Image Colorization on Urban Landscapes

José Chacón Mejía
josejavier.chaconmejia@studenti.unipd.it

Joan Verguizas I Moliner
joan.verguizasimoliner@studenti.unipd.it

Abstract

Image colorization involves the task of predicting RGB colors for grayscale images. Recent advancements in Deep Learning have enhanced progress in this task over the past several years. This research sets out to examine how different CNN-based strategies, particularly comparing standard CNN models against models benefiting from pre-trained models like Vision Transformers (ViT) to extract features, perform in coloring urban scenes from a subset of the SUN dataset. Our study aims to discern how newer architectures influence the quality of colorized images. Evaluation metrics such as PSNR, LPIPS, and SSIM, alongside qualitative assessments of historical urban images, form the basis of our analysis. Our results showed than the quality of colorization varied through classes.Image regions depicting skies and vegetation, showed better generalization compared to others where the generalization was poor. While pre-trained models helped increase the performance of our architecture, the ViT model built better colorized representations according to the LPIPS metric.

1. Introduction

Since the early 2010s, the field of Deep Learning has revolutionized numerous tasks in Computer Vision, transitioning many from traditional statistical or machine learning methodologies to those based on neural networks. Image Colorization, the process of inferring RGB values for grayscale images, distinguishes itself among these tasks.

Primarily known for its role in image restoration, particularly with historical photographs, Image Colorization extends beyond mere restoration to encompass image enhancement. This distinction underscores the process's ability to not only recover but also enrich visual content, as highlighted in [1]. In the current landscape, a diversity of techniques exist to address Image Colorization. These range from the application of Convolutional Neural Networks (CNNs) [5] to the innovative use of Generative Adversarial Networks (GANs) [18]. Additionally, the field has seen the incorporation of real-time user feedback [22] and the integration of multi-modal data in text-based methods [12], enriching the training process and opening new avenues for nuanced colorization. Another recent approach, as explained in [9] , employs a dual encoder architecture to establish correlations between color and multi-scale semantic representations through cross-attention mechanism.

As it could be seen, recent breakthroughs in computer vision and across various AI domains are propelling advancements in image colorization techniques. Drawing inspiration from these methods, our objective is to conduct a comparative analysis of CNN-based colorization techniques, contrasting a traditional CNN approach with an approach where an advanced pre-trained model such as Vision Transformers (ViT) is used. Our analysis will focus on the colorization of images representing urban landscapes such as alleys, streets and diverse buildings extracted from the SUN dataset. We began with an examination of a baseline approach characterized by multiple convolutional layers [19] .Subsequently, we will explore alternative methods employing an encoder-decoder architecture complemented by a feature extractor module, as outlined in [3]. Through this comparison, we intend to evaluate how more recent architectures, including Inception and Vision Transformers, can enhance the accuracy and quality of colorized images.To achieve this, we used metrics such as PSNR(Peak Signal Noise to Ratio), LPIPS(Learned Perceptual Image Patch Similarity) [21] and SSIM (Structural Similarity Index Measure). In addition to this, we complement our approach by conducting a qualitative evaluation of our trained models using historical images of Padova.

2. Related Work

In this section we will have a brief review of the models that can be considered for the task of image colorization. A more in-depth overview on this topic can be found at [2].

2.1. Deep Colorization

The Deep Colorization model proposed by Cheng et al. [4] was the first model that introduced the use of a Convolutional Neural Network for the task of image colorization. Although, most of their approach was still based on the framework used during the early 2000s for computer vision tasks. As that, the model will perform feature extrac-

tion at low, mid and high levels using pixel patches features, DAISY descriptors and finally a CNN model respectively. Notice that the use of a CNN is not in the architecture perse but instead is used for high level feature extraction. The CNN function in this architecture is therefore to perform a semantic segmentation task which already had been proved to be effective in [11]. It is used to perform a sub-task rather than to leveraging the power of representation learning.

The extracted features are then used as input to a fully connected neural network with 3 hidden layers. Finally, refined chrominance and joint bilateral filtering are performed to improve the quality of the resulting image. Again as we can see this model does not take advantage of the potential that deep learning networks and convolutional neural networks have for learning. This shows the scepticism towards the predictive power of deep learning models at that time by researches.

2.2. Colourful Colarization

Colorful image Colorization [20] was one of the first Convolutional Neural Networks architectures used for the task of image colorization. The model consist in 8 sets of blocks of layers each one of them consisting in either two or three convolutional layers followed by a ReLU layer for activation and a Batch Normalization layer. As many other architecture, they will take as input the first canal of a LAB image (the lightness) but in this case the architecture will return a probability distribution that we will be able to use to sample the values of the predicted AB channels.

This paper is not the only one to have developed architectures which leverage large-scale data and CNNs. Larsson et al. [10] and Iizuka et al. [8] both developed similar systems. The methods differ in specific characteristic of their CNN architectures and the loss functions they applied.

While considering the method to adopt in order to solve the image colorization problem, we think about implementing Zhang method this though has not been possible due to the huge GPU RAM required to run this model which has made practically infeasible. Specifically, the huge number of parameters (> 32 millions) surpassed by a large number our models number of parameters.

2.3. Scribbler

The Scribbler network [13] is a kind of user guided network. These kinds of networks are characterized by that they require the input of the user in some way. To solve the problem of image colorization, scribbler networks uses an end-to-end feed-forward deep generative adversarial architecture. To guide the color patterns user input of sketches and color strokes is needed. This will enhance the quality of the resulting image.

The generator part of the network follows a encoder-decoder structure. The encoder will have three layers that uses to learn hierarchical information about the data by compressing it. After that, 7 residual blocks are preceded by three more layers of the decoder that will be used to up-sample the data. All layers are followed by batch normalization and a ReLU activation function. To the outcome of the final layer a $\tanh$ is applied to get the output of the model.

3. Dataset

In this work we have put our focus in performing the task of image colorization to images that belong to an urban landscape. In order to do that we have used a subset of the SUN dataset. This dataset called Scene UNderstanding (SUN) database [17] [16] was created as a benchmark for scene understanding tasks and consist of $130,519$ images grouped in 899 categories. Usually only the most well-sampled categories are used (the ones with over 100 samples) from which the SUN397 is constructed with 397 categories and a total of $108,754$ images.

Since our available computational resources are limited, we have select a subset of the 397SUN dataset. Specifically, this subset has been set manually by choosing the categories that may be found in urban landscapes. This decision was driven by the fact that a common application for image colorization is to restore images from the past. Often these images consist in scenery from many decades ago of the same places that nowadays people are still living in it. Therefore we consider that performing image colorization using the categories from SUN397 that relate to urban categories can be a good application.

At the end, 32 categories that add up to 12950 images have been used to work with our model. The amount of images that each category has varies and is indicated in 5. For each of the aforementioned categories we will perform a train, validation and test split which then will be grouped with all the other images for the training, validation and testing parts of the pipeline.

Finally, before given the data as input to the model a set of image preprocessing steps are performed. The first one of which is to resize the images of the dataset to 256×256 size, following since our original color images are encoded using a RBG color model we will need to process a very small fraction of them that are encoded using RGBA color model. Next, due to the image colorization task we are performing, we will need to convert our images from RGB to CIELAB images. This last model consist on three channels the first one called lightness which ranges from 0 to 100 and represents the gray-scale of the image. Meanwhile the A and B channels represents color scales both ranging between -128 and 127. The input to our models therefore will be the lightness of the images from which we will try to predict the optimal values of "AB" in order to be able to perform the task of image colorization effectively. Finally all

channels will be normalized to fall inside the $[-1, 1]$ range either for the input L channel or the AB channels that we use for validation.

4. Method

4.1. Overview

The task of image colorization can be formalize as given an image in the CIELAB color space, we can use the L channel that represents the lightness of the color that is the grayscale value to recover a full RGB color image by predicting the AB channels that represent the chrominance. Given an input image the L channel is extracted $x_L \in \mathbb{R}^{H \times W \times 1}$, therefore our colorization network aims to predict the two missing color channels $\hat{y}_{AB} \in \mathbb{R}^{H \times W \times 2}$. In order to perform this operation we have chosen to work with encoder-decoder architectures. In the following sections we describe the architectures that were explored in this project. As well as why using this type of architecture makes sense to solve a image colorization task.

4.2. CNN-encoder-decoder-based model

Make sense to use a CNN encoder-decoder based architecture to solve a image colorization task. This is due that the encoder can be seen as the part of the network that extracts high-level features from the input gray-scale image through successive convolutional or pooling layers. These features may represent different levels of abstraction, which are crucial for understanding the semantics of the input image. Meanwhile, the decoder through another set of layers, is able to successfully reconstruct the colorized image from these features. The encoder-decoder architecture then enables the model to capture the essential information needed for colorization and generate accurate colorizations of the input data.

We have dealt with 3 types of architectures based on CNN encoder-decoder. The first of them is a very simple vanilla CNN encoder-decoder architecture called Hourglass architecture we will call this the baseline model, were both the encoder and the decoder is based on 3 convolutional layers followed by a ReLU activation function. Additionally, we have the Inception-V3 and the Vision Transformer models that will be explained in the following subsections. The last architecture is another vanilla encoder-decoder CNN this time using the same architecture as the V3-inception after substracting the inception part.

Let's now see a brief review of the parts of the encoder-decoder architecture as well as the features of the models we just mentioned.

4.2.1 Encoder

The Encoder operates on H $\times$ W grayscale images, producing a feature representation of size $\frac{H}{8} \times \frac{W}{8} \times 512$. This is achieved through applying 8 convolutional layers with 3 $\times$ 3 kernels. Padding is employed to maintain the input size of each layer. A stride of 2 was used in the first, third, and fifth layers, reducing both volume dimensions and computational workload.

4.2.2 Feature Extractor

In this project, two pre-trained models were used, Inception-V3 [15] and Vision Transformers(VIT) [7]. In both cases, we extracted the output of the last hidden layer consisting of a vector of 1000 elements. Therefore these high-level features could contribute to improve the colorization quality.

- **Inception-V3**: CNN consisting of 48 convolutional layers. The pretrained model was trained on more than a million images from ImageNet [6]. In order to use it, our images were resized to 299x299
- **Vision Transformer (ViT)**: A model for image classification that employs a Transformer-like architecture over patches of the image. The chosen model checkpoint was "google/vit-base-patch16-224". It was pretrained on ImageNet at resolution 224x224. Therefore, our images were resized to satisfy the model requirements.

4.2.3 Fusion Module

The fusion module serves as the component responsible for integrating the outputs from both the encoder and the feature vector extracted from either Inception or Vision Transformer along the depth axis, as introduced in [16]. This is obtained by concatenating the feature vector, which helps to propagate the semantic information through all regions of the image. Then, dimensionality reduction is applied by using 256 filters of size 1x1 to derive the output volume $v \in \mathbb{R}^{H/8 \times W/8 \times 256}$, which serves as input of the decoder.

4.2.4 Decoder

As usual in encoder-decoder architectures, upsampling is performed after each convolutional layer. In this case, an upsample of scale 2 is performed after the first, third and fifth convolutional layer. In the end, the input volume of $H/8 \times W/8 \times 256$ will be transformed into an output of $H \times W \times 2$, which represent the model prediction of the AB channels.

5. Experiments

5.1. Training

5.1.1 CNN-encoder-decoder-based model

We trained three models independently, for the two most prominent of them (inception and vision transformer) employing identical architectures but differing solely in the feature extractor component. One model utilizes the Inception pre-trained model, while the other employs the Vision Transformer.

Prior to each run, the tensor representations of training images,preprocessed in advance and stored in the cloud, were loaded to reduce the computational workload of preprocessing training images in each run , which slowed down considerably the training phase. This preprocessing consisted of resizing each images so it could satisfy the resolution requirement of the encoder module (256x256) that is used in all four architectures, the Inception pre-trained model (224x224) and the Vision Transformer pre-trained model (299x299). After that they were converted to the CIELAB channel and had their L and AB channel extracted.

For all runs,the whole network is trained for 50 epochs with a batch size of 64. Adam Optimizer was used with learning rate $\eta = 0.0001$ and weight decay 1e-6. All experiments were conducted on 1 NVIDIA T4 GPU.

Hyperparameter	Value
Epochs	50
Learning Rate	1e-4
Batch size	64
Weight decay	1e-6

Table 1. Hyperparameters for both runs of the encoder-decoder model

5.1.2 CNN baseline

In order to a have a better understanding of the improvement the models described in the previous section brings in the colorizing task, we trained an additional model which follows a similar architecture but without a feature extractor component and a fusion module. The choice of hyperparameters is the same as in Table 1

5.2. Results

As expressed in 2, both models that used pre-trained models has similar performance. They also had better performance than the baseline model which was expected. In addition to this,the CNN + ViT model showed a better score in the Learned Perceptual Image Patch Similarity (LPIPS) metric, which could be understand due to ViT strength in capturing and learning contextual information within images.

Model	PSNR	SSIM	LPIPS	MSE
CNN Baseline	22.0465	0.8712	0.1725	0.0091
CNN + Inception	25.0957	0.9390	0.1158	0.0073
CNN + ViT	24.9997	0.9375	0.1106	0.0076

Table 2. Model Performance Metrics

For scene semantics in our dataset, the model successfully learned a limited number of color-object relationships. Notably, it accurately identified that grass tends to be green and that the sky is typically blue, as illustrated in Figure 1. However, the model's performance was notably poorer for other common elements. In these cases, the colorization performance was suboptimal, and it was evident that the model's predictions for the AB values were conservative, likely to avoid increasing the loss function. In 3 could be seen that the classes that has abundant representation of skies and vegetation turned out to be the ones with better metrics performance.

Class	**PSNR**	**SSIM**	**LPIPS**	**MSE**
Basilica	27.4201	0.9555	0.0708	0.0033
Skyscraper	25.6794	0.9468	0.0772	0.0050
Cathedral	26.0213	0.9444	0.0861	0.0053
Street	27.2297	0.9635	0.0889	0.0043

Table 3. Classes with best metrics on test set

Class	**PSNR**	**SSIM**	**LPIPS**	**MSE**
Phone booth	22.5894	0.8602	0.1418	0.0117
Market	22.1302	0.9117	0.1997	0.0133
Playground	22.1466	0.9111	0.1791	0.0134
Balcony	23.3941	0.9144	0.1128	0.0135

Table 4. Table of values with 4 decimals.

5.3. Failure cases

Through a qualitative analysis of the models prediction, we observed that the model fail to generalize when there are many instances of a object in the image, as explained in [14]. Besides, we could seen that the model tended to over represent pixels with green colors due to what it learned during the training phase. In addition, when faced with images with different patterns the colorization did not go well as it could be seen in the third picture of 2

5.4. Testing models on Padova pictures

In addition to our main experiments, we tested the model on pictures of Padova main urban landscape attractions. While old buildings such as Porta Portello got an accurate

Figure 1. Visual comparison of the two colorization models trained from our subset of SUN Dataset

Figure 2. Examples where the model failed to generalize

representation. Surprisingly, Pratto della Valle wasn't colorized well maybe for the perspective of the picture and the presence of the concrete around the vegetation.

Figure 3. Colorization on a Porta Portello picture

6. Conclusion

The task of performing image colorization has proved successful when using encoder-decoder CNN based architectures. Although some results of the model regarding problems with images where there is occlusion or cluttering also appear. Additionally, the models sometimes does not generalize that good while it is robust to building and structures it may fail when the images have some non-standard pattern. Additionally, going back to the architecture features it is demonstrated that the pretrained models help in making the image representation better. This is shown when using the LPIPS metric that measure the perceptual similarity between images and where pretrain models score higher. Using other type of metrics such as the standard MSE also show that where the baseline model as expected is the one that scores worse.

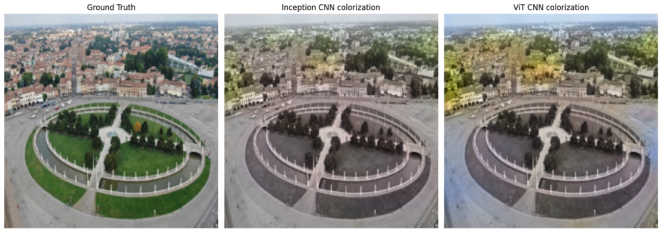

Figure 4. Colorization on a Prato della Valle picture

Figure 5. Colorization on a Basilica di sant'Antonio di Padova picture

References

[1] Saeed Anwar, Muhammad Tahir, Chongyi Li, Ajmal Mian, Fahad Shahbaz Khan, and Abdul Wahab Muzaffar. Image colorization: A survey and dataset. *arXiv preprint arXiv:2008.10774*, 2020.

[2] Saeed Anwar, Muhammad Tahir, Chongyi Li, Ajmal Mian, Fahad Shahbaz Khan, and Abdul Wahab Muzaffar. Image colorization: A survey and dataset, 2020.

[3] Federico Baldassarre, Diego González Morín, and Lucas Rodés-Guirao. Deep koalarization: Image colorization using cnns and inception-resnet-v2, 2017.

[4] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. In *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Dec. 2015.

[5] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization, 2016.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2009.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner,

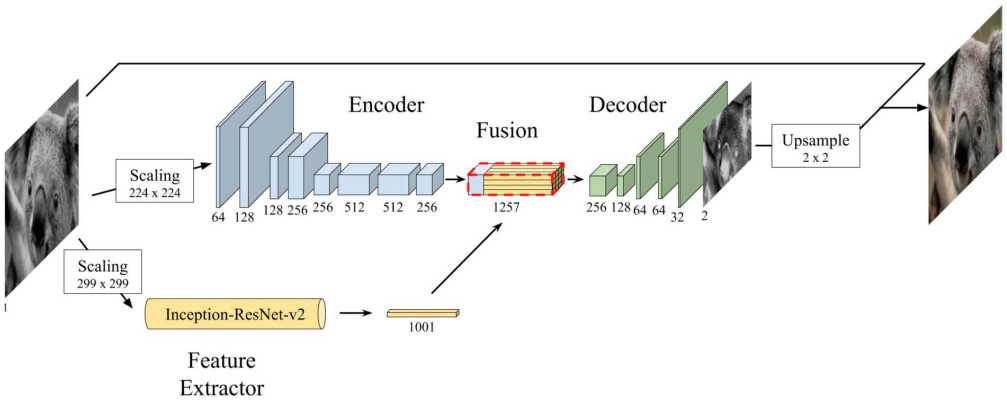

Figure 6. CNN + Feature Extractor architecture

Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[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 (Proc. of SIGGRAPH 2016)*, 35(4):110, 2016.

[9] Xiaoyang Kang, Tao Yang, Wenqi Ouyang, Peiran Ren, Lingzhi Li, and Xuansong Xie. Ddcolor: Towards photo-realistic image colorization via dual decoders. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 328–338, 2023.

[10] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization, 2016.

[11] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation, 2014.

[12] Varun Manjunatha, Mohit Iyyer, Jordan Boyd-Graber, and Larry Davis. Learning to color from language. 2018.

[13] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color, 2016.

[14] Jheng-Wei Su, Hung-Kuo Chu, and Jia-Bin Huang. Instance-aware image colorization, 2020.

[15] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[16] Jianxiong Xiao, Krista A. Ehinger, James Hays, Antonio Torralba, and Aude Oliva. Sun database: Exploring a large collection of scene categories. *International Journal of Computer Vision*, 119(1):3–22, Aug. 2014.

[17] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, June 2010.

[18] Seungjoo Yoo, Hyojin Bahng, Sunghyo Chung, Junsoo Lee, Jaehyuk Chang, and Jaegul Choo. Coloring with limited data: Few-shot colorization via memory-augmented networks, 2019.

[19] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization, 2016.

[20] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization, 2016.

[21] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018.

[22] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros. Real-time user-guided image colorization with learned deep priors. *ACM Transactions on Graphics (TOG)*, 9(4), 2017.

7. Appendix

Classes	Number of Images
Alley	324
Amphitheater	316
Apartment Building	527
Arch	218
Balcony	125
Basilica	295
Bazaar	111
Bow Window	131
Bridge	847
Building Facade	325
Canal	490
Castle	1100
Cathedral	635
Crosswalk	186
Doorway	550
Fountain	212
Hospital	120
House	955
Market	839
Medina	104
Mosque	536
Office Building	300
Palace	223
Park	143
Parking Lot	439
Phone Booth	316
Playground	898
Plaza	228
Residential Neighborhood	110
Skatepark	100
Skyscraper	801
Street	458

Table 5. The table displays the number of instances for every class in our dataset.