COLORIZATION: RECOLORIZING BLACK AND WHITE IMAGES USING DEEP LEARNING

Mario Vázquez (s223268), Nil Palau (s222953), Sergio Reinosa (s223177), and Victor Lyra (s220285)

DTU - Danmarks Tekniske Universitet 2023

ABSTRACT

The aim of the project is to take grayscale images and generate coloured versions by predicting the RGB values for each pixel as accurately as possible. Additionally, we intend to compare the results of the colouring process with the original RGB images to assess the accuracy of our architecture, especially when applying various data augmentation techniques (e.g., to prevent the neural network from colouring the top part of the images blue due to the presence of sky in many pictures). Finally, we plan to modify the architecture to achieve improved results.

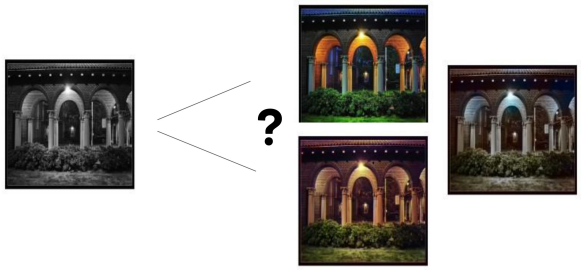

Fig. 1: Possible results for colorizing a grayscale image

1. INTRODUCTION

In the innovative and dynamic world of computer vision, one of the captivating challenges that has gathered a huge attention recently is the colorization of grayscale images. In this report it is going to be documented the work done during the final project of the Deep Learning course. The project that has been chosen is: Colorization of grayscale images.

The goal is simple, transform a grayscale image, with only 1 channel, into a color image with 3 channels, RGB. At the same time the objective pursued is that this images are not just colorized, but that they can resemble reality, so one of the main goals is to have plausible representations.

This last objective mentioned above is the biggest challenge that the technique of colorization challenges. The reason behind it is that the conversion from grayscale to RGB is not a exhaustive function. This means that there is not a 1 to 1 correspondence, or, said in other words; that multiple colorizations can be deemed equally valid for a single grayscale image grayscale value. This can be seen in Figure 1, where is shown 1 grayscale image and 3 possible colored representations of it.

In the recent years the topic of deep learning-based image colorization techniques (DLIC) has gained a lot of popularity between the research community and many studies have been done on it. In this case, the powerful feature extraction ability of deep learning in image processing has shown great application potential and has hugely propeled the development of DLIC. The first DLIC method was proposed in 2015 [1] and have shown superior performance over conventional solutions and are constantly improving to the state of the art. Since 2015 there has been different approaches to this problem, beign the more used ones Convolutional Neural Networks (CNN) based methods, CapsNet-based methods, GANs-based methods and Transformer-Based methods. All of these techniques have leaded to big improvements within the field and great results [2].

The applications that this technology can have are many, being some of them colorization of historical footage, historical photograph restoration or animation scene design.

2. DATASET

The dataset utilized for this project is the MIT Places dataset [3], which comprises approximately 40,000 images. These are partitioned into a training set of about 30,000 images and a testing set of 10,000 images. Each image has a resolution of 256x256 pixels.

Originally made for scene recognition tasks, the MIT Places dataset offers a diverse collection of images across various scenes and situations capturing from natural landscapes to urban settings, each offering its unique palette and lighting conditions. This diversity ensures a broad spectrum of environments is represented, providing the model with a wide range of colors and contexts for learning.

Moreover, the dataset's comprehensive nature provides an opportunity to challenge and evaluate the model's ability to

handle diverse and complex colorization tasks. The range of scenes ensures that the model encounters and learns from a variety of lighting conditions and object arrangements. This variety not only helps in the generalization of the model but also in its capability to adapt to different color distributions and texture details.

2.1. Color space

The color space that has been used in this project to perform the training of the neural network is CIELAB. This is a widely used color space in deep learning-based image colorization. Unlike RGB, CIE LAB is perceptually uniform, ensuring consistent color differences aligning with human vision. Its separation into luminance (L*) and chromaticity channels (a* and b*) enhances the model's ability to understand color relationships. This uniformity contributes to more realistic colorizations, making CIE LAB one of the best color spaces for applications where replicating human-like color perception is crucial.

In the training process, the luminance channel (L*) serves as the input to the network, while the chromaticity channels (a* and b*) constitute the ground truth. This setup allows for the original grayscale image (L*) to be merged with the predicted chromaticity channels, subsequently converting the combined LAB image back to the RGB color space for display and evaluation purposes.

3. MODEL

In this section, it is described the model architecture that has been used in the project. As it has been mentioned in the introduction, there is many approaches that have gotten popular in the last years when it comes to image colorization, having methods that rely on GANs, transformers, CapsNet or CNNs. This last one is the one that have been chosen for our project, CNNs.

The reason to choose this type of arquitecture is that the team is more familiarized with this kind of Neural Network. So, the understanding about this kind of network is going to be more deep and the changes that can be done in the model are going to be in a more low level.

About the model, this follows a structure that can be considered very similar to the one of a encoder-decoder CNN. Having a first part that reduces the size of the image and extract features and a second part that tries to reconstruct the original image by an upsampling and a feature expansion.

The input of the neural network a colored image. This image is transformed to CIELAB and the L,A and B channels are splitted. The L channel, which is similar to a grayscale image, is the one that goes into the CNN, while the channels A and B are taken as a ground truth that is used to compare with the output of the network.

The first part of the neural network uses a ResNet, which is a type of architecture based on convolutional layers and pooling layers. It is an architecture designed to extract features from images in a way that allows the network to extract more complex representations. In addition, ResNet also implements an interesting variation; it introduces residual blocks, which allow the network to skip some of the connections during training. The advantage of this is that the network is more likely to learn the difference between the input and the desired output in a layer instead of a direct mapping from input to output. In the case of this project the ResNet used is called ResNet-18, which indicates that it has 18 layers.

The second part of the neural network acts as a decoder, taking the features obtained from the ResNet and resizing the image obtained to the original size. In order to do this, the network uses deconvolutional layers as well as upsampling layers. The output obtained at the end of the network is a image that has the same size as the input image and some colors in it. This image is compared with the ground truth, which is the original image, and it evaluates the performance using a loss function.

In Figure 2, it can be seen the structure of the CNN.

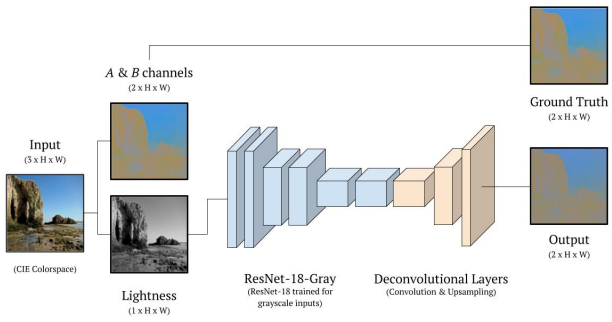

Fig. 2: CNN Arquitecture Representation

4. LOSS FUNCTIONS

For our task, the choice of a loss function is extremely important as it quantifies the difference between the predicted image and the ground truth, subsequently guiding the optimization of the model. A well-chosen loss function will enhance the model's performance, ensuring that it not only reproduces colors accurately but also maintains the perceptual integrity of the image. In this context, we explored several loss functions to identify those most conducive to high-quality colorization. Our investigation into these functions aligns with recent advancements in the field, as discussed in [2].

The following subsections delve into the specific loss functions we evaluated.

4.1. L2 loss

The L2 loss, also known as Mean Squared Error (MSE), calculates the sum of the pixel-wise square differences between the target and predicted images. This is a widely used loss function that tends to converge very fast. On the other hand, this loss is not robust to the inherent multimodal properties of the colorization problem [4].

4.2. Smooth-L1 Loss

The Smooth-L1 loss is a combination of the L1 and L2 losses, which is calculated by:

$$L_\delta(\hat{Y}, Y) = \begin{cases} \frac{1}{2}(\hat{Y} - Y)^2 & for \left|\hat{Y} - Y\right| \leq \delta \\ \delta \left|\hat{Y} - Y\right| - \frac{1}{2}\delta & otherwise \end{cases} \quad (1)$$

where $\hat{Y}$ is the colorized image, Y is the ground truth, and, usually, $\delta = 1$.

4.3. Perceptual Loss

The Perceptual loss was originally designed for image super-resolution and style transfer tasks. It works by passing both the predicted and the target image through a pre-trained network extracting outputs from some of the intermediate layers and comparing them. This loss can make up for defects of pixel-level loss functions and better measure the perceptual and semantic difference between the images.

To use it in our solution, we used a pre-trained VGG network and extracted the features of 4 intermediate layers. But, from our preliminary tests, this loss function ended up using a lot of memory and considerably slowing down the training loop while also not giving such good results. Because of that, we ended up not proceeding to use this loss in future tests.

4.4. Total Variance Loss

Also called TV loss, this loss is defined as the sum of absolute differences between adjacent pixels in the predicted image. It can be calculated by:

$$L_{tv} = \sqrt{(y_{i+1,j} - y_{i,j})^2 + (y_{i,j+1} - y_{i,j})^2} \quad (2)$$

this suppresses the noise in the generated image by improving its smoothness level. This loss can not be used on its own as it serves as a regularization factor for the network.

4.5. Color Weighted Loss

The Color Weighted loss gives more importance to infrequent colors in the training set. Most of the colors of the a* and b* channels seen during training are of values between -25 and 25 (Figure 3), this pattern is learned by the network and it ends up predicting very dull images without any bright colors.

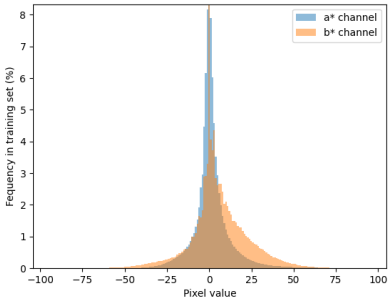

Fig. 3: Pixel value frequency in a* and b* channels in the training set.

This loss function divides the color values into different bins and uses the inverse of its frequency in the training set as a scaling factor for when the network is trying to predict a color that belongs in that bin. In other words, it multiplies the difference between the predicted and the ground truth color by a large number if the target color has a small frequency in the training set, or it multiplies it by a small number otherwise.

5. METRICS

After obtaining results by rebuilding the image using the A and B channel outputed by the different CNN's we have tested, we also wanted to further evaluate the results.

A widely used method for evaluating the final results when facing challenges of recolorizing images, is by using human evaluators to dictaminate a subjective evaluation. So the first phase for discarting models consisted of visually checking the output.

We chose three different metrics for evaluating: Mean Squared Error (MSE) after reconstruction, Structural Similarity Index (SSIM) and Peak Signal-to-Noise Ratio (PSNR).

5.1. Mean Squared Error (MSE)

The Mean Squared Error we measure the average squared difference between corresponding pixel values of the ground truth image and the colorized images. This metric is widely used in image quality assessment.

$$\text{MSE}(x, y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2$$

Where the variables are:

- x and y: The two images.
- N: The total number of pixels in the images (255x255x3).

- x_i and y_i: The pixel value at position i in images x and y.

The MSE is calculated as the average of the squared differences between corresponding pixel values in the two images. It quantifies the average squared difference between the original and reconstructed (or compared) images, with a lower MSE indicating better similarity.

5.2. Structural Similarity Index (SSIM)

The Structural Similarity Index metric measures the similarity between two images. It takes into account different aspects such as luminance, contrast, and structure. The value ranges from -1 to 1, where 1 indicates perfect similarity.

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

Where the variables are:

- x and y: The two images.
- μ_x and μ_y: The mean intensities of x and y.
- σ_x^2 and σ_y^2: The variances of x and y.
- σ_{xy}: The covariance between x and y.
- c_1 and c_2: Constants to avoid instability when the denominators are close to zero.

5.3. Peak Signal-to-Noise Ratio (PSNR)

This is a metric that measures the quality of an image by comparing the original image with a compressed or reconstructed version, we thought that using this index would be helpful to further analyze the reconstruction quality once colorized. The value it outputs is expressed in decibels (dB) in a logarithmic scale, where exactly similar images will have an infinite value of decibels and if the opposite, a value of 0 decibels.

$$\text{PSNR}(x, y) = 10 \cdot \log_{10}\left(\frac{\text{MAX}^2}{\text{MSE}(x, y)}\right)$$

Where the variables are:

- x and y: The two images.
- MAX: The maximum possible pixel value in the images (in our case, 255 since they were converted to RGB before).
- MSE(x, y): The Mean Squared Error between both images.

6. TRAINING

Our training strategy focused on refining the decoder component of the model, outlined in Section 3. Leveraging the pretrained weights of the ResNet-18 encoder, we concentrated our efforts on optimizing the decoder for proficient image colorization. Key aspects of our training process are outlined below.

As described in Section 4, a number of different loss functions were used to train the model and compare the results.

6.1. Architecture modelling

The initial step involved crafting a decoder architecture that not only delivered satisfactory results but also restored images to their original dimensions. Extensive experimentation was conducted, exploring different upsampling methods, adjusting convolutional layers, varying features, and implementing diverse regularization techniques. Ultimately, a configuration with four convolutional blocks, incorporating batch normalization, ReLU activation, and upsampling layers, emerged as the most effective.

6.2. Hyperparameters

We employed the Adam optimizer for training, configuring the model to run for 100 epochs with batch sizes of either 128 or 64. The learning rate varied between 1×10^{-3} and 1×10^{-4}, introducing the necessary variability for robust model adaptation.

6.3. Training process

Encountering challenges during training prompted a thoughtful approach. Traditional loss function comparisons proved challenging due to significant magnitude differences. Consequently, the focus shifted towards evaluating RGB reconstruction outputs. Surprisingly, early stopping mechanisms were found ineffective, as improvements in output quality did not always correlate with reductions in loss values.

Visual inspections of recurrent outputs facilitated informed decisions on experiment viability, streamlining the training process. In Figure 4 we can see an example of these recurrent outputs of the same image during different epochs, where the learning progress of the CNN can be acknowledged

For these reasons, we decided to evaluate the results after the RGB reconstruction of our outputs with the metrics described in section 5, instead of presenting metrics and results of the training and the A and B channels images.

6.4. Computational Resources

Our image colorization model was trained on the DTU high-performance computing cluster, utilizing three nodes, each equipped with four Nvidia Tesla V100 GPUs, boasting 32GB

Fig. 4: Visual example of the CNN learning progression over different epochs.

of memory each. The computational resources played a pivotal role in meeting the demanding requirements of training deep learning models on a substantial dataset. On average, each model configuration took approximately 3 hours to complete 100 epochs.

6.5. Results

The training phase resulted in a well-performing model, capturing intricate details in colorization. The success observed during training motivates further exploration in subsequent project phases.

Table 1 presents a comprehensive overview of the quantitative results obtained from our experiments with best results, with a comparative reference to Zhang et al.'s model. Notably, the choice of loss function and hyperparameter configuration played a pivotal role in shaping the colorization outcomes.

Network	Loss Function	Epochs	Learning Rate	SSIM	PSNR	MSE
Zhang, et al.	-	-	-	0.0241	6.79	15296.60
Ours	MSE	100	1e-4	0.1264	9.22	8799.35
Ours	MSE	98	1e-3	0.8197	17.79	1214.07
Ours	MSE	101	1e-4	0.8244	18.39	1076.04
Ours	MSE	93	1e-4	0.1267	9.25	8720.09
Ours	MSE	100	1e-4	0.8235	18.71	1024.60
Ours	L1	100	1e-3	0.8148	18.30	1137.31
Ours	L1	100	1e-3	0.8148	18.30	1137.31
Ours	L1Smooth	100	1e-3	0.8185	18.33	1092.50
Ours	L1Smooth	100	1e-4	0.8220	18.80	1027.55
Ours	WeightedColorLoss	100	1e-3	0.8163	18.77	994.90
Ours	WeightedColorLoss	100	1e-4	0.8188	18.90	978.10

Table 1: Quantitative results comparing our method with the original color using different metrics.

Noteworthy insights from our experiments include the impact of varying batch sizes on overall model performance. Rows 4 and 5 illustrate this point, where the worst results in column 4 had a batch size of 64.

Additionally, the comparison of traditional loss functions (MSE, L1, L1Smooth) highlighted the superior performance of the Weighted Color Loss. By giving more attention to infrequent colors, this loss function achieved the most effective image colorization, surpassing more common alternatives like MSE.

In conclusion, our exploration of loss functions and hyperparameters during training has unveiled crucial nuances in image colorization outcomes. The performance variations observed in response to batch size changes underscore the importance of thoughtful configuration. Importantly, the Weighted Color Loss emerged as a powerful tool, emphasizing the significance of addressing infrequent colors for superior colorization results. These findings not only contribute valuable insights to the field but also lay the groundwork for informed decisions in subsequent stages of the project.

7. CONCLUSIONS

In Figure 5, we present a visual comparison of significant results, juxtaposing the original images with colorizations generated by Zhang et al.'s model and our best models utilizing Mean Squared Error (MSE) and Color Weighted Loss as loss functions.

It is noteworthy that Zhang et al.'s results exhibit more vivid colors, attributed to their training on a class-balanced dataset with a diverse color palette. This approach allows their model to effectively colorize small or intricate objects, evident in instances such as the microscope slide in the third photo. However, this richness in color sometimes leads to inaccuracies, as seen in the sky in the second row and the flowers in the seventh row.

Conversely, our models adopt a more conservative approach to colorization, resulting in less vibrant outputs. The MSE-trained model struggles with colorizing images featuring unconventional color palettes beyond typical natural landscapes. Notably, predominant colors during training, such as blue, green, and brown, limit the model's ability to accurately reproduce diverse color schemes.

On the contrary, the model trained with Color Weighted Loss aligns with our expectations. It successfully identifies diverse scenes, including landscapes, indoor settings, and various setups. In indoor images (first and fourth rows), the model discerns the absence of a sky or ground, adapting its color predictions accordingly. In landscape images, it introduces varied textures to avoid monochromatic colorization of individual elements. The last row highlights the model's proficiency in differentiating people from the background, enhancing overall recognition compared to other models.

In summary, our exploration into image colorization using convolutional neural networks has revealed a nuanced interplay between vividness and accuracy in the generated out-

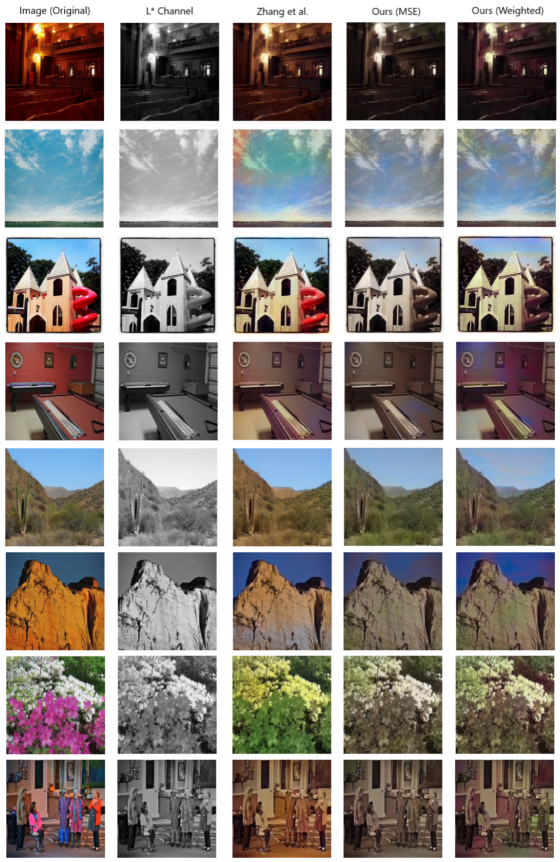

Fig. 5: Results comparison. From left to right: original image, greyscale image, results from [4], our results with MSE loss, and our results with Color weighted loss.

puts. The comparison with Zhang et al.'s model showcases their remarkable ability to infuse images with vibrant colors, attributed to their training on a well-balanced dataset. This richness, however, comes at the cost of occasional inaccuracies, as demonstrated by colorization artifacts in specific regions. Conversely, our models, particularly the one trained with Color Weighted Loss, exhibit a more restrained yet contextually faithful colorization. This approach succeeds in diverse scenarios, showcasing adaptability in recognizing varied scenes, textures, and spatial contexts. The choice between these approaches ultimately hinges on the specific application requirements, whether emphasizing visual richness or prioritizing accuracy in the colorization process. Our findings contribute to the broader discourse on balancing aesthetic appeal with fidelity in image colorization tasks, offering insights for future developments in this evolving domain.

8. FUTURE WORK

In this section, we will discuss different future works that could improve the project and that we did not have enough project time to apply them.

8.1. Class balancing

After seeing the results in Table 1 it has been clear than one of the best performing loss functions and models in general was the ones using Weighted color loss. This method, as it has been explained before, it gives major importance to less frequent colors.

This tells us that a proper dataset class balancing could be beneficial in some extent if applied together with widely used loss functions such as Minimum square error or L1. Even tho class balancing for picture-based datasets is not as trivial as it could be with datasets of other sorts, we have found a good amount of interesting tools to achieve so.

8.2. Transfer learning

By first training the model on a big dataset, we could use that pre-trained model and retrain it on a more specific dataset.

It can be guessed that using first the model on a bigger dataset, and then using it on a case-specific one, such as landscapes, could greatly improve the quality of the results obtained.

8.3. Cross-modal colorization

As we have seen in modern colorization techniques, using other data inputs such as a textual description, have resulted to be very useful during colorization processes. Building a context-aware colorization model would be a nice next step to improve our current model.

8.4. Real-time video colorization

Another interesting application could be trying to use a lighter CNN model for doing a real-time application to colorize videos on the fly.

Right now we are using a Resnet-18, so using a variant of this one, such as the Resnet-9, could make it possible to achieve this performance.

Thus, by using an architecture capable of outputting images at the same pace as they are played in the video, we could create real-time video colorization.

9. REFERENCES

[1] Zezhou Cheng, Qingxiong Yang, and Bin Sheng, "Deep colorization," December 2015.

[2] Shanshan Huang, Xin Jin, Qian Jiang, and Li Liu, "Deep learning for image colorization: Current and future prospects," *Engineering Applications of Artificial Intelligence*, vol. 114, pp. 105006, 2022.

[3] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva, "Learning deep features for scene recognition using places database," *Advances in neural information processing systems*, vol. 27, 2014.

[4] 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*. Springer, 2016, pp. 649–666.