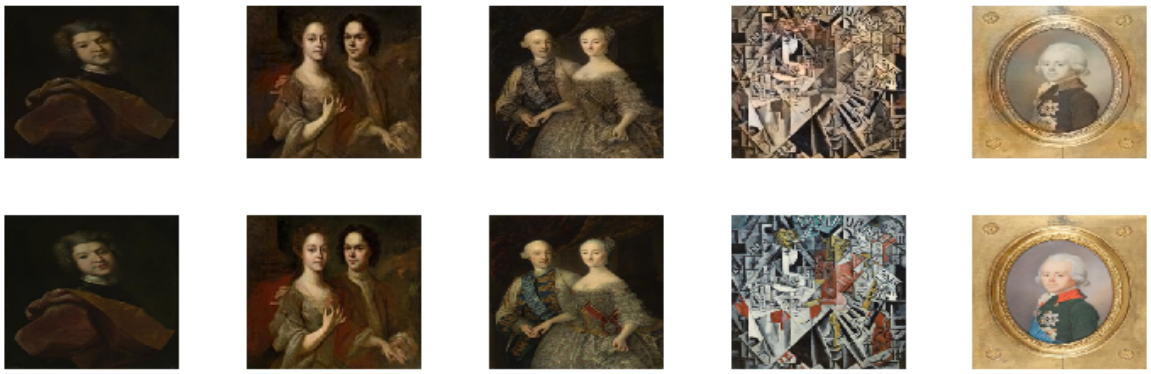
1. Dataset Preparation:
(DATA)We have used this dataset that consists of different Victorian arts and paintings. We have chosen this dataset because we often find models that colour typical images, but one of the essential applications of colourising black and white images is restoring old photographs thus, training the model on a dataset consisting of old pictures having the tinted shade will actually address that problem particularly.

2. Model Development and Training:
There were several options, such as a Convolutional Neural Network (CNN) or a Generative Adversarial Network (GAN), for choosing the ML model to perform the desired operations. We chose to work with a CNN, followed by inceptions from ResNet. We then split the dataset into training and validation sets, followed by choosing the particular parameters for the model decided. Then training the model for epochs and optimise the model's parameters to minimise the difference between the predicted colours and the ground truth colours.

```
Model: "model"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #     Connected to
=================================================================================================
 input_3 (InputLayer)        [(None, 256, 256, 1       0           []
                             )]

 conv2d_203 (Conv2D)         (None, 256, 256, 12       1280        ['input_3[0][0]']
                             8)

 max_pooling2d_4 (MaxPooling2D)  (None, 128, 128, 12   0           ['conv2d_203[0][0]']
                             8)

 conv2d_204 (Conv2D)         (None, 128, 128, 12       262272      ['max_pooling2d_4[0][0]']
                             8)

 conv2d_205 (Conv2D)         (None, 128, 128, 12       147584      ['conv2d_204[0][0]']
                             8)

 max_pooling2d_5 (MaxPooling2D)  (None, 64, 64, 128)   0           ['conv2d_205[0][0]']

 conv2d_206 (Conv2D)         (None, 64, 64, 256)       524544      ['max_pooling2d_5[0][0]']

 conv2d_207 (Conv2D)         (None, 64, 64, 256)       590080      ['conv2d_206[0][0]']

 max_pooling2d_6 (MaxPooling2D)  (None, 32, 32, 256)   0           ['conv2d_207[0][0]']

 conv2d_208 (Conv2D)         (None, 32, 32, 256)       1048832     ['max_pooling2d_6[0][0]']

 input_2 (InputLayer)        [(None, 1000)]            0           []

 conv2d_209 (Conv2D)         (None, 32, 32, 256)       590080      ['conv2d_208[0][0]']

 repeat_vector (RepeatVector)   (None, 1024, 1000)     0           ['input_2[0][0]']

 conv2d_210 (Conv2D)         (None, 32, 32, 256)       590080      ['conv2d_209[0][0]']

 reshape (Reshape)           (None, 32, 32, 1000       0           ['repeat_vector[0][0]']
                             )

 concatenate (Concatenate)   (None, 32, 32, 1256       0           ['conv2d_210[0][0]',
                             )                                      'reshape[0][0]']

 conv2d_211 (Conv2D)         (None, 32, 32, 256)       321792      ['concatenate[0][0]']

 conv2d_212 (Conv2D)         (None, 32, 32, 128)       295040      ['conv2d_211[0][0]']

 conv2d_213 (Conv2D)         (None, 32, 32, 64)        73792       ['conv2d_212[0][0]']

 up_sampling2d (UpSampling2D)   (None, 64, 64, 64)     0           ['conv2d_213[0][0]']

 conv2d_214 (Conv2D)         (None, 64, 64, 128)       73856       ['up_sampling2d[0][0]']

 up_sampling2d_1 (UpSampling2D)  (None, 128, 128, 12   0           ['conv2d_214[0][0]']
                             8)

 conv2d_215 (Conv2D)         (None, 128, 128, 64       131136      ['up_sampling2d_1[0][0]']
                             )

 conv2d_216 (Conv2D)         (None, 128, 128, 64       36928       ['conv2d_215[0][0]']
                             )

 conv2d_217 (Conv2D)         (None, 128, 128, 32       8224        ['conv2d_216[0][0]']
                             )

 conv2d_218 (Conv2D)         (None, 128, 128, 2)       578         ['conv2d_217[0][0]']

 up_sampling2d_2 (UpSampling2D)  (None, 256, 256, 2)   0           ['conv2d_218[0][0]']

=================================================================================================
Total params: 4,696,098
Trainable params: 4,696,098
Non-trainable params: 0
_________________________________________________________________________________________________
```

Model Architecture used for fine tuning

3. Model Evaluation:
We compared the results produced from the model to the ground truth and observed that the results produced are okay for the computational limitations provided to us.



Here the first row depicts the predicted output by our model while the second row is the ground truth.

4. Model Optimization for Raspberry Pi:
Once the model is trained and evaluated, it must be optimised for deployment on a resource-constrained device like Raspberry Pi. Since the Raspi we have chosen to work with has the memory constraint of 1 GB, we cannot deploy our trained model into the Raspi. Thus we were required to convert the model into Tensorflowlite model is smaller and easier to deploy on the Raspberry Pi.

5. Deployment on the Raspberry Pi:
Transfer the optimised model to the Raspberry Pi[1]. Then we install the necessary libraries and dependencies on the device, such as Tensorflow, Tflite, Numpy, and PIL.
Then we load the Tflite converted model on the Raspberry Pi that can accept black and white images as input and generate colourized versions using the trained model.

[1] The Raspberry Pi used has the following specifications: 1.4GHz 64-bit quad-core processor, dual-band wireless LAN,Bluetooth4.2/BLE, faster Ethernet, and Power-over-Ethernet support (with separate PoE HAT).

```python
import tensorflow as tf

# Convert the model
converter = tf.lite.TFLiteConverter.from_saved_model("/content/drive/MyDrive/dataset/dataset_updated/") # path to the SavedModel directory
tflite_model = converter.convert()

# Save the model.
with open('model.tflite', 'wb') as f:
  f.write(tflite_model)
```

Converting tensorflow model to tflite model

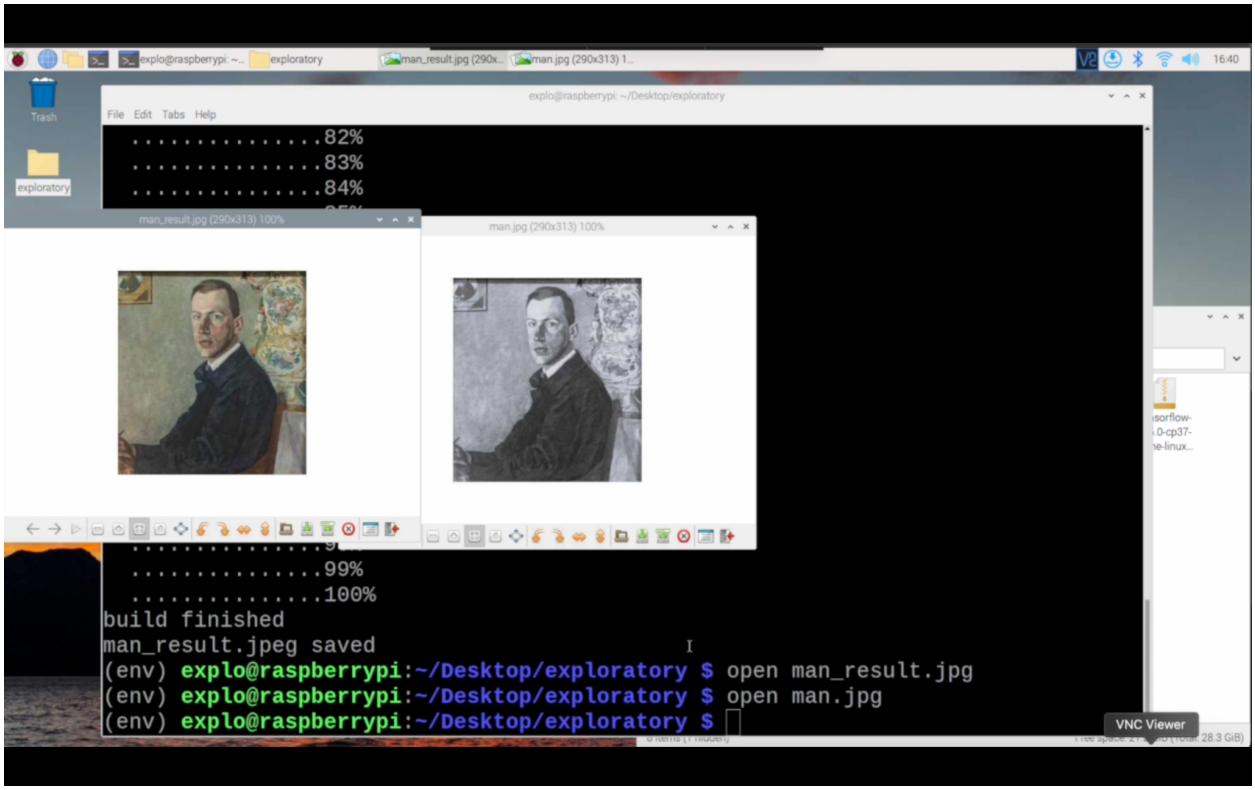
7. Testing and Iteration:

We now validate the deployed model by testing it with various black-and-white images. This could be further optimised[2].

Thus we developed the ML model for colourizing black and white images and deployed it on a Raspberry Pi, which performs colourization tasks locally on the device.

[2] If the model's inference on the Raspberry Pi is slow, consider techniques like model quantization, model pruning, or optimizing the code for the specific hardware architecture of the Raspberry Pi to improve the inference speed.

Preliminary Results and Insights



Result of the colourisation on Raspberry pi.

The model was successfully able to convert black and white images into its coloured version. When tried on general images this model will often lead to some errors due to its restriction of being trained on only Victorian images. This could be improved by improving the computation power,however, this comes at the cost of the size of the device.

Inferences, Discussions, and Conclusions :

Although this technique is not intended for clinical diagnosis with medical image data, it might be used in anatomical illustrations or to enhance scientific presentations. Although we show that the algorithm works well in several image domains, we do not claim that the technique will work on most images. When one considers only a Small neighbourhood size around a pixel, it is often impossible to determine whether that neighbourhood belongs to one texture or another. However, by using high-resolution images and more prominent neighbourhoods we can obtain improved results. Further, more images can be colourized using the primary method provided but with better texture classification methods at the expense of simplicity and computation time. The algorithm's running time for one image can range from 15 seconds to 4 minutes on a Pentium III 900 MHz CPU using optimised MATLAB code. Running time will vary depending on the number of samples used for comparison, the number of batches, the neighbourhood size, and the size of the images. Most images can be colourised reasonably well in under a minute.

In this project, we have formulated a new, general, fast, and user-friendly approach to the problem of colourizing grayscale images. While standard methods accomplish this task by assigning pixel Colors via a global color palette, our technique empowers the user to first select a suitable color image and then transfer the color Mood of this image to the grey level image at hand. We have intentionally kept the basic technique simple and general by not requiring registration between the images or incorporating spatial information. Our technique can applied to a larger class of images by adding a small amount of user guidance . Currently, the L2 is used to measure texture similarity within the image. In the future, the technique can be substantially improved by using more sophisticated measures of texture similarity.

References and Bibliography

- https://arxiv.org/abs/1705.02999 -"Real-Time User-Guided Image Colorization with Learned Deep Priors" by Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S. Lin, Tianhe Yu, and Alexei A. Efros (2017)
- https://www.researchgate.net/publication/314942807_Colorization_as_a_Proxy_Task_for_Visual_Understanding -"Colorization as a Proxy Task for Visual Understanding" by Richard Zhang, Phillip Isola, and Alexei A. Efros (2017)
- https://waseda.elsevierpure.com/en/publications/let-there-be-color-joint-end-to-end-learning-of-global-and-local- -"Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification" by Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa (2016)
- https://www.researchgate.net/publication/321744935_Deep_Koalarization_Image_Colorization_using_CNNs_and_Inception-ResNet-v2 -"Deep Koalarization: Image Colorization using CNNs and Inception-ResNet-v2" by Gustav Larsson, Michael Maire, and Gregory Shakhnarovich (2017)
- https://arxiv.org/abs/1603.08511 -"Colorful Image Colorization" by Richard Zhang, Phillip Isola, and Alexei A. Efros (2016)