Image Colorization

Sofia Bazzan
sofia.bazzan@studenti.unipd.it

Diletta Pasin
diletta.pasin@studenti.unipd.it

Abstract

The image colorization task involves transforming a grayscale image into its corresponding colored version. Given the inherent ill-posed and under-constrained nature of the problem, characterized by multi-modal uncertainty, various techniques have been proposed. This study focuses on automating colorization using deep learning models, with a specific emphasis on the Pix2Pix architecture—a conditional Generative Adversarial Network (cGAN) designed for diverse image-to-image translation tasks. Our approach involves utilizing a comprehensive dataset comprising more than 8000 *images of flowers of* 102 *different species. Prior to employing Pix2Pix, we implemented several traditional neural network architectures, including a deep neural network, autoencoder, ResNet, U-Net, and a dense neural network. Experimental results demonstrate the effectiveness of our methodology in colorizing flower images. The use of diverse neural network architectures, including Pix2Pix, provides insights into their comparative performance. The outcomes contribute to advancing the field of image colorization, highlighting the potential and limitations of various approaches.*

1. Introduction

The image colorization task involves predicting plausible color channels for a given grayscale image, transforming it into a realistic colored version. This challenging problem in computer vision and graphics has practical applications such as restoring legacy photos/videos or image compression. However, predicting color channels from a single channel makes the task ill-posed, and its multi-modal nature, where objects can have different colors within the same image, adds complexity. Early attempts relied on user guidance through color scribbles or reference images. With the rise of deep learning, the focus shifted to end-to-end colorization using deep neural architectures.

In this project, we initially experimented with implementing and training our models on our dataset using approaches such as autoencoder and deep neural networks. Subsequently, we transitioned towards more complex models such as U-Net, DenseNet, and finally *Pix2Pix*. This progression was motivated by the need to address more intricate challenges in image colorization, such as preserving details and generating realistic colors.

To evaluate the performance of the models, in addition to the traditional MSE (mean squared error) and MAE (mean absolute error) we employed metrics such as PSNR (peak signal-to-noise ratio), SSIM (structural similarity index) and a metric proposed by *Hasler et al*. [5] for measuring image colorfulness. Through iterative experimentation and refinement, we demonstrated significant improvements in colorization quality, particularly with the *Pix2Pix* model, leveraging its ability to learn the direct relationship between grayscale and color images through conditional adversarial training.

2. Related Works

For an in-depth exploration of image colorization techniques and datasets, refer to [2], which provides a comprehensive survey. This summary highlights the major contributions in the field.

2.1. Scribble-based colorization

These approaches incorporate user-provided cues in the form of color scribbles, which are then extended to neighboring pixels based on various low-level similarity metrics such as luminance similarity [12] or texture similarity [17]. For example, *Levin et al.* [12] introduced an optimization-based system, assuming that adjacent pixels with the same luminance might share similar colors. While these methods can yield impressive outcomes, they often demand substantial user involvement, requiring explicit user indications for different color regions. To mitigate the manual effort, *Zhang et al*. [27] introduced an interactive colorization model. This model combines user hints with a deep neural network to ensure plausible colors for image parts lacking explicit scribbles.

2.2. Example-based Colorization

To alleviate the manual input burden on users, example-based techniques leverage color information from a reference image, either provided by the user or sourced from

the Internet, to guide the colorization process [24, 19, 4, 8]. These methods primarily establish correspondences between the reference and input images using various low-level similarity metrics at the pixel level [24], semantic segment level [8], or super-pixel level [4]. However, these approaches heavily rely on references closely resembling grayscale input. Consequently, when the given images differ significantly from the input, the colors in the output often appear unnatural. Additionally, researchers have explored alternative reference types, such as words [3] and complete sentences [30], to provide guidance for the colorization process.

2.3. Fully Automatic Colorization

In recent years, deep convolutional neural networks (CNNs) have become the leading method for color prediction, thanks to advances in deep learning and access to large datasets like *ImageNet* [21]. Notably, *Larson et al*. [10] and *Zhang et al*. [26] utilized the VGG-Net architecture [22] for their colorization networks.

Larson et al. used hypercolumns from a pre-trained VGG-Net to enrich grayscale images with colors [10], while *Zhang et al*. approached colorization as a classification task, resulting in vibrant and diverse colorization schemes [26]. Based on these results we implemented as one of our first models, similarly to Zhang's work, a CNN that employ stacked convolutional layers with ReLU activation and Batch Normalization, but that instead of predicting "ab" pairs representing the empirical probability distribution, ours directly predicts RGB values from grayscale inputs

Inspired by these works, we also implemented the U-Net [20] architecture and the *Pix2Pix* model [9]. The U-Net-based encoder-decoder architecture captures both local and global image features, facilitating high-quality colorization. Our *Pix2Pix* implementation follows Isola's design, with a generator adopting a UNet structure and a PatchGAN discriminator.

3. Dataset

We curated a dataset using the **102 Category Flower Dataset** [16], a collection developed by Maria-Elena Nilsback and Andrew Zisserman. The dataset encompasses 102 distinct flower categories, each containing a variable number of images, ranging from 40 to 258, showcasing commonly found flowers in the United Kingdom. The images within the dataset exhibit considerable diversity in scale, pose, and lighting, capturing variations within categories and similarities across closely related ones. The decision to focus on flowers for this task was influenced by the diverse forms and colors they exhibit, rendering them particularly intriguing for our purposes. Given the relatively small size of the dataset, opting for images within the same class, i.e., flower, was a strategic choice. This approach not only enhances the colorization task but also maintains a level of complexity essential for meaningful experimentation.

To prepare the dataset for model training, we performed preprocessing steps, focusing on standardizing image dimensions to 128x128 pixels. In the subsequent stages, before training each model, we implemented a training function using TensorFlow. During the training phase, additional data augmentation techniques were applied to enhance the model's ability to handle diverse scenarios.

Specifically, a random hue shift was introduced to the images, providing a varied representation of colors. This augmentation technique aimed to improve the model's robustness and adaptability to different color distributions. While exploring various data augmentation techniques [13], such as contrast shift, we found that the hue shift yielded the best results. This choice was guided by the consideration that flowers exhibit a wide range of hues and colors, making the hue shift augmentation particularly suitable for capturing these variations effectively. Additionally, we normalized the pixel values to fall within the [0, 1] range by dividing by 255. The resulting dataset was then cached, shuffled with a buffer size of 1024 for improved randomness, and organized into batches of a specified size.

We divided our dataset into 60% for training, 20% for validation, and 20% for the test set to ensure a comprehensive evaluation of model performance. The validation dataset underwent similar preprocessing steps, ensuring consistency in data representation during the evaluation phase. These steps collectively contribute to the effective training and evaluation of the models over 30 epochs. The chosen preprocessing techniques aim to strike a balance between maintaining the integrity of the original flower dataset and enhancing the models' ability to generalize to diverse flower images.

4. Methods

4.1. Convolutional Auto-Encoder

The architecture follows the standard encoder-decoder structure. We implement the encoder section with three sets of three consecutive convolutional layers. For stability and accelerated training, each convolutional layer is followed by batch normalization. To introduce non-linearity, a leaky ReLU activation function is applied. MaxPooling2D layers are strategically used to reduce the spatial dimensions, aiding in the extraction of crucial features.

The decoder section mirrors the encoder's structure. It consists of three sets of three consecutive convolutional layers, each followed by batch normalization and leaky ReLU activation. To restore spatial dimensions, we employ UpSampling2D layers.

The output layer is a Conv2D layer with three filters and

a sigmoid activation function. This final layer is responsible for reconstructing the colorized version of the input image.

Moving to the model compilation, for this architecture and all the future ones, we use the Adam optimizer with a learning rate of $1e^{-3}$. Adam's effectiveness in gradient-based optimization makes it a suitable choice. The loss function employed is Mean Squared Error (MSE), measuring the average squared difference between predicted and actual values. This aligns with our goal of minimizing reconstruction errors.

Our implementation closely follows the approach outlined in the paper by *Zhang et al*. [26], ensuring consistency and compatibility with established methodologies in the field of image colorization.

4.2. Deep CNN

We've developed a deep Convolutional Neural Network (CNN) architecture tailored for grayscale images sized at 128x128 pixels. Our design begins with an input layer specific to this image format, followed by six convolutional layers for feature extraction. Each convolutional layer is accompanied by Batch Normalization and Leaky ReLU activation for stable training and non-linearity introduction.

Inspired by the success of *Zhang et al*. in automatic image colorization [26], we investigated the effectiveness of CNN-based strategies for our colorization task.

A notable aspect of our architecture is the incorporation of skip connections after every set of three convolutional layers. These connections allow concatenation of the current layer's output with that of corresponding layers from earlier stages, preserving low-level features throughout the network. This strategy aligns with the U-Net architecture proposed by *Ronneberger et al*. [20], widely adopted in image-to-image translation tasks, including colorization.

The final segment comprises the output layer, featuring a convolutional layer with three filters and a sigmoid activation function. This layer generates a colorized version of the input grayscale image, with three filters aligning with RGB color channels.

4.3. U-Net

Our U-net architecture represents a fusion of established methodologies and innovative elements tailored for image segmentation tasks. Inspired by seminal works such as the U-Net architecture proposed by *Ronneberger et al*. (2015) [20], our model begins with an input layer designed for grayscale images, sized at 128x128 pixels. This foundational layer is complemented by batch normalization to stabilize training, followed by a 3x3 convolutional layer with 16 filters and a linear activation function, introducing non-linearity through leaky ReLU activations.

A distinguishing feature of the architecture lies in the integration of residual layers within the down-sampling path. These residual layers, comprised of convolutional blocks with leaky ReLU activations, batch normalization, and max-pooling operations, strategically capture complex hierarchical features, enhancing the network's capacity. As the down-sampling process ensues, a series of max-pooling layers effectively diminish the spatial dimensions of the feature maps.

In the up-sampling trajectory, residual layers are again employed to retain essential information and promote feature reuse. The spatial dimensions of the feature maps are gradually increased using 2x up-sampling layers. The architectural cascade concludes with a succession of 3x3 convolutional layers featuring linear activation, batch normalization, and leaky ReLU activations. The final layer, equipped with a 3x3 convolutional veneer and a sigmoid activation function, ensures that the output pixel values adhere to the range of 0 to 1, suitable for image reconstruction tasks.

Our architectural blueprint integrates insights from prior works, leveraging the advancements brought forth by the U-Net architecture and UNet++ by *Zhou et al*. (2019) [28] . While drawing inspiration from these established methodologies, our model introduces novel elements, such as the incorporation of residual layers within the down-sampling trajectory, to further augment feature representation and segmentation accuracy.

4.4. ResNet

The ResNet architecture, engineered for grayscale images of 128x128 pixels, begins with batch normalization to stabilize training, followed by a 1x1 convolutional layer with 256 filters and linear activation to transform input features.

Introduced by *He et al*. (2015) [6], ResNet shares fundamental characteristics with our model, utilizing residual connections to address training challenges. These connections alleviate the vanishing gradient problem, enabling deeper and more stable networks.

A key innovation of ResNet is its implementation of residual layers, realized as custom Keras layers named Residual Layer. Each Residual Layer contains a feed-forward neural network (FFNN) with batch normalization, leaky ReLU activation, and convolutional operations. During forward pass, the output of the FFNN is added to the input tensor, facilitating gradient flow during backpropagation.

The ResNet architecture unfolds with a series of residual layers, each comprising batch normalization, leaky ReLU activation, 1x1 convolution, and another leaky ReLU activation before final 1x1 convolution. This sequence fosters hierarchical feature abstraction.

The narrative culminates with 20 iterations of identical residual layer structures, significantly deepening the network. The architecture concludes with batch normalization,

leaky ReLU activation, a 3x3 convolutional layer, and a sigmoid activation function to ensure output values within the [0, 1] range, crucial for pixel value reconstruction.

4.5. DenseNet

The architecture begins with an input layer for grayscale images, each with dimensions of 128x128 pixels. The first layer is a 1x1 convolutional layer with 256 filters, serving as a feature extractor while preserving the spatial dimensions of the input.

Following the initial layer, a series of densely connected blocks are constructed. Each block comprises a combination of 1x1 and 4x4 convolutional layers, contributing to the extraction of hierarchical features. Batch normalization is applied after each convolutional operation to enhance training stability, while the rectified linear unit (ReLU) activation function introduces non-linearity to the model.

After the dense blocks, a transition layer follows, consisting of a 1x1 convolutional layer to compress the feature maps and reduce the computational load for subsequent layers. The transition layer maintains a balance between model complexity and efficiency.

The final layers involve a 1x1 convolutional layer followed by batch normalization and ReLU activation, leading to a 3x3 convolutional layer with a sigmoid activation function. This configuration reconstructs the colorized version of the input image, with the sigmoid activation ensuring pixel values are within the range of 0 to 1.

While the architecture shares similarities with DenseNet proposed by *Huang et al.* (2017) [7], which emphasizes dense connectivity patterns for gradient flow and feature reuse across layers, our model leverages dense connections to improve feature propagation and network performance in image reconstruction tasks.

4.6. Pix2Pix

We finally adopted an architecture inspired by the *Pix2Pix* model proposed by *Isola et al.* (2017) [9]. The *Pix2Pix* model, a conditional Generative Adversarial Network (cGAN), conditions the generation of the output image on an additional input, typically a black and white (b/w) image. Similarly, in our model, the generator and discriminator utilize Convolution-BatchNorm-ReLu (CBR) layers, following the architecture outlined in [18].

In our implementation, we observed that the noise vector is not utilized as input, consistent with the findings reported in the *Pix2Pix* literature. Instead, noise is introduced via dropout applied to several layers of the generator, a technique that we found beneficial for our task.

Our generator architecture follows an encoder-decoder structure with skip connections, akin to the "U-Net" architecture. This design choice, while sharing similarities with *Pix2Pix*, aims to preserve essential features while altering the surface appearance of the output image. However, unlike the *Pix2Pix* model, we did not incorporate leaky ReLU activations into our encoder architecture.

Regarding the discriminator, we designed it to classify overlapping patches of the image, similar to the patchGAN concept introduced in *Pix2Pix*. However, our experimentation led us to use different patch sizes, such as 64x64 or 128x128, as we found this approach yielded optimal results for our specific task.

4.7. Objective

4.7.1 MSE Loss

The primary loss used for training the deep neural network models, including ResNet, DenseNet, U-Net, and Autoencoder, is the Mean Squared Error (MSE) loss. The MSE loss quantifies the pixel-wise difference between the predicted colorized image $\hat{Y}$ and the ground truth color image Y:

$$\text{MSE Loss} = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2$$

Here, N is the total number of pixels in the images.

4.7.2 Adversarial and Reconstruction Loss

For the *Pix2Pix* model, the loss function is a combination of adversarial loss and reconstruction loss. The adversarial loss includes both Mean Squared Error (MSE) and Binary Cross-Entropy (BCE) formulations, showcasing the exploration of both approaches during experimentation. MSE is used to quantify the pixel-wise difference for adversarial loss:

$$\text{Adv Loss (MSE)} = \frac{1}{N}\sum_{i=1}^{N}(\text{Preds}_{1_i} - 1)^2 + \frac{1}{N}\sum_{i=1}^{N}\text{Preds}_{0_i}^2$$

Additionally, Binary Cross-Entropy (BCE) is included in the adversarial loss:

$$\text{BCE} = -\frac{1}{N}\sum_{i=1}^{N}\Big(\log(\text{Preds}_{1_i} + \epsilon) + \log(1 - \text{Preds}_{0_i} + \epsilon)\Big)$$

Here, Preds_{1_i} and Preds_{0_i} are the discriminator's predictions for real and generated images, and ϵ is a small constant to prevent logarithmic divergence.

The reconstruction loss measures the pixel-wise difference between the generated RGB images $\hat{Y}$ and the ground truth RGB images Y. This loss encourages the generator to faithfully reconstruct the input images during the translation process:

$$\text{Reconstruction Loss} = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2$$

4.8. Training

We implemented functions to train neural networks for different tasks. For autoencoder training, we developed a custom training function that leverages TensorFlow's capabilities for efficient computation. This function organizes the training process into epochs and conducts custom training steps for each batch of the dataset. During training, the network processes input data and computes the mean squared error loss between predicted and actual outputs. To enhance data diversity and improve model robustness, we incorporated data augmentation techniques such as random hue shifting. Additionally, we ensured data consistency and stability by normalizing the images, aligning their pixel values across the dataset. We used automatic differentiation to calculate gradients and optimize the model parameters using an optimizer. The training loop records losses for both training and validation datasets, enabling further analysis.

Similarly, we developed a training function tailored for Pix2Pix-style Generative Adversarial Networks (GANs). This function alternates between updating the discriminator and the generator networks. The discriminator learns to distinguish between real and generated images, while the generator aims to produce realistic images. Similar to the autoencoder training, we utilized TensorFlow's features for efficient computation and applied data preprocessing techniques, such as random hue shifting for data augmentation.

These training procedures enable our models to learn from the provided datasets and improve their performance over successive epochs. The recorded losses serve as indicators of model performance and guide further optimization efforts.

4.9. Evaluation Metrics

In our experimentation, we followed established colorization protocols, employing quantitative metrics such as MSE (Mean Squared Error), MAE (Mean Absolute Error), PSNR (Peak Signal-to-Noise Ratio), SSIM (Structural Similarity Index) [29]. However, it is essential to note certain inherent challenges with these metrics. MAE/MSE operates under the assumption of a single correct output for each pixel, which may not align with the nature of the colorization problem. Similarly, PSNR, being based on MSE, shares these limitations.

Furthermore, SSIM captures only statistics of the output and is unable to adequately represent the credibility of the recolorization process. These limitations underscore the need for a comprehensive evaluation approach that considers the nuances of the colorization task, acknowledging the inherent ambiguity in determining a singular correct output for each pixel.

	Colorfulness	PSNR	SSIM	MSE	MAE
Autoenc	15916.6	45697.3	347.5	5081997	**5885808**
Deep NN	**22045.0**	**45700.4**	428.8	5080131	5976311
Dense net	20934.3	45683.7	377.0	5091630	5902259
Resnet	21575.1	45700.1	401.7	5080246	5945178
Unet	20723.7	45657.8	**434.7**	**5061775**	5896118
Pix2Pix	16742.3	45695.1	429.0	5083569	6003876

Table 1. Performance of the trained models measured with Colorfulness, PSNR, SSIM, MSE and MAE. We can see that there is not a single model that performs better in each metric.

Acknowledging that MSE, MAE, PSNR and SSIM may penalize images that are reasonably well colorized but differ in color from the ground truth, an additional metric proposed by *Hasler et al*. [5] was incorporated. This metric assesses the colorfulness of natural images by linearly combining the mean and standard deviation across opponent color channels, specifically the yellow-blue and red-green channels, in the given RGB image.

The limitation of the metric proposed in [5] lies in the fact that if an image predominantly consists of black, the metric will penalize it due to its perceived lack of color, even if the model-generated image is, in reality, accurate.

In table 1 we reported the obtained results. The results have been rounded to one decimal place for PNSR, SSIM and Colorfulness, and to the units in MAE and MSE.

As it is possible to notice from the provided table, there isn't one single implemented model that outperforms the others in all the metrics and this is due to the nature of the metrics. Since those results and considerations previously done, we relied on the "Turing test", evaluating results visually on a subset of the test-set.

5. Results

In addition to quantitative metrics, we conducted a visual analysis of the colorization results obtained from different models. Figure 1 presents a visual comparison of the colorized images produced by each model on a sample subset of the test set: from the visual inspection of the results, it is evident that the UNet and *Pix2Pix* models consistently produced colorized images with higher fidelity and realism compared to all the other implemented models.

The superior performance of UNet and *Pix2Pix* can be attributed to several factors. Firstly, the architectural design of UNet and *Pix2Pix* is specifically tailored for image-to-image translation tasks, such as colorization. The presence of skip connections in UNet facilitates the preservation of spatial information, while the conditional adversarial training mechanism in *Pix2Pix* enables the generation of high-quality colorized images with improved texture and detail.

Furthermore, the utilization of specialized loss functions, such as adversarial loss in *Pix2Pix*, allows the models to op-

Figure 1. Visual comparison of the colorization done by the different models. We can observe that the *Pix2Pix* model achieve the best result producing colorized images which are nearly similar to the ground truth.

timize for perceptually realistic colorization outputs, resulting in visually pleasing results. Despite being more complex architectures, UNet and *Pix2Pix* effectively balance model complexity with training data size, leading to robust performance across various colorization scenarios.

5.1. Failure cases

While the model demonstrates strong performance across multiple images, there are still prevalent instances of failure commonly observed in automatic colorization models, contributing to the perception of poorly colorized images. Figure 2 illustrates various common failure cases, which can be categorized as follows:

- **Inconsistent Coloring**: Spatially inconsistent colors, such as patches of different colors in the background or on the depicted object.
- **Color Bleeding**: The color of an object spills over its intended boundary in this scenario.
- **Greenish Colors**: This is the most frequent case, where the image is predominantly colorized with shades of green.

One of the primary sources of failure stems from the model's bias towards greenish colors, likely influenced by their prevalence in the training dataset.

Figure 2. Forms of failure commonly seen in colorized results. From top to bottom: color inconsistency, color bleeding and greenish colorization.

It's important to note that, for brevity, we have chosen to present failure cases only for the Pix2Pix and Unet models, which exhibited superior performance. However, it's essential to acknowledge that similar types of errors were observed, to a greater extent, in other models with lower performance.

5.2. Considered variation

In our pursuit of optimizing colorization performance, we explored several variations and alternative approaches. Initially, as a data augmentation technique, we experimented with randomly shifting the contrast instead of hue or applying both transformations simultaneously. However, despite these attempts, we found that neither approach significantly improved performance over the baseline.

Furthermore, in an attempt to enhance the learning process of the Pix2Pix model, we replaced the Mean Squared Error (MSE) loss function with Binary Cross-Entropy (BCE) loss. However, our experimentation revealed that the results obtained with BCE loss were comparable, if not slightly inferior, to those obtained with MSE loss.

Additionally, it's worth noting that in the early stages of our experimentation, we attempted to use the flower images from the ImageNet dataset, incorporating a random flip as a data augmentation technique. However, due to the predominance of white and yellow flowers in the dataset, we encountered limitations in achieving diverse colorization outcomes. Consequently, we transitioned to a larger and more varied dataset to better capture the complexities of natural color distributions.

5.3. Comparisons with the state of the art

Figure 3 presents a comparative analysis involving state-of-the-art colorization models. We decided to compare *pix2pix* with popular online tool called DeOldify[1] which is again based on cGAN with a "U-net" generator but also

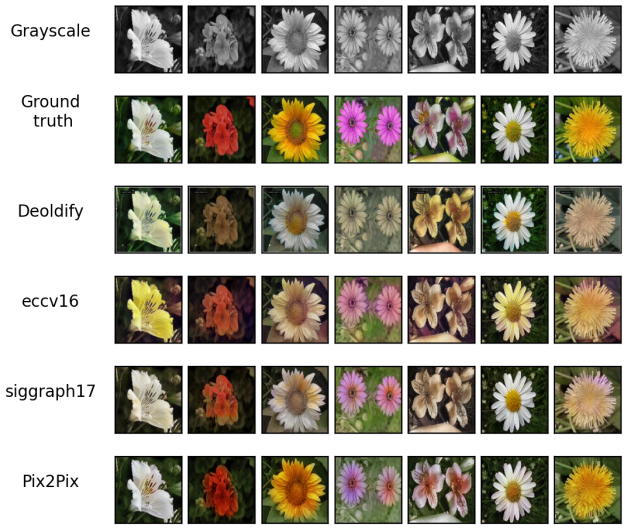

Figure 3. Visual comparison with some state-of-art colorization models. The *pix2pix* yields better colorization in all the test samples.

add a self-attention layer [25] and spectral normalization [14] to both the generator and discriminator, and with the *ecvv16* and *siggraph17* models presented in [26]. The results illustrate that the *Pix2Pix* model, achieves superior colorization performance in the majority of examples. It is crucial to note that this comparison may be perceived as somewhat biased towards *Pix2Pix*, as the other models were originally trained and optimized for generic natural images. In contrast, the proposed model underwent training exclusively on a dataset comprised solely of flower images.

6. Conclusions

In this study, we conducted extensive experimentation to explore various neural network architectures for the task of image colorization. Our results indicate that among the different neural networks evaluated, the U-Net architecture consistently demonstrated the best performance. However, in terms of overall performance, the *Pix2Pix* model emerged as the most effective, producing colorized images that closely resembled the ground truth.

Moving forward, there are several promising avenues for future research and extensions. One potential direction is to explore modifications to the generator or discriminator of the Pix2Pix model, aiming to further enhance its performance. Incorporating techniques such as attention mechanisms or self-attention layers, like in [11] could potentially improve the effectiveness of existing models by enabling them to focus on relevant image regions during the colorization process. Furthermore, exploring novel loss functions [23] or regularization techniques [15] specifically tailored to the colorization task could lead to additional performance improvements.

Through our experimentation, we have gained valuable insights into the strengths and limitations of different neural network architectures for image colorization. These findings underscore the importance of selecting appropriate models and loss functions tailored to the complexities of the colorization task.

References

[1] Jason Antic. jantic/deoldify:a deep learning based-project for colorizing and restoring old images (and video!). Available online: `https://github.com/jantic/deoldify`, 2020.

[2] 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.

[3] Hyojin Bahng, Seungjoo Yoo, Wonwoong Cho, David Keetae Park, Ziming Wu, Xiaojuan Ma, and Jaegul Choo. Coloring with words: Guiding image colorization through text-based palette generation. In *Proceedings of the european conference on computer vision (eccv)*, pages 431–447, 2018.

[4] Raj Kumar Gupta, Alex Yong-Sang Chia, Deepu Rajan, Ee Sin Ng, and Huang Zhiyong. Image colorization using similar images. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 369–378, 2012.

[5] David Hasler and Sabine Suesstrunk. Measuring colourfulness in natural images. *Proceedings of SPIE - The International Society for Optical Engineering*, 5007:87–95, 2003.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[8] Revital Ironi, Daniel Cohen-Or, and Dani Lischinski. Colorization by example. *Rendering techniques*, 29:201–210, 2005.

[9] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of*

the IEEE conference on computer vision and pattern recognition, pages 1125–1134, 2017.

[10] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 577–593. Springer, 2016.

[11] Yingtao Lei, Weiwei Du, and Qinghua Hu. Face sketch-to-photo transformation with multi-scale self-attention gan. *Neurocomputing*, 396:13–23, 2020.

[12] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. In *ACM SIGGRAPH 2004 Papers*, pages 689–694. 2004.

[13] Agnieszka Mikołajczyk and Michał Grochowski. Data augmentation for improving deep learning in image classification problem. In *2018 international interdisciplinary PhD workshop (IIPhDW)*, pages 117–122. IEEE, 2018.

[14] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[15] Aamir Mustafa and Rafał K Mantiuk. Transformation consistency regularization–a semi-supervised paradigm for image-to-image translation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pages 599–615. Springer, 2020.

[16] Maria-Elena Nilsback and Andrew Zisserman. 102 category flower dataset. Available online: `https://www.robots.ox.ac.uk/~vgg/data/flowers/102/`, 2008.

[17] Yingge Qu, Tien-Tsin Wong, and Pheng-Ann Heng. Manga colorization. *ACM Transactions on Graphics (ToG)*, 25(3):1214–1220, 2006.

[18] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[19] Irony Revital, D Cohen-Or, and D Lischinski. Colorization by example. In *Eurographics Symposium on Rendering Techniques, Konstanz*, pages 201–210, 2005.

[20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[22] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[23] Arief Rachman Sutanto and Dae-Ki Kang. A novel diminish smooth l1 loss model with generative adversarial network. In *Intelligent Human Computer Interaction: 12th International Conference, IHCI 2020, Daegu, South Korea, November 24–26, 2020, Proceedings, Part I 12*, pages 361–368. Springer, 2021.

[24] Welsh Tomihisa. Transferring color to grayscale images. *Proc. SIGGRAPH2002, August*, 2002.

[25] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, 2019.

[26] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pages 649–666. Springer, 2016.

[27] 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. *arXiv preprint arXiv:1705.02999*, 2017.

[28] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: Redesigning skip connections to exploit multiscale features in image segmentation. *IEEE transactions on medical imaging*, 39(6):1856–1867, 2019.

[29] H. R. Sheikh Zhou Wang, A. C. Bovik and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[30] Changqing Zou, Haoran Mo, Chengying Gao, Ruofei Du, and Hongbo Fu. Language-based colorization of scene sketches. *ACM Transactions on Graphics (TOG)*, 38(6):1–16, 2019.