Comparative Analysis of Colorization Techniques for Grayscale Images

Final Report
February 10, 2024

Rasmussen, Magne Egede s183963
Pepping, Bartholomeus Diederik s223301
Lam, Hui Yin (Henry) s231366
Gummesen, Thomas Skøtt s183828

Danmarks
Tekniske Universitet

Supervisor: Thierry Silvio Claude Soreze, Project Manager, DOLL Virtual Lab
DTU - Department of Electrical & Photonics Engineering Diode Lasers & LED Systems

Abstract

In modern communication, visual enhancement is key. Thus, to preserve cultural heritage, colorization of otherwise grayscale images plays a pivotal role: Colorization breathes new life into the historic images making them more relatable & relevant to audiences now a days.

From a modelling perspective, automatic image colorization is complex as the problem is essentially ill-posed. Historically, significant user interaction has therefore been applied either by providing color at specific pixels or linking to reference images. Both methods are time- and energy-consuming. Following more recent research, this paper proposes several fully automatic colorization techniques building on fundamentally different ideas and mathematical fields.

Specifically, we perform a comparison study between gray-box models as regression and Bayesian techniques to the completely black-boxed neural networks. Based on a sufficient data set with different landscape Untill now, we have focussed on general data sets with many different sceneries, we conclude in the paper that deep learning methods do have advantages both in respect to numerical consideration, visual appeal, and quantitative color image metrics. However, the scenario changes for smaller data sets or for more specific objectives, where classical grey box modelling of automatic image colorization still can find use cases.

Contents

1 Introduction

In this project, we search for widespread methods to solve the automatic image colorization problem. In essence, the idea of automatic image colorization constructs an ill-posed inverse problem. There is no unique colorization of a grayscale image as we go from a 'sparse' 1-D pixel space to a 3-D space [1].

Image colorization has been of research interest for decades, trying to preserve historical events and enhance visual appeal. Initially, user-inputted color priors methods have been applied in color annotations or reference images. In alignment with more recent literature [1, 2], we attempt to solve the more complex but more applicable fully automatic colorization based on a large data set of images. We address critical considerations for automatic coloring by utilizing different modelling approaches, pre-processing ideas, and global and local outlooks. Further, we attempt to elucidate the pros and cons of classical image analysis approaches and deep neural networks.

As colors are continuous variables, we start by employing a regression technique as a classical way of automatic image colorization [3, 4]. Specifically, we look at Support Vector Regression along Markov Random Fields for smoothing in SLIC-based segments (SVR). There are a couple of issues with regression methods. In real-world images, many objects have similar local descriptors but contain different colors. For example, having multiple different colored cars in our training data would, with a regression method, imply that a grayscaled car is likely to be colored gray. To make multi-modality in image feasible, we provide a probability-based, non-parametric method using Gaussian Parzen Window estimators along spatial coherence (PWM). Graph-cut algorithms connect the estimators and the coherence to obtain local and global relevant colorizations [1, 5].

Finally, we approach the image colorization problem utilizing Deep Neural Networks by exploring two machine learning approaches: CNN and cGAN [6, 7]. The first method involves using a CNN model with transfer learning, which is beneficial in dealing with time constraints. Meanwhile, the second method employs a cGAN, a deep learning model comprising two distinct components: the generator and the discriminator. In the cGAN model, the generator creates realistic-looking images that can deceive the discriminator. On the other hand, the discriminator is responsible for classifying and distinguishing between real and generated (fake) images.

The report is a classic research article: We provide an overview of our dataset in section 2. In section 3 we deep dive into the three colorization techniques both theoretically and application-wise. Through section 4, we provide results both visually & using different comparable metrics. They are discussed in section 5 where we consider time limitations & alternative approaches. The report culminates with a conclusion and relevant next steps.

2 | Data Exploration

2.1 Dataset

Our dataset comprises two folders containing 7,129 images of scenes such as streets, buildings, mountains, glaciers, and trees [8]. The dataset includes color and grayscale versions of these images, allowing for a direct comparison of results across different methods. Most of the images has a size of 150 × 150 pixels, while some of the images were smaller. We randomly selected 80% of the images as training images and the remaining 20% as testing images. The same distinction between training and testing images is used for the three different methods to avoid having the choice of images affecting the relative performance of the different methods. However, for the SVR and PWM, only a subset of images is used for training, as the methods scale poorly with the size of the training set.

2.2 LAB Conversion

Accurately measuring color similarity and associating saturated colors with corresponding grayscale levels are crucial in image colorization. In the classic RGB color space, every pixel is represented by three color values: Red, Green, and Blue. Hence, in an 8-bit image, each channel denoted (R, G, B) ranges from 0 to 255. The combination of these three channels determines the overall brightness of the image. On the contrary, grayscale images have only the intensity value for each pixel, resulting in a monochromatic image representation without any color information.

Figure 2.1 – LAB 3D-space [9]

By utilizing LAB color space, it has the ability to separate the luminance or lightness component (*L channel*) from the color components (a and b channels). From the visualization of the LAB 3D-space [9], the *L channel* solely represents the grayscale axis, capturing only the intensity or brightness information of the image from 0 to 100. On the other hand, the a and b channels encode the orthogonal color axes, representing the green-red and blue-yellow color information from -128 to 128, respectively. This separation enables the model to handle color variations more effectively, especially in diverse lighting conditions. As a result, the colorization model becomes more robust against changes in lightness, a common challenge faced in colorization tasks.

Furthermore, the LAB color space's underlying metric is designed to approximate human perception of color distances as accurately as possible. This means that the differences in color within the LAB space correspond more closely to how humans perceive color variations, aligning colorizations with human visual expectations. This property is critical in achieving believable and visually pleasing colorizations, as the color differences are represented in a way that corresponds well with human perception, enhancing the overall quality of the colorized images.

3 | Methods

In this paper we compare three different approaches different approaches: Support Vector Regression, Parzen window & spatial coherence and deep learning. Within the deep learning part, two different approaches will be compared, namely, convolutional neural network with transfer learning from VGG16 model and a conditional generative adversarial network. ALl modelling approaches are discussed in detail in the three sections below.

3.1 Support Vector Regression

A subset of the 80% training data is used to train our SVR method, the reason for this is poor scaling of SVR with the number of samples SVR fits for [10]. 5%, 10% and 20% of training images is used, which allowed for a comparison on the quality of the method for varying number of training images. In the training process, each colored image has been smoothed with a 3×3 Gaussian kernel with $\sigma = 1$. On each coloured image in the training set we use the SLIC superpixel algorithm to segment the image into segments of similar grayscale brightness. The number of segments specified is 40 however, the SLIC algorithm will sometime segment into fewer segments than specified. After the image has been segmented, the image is converted to LAB color space by using the module `skimage.color.rgb2lab`, and the scales of the a and a channels are reduced to be between -1 and 1. The a and b channels are assumed to be independent of each other.

The center of each segment, and the arithmetic mean of a and b of each segment are calculated. A 2D Fourier transformation is performed on the L channel in a square around the center of each segment, which size is dependent on image size and the number of segments we specified in the SLIC algorithm. Each 2D FFT is reshaped into a vector, where an array containing the 2D FFT information is generated from each of the vectors generated by the 2D FFT of each segment. This array together with the average a and b for each segment is calculated for every training image.

The python module `sklearn.svm.SVR` is used to generate a Support Vector Regression model, with the average a of each segment of each image being fitted against the array containing 2DFFT for each segment for each image. The same process is performed with b. In order to test the regression model the same method to segment the grayscale image used, and the arithmetic mean of a and b of each segment is identified. The same array

containing 2D FFT information about the L channel is also generated. The model generated by `sklearn.svm.SVR` is then used to predict the a and b channels of each segment. The three terms of the LAB image had been scaled down in the training process, so now with the predicted a and b values, we scale them back up and convert them back into RGB.

In this subsection, it has been attempted to recreate the same process as used in [11]. Hence, an attempt was made to use Markov Random Field in order to smooth across adjacent segments, but the attempts at implementing it did not improve the quality of the results. So Markov Random Field was cut from the process.

3.2 Parzen Window Methods & Spatial Coherence

In this section, we move away from classical regression colorization methods to make (the natural) multi-modality feasible in color selection. Inspired by [1], we infer Parzen Window estimators to learn local color predictors and color variations. The likelihood of color variation from the training data we directly utilize to model the spatial coherency. The probabilistic, non-parametric, scalable and 'machine learning'-inspired method is used as a standalone, but Parzen Window estimators can also be directly employed in graph-cut algorithms [1].

Graph-cut algorithms are optimization techniques based on probabilistic density functions widely used for complete image results [12]. By combining local descriptors and predictions with spatial coherency functions for local pixels, graph-cut algorithms target to provide global interaction across pixel coloring. Hence, graph-cut algorithms are an optimization of coloring grayscale images with respect to both local & global predictors.

Parzen Window Methods (PWM)

Before introducing a Gaussian Parzen Window model we briefly state some of the performed pre-processing work. For a Gaussian Parzen Window model, the grayscale pixel intensity alone is not sufficient. We seek more information from each grayscale pixel, which leads to the use of SIFT (Scale-Invariant Feature Transform) [13]. The SIFT algorithm is based on the key idea of extracting robust pixel features that are invariant to changes in scale and rotation. Fixing a specific pixel, the algorithm uses the gradient magnitude and orientation within a local neighborhood to produce 128 pixel feature variables. With the now 128 features for each pixel in hand, we perform PCA to condense information by only looking at the 10 most important eigenvectors. It is worth highlighting that PCA is performed on non-linear relationships between features (non-linear data) [14]. Hence, the performance is sub-optimal. Generally, the application of PCA to SIFT features is not common practice, also because SIFT descriptors are already designed to be compact. Having performed PCA, the explained variance of the first 10 components only accumulates to 50%. Hence, there is a large amount of information covered in the 128 features from SIFT that we lose performing the dimension reduction.

For each pixel descriptor, we add 3 supplementary features, namely pixel intensity, the local weighted standard intensity deviation and a smooth version of its Laplacian based on a 5×5 grid. The features are added to include direct physical interpretations of grayscale pixels. For each gray pixel $\mathbf{q}$ (away from the boundary) we can now compute a local description $\mathbf{v}(\mathbf{q})$ uniquely identified from $\mathbf{q}$ consisting of 13 condensed informative features.

With the local descriptors available, we target to connect the trained descriptors $\mathbf{v}(\mathbf{q})$ to discretized colors from the LAB color space in order to establish conditional probabilities $\mathbb{P}(c_i|\mathbf{v})$. Since we are transforming from a continuous space to a discretized, we need to define a set of color bins & corresponding centroids. To avoid unused color bins for our images, we discretize applying Kmeans instead of a regular grid. For the specific approach, the reader is referred to Appendix 8.B. In the upcoming Parzen Window formulas, we let B_i note color bin i along a corresponding centroid color c_i. The bin set size is chosen to be 64 as illustrated in figure 3.1.

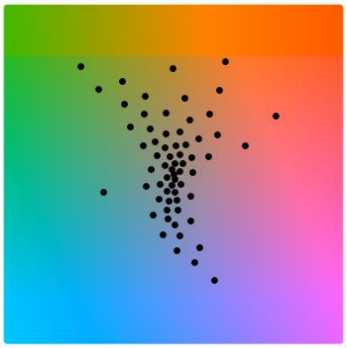


Figure 3.1 – 64 selected color bins from performing K-means on training set's a and b channels.

Let $\mathbf{v}$ denote the descriptor for a greyscale pixel & $\mathbf{e}_j = (\mathbf{w}_j, c(j))$ sets of descriptor for known discretized colors. Now, Parzen Window estimators are given by:

$$\mathbb{P}(c_i|\mathbf{v}) = \frac{\sum_{\{j\,:\,c(j)\in B_i\}} k(\mathbf{w_j}, \mathbf{v})}{\sum_j k(\mathbf{w_j}, \mathbf{v})} \tag{3.2.1}$$

with kernel $k(\mathbf{w_j}, \mathbf{v})$. Specifically, for Gaussian Parzen Window estimators the kernel function is $k(\mathbf{w_j}, \mathbf{v}) = e^{-||\mathbf{w_j}-\mathbf{v}||_2^2/2\sigma^2}$ where the standard deviation σ is a hyperparameter. A proper selection of σ is key for the Parzen Window method to function. It is dependent on the number of training pixels and implies different steepness of neighbouring losses. For large values of σ, it follows that the obtained kernel values become closer and contrariwise. For calculating the conditional probability in eq. (3.2.1), we utilize every pixel available. In application, such progress is too slow. Here, we restrict to sums consisting only of the K-nearest neighbours of v found by utilizing KD-trees to ramp up speed for large datasets.

Spatial Coherence

Instead of choosing a classic prior for spatial coherence, that could be for example detection of edges or corners, we utilize the Parzen Window framework above. From the set $\mathbf{e}_j = (\mathbf{w}_j, c(j))$ we calculate the norm of the gradient, g_j, for each pixel j based on the a & b channels. With $k(\cdot,\cdot)$ the Gaussian kernel, the expected variation of a grayscale pixel is:

$$g(\mathbf{v}) = \frac{\sum_j k(\mathbf{w_j}, \mathbf{v}) g_j}{\sum_j k(\mathbf{w_j}, \mathbf{v})} \tag{3.2.2}$$

Graph-cut Algorithms

Now, from eq. (3.2.1) and eq. (3.2.2) we have built the foundation such that at any grey pixel, we can provide estimated scores of all possible colors and further the probability of color variation. As mentioned initially, this is the two inputs for graph-cut algorithms. Hence, the best color might not be the one with highest conditional probability from eq. (3.2.1), no it is also affected by the probability distributions in neighborhoods. As every pixel has a neighbourhood, the method including color variation implies global interaction across all pixels. Hence, the Graph-cut algorithms are applicable for global results and

function as alternatives to diffusion/'random walk' methods. The optimization problem for image colorization we formulate as the minimization problem with objective:

$$\sum_{\mathbf{p}} V_{\mathbf{p}}(c(\mathbf{p})) + \rho \sum_{\mathbf{p}\sim\mathbf{q}} \frac{||c(\mathbf{p}) - c(\mathbf{q})||_2}{g_{\mathbf{p},\mathbf{q}}} \tag{3.2.3}$$

In eq. (3.2.3) we use new notation. First, $V_{\mathbf{p}}(c(\mathbf{p}))$ is a cost function of choosing color $c(\mathbf{p})$ for pixel $\mathbf{p}$. With Parzen Window estimators, we define the local cost $V_{\mathbf{p}}(c(\mathbf{p}))$ as:

$$V_{\mathbf{p}}(c(\mathbf{p})) = -\log\left(\mathbb{P}(\mathbf{v}(\mathbf{p}))\,\mathbb{P}(c(\mathbf{p})|\mathbf{v}(\mathbf{p}))\right) \tag{3.2.4}$$

where $\mathbb{P}(\mathbf{v}(\mathbf{p}))$ describes the reliability of a color probability estimation. If $\mathbf{v}$ is unlikely, we do not have many close pixels to help colorize. Hence, $\mathbb{P}(\mathbf{v}) \propto \sum_j k(\mathbf{w_j}, \mathbf{v})$. Now, $\mathbf{p} \sim \mathbf{q}$ denotes that the two pixels are neighbors, here we are considering 8-neighborhoods. Further,

$$g_{\mathbf{p},\mathbf{q}} = 2\left(g(\mathbf{v}(\mathbf{p}))^{-1} + g(\mathbf{v}(\mathbf{q}))^{-1}\right)^{-1} \tag{3.2.5}$$

is the harmonic mean of the estimated color variation at neighbooring pixels $\mathbf{p}$ and $\mathbf{q}$. Finally, ρ is a hyperparameter that can be tuned in favour of respectively local and global correctness. The reader is referred to [5, 12] for more on graph-cut algorithms.

3.3 Deep Learning Methods

Convolutional Neural Network

This method explores a Convolutional Neural Network (CNN) approach to colorize grayscale images using transfer learning with the VGG16 model.

A Convolutional Neural Network (CNN) is a deep learning architecture used in computer vision tasks. It consists of essential building blocks such as convolutional layers, activation layers (ReLU), pooling, and fully-connected layers. The convolutional layer applies filters to the input image, generating feature maps that capture visual patterns. ReLU introduces non-linearity, pooling reduces spatial dimensions, and the fully-connected layer makes predictions. CNNs excel at image classification, object detection, and segmentation.

Transfer Learning is a powerful machine learning technique that addresses the limitations of isolated learning paradigms by sharing learned features across different tasks and domains. In the specific case of automatic image colorization, we can leverage pre-trained convolutional neural networks (CNNs) like VGG16 to improve colorization performance significantly. We can extract visual features from the grayscale images by removing the fully-connected layer and keeping the convolutional and pooling layers. The pre-trained VGG16 model, which has learned rich features from many images, can detect important patterns and color-related information effectively. This transfer learning approach enables us to enhance the colorization process, producing high-quality and realistic colorized images with less need for extensive training data.

Model Architecture

Using transfer learning with VGG16, we first create a new CNN model by incorporating the feature extraction layers from VGG16. These layers are set as non-trainable to retain

the pre-trained weights. Data augmentation techniques are applied during training using Keras' ImageDataGenerator to enhance the model's generalization. The grayscale input images are then converted to the LAB color space, separating the L channel (representing lightness) and AB channels (representing color components). The AB channels are normalized by dividing by 128, while the L channel is reshaped to form a three-channel input. This prepares the data for the colorization process, where the model uses the extracted features to produce realistic and accurate colorized images.

Training Process

The training process for our image colorization model involves supervised learning with a large dataset of grayscale and corresponding colorized image pairs. Before training, the dataset is transformed into the Lab color space, where the grayscale images' L channel serves as the input, and the a and b channels are treated as target color components. The model is trained using the Adam optimizer with the mean squared error (MSE) loss function to minimize the difference between the predicted and ground truth color components. To evaluate the model's performance, we adopt K-Fold Cross-Validation, setting k to 5. The dataset is randomly split into k folds, and the model is trained and validated on different subsets in each fold. Early stopping is implemented during training to prevent overfitting, ensuring the model generalizes well to unseen data. The training process results in a high-performance model capable of colorizing grayscale images accurately and realistically.

Results

The experimental results encompassing training, validation loss, and accuracy for each fold have been diligently recorded. The figure 8.3 shown in Appendix 8.C provides the graphical representation of loss plots for the Convolutional Neural Network (CNN) model. A comprehensive table encapsulates the results below for further clarity and reference.

Table 3.1 – Training and Validation Results

Fold	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
1	0.0051	0.7632	0.0041	0.7757
2	0.0051	0.7531	0.0042	0.7732
3	0.0051	0.7562	0.0044	0.7719
4	0.0050	0.7618	0.0041	0.7829
5	0.0040	0.7633	0.0088	0.6796
Average	0.0048	0.7595	0.0051	0.7566

One possible reason for the premature convergence of the CNN model is the limited model complexity. The chosen architecture of the CNN model may not be complex enough to effectively capture the intricate colorization patterns from grayscale images. As a result, the model may fail to learn the diverse color relationships in the data, leading to premature convergence and suboptimal colorization results. Another factor that could contribute to premature convergence is the insufficient size or diversity of the training dataset. The model may not be exposed to a wide range of colorization scenarios and patterns, limiting generalization capability. Augmenting the dataset or using a larger and more diverse training dataset could help the model learn a broader range of colorization patterns and prevent premature convergence.

The choice of the Rectified Linear Unit (ReLU) activation function in the CNN model for colorization is based on its effectiveness in deep learning architectures, even when dealing with grayscale images with symmetrical positive and negative parts. ReLU's popularity stems from its computational efficiency and simplicity, making it well-suited for complex tasks like image colorization. By setting negative inputs to zero, ReLU involves straightforward thresholding, resulting in faster computations during forward and backward passes. This computational efficiency is advantageous in deep neural networks with numerous layers, reducing training time and facilitating real-time inference.

Moreover, ReLU introduces non-linearity into the model, a crucial aspect for learning complex, nonlinear relationships in the data. Although grayscale images have symmetrical positive and negative values, the process of colorization inherently involves nonlinear mapping between grayscale and color components. ReLU's non-linearity enables the model to capture intricate color patterns and relationships effectively, facilitating the generation of accurate and realistic colorized images. Additionally, ReLU helps mitigate the vanishing gradient problem that can hinder training in deep networks. As ReLU has a constant gradient of 1 for positive inputs, it allows gradients to flow freely during training, preventing them from becoming too small and promoting stable optimization.

General Adversarial Network

The conditional general adversarial network (cGAN) is the fourth approach tried to colorize black and white images. The cGAN is a variation one a normal GAN that allows to specify additional information to control the output of the model [15]. The cGAN consists of two sub-modules called a generator and discriminator. The generator will generate a potential colour image based on black and white input image. The discriminator functions as a judge to determine whether the generated image is real or fake.

The generator in this paper is a convolutional encoder decoder (CAE) with 3 Conv2D layers in the ecoder and three to decode it back to the original size. The input is provided as a (150,150,1) and it outputs a (150,150,3) coloured image .
The discriminator is a basic convolutional neural network that categorizes the images as real or fake. The training of these models are sequential. Initially, fake images are generated these are fed into the CNN. After which a half batch amount of real images are fed to the discriminator. As a subsequent training step, the generator is trained through the cGAN. The discriminator is not updated in this step .

A very realistic deep learning network created for this type of work is pix2pix [16] was trained on the dataset used in the ImageNet Large Scale Visual Recognition Challenge which contains 1.4 million images [17]. Indicating the required amount of images for a algorithm of this type.

4 | Results

4.1 Visual Results For Automatic Colorization Methods

(a) Color image

(b) Grayscale image

(c) SVR 5% training

(d) SVR 20% train

(e) PWM 5% training, 100 neighbours in KD-tree image

(f) PWM 5% training, 500 neighbours in KD-tree

(g) CNN - Convolutional Neural Network Method

Figure 4.1 – Image No. 16 in our dataset - Test image.

We ensured that none of our models used any testing data for training. In fig. 4.1, we have shown our final results for a test image (labeled 16) in our data set. The selected test image is taken in a city at nightlight. Our data contains various scenery, from nature to cities. In Appendix 8.D, the reader can see examples from nature and daylight city images.

In comparing the two images generated through the SVR method, we see little difference in the output when comparing using 5% or 20 % of the images for training data. And since the SVR algorithm used scales worse than quadratically with the amount of data [18], SVR appears well suited to smaller data sets. However, the coloring of these images will be blocky, as each segment will be given the same A and B values in the LAB colorspace. This can be mitigated with training and testing with a smaller size of each segment. From fig. 8.4, a common issue of this method is elucidated, where the colors of textures bleed in, i.e., the sky close to the ground becomes a color between that of the sky and of the ground. In figure 8.5, the SVR algorithm is poor at dealing with a cloudy sky.

For the Bayesian Parzen Window method, we have yet to obtain results with the smoothing part from the graph-cut algorithm included. As a direct implication of the lack of smoothing, we see several patches of varying colors. Finally, changing hyperparameter K between 100 and 500, we do not see much difference in the output quality. Hence, from the KDtree structures, choosing only the 100 closest known $\mathbf{w_j}$ pixels to the unknown grayscale pixel $\mathbf{v}$ when evaluating conditional probabilities seems applicable.

The observed disparity in the CNN deep learning neural network's performance, as depicted in figures 4.1, 8.4, 8.5, and 8.6, can be attributed to several factors. Nature images typically offer more distinguishable features, diverse color variations, and isolated objects, making them easier for the network to learn and generalize from the training data. On the other hand, cityscape images present greater complexity with intricate scenes, limited color palettes, and more uniform textures, challenging the network's ability to identify and color the various elements accurately. To improve the network's cityscape image coloring, strategies like data augmentation, balanced training data, domain-specific architectural adjustments, and user feedback can address these challenges and enhance the model's performance in nature and cityscape domains.

4.2 Comparison in Metrics Between Methods

To compare different methods of automatic image colorization, we used four commonly used metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Structural Similarity Index (SSIM), and Peak Signal-to-Noise Ratio (PSNR). MAE and RMSE scores are directly implicated by the chosen color pixel by pixel. The target is to obtain as small errors as possible for MAE and RMSE. Contrary, SSIM and PSNR look into the general structure and noise similarities between the original colorization and the reconstructed image. Here, larger values indicate better image quality

MAE and RMSE are found in LAB color space (as required to measure distances in colors), while SSIM and PSNR are calculated based on the RGB images. Their values and their standard deviations are shown in table 4.1.

Table 4.1 – Metric Comparison for SVR, PWM, and CNN. STD notes the standard variation

	SVR	SVR-STD	PWM	PWM-STD	CNN	CNN-STD
Mean Absolute Error A	7.23	4.42	6.64	2.58	5.75	4.49
Mean Absolute Error B	15.6	7.02	16.6	8.40	13.1	6.61
Root Mean Square Error A	8.61	4.97	9.37	3.26	7.35	5.14
Root Mean Square Error B	18.6	7.92	21.6	8.83	16.1	7.25
SSIM	0.91	0.06	0.74	0.10	0.94	0.05
PSNR	29.9	0.81	19.1	3.26	23.0	3.83

Based on the outputted metric scores in table 4.1, our implementation of CNN deviated less from the true colors of the images than SVR and PWM. The result aligns with the visual interpretations from the previous subsection.

It is worth noting that due to time limitations, SVR and PWM are validated only on smaller sets of test images that contain at least 100 test images selected randomly.

5 Discussion

Broadly we have applied two different techniques which have been defined as deep learning and non deep learning techniques. One of the key differences noted in this paper was the different requirements for training volume between the deep learning methods and the non-deep learning methods. Especially the SVR needed to be trained at a reduced value of already limit pictures. Whereas, it seemed that the limited available images in the data set actually reduced the effectiveness of the transfer learning network. Potentially contributing towards its premature convergence.

Images colorized by the Probabilistic-diffusion and the SVR methods have color issues upon their generation. For the Bayesian model, issues persist with patchy colors that potentially can be mitigated by smoothing methods. Whereas, for SVR potential avenues to be explored are differences in the segments selected. Ensuring that the colors at the boundary do not spill over to the next. In general, we observe that a larger training data set is not necessarily correlated with better results. Hence, it can be discussed if the methods are even appropriate for large data sets with varying scenery. In appendix 8.E we have included a section on Parzen Window models for extended reference image objectives. The section shows that depending on your objective, classical methods might still find relevancy, as Parzen Window models perform well on biased small data sets.

For the cGAN the amount of implementation time for a function network came as a surprise. However, as this is the current state of the art in image translation, it is a subject that should be more explored. Moreover, CNN does perform best on the most utilized metrics in this paper. Additionally, it does provide more "natural" images, as the two other compared methods more problematic flaws such as patches or blocks that have the incorrect color.

6 Conclusion

In this paper, four different methods for image colorization have been compared to each other. The four techniques were support vector machines, Parzen windows, transfer learning, and a cGAN. Unfortunately, due to time constraints, the cGAN has not been fully implemented, therefore, it is not considered in the final comparison.

When inspecting the images from the Parzen windows and SVR methods both have shown to have their individual issues with the colorization. The Parzen windows are described as patchy, whereas, the SVR is blocky.

To compare the different models MAE, RMSE, PSNR, and SSIM have been utilized. The CNN deep learning model has performed the best in most of the metrics indicating that this is the area where additional research must be done. However, the other techniques might be more beneficial when limited data is available.

7 | Next Steps

In this section, we briefly cover the appropriate next steps for each of the methods introduced in this paper. As the project period has been 1.5 weeks, there are a lot of future next steps that are not listed below, including fully different automatic coloring methodologies, setting specific colorization targets/objectives, etc.

SVR The scripts written to perform SVR scale poorly with the size of the training set. A weaker version of SVR scales linearly with the amount of data that exists, `sklearn.svm.LinearSVR`. This method was not tested, but if the quality of the images is similar with this method, then it could allow to use much larger training sets in this method. Additionally successfully implementing Markov Random Fields would also improve the quality of the image colorization.

PWM A pixel-by-pixel version of the Bayesian Method of Parzen Windows has been shown to be slow on large data sets. Further, without any spatial coherence added through Graph-cut algorithms, the resulting images elucidate noise full sections. Combined there are two steps to go ahead: First, a fully integrated local/global threshold optimization along the current implementation. Secondly, it could be interesting to go from pixel level to section level for example using SLIC.

CNN To enhance the colorization CNN model, the next steps include fine-tuning by training additional layers while keeping the VGG16 layers non-trainable, hyperparameter tuning, applying data augmentation techniques, exploring different model architectures, and incorporating a larger and diverse dataset. These efforts aim to improve model performance, convergence, generalization, and ability to produce realistic and accurate colorized images.

cGAN To ensure that the cGAN is evaluated against the other methods. Therefore, the model should be developed further. When the model functions as expected, then proper training and validation should be performed and compared to the other methods. However, when the model functions as expected, one of the key considerations to expand upon is the amount of training data.

8 | Appendix

Appendix A: Author Contributions

Name	Method
Thomas	Support Vector Method
Magne	Parzen Window Method
Henry	CNN Method
Bart	GAN Method

Table 8.1 – Methods and Contributors

Our code is available at https://github.com/MagneEgede/Colorization.

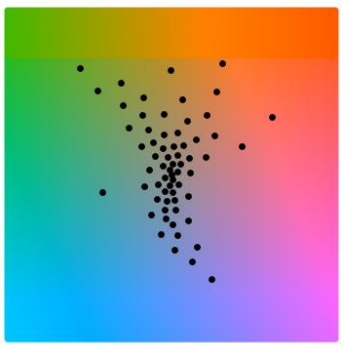
Appendix B: Color Space Discretization - Parzen Window

A critical step in the modelling approach for the Bayesian Parzen Window method is to infer probabilistic distributions for discrete colors given from their local descriptors of a pixel. In other words, having found local descriptor information based on SIFT etc. the target is to provide a probability distribution of likely color selections. To avoid working with probability densities and to be able to classify local descriptors rapidly, we, as mentioned, discretize the 2D a & b channels and afterwards provide a projection method for continuous-valued colors. The discretization part can be performed in numerous ways. The most intuitive way is to construct a grid in $[-128, 128] \times [-128, 128]$ space of the two channels. But practically, the approach is lacking as we obtain many empty bins, where no pixels in the data set are located. Instead, we utilize the K-means method on our training data to provide better segmentation choices. Adding more bins in zones with a high density of pixels allows the modelling to be more nuanced in major parts of the pixel set. With respect to the projection from the continuous-valued colors, we assign the average color (the centroid) to each bin. K-means requires an initial input of a number of bins needed K, i.e. number of discrete colours available for recolorization which again is a hyperparameter. Based on the results in fig. 8.1, 64 color bins are sufficient to at least produce a good approximation of the image. Finally, fig. 8.2 shows that based on the scenery in the set, the discrete color selection varies to include as much nuance as possible.

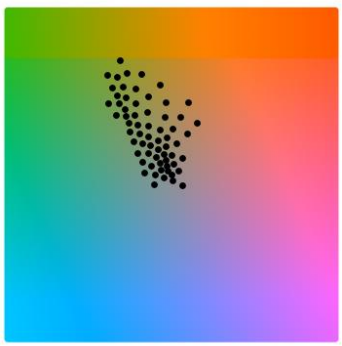
(a) Original given continuously colored image - Image 0 in data set.

(b) Discretized projection based on training data segmented into 64 bins.

Figure 8.1 – Continuously colored image and discretized projection



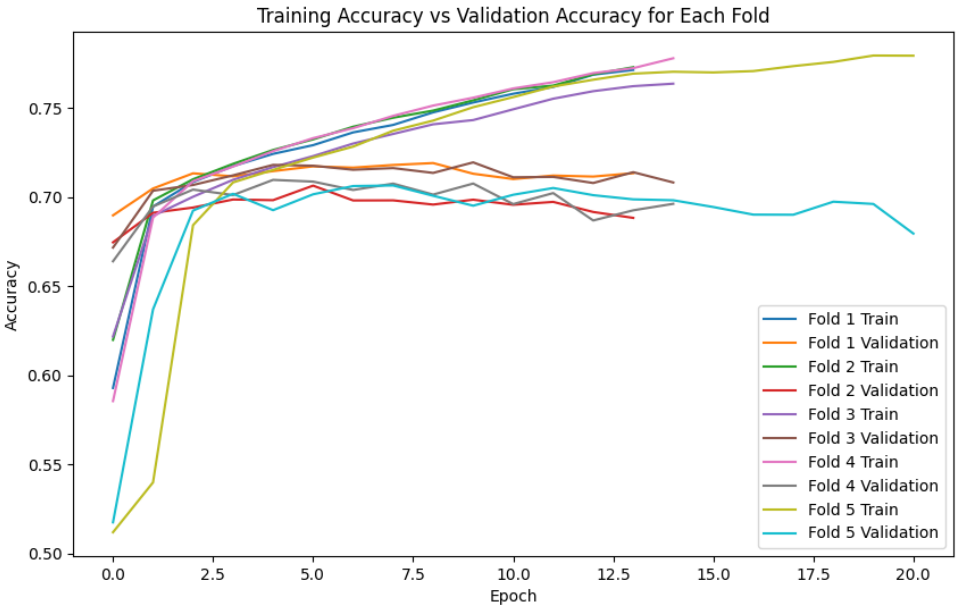
(a) Non-biased set



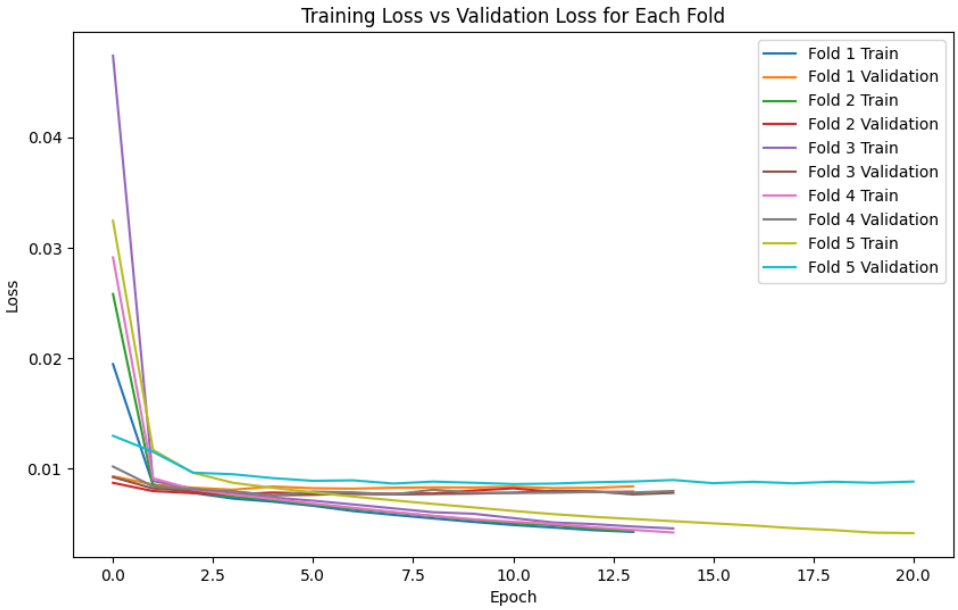
(b) Biased set - only forest images

Figure 8.2 – Discretized colors variation between non-biased and biased training data set. For chosen forest images, see Appendix 8.E, the available discrete colors are nuances of green.

Appendix C: Training & Testing Losses Plots for CNN model



(a) Training Accuracy vs Validation Accuracy for each fold.



(b) Training Loss vs Validation Loss for each fold.

Figure 8.3 – Losses plots for the CNN model.

Appendix D: Extension of Visual Results

(a) Color image

(b) Grayscale image

(c) SVR 5% training

(d) SVR 20% train

(e) PWM 5% training, 100 neighbours in KD-tree

(f) PWM 5% training, 500 neighbours in KD-tree

(g) CNN - Convolutional Neural Network Method

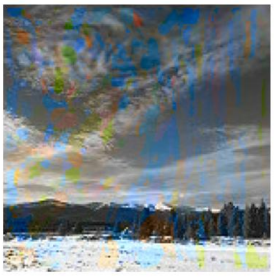
Figure 8.4 – Image No. 17 in our dataset.

(a) Color image

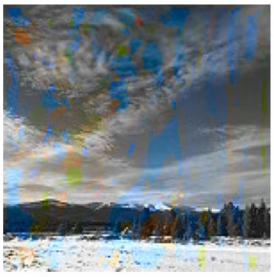
(b) Grayscale image

(c) SVR 5% training

(d) SVR 20% train



(e) PWM 5% training, 100 neighbours in KD-tree



(f) PWM 5% training, 500 neighbours in KD-tree

(g) CNN - Convolutional Neural Network Method

Figure 8.5 – Image No. 18 in our dataset.

(a) Color image

(b) Grayscale image

(c) SVR 5% training

(d) SVR 20% train

(e) PWM 5% training, 100 neighbours in KD-tree

(f) PWM 5% training, 500 neighbours in KD-tree

(g) CNN - Convolutional Neural Network Method

Figure 8.6 – Image No. 19 in our dataset.

Appendix E: Parzen Window method for reference images

Until now, we have focused on general data sets with many different sceneries. It is our conclusion through the 1.5 weeks project period that deep learning methods seem to have the upper hand both in numerical, visual and quantitative results. Within the project period, we did tho observe that the machine learning grey-box modelling methods perform equally well on small data sets compared to larger ones. This led to the idea of testing Parzen Window models on small biased data sets. In essence, this is equivalent to an extended reference image method. The results are shown below for 'reference images' of forests. One sees in fig. 8.7 that for the reference image techniques, Parzen Window models do perform better, especially considering that we have not added smoothing through graph-cut algorithms yet.

(a) Color image **(b)** Grayscale image **(c)** Reconstructed Colorization

Figure 8.7 – Results for Parzen Window Method on biased training set of only 6 images.

The biased training set for the results in fig. 8.7 consists of only six images of different forests. They are all shown in fig. 8.8 and do not all directly reminisce the color image from fig. 8.7. To improve the results of the reconstructed image even further, next steps will be to include images with similar lighting reflections in the training set (even if they do not include forests). Finally, Parzen Window methods are also flexible for the historic solution of user-inputted pixels. At the selected pixel, the conditional probability is set to 1 and the same procedure can be applied.

Figure 8.8 – Biased training set for 'reference image'-technique with Parzen Window methods.

Bibliography

[1] G. Charpiat, I. Bezrukov, M. Hofmann, Y. Altun, B. Schölkopf, and R. Lukac, "Machine learning methods for automatic image colorization," *Computational Photography: Methods and Applications, 395-418 (2010)*, 01 2011.

[2] E. M. Farella, S. Malek, and F. Remondino, "Colorizing the past: Deep learning for the automatic colorization of historical aerial images," *Journal of Imaging*, vol. 8, no. 10, 2022. [Online]. Available: https://www.mdpi.com/2313-433X/8/10/269

[3] S. Liu and X. Zhang, "Automatic grayscale image colorization using histogram regression," *Pattern Recognition Letters*, vol. 33, p. 1673–1681, 10 2012.

[4] H. Ho and V. Ramesh, "Automatic image colorization." [Online]. Available: https://cs229.stanford.edu/proj2015/150_report.pdf

[5] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for markov random fields with smoothness-based priors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 6, pp. 1068–1080, 2008.

[6] C. Qiu, H. Cao, Q. Ren, R. Li, and Y. Qiu, "Automatic image colorization with convolutional neural networks and generative adversarial networks," 2021.

[7] T. Deng, "Image colorization based on deep learning."

[8] B. Mamba. Landscape image colorization dataset, version 1. Retrieved July 19, 2023. [Online]. Available: https://www.kaggle.com/datasets/theblackmamba31/landscape-image-colorization.

[9] M. Alassaf, K. Kowsari, and J. Hahn, "Automatic, real time, unsupervised spatio-temporal 3d object detection using rgb-d cameras," 07 2015.

[10] scikit learn. sklearn.svm. Retrieved July 18, 2023. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

[11] H. Ho and V. Ramesh. Automatic image colorization. Retrieved July 13, 2023. [Online]. Available: https://cs229.stanford.edu/proj2015/150_report.pdf

[12] S. Sinha, "Graph cut algorithms in vision, graphics and machine learning an integrative paper," 01 2004.

[13] O. S. C. Vision. Introduction to sift (scale-invariant feature transform). Retrieved July 17, 2023. [Online]. Available: https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html

[14] scikit learn. sklearn.decomposition.pca. Retrieved July 17, 2023. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[15] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[16] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, pp. 211–252, 2015.

[18] scikit-learn.org. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html