IMAGE COLORIZATION USING TRANSFER LEARNING

Shashank
M11115809@mail.ntust.edu.tw
NTUST,Taiwan

Megha Dey Sarkar
M11115811@mail.ntust.ed.tw
NTUST,Taiwan

Abstract

Grayscale photos were transformed into aesthetically appealing color images through the process of colorization. We attempted to use the technique of transfer learning and autoencoders to convert grayscale images into colorful ones, enhancing their visual appeal. After exploring various network architectures, color spaces, and image handling techniques, we finally experimented with using the LAB format image, which covered almost all combinations of color palettes, along with the concept of autoencoders to achieve our goal. The primary objective was to convince the viewer that the result was authentic.

1. Introduction

Black and white image colorization has been the subject of extensive study in the fields of computer vision and machine learning over the past decade. The process involved creating a colorized image that captured the semantic colors and tones of the given grayscale input image. Image colorization found applications in various areas, including medical imaging, night vision cameras, denoising, and the restoration of obsolete images. Essentially, colorization involved assuming the presence of color information. Assigning three-dimensional RGB (Red, Green, and Blue) color information to each pixel of a grayscale image based on intensity in a visually pleasing and believable manner was technically challenging. However, image colorization systems primarily relied on human input due to the ambiguous nature of colors.

Before the invention of color cameras, black and white images were used to document memories, but these images lacked the full story conveyed by colors. Traditional methods of image colorization, such as point-based colorization, masking, or segmentation, were time-consuming and expensive as they required human intervention to determine the colors. In our project, we employed an automated data-driven technique called autoencoders for image colorization. Additionally, we utilized transfer learning, leveraging pre-trained models and their knowledge from large color image datasets to enhance the colorization process.Our project aimed to construct an effective convolutional neural network for converting grayscale images into RGB images, thereby bringing colors to life in black and white photographs. (Fig-1).

Fig 1. Sample Grayscale image (left) and RGB image (right).

2. Colorization Methods

The use of colorization in research articles has exhibited significant variation. Categorizing the diverse problem-solving approaches into appropriate categories has proven to be a challenging task due to the creativity and diversity of these approaches. Many current articles classify colorization techniques based on the level of user engagement required for problem solution and the method of data retrieval [1].

Federico Baldassarre et al. [2] employed deep learning methods to colorize grayscale photos. They utilized high-level features extracted from InceptionResNetV2, pre-trained on Imagenet, and integrated them with a deep convolutional neural network. The authors developed a comprehensive convolutional architecture capable of processing photos with varying sizes and aspect ratios.

Richard Zhang et al. [3] proposed a fully automated method for colorization, which differed from earlier techniques that often resulted in desaturated colorization and required extensive user input. The author presented the colorization problem as a classification challenge and employed class-rebalancing during training to produce more vibrant and colorful results.

Gustav Larsson et al. [4] concentrated on developing a fully automatic image colorization system. They aimed to

incorporate both low-level and semantic representations in their approach and trained the model to predict per-pixel color histograms rather than using a regression-based approach.

In our project, we utilized autoencoders that employed backpropagation with the target value set to be equal to the input. Our model's purpose was not to predict the future but rather to reconstruct the present state. Essentially, we aimed to train our model to "recall" an input. After collecting the images, the next step involved selecting the color space. The choice of color space depended on the algorithm's application scenario. The RGB color space, which represents the color spectrum recognized by the human eye and encompasses a wide range of natural colors, was commonly used. However, adapting the RGB color space to perceived details was challenging as it combined descriptions and definitions of aspects of human-eye perception, such as brightness, hue, and saturation. To address this challenge and eliminate dependencies among color components, we opted for the LAB color space. LAB color space is considered one of the most realistic color spaces. It employs three values (L, a, and b) to designate colors and uses a 3-axis system. The a-axis represents the range from green to red, the b-axis represents the range from blue to yellow, and the Lightness axis (Fig-2) determines the overall lightness or darkness of the color.

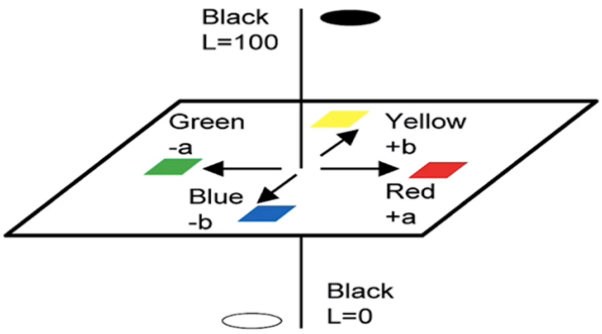

Fig 2. Color axis in LAB color space.

After converting the image format to LAB, it was utilized as the input for the neural network, as discussed in subsequent sections. Another modification made to the autoencoder component was the replacement of the encoder section with VGG-16, which served as a feature extractor. The pre-trained weights of VGG-16 were utilized to extract features from the input images.

Fig 3: Image in RGB(Left) and LAB(right) format

3. Approach

We constructed a model utilizing an encoder-decoder architecture. Furthermore, we employed transfer learning techniques and utilized VGG-16 as the feature extractor. In our approach, the encoder part remained untrained, with no weight updates taking place. On the other hand, the decoder was trained using the backpropagation algorithm and utilized to regenerate colorful images.

3.1. General pipeline and Dataset

In our model, we processed images with a pixel dimension of 224 x 224 and three channels representing red, green, and blue in the RGB color space. Prior to inserting these images into the first convolutional layer of VGG-16, they were converted to the LAB format. During testing, grayscale images of dimensions 224x224 were inputted to regenerate the images with colors.
The dataset consisted of over 7000 RGB landscape photos sourced from Kaggle. This diverse collection encompassed streets, buildings, mountains, the sea, beaches, and more. The main objective was to transform these RGB images into the LAB format. Unlike RGB or CMYK color spaces, the LAB formaa t employed a 3-axis system to specify color.

3.2. Autoencoders

Autoencoders were utilized in our model, which are unsupervised artificial neural networks capable of effectively compressing and encoding data. They learn to reconstruct data from the compressed, encoded representation, aiming to achieve a representation similar to the original input. Autoencoders can handle non-linear transformations using non-linear activation functions and multiple layers. These models are preferred over PCA (Principal Component Analysis) as they can efficiently learn multiple layers, unlike PCA which focuses on learning a single large transformation. Additionally, transfer learning can be applied by utilizing pre-trained layers from another model to enhance the encoder and decoder components. In our model, autoencoders played a crucial role in learning features and subsequently regenerating images with colors.

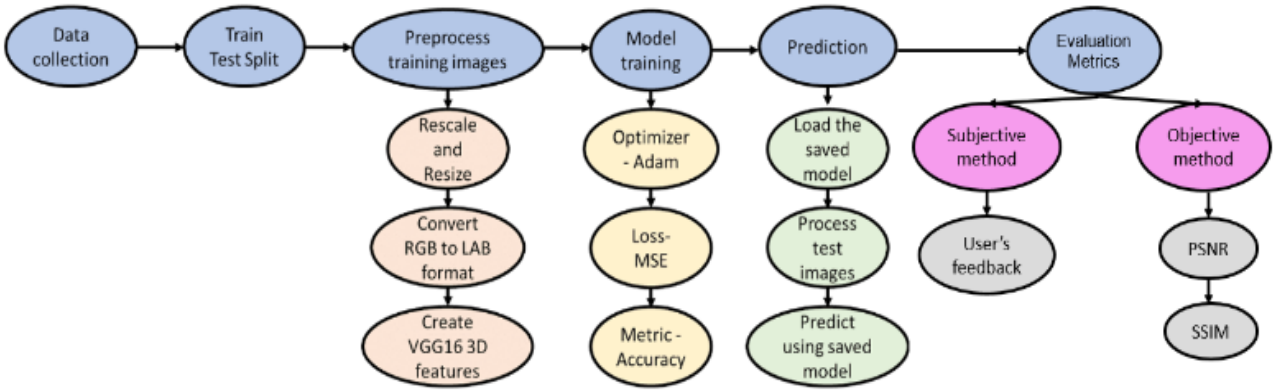

Fig 4: Implementation Flow Chart

3.3. Transfer learning

Transfer Learning [5] is a machine learning approach where a model developed for one task is utilized as a starting point for a model in another task. It is particularly popular in deep learning due to the extensive resources required to train deep learning models or the large and challenging datasets on which they are trained. In our case, we employed a pre-trained instance of VGG-16, which had been trained on the ImageNet dataset. The extracted features from this pre-trained model were found to be highly relevant to the features present in our Landscape dataset. VGG-16 was utilized as the backbone, which refers to the feature extractor network. The backbone, typically VGG16 or VGG19, is a standard convolutional neural network that is responsible for extracting features. The early layers of the backbone detect low-level features such as edges and corners, while the later layers progressively detect higher-level features like cars, people, and the sky. This backbone allows us to obtain a feature map representation from the input.

3.4. Activation function

The activation function plays a crucial role in converting the summed weighted input of a node into its corresponding output or activation. In our model, we employed the rectified linear activation function, commonly known as ReLU. This activation function outputs the input directly if it is positive; otherwise, it outputs zero. Mathematically, the rectified linear unit can be represented as follows:

$$f(x) = \max(0, x)$$

The rectified linear unit (ReLU) overcomes the issues of sensitivity and saturation found in other activation functions such as sigmoid and tanh. Additionally, ReLU computations are significantly faster compared to many other commonly used activation functions. As a result, ReLU has become the standard choice for convolutional neural networks, owing to its ability to address these challenges effectively.

3.5. Baseline architecture

The architecture of our model comprises a left-side encoding process and a right-side decoding procedure [Fig 5]. For the encoder part, we utilized the VGG-16 model while excluding the dense layers and focusing solely on the convolutional layers. The encoder section was not trained, and we utilized the pre-trained weights. Our input images were resized to 224x224, matching the default size accepted by VGG-16. Subsequently, the images were downsampled to 7x7x512, and the corresponding feature maps were extracted. The output from the encoder was then passed to the decoder, which required training. To achieve this, we employed backpropagation and upsampled the images back to the size of 224x224.

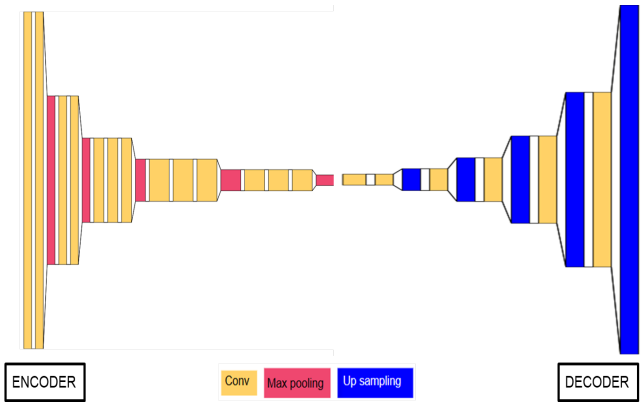

Fig 5: Encoder-decoder architecture.

4. Experiments

4.1 Evaluation Metrics

When evaluating the effectiveness of image colorization algorithms or processes, various evaluation metrics can be considered. In our case, we considered two evaluation criteria: subjective evaluation and objective evaluation.

4.1.1. Subjective Evaluation

Subjective evaluation involves the visual inspection and assessment of the colored images by human observers. Since colorization can be subjective and vary among individuals, this evaluation approach allows us to gather user preferences and perspectives. By collecting feedback from a diverse group of users, developers can gain insights into the arbitrary preferences of the target audience and potentially adjust the colorization algorithm accordingly. In our study, we asked 12 individuals to evaluate the generated images in comparison to the original images, categorizing them into four classes: good, modest, ambiguous, and failed. Selected results from this evaluation are depicted in Figures 6 and 7.

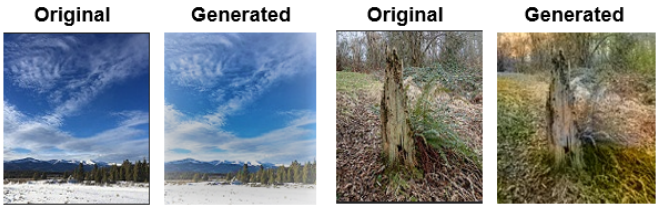

Fig 6: Figure depicting good results (left) and modest results(right)

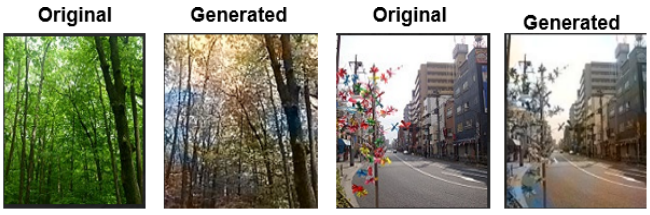

Fig 7: Figure depicting ambiguous(left) and failed (right) results.

4.1.2. Objective Evaluation

The color disparities between the original and colorized photos can be quantitatively measured using objective assessment metrics. In our experiment we have use 2 metrics: 1) PSNR and 2) SSIM

PSNR:

Peak Signal-to-Noise Ratio (PSNR) is a metric used to assess how well a picture has been processed or reconstructed in comparison to the original. The color accuracy of the colorized image is more closely matched to the original image the higher the PSNR value. The formula for PSNR is:

$$PSNR = 20 * log_{10}(\text{MAX}) - 20 * log_{10}(\text{MSE})$$

where MAX is the maximum possible pixel value of the image (usually 255 for 8-bit images). MSE is the mean squared error between the original and reconstructed images.

SSIM:

Another metric used to evaluate the similarity between two images is the Structural Similarity Index (SSIM). Luminance, contrast, and structural similarity are taken into account when determining how well a picture is received. A perfect match is represented by an SSIM value of 1, whereas no similarity is represented by a value of -1.

The formula for SSIM is as follows:

$$SSIM(x,y) = \frac{(2\mu_x \mu_y + c_1)(2\alpha_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\alpha_x^2 + \alpha_y^2 + c_2)}$$

Where x and y are the input images being compared. μ_x and μ_y are the average values (means) of x and y, respectively. α_x^2 and α_y^2 are the variances of x and y, respectively. α_{xy} is the covariance of x and y. c_1 and c_2 are small constants added to avoid division by zero.

5. Results

Figure 8 showing the results generated by our model.The model was given grayscale landscape images as testing image and the decoder has been trained to regenerate the colored image from it.

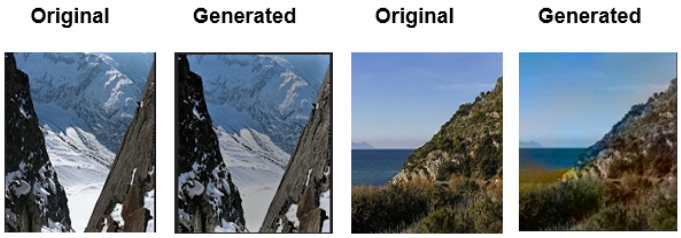

Fig 8: Original image along with generated image.

We have trained our model using 3 different sets of images and epochs as described in the table1 :

NO. OF TRAINING IMAGES	NO. OF EPOCHS RUN	TRAINING ACCURACY	AVERAGE SSIM	AVERAGE PSNR
700	200	84.99%	0.944	25.448
700	500	86.99%	0.927	25.448
*1000	500	87.32%	0.924	22.994

Table 1: Experimental results showing accuracy,PSNR,SSIM

In the first experiment, we have considered 700 images and trained for 200 epochs and training accuracy is calculated

based upon MSE loss. The average SSIM came out to be 0.944 which is desirable. The average PSNR is 25.448. As we increased the number of epochs to 500 in the second experiment, the training accuracy also increased by 2%. In the third experiment, we have used 1000 images for 500 epochs which resulted in 87.32% accuracy and 0.924 average SSIM & 22.994 average PSNR. The reason for the decrease in SSIM and PSNR could be due to the unseen data we have used for testing our model.

6. Conclusion and future work

In conclusion, we have successfully built a image colorization model which can generate colorful image from the grayscale images. However, there are still areas for improvement. Obtaining consistent and accurate colorization across many image types can be difficult, especially when it comes to things with a variety of colors and textures, like trees. For more accurate and balanced outcomes, it is also necessary to address the colorization process' predisposition toward sepia tones.

Future research should focus on determining the root reasons of the sepia color prejudice and developing countermeasures. Furthermore, investigating the usage of Generative Adversarial Networks (GANs) may lead to improved colorization results. In light of the resource constraints associated with training with big datasets and a finite number of epochs, it would also be beneficial to maximize the computing efficiency of large-scale colorization. We can improve the effectiveness and application of picture colorization in the area of computer vision and image processing by solving these issues and conducting additional research.

Source Code:
https://github.com/MeghaDeySarkar/ComputerVisionCourswork/tree/d439f5d2dd73a0483979cd39f13983d92167be3e/FinalProject

References

[1] G. Larsson, M. Maire and G. Shakhnarovich, "Learning representations for automatic colorization", *ECCV*, pp. 577-593, 2016.

[2] F. Baldassarre, D. G. Morín and L. Rodés-Guirao, Deep koalarization: Image colorization using CNNs and inception-resnet-v2, no. June 2017, pp. 1-12.

[3] R. Zhang, P. Isola and A. A. Efros, "Colorful image colorization", *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9907, pp. 649-666, 2016.

[4] G. Larsson, M. Maire and G. Shakhnarovich, "Learning representations for automatic colorization", *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9908, pp. 577-593, 2016.

[5] Pan, Sinno Jialin and Qiang Yang. “A Survey on Transfer Learning.” *IEEE Transactions on Knowledge and Data Engineering* 22 (2010): 1345-1359.