BLACK & WHITE IMAGE COLORISATION USING DEEP LEARNING

COURSE PROJECT REPORT

Submitted by

SHAKTHI VELU A [RA2152008010019]

MASTER OF BUSINESS ADMINISTRATION IN ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

COLLEGE OF MANAGEMENT

Faculty in charge

Dr. S. Prithi

Associate Professor, Department of Computational Intelligence,SRM

S. R.M. Nagar, Kattankulathur, Chengalpet District

3rd Semester
November 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY COLLEGE OF MANAGEMENT

BONAFIDE CERTIFICATE

This is to certify that this is the bonafide record of work done by **SHAKTHI VELU A RA2152008010019]** ofsecond year M.B.A. Degree Course**,** College of Management, SRM Institute of Science And Technology, Kattankulathur for **MBL21306L Deep Learning - 1** during the academic year 2022 -2023(odd semester).

Dr. S. Prithi
Faculty in-Charge
Coordinator

Dr. G. Kumar
Course

Dr. P. Subhashree
NatarajanDean - COM

Internal Examiner-I

Internal Examiner-II

ABSTRACT

This review paper will present a few of the several approaches that have been investigated for the image colorization process. Since the beginning of time, there have been numerous ways to colourise images. The current fashion is toward completely automatic image colorization methods. This paper presents an overview of several different approaches that have been attempted and used, along with their benefits and drawbacks, and a comparison of them.

TABLE OF CONTENTS

ABBREVIATIONS

AES	**Advanced Encryption Standard**
ANN	**Artificial Neural Network**
CSS	**Cascading Style Sheet**
CV	**Computer Vision**
DB	Data Base
DNA	**Deoxyribo Neucleic Acid**
SQL	**Structured Query Language**
SVM	**Support Vector Machine**
UI	**User Interface**
CONV	**Convolutional**

INTRODUCTION

What is Deep Learning?

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called **artificial neural networks**.
If you are just starting out in the field of deep learning or you had some experience with neural networks some time ago, you may be confused. I know I was confused initially and so were many of my colleagues and friends who learned and used neural networks in the 1990s and early 2000s.

The leaders and experts in the field have ideas of what deep learning is and these specific and nuanced perspectives shed a lot of light on what deep learning is all about.

How does deep learning attain such impressive results?

In a word, accuracy. Deep learning achieves recognition accuracy at higher levels than ever before. This helps consumer electronics meet user expectations, and it is crucial for safety-critical applications like driverless cars. Recent advances in deep learning have improved to the point where deep learning outperforms humans in some tasks like classifying objects in images.

There are two main reasons it has only recently become useful:

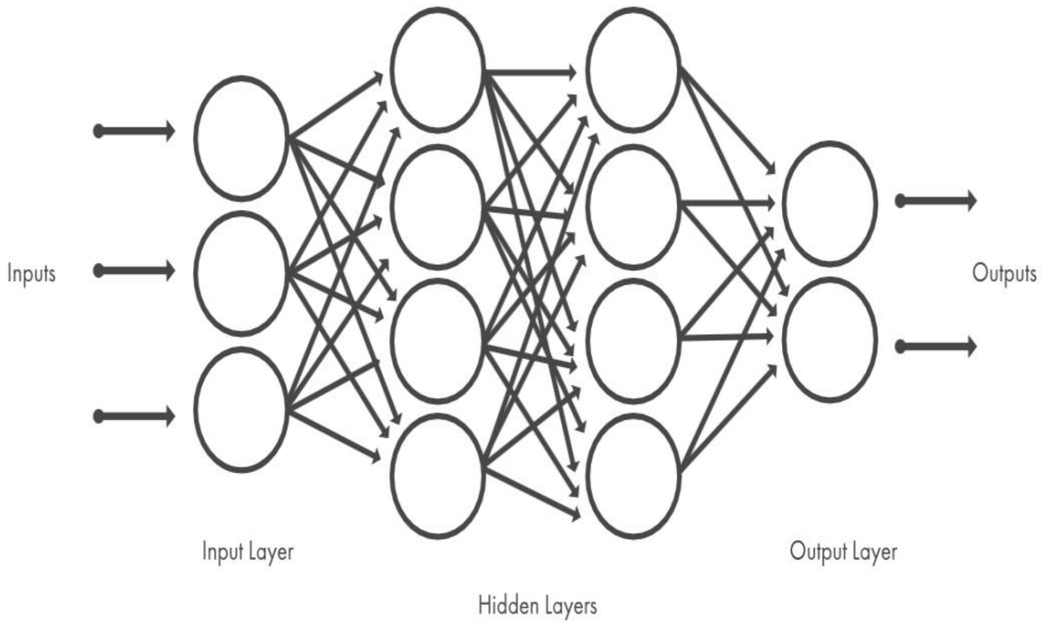
1. Deep learning requires large amounts of **labeled data**. For example, driverless car development requires millions of images and thousands of hours of video.
2. Deep learning requires substantial **computing power**. High-performance GPUs have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less.

How Deep Learning Works

Most deep learning methods use neural network architecture , which is why deep learning models are often referred to as **deep neural networks**.

The term “deep” usually refers to the number of hidden layers in the neural network. Traditional neural networks only contain 2-3 hidden layers, while deep networks can have as many as 150.

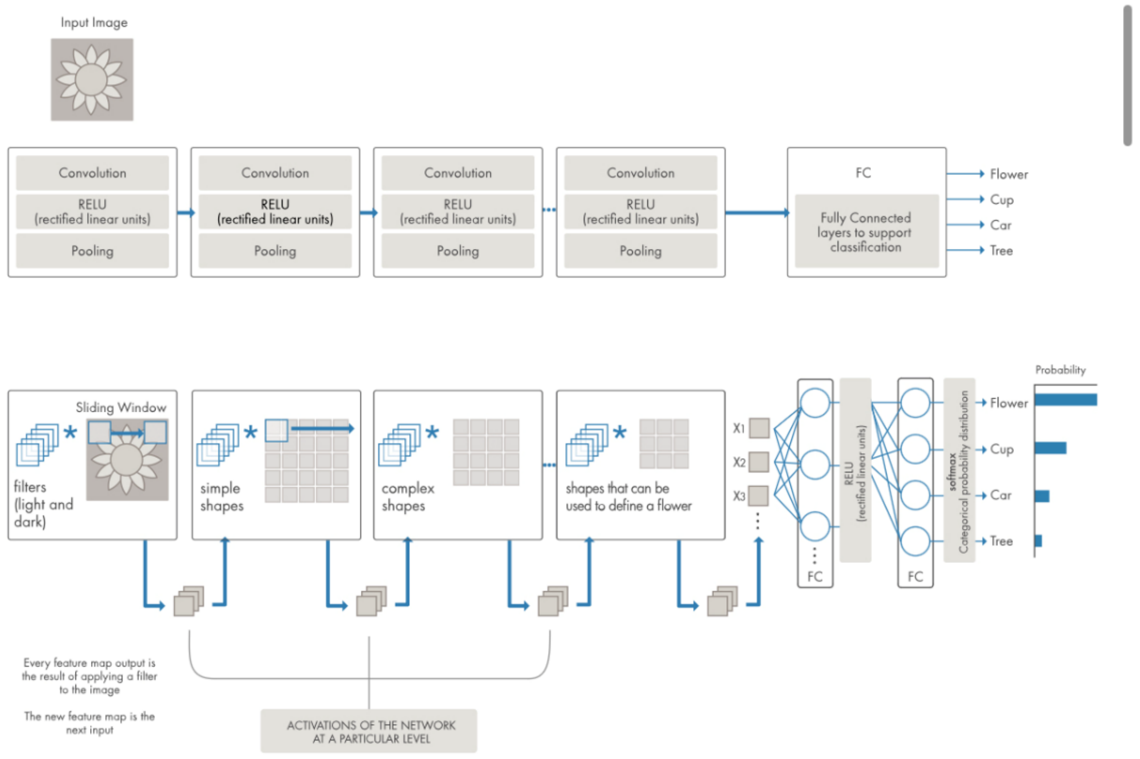
Deep learning models are trained by using large sets of labeled data and neural network architectures that learn features directly from the data without the need for manual feature extraction.



One of the most popular types of deep neural networks is known as convolutional neural networks **(CNN** or convent**)**. A CNN convolves learned features with input data, and uses 2D convolutional layers, making this architecture well suited to processing 2D data, such as images.

CNNs eliminate the need for manual feature extraction, so you do not need to identify features used to classify images. The CNN works by extracting features directly from images. The relevant features are not pre-trained; they are learned while the network trains on a collection of images. This automated feature extraction makes deep learning models highly accurate for Computer Vision task such as object classification.

CNNs learn to detect different features of an image using tens or hundreds of hidden layers. Every hidden layer increases the complexity of the learned image features. For example, the first hidden layer could learn how to detect edges, and the last learns how to detect more complex shapes specifically catered to the shape of the object we are trying to recognise.

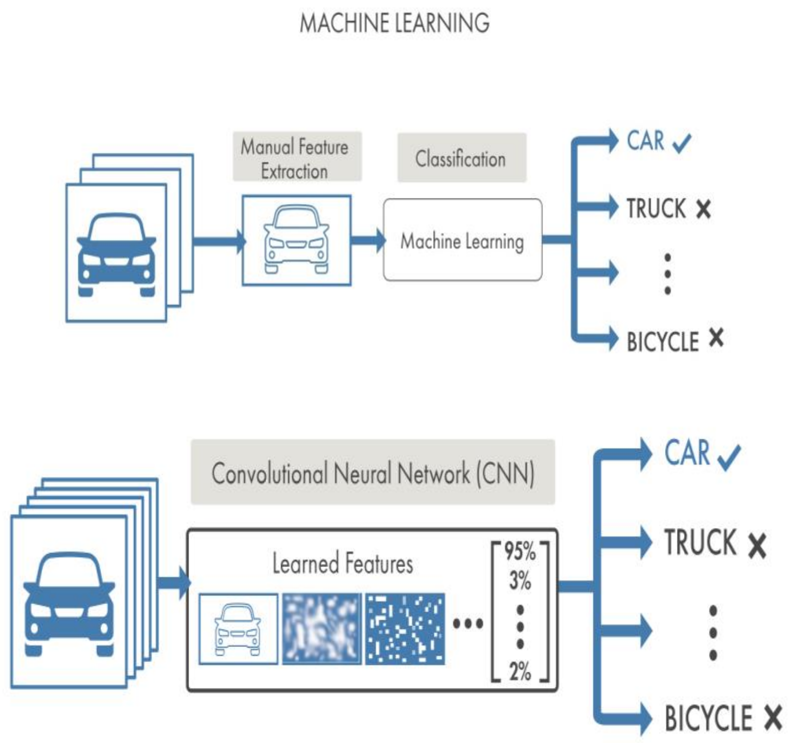


Input Image
Convolution
RELU (rectified linear units)
Pooling
FC
Fully Connected layers to support classification
Flower
Cup
Car
Tree
Sliding Window
filters (light and dark)
simple shapes
complex shapes
shapes that can be used to define a flower
X1
X2
X3
RELU (rectified linear units)
softmax
Categorical probability distribution
Probability
Every feature map output is the result of applying a filter to the image
The new feature map is the next input
ACTIVATIONS OF THE NETWORK AT A PARTICULAR LEVEL

What's the Difference Between Machine Learning and Deep Learning?

Deep learning is a specialised form of machine learning. A machine learning workflow starts with relevant features being manually extracted from images. The features are then used to create a model that categorises the objects in the image. With a deep learning workflow, relevant features are automatically extracted from images. In addition, deep learning performs “end-to-end learning” – where a network is given raw data and a task to perform, such as classification, and it learns how to do this automatically.

Another key difference is deep learning algorithms scale with data, whereas shallow learning converges. Shallow learning refers to machine learning methods that plateau at a certain level of performance when you add mor examples and training data to the network. key advantage of deep learning networks is that they often continue to improve as the size of your data increases



WHAT IS IMAGE COLORIZATION ?

Image colorization is the practice of adding colors to grayscale photos in order to improve its aesthetic appeal and visual impact. This might be used to colorizer old pictures and movies to make them more appealing, to analyzed CCTV footage to better understand the scene and draw conclusions, to colorizer MRI scans to make it simpler for doctors to diagnose issues, and possibly even to do so.

There have been various methods used to colorise images, but it used to be a very sophisticated task, requiring a lot of human intervention. Hence people have been trying different methods to ease this process.

Image colorization techniques can be broadly categorized into four categories: manual approach, scribble-based approach, example-based approach, and learning-based approach. A system that might colourised photos entirely automatically is currently being developed utilizing deep learning techniques.

In this paper, we summarise the different techniques that have been tested along with their comparison of results.

The organisation of this document is as follows. In Section 2 (Literature Survey), a detail study of existing methods are done. In Section 3 Overview of Literature Survey), we have discussed about the different methods and its limitations. Discussed in Section 4 (Conclusion) we have concluded.

Generally, the idea of coloring a grayscale image is a task that is simple for the human mind, we learn from an early age to fill in missing colors in coloring books, by remembering that grass is green, the sky is blue with white clouds or that an apple can be red or green.
In some domains, automatic colorization can be very useful even without semantic understanding of the image, the simple act of adding color can increase the amount of information that we gather from an image. For example, it is commonly used in medical imaging to improve visual quality

when viewed by human eye. Majority of equipment used for medical imaging captures grayscale images, and these images may contain parts that are difficult to interpret, due to inability of the eye of an average person to distinguish more than a few hues of gray [1].

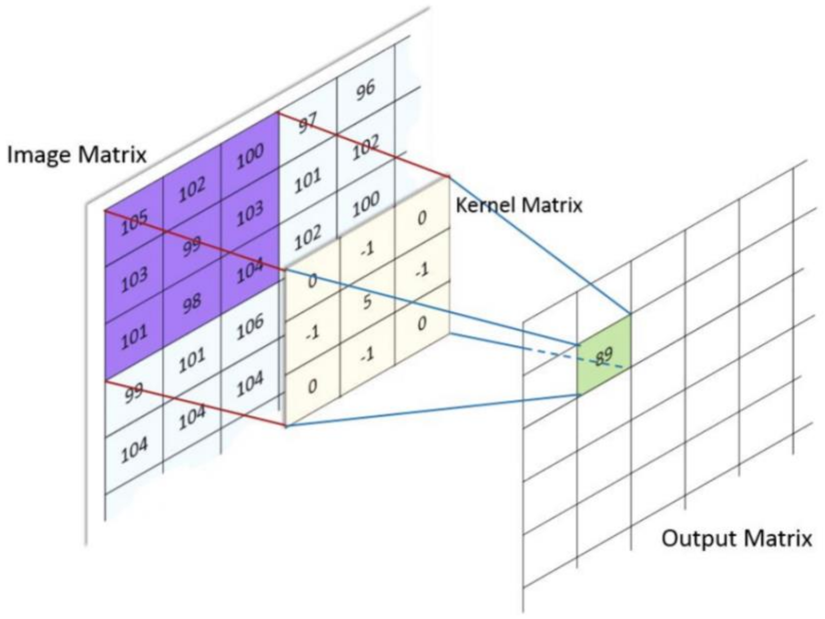
As a computer vision task, several user-assisted methods have been proposed for colo- rization, be it for natural or hand-drawn images, and expect supplied localized color hints, or provided reference images that are semantically similar to the target image that we can transfer color from, or even just keywords describing the image, to search the web for the reference images automatically.
However, the high-level understanding of scene composition and object relations re- quired for colorization of more complex images remains the reason developing new, fully automated solutions is problematic.
Recently, the approach to this task has shifted significantly from the human-assisted methods to fully automated solutions, implemented predominantly by convolutional neural networks. Research in this area seeks to make automated colorization cheaper and less time consuming, and by that, allowing its application on larger scale.
With the advent of deep convolutional neural networks, the task has been getting increased amounts of attention as a representative issue for complete visual understanding of artificial intelligence, similar to what many thought object recognition to be previously, since to truly convincingly colorize a target image, the method needs to be able to correctly solve a number of subtasks similar to segmentation, classification and localization.

Convolutional layer

Convolutional (conv) layers are the basic layers used in CNNs (hence the name). They represent the convolution operation performed on input with weights as parameters of the convolutional kernel, in the 2 dimensional case:

$$y_{i,j} = \sum_{m=i-\lfloor \frac{k}{2} \rfloor}^{i+\lfloor \frac{k}{2} \rfloor} \sum_{n=j-\lfloor \frac{k}{2} \rfloor}^{j+\lfloor \frac{k}{2} \rfloor} x_{m,n} W_{m-i+\lfloor \frac{k}{2} \rfloor, n-j+\lfloor \frac{k}{2} \rfloor} + b_{m-i+\lfloor \frac{k}{2} \rfloor, n-j+\lfloor \frac{k}{2} \rfloor}$$

where k is the size of the convolution kernel (also called filter), assuming it is odd (com- putation is ambiguous for even sizes and is therefore implementation dependent) and W and b simply run from the first index to the last. As we can see, the weights are stored per kernel and not per input, meaning that the layer does not need to have a fixed size input, acting as a sliding window. The weights are spatially identical for all parts of the input, which makes convolutional layers shift invariant.



The two dimensions are often called width and height, just like an image.

LITERATURE SURVEY

Colorization basically involves assigning realistic colors to grey-scale image. Convolutional neural networks are specifically designed to deal with image data. Many authors have done promising work on this idea.

Domonkos Varga proposed the idea of automatic coloring of Pictures images, since they are very different from natural images, they pose a difficulty as their colors depend on artist to artist. So, the data-set was specifically trained for Pictures images, about 100000 images, 70% of which were used in training and rest for validation. But unfortunately, the color uncertainty in Picturess is much higher than in natural images and evaluation is subjective and slow.

Shweta Salve proposed another similar approach, employ- ing the use of Google's image classifier, Inception ResNet V2. The system model is divided into 4 parts, Encoder, Feature extractor, Fusion layer and Decoder. The system is able to produce acceptable outputs, given enough resources, CPU, Memory, and large data-set. This is mainly proof of concept implementation.

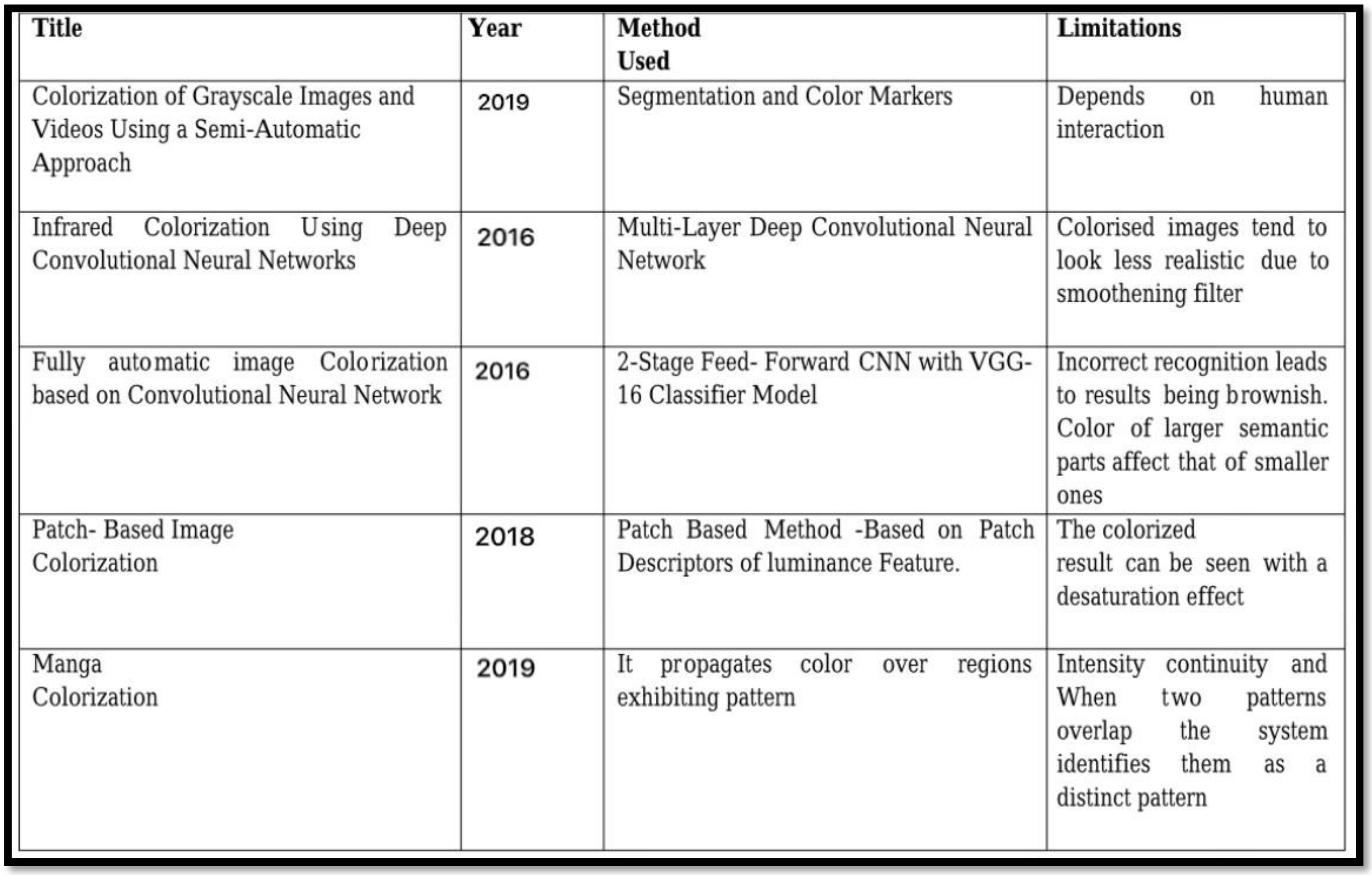
Yu Chen proposed a approach to mainly address the problem of coloring Chinese films from past time. They used existing data-set with their data-set of Chinese images, fine- tuning the overall model.
The network makes use of multi- scale convolution kernels, combining low and middle features extracted from VGG-16.

V.K. Putri proposed a method to convert plain sketches into colorful images. It uses sketch inversion model and color prediction in CIELab color space. This approach is able to handle hand-drawn sketches including various geometric transformations. The limitation found was that, data-set is very limited but it works well for uncontrolled conditions.

Richard Zhang has proposed a optimized solution by using huge data-set and single feed-forward pass in CNN. Their main focus lies on training part. They used human subjects to test the results and were able to fool 32% of them.

can have various number of neurons. The various attempts used various architectures . In some papers, generally number of neurons is same as the dimension of the feature descriptor extracted from each pixel coordinates in a gray-scale image.

EXPLANATION OF METHODS USED :



Title	Year	Method Used	Limitations
Colorization of Grayscale Images and Videos Using a Semi-Automatic Approach	2019	Segmentation and Color Markers	Depends on human interaction
Infrared Colorization Using Deep Convolutional Neural Networks	2016	Multi-Layer Deep Convolutional Neural Network	Colorised images tend to look less realistic due to smoothening filter
Fully automatic image Colorization based on Convolutional Neural Network	2016	2-Stage Feed- Forward CNN with VGG-16 Classifier Model	Incorrect recognition leads to results being brownish. Color of larger semantic parts affect that of smaller ones
Patch- Based Image Colorization	2018	Patch Based Method -Based on Patch Descriptors of luminance Feature.	The colorized result can be seen with a desaturation effect
Manga Colorization	2019	It propagates color over regions exhibiting pattern	Intensity continuity and When two patterns overlap the system identifies them as a distinct pattern

- **<u>Colorization of Grayscale Images and Videos using a Semi-Automatic Approach</u>**

This paper presents a semi-automatic approach for Colorization. They make use of segmentation, and color different areas of images should be colorized. The algorithm adds color to each pixel by considering the position of color markers. It first segments the image and then colorises it. They also attempt to colorise videos, by some frames, colorising them, then transferring the color to other frames. Keyframes are selected local minima of block motion. Segmentation is done using rain water simulation technique of watershed segmentation 1]. This method of segmentation leads to over-segmentation merging operations. of a unique segment, each with a marker after the process. Their results looked visually good for a large number of images.

- **<u>Infrared Colorization Using Deep Convolutional Neural Networks</u>**

The paper, deals with Colorization of Near-infrared (NIR) images of road scenes captured from cameras of cars. They make use of a multi-scale deep convolutional neural network. The approach consists of 3 parts, namely pre-processing, inference, and post-processing.
In pre-processing stage, they make up an image pyramid of the input image in multiple resolutions.
Then for inference, each element of the pyramid passes through several convolutional layers and max pooling layers.

After this, the result of each element passing through the deep CNN is merged using a fully connected layer.
A bilateral filter is applied onto the result of above, as post-processing, to reduce noise produced in the result.
The disadvantage of this method is that the resulting colorised images looks more like paintings that real world pictures.

- **Fully automatic image Colorization based on Convolutional Neural Network**

In this paper, a feed-forward, 2 stage architecture based on Convolutional Neural Network is used, to predict U and Color channels of an input Grayscale image. Colorization is looked at as a regression problem and is resolved using Anstey make use of the pre-trained VGG-16 classifier, which is already trained on a million images. The architecture consist of the VGG-16 model along with a 2 stage CNN that outputs the predicted U and V color channels of the image. They have used the YUV color model because it has the minimum correlation between the 3 coordinate axes. It produced very good results for some images. They evaluated the performance of the system using Quaternion Structural Similarity Index Measure (OSSIMI8) and obtained better values
of QSSIM for their results than many of the previous methods which they compared with.
Drawbacks of this method are that, if the system cannot clearly identify semantic information in the image, it tends to blur the output with a sepia or brownish tone. Also, sometimes the color information of the bigger semantic regions of the image get transferred to the smaller semantic regions

- **Patch-Based Image Colorization**

In this paper, a simple patch-based image Colorization based on an input image as a color example.This method which is based on patch descriptors of luminance features Anda color prediction model with a general distance selection strategy. A Total Variation (TV) regularization is also per-formed on the colorized image to ensure the spatial color coherency of the final result. Experiments show the potentiality of our proposition in order to automatically colorize Grayscale images.
The color prediction is performed from the luminance channel of both images. One drawback of automatic color-inaction is the spatial coherency during the
color transfer leading to possible inconsistent
Colorization in the final result. The image Colorization scheme is described based on patch features as pixels descriptors to capture image textures or complex structures.

- **<u>Manga Colorization</u>**

This Paper 7, Colorization technique that applies color over certain to the pattern continuity as well as that region. This black and white manga or Picturess which has drawing with the intensive number of strokes, halftoning and screening. intensity introduce many kinds of difficulties to Colorization methods which depend on intensity once the user marks on the drawing, a local, statistical-based pattern feature mechanism obtained with Gabor wavelet tarsi's applied to find the pattern-continuity. The boundary is then propagated by the level set method that monitors the level of. regions with same can be segmented by a single scribble. On these segmented regions, various Colorization techniques can be applied to replace Colors, colorize with stroke preservation, or even convert pattern shading
This method starts by drawing the desired interested regions. It automatically propagates the color within the pattern-continuous regions. The coloring stops change at the abrupt change, no fixed outline.

1. **Global features:** Most of the methods utilize the global features to form an image filter, and then use this filter to select similar images from a large image set automatically. However, some models produced unnatural colorization result due to global similarity but semantic difference.

2. **Data Set size:** Parametric and Non- Parametric models use different sizes of data sets for training the CNN. The Parametric model use very large data set to to train the CNN and produce more accurate result while the Non- Parametric models rely more on the input hints and reference image and use smaller data set for training the CNN.
3. **Semantic Information:** Semantic information has a significant and unavoidable role in deep image colorization. For effective colorization of images, the system must have information of the semantic composition of the image and its localization. For instance, leaves on a

tree may be colored some kind of green in spring, but they should be colored brown for a scene set in autumn. VGG-16 CNN model was used by majority of approaches to extract semantic information about the image before applying colorization techniques.

4. **Feature Extraction:** Features of the image are obtained through by integrating pre-trained neural networks to extract information about objects, shapes and use this context to assign color values to the objects. Some approaches used using Inception ResNet V2 classifier or Tensorflow to serve this purpose.

SYSTEM ARCHITECTURE AND DESIGN

In this chapter, we will describe the used CNN architectures, including individual layers and their hyper parameters. We choose to train two different architectures.

The first CNN architecture is a classical CNN with convolutional layers stacked on top of one another in a straight-forward way. The second architecture draws inspiration from the ResNet architecture, providing shortcut connections in between layers that are not directly on top of each other in the convolutional layer ”pipeline”.

Both networks are capable of processing images of arbitrary size, downsampling the output by a factor of 4, producing same output form for easy comparison, but work best on the image size they were trained on - 224 × 224.

In Chapter 8, we compare the results produced by both networks, and attempt to point out their strengths and weaknesses on two different data sets.

Pooling layers:

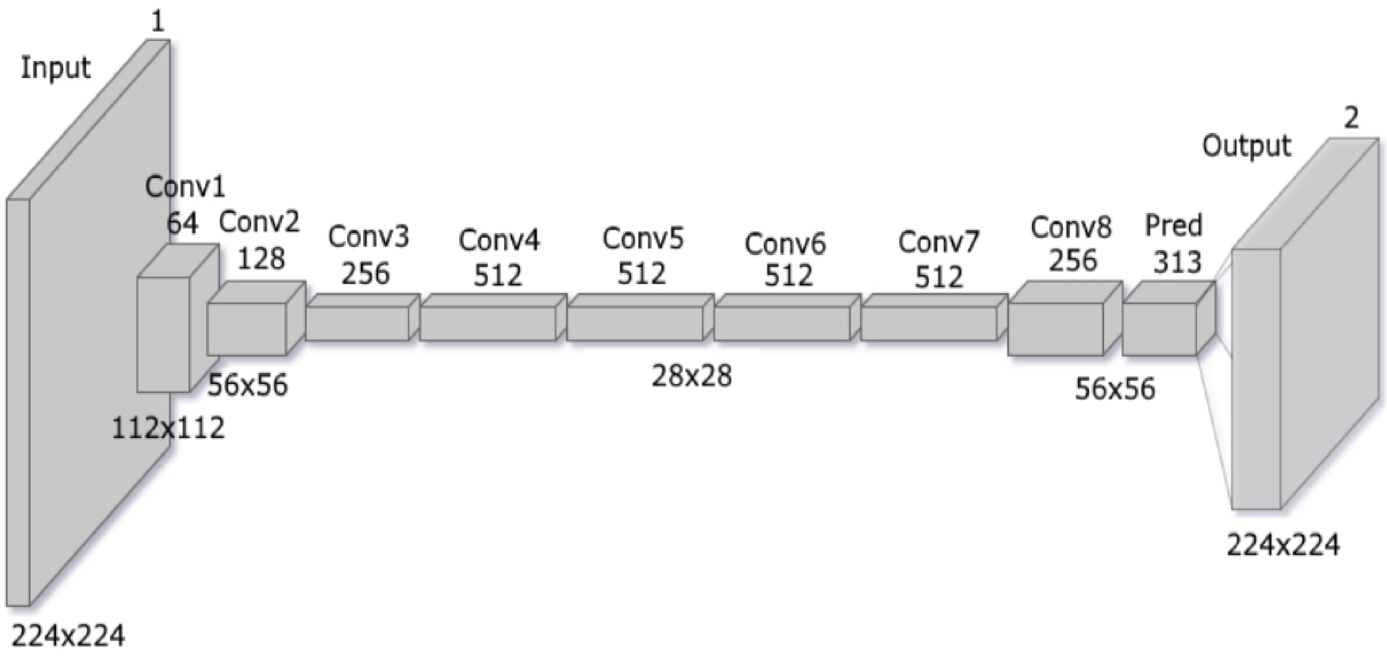
An important feature that both network architectures have in common is that they employ no pooling layers. Part of the reasoning for this decision was given in Section 2.2.3, pointing out that pooling layers are a form of downsampling. Even though some downsampling is required to effectively expand the network’s receptive field, we found pooling to have adverse effects on the results in both architectures, which we attribute to the fact that pooling layers drop too much information about the inputs.

Instead, downsampling is achieved through increasing the stride of some of the convo- lutional layers. This is a more natural fit for the task of colorization, as more information is preserved and can reach the later layers of the network.

Plain CNN model:

The first model that we train variants of is identical to the model used by Zhang et al. for natural image colorization on ImageNet. It is a model composed of stacked convolutional layers only (hence plain). A general overview of network architecture can be seen, and a detailed listing of layers and their hyper parameters.

Each block of conv layers refers to a groupping of 2 to 3 convolutional layers followed by rectified linear unit activation for each individual conv layer. Between blocks, a batch normalization layer is inserted to help prevent exploding or vanishing gradient problems and speed up convergence. All convolutional layers learn filters of size 3 ×3.



PLAIN CNN NETWORK ARCHITECTURE

In total, there are 22 convolutional layers split into 8 blocks, plus the prediction layer at the end. Initial layers contain a lower number of output channels, to simulate the use of a moderate number of low level features, usually very similar to filters that are used by many corner or edge detection algorithms. This can be more difficult to recognize and have a less noticeable effect when using small filter sizes, but encodes information equally well.

The initial layers function as feature extractors, while the later layers have the role of information encoding.

The resolution is quickly reduced to half of input size and quarter of input size after 2 and 4 layers, respectively. It is further reduced to one eighth of input size in the conv3 block and later upsampled back to one fourth by block conv8 via learned upsampling filters. One fourth of input size is the total prediction output resolution.

Using lower resolution for the densest layers helps prevent overfitting and requires the model to learn effective encoding of the input, allowing denser concentration of information in the encoding layers, which are in blocks conv4 through conv7, as well as rapid growth of effective receptive field, shown in the ERF column. This is where most of the network's parameters are located, over 27.13 million of the total 32.23 million parameters learned by the network.

Blocks conv5 and conv6 have dilated convolutions with dilation set to 1, further increasing the growth of receptive field. Every conv layer has input zero padding set such that its convolutions have valid ranges to operate over all of the previous layer's outputs, which translates to 2 pixels of padding in layers with dilation set to 1 and 1 pixel of padding in the remaining layers.

Block	**Layer id**	**R**	**O**	**S**	**D**	**ERF**	**P**
-	data	224	1	-	-	-	-
conv1	conv1_1	224	64	1	0	3	0.5
	conv1_2	112	64	2	0	5	36.8
conv2	conv2_1	112	128	1	0	9	73.7
	conv2_2	56	128	2	0	13	147.4
conv3	conv3_1	56	256	1	0	21	294.9
	conv3_2	56	256	1	0	29	589.8
	conv3_3	28	256	2	0	37	589.8
conv4	conv4_1	28	512	1	0	53	1179.6
	conv4_2	28	512	1	0	69	2359.2
	conv4_3	28	512	1	0	85	2359.2
conv5	conv5_1	28	512	1	1	117	2359.2
	conv5_2	28	512	1	1	149	2359.2
	conv5_3	28	512	1	1	181	2359.2
conv6	conv6_1	28	512	1	1	213	2359.2
	conv6_2	28	512	1	1	245	2359.2
	conv6_3	28	512	1	1	277	2359.2
conv7	conv7_1	28	256	1	0	293	2359.2
	conv7_2	28	256	1	0	309	2359.2
	conv7_3	28	256	1	0	325	2359.2
conv8	conv8_1	56	128	0.5	0	341	2097.1
	conv8_2	56	128	1	0	349	589.8
	conv8_3	56	128	1	0	357	589.8
pred	pred_313	56	313	1	0	357	80.1

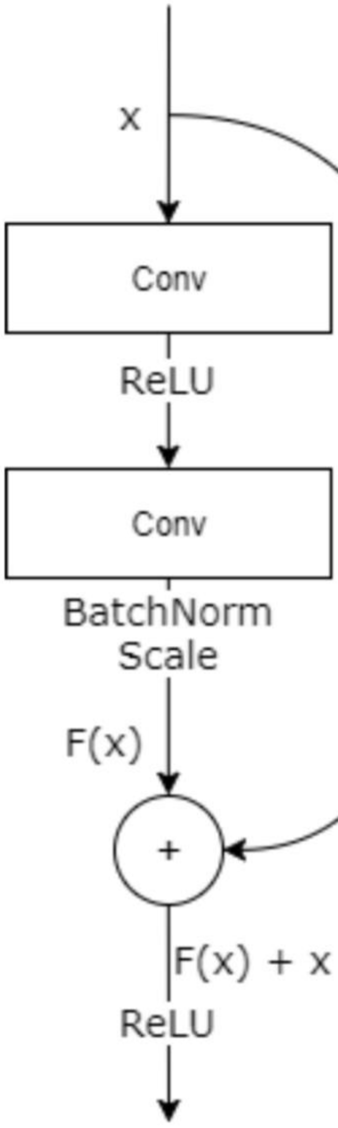
Plain CNN network architecture. R spatial resolution of output, O number of channels in output, S layer stride, D layer dilation, ERF effective receptive field with regards to data layer, P number of learned parameters in thousands rounded off

Residual CNN model

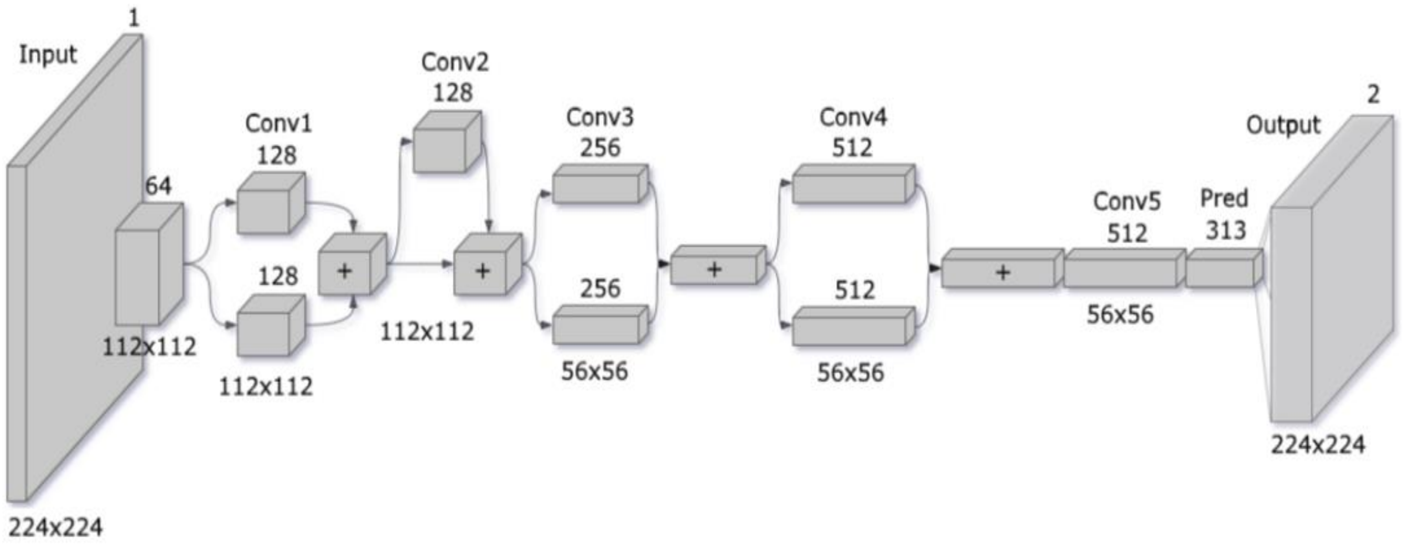
The second model that we introduce and train is one that is inspired by the success of the ResNet models in area of image classification. These models show much faster convergence rates as well as improved performance.

As more research was carried out in the field of training various deep CNNs archi- tectures, it became apparent that the problems vanishing or exploding gradients, which can mostly be resolved by applying normalization, were not the only factors hampering convergence and accuracy of networks.

In particular, a problem of accuracy degradation was uncovered during the training stage, accuracy gets saturated, as is expected when training converges. However, if training continues past the saturation point, accuracy starts degrading rapidly. A similar phenomenon is known as overfitting, but this is not the cause of degradation, since adding more layers to the model results in increased training error as well as reduced accuracy, the opposite of what would be expected in the case of overfitting. With increased model



Building block of residual learning, the number of conv layers between shortcut connections generally does not exceed 3-4 layers variance, overfitting would result in further reduced training error.



This indicates that deeper models are more difficult to train, even when vanishing/exploding

gradient problem is accounted for by normalization, with the likely culprit being the num- ber of stacked non-linearity functions. In plain convolutional networks, the expectation is that each few stacked layers (blocks in the Plain CNN architecture) learn the under- lying mapping produced by the following block of layers, denoted as H(x), which gets progressively more difficult to learn effectively as we add more layers.

Several techniques to alleviate this problem have been proposed, such as using better optimizing algorithms like Adam or improved initialization techniques.

ResNet models choose to solve this problem by using a different approach. Instead of expecting every block to learn the mapping H(x), the stacked layers are allowed to fit mapping $F(x) = H(x) - x$. In [6], the authors hypothesize that it is inherently easier to optimize this residual mapping

In the CNN, the modification of the mapping is realized with ”shortcut connections” between blocks of layers, After a block of layers has computed its outputs, the original inputs are added to it via element-wise summing.

Block	Branch	Layer id	R	O	S	D	ERF	P	BN&S
-	-	data	224	1	-	-	-	-	-
-	-	conv0	224	64	1	0	3	0.5	-
conv1	top	conv1_branch2a	112	128	1	0	5	73.7	-
	top	conv1_branch2b	112	256	2	0	9	294.9	Yes
	bottom	conv1_branch1	112	256	1	0	5	147.5	Yes
sum1	-	res1	112	256	-	-	13	0	-
conv2	top	conv2_branch2a	112	256	1	0	13	589.8	-
	top	conv2_branch2a	112	256	1	0	17	589.8	Yes
sum2	-	res2	112	256	-	-	21	0	-
conv3	top	conv3_branch2a	56	512	2	0	21	1179.6	-
	top	conv3_branch2b	56	512	1	1	37	2359.2	-
	top	conv3_branch2c	56	256	1	1	53	1179.6	Yes
	bottom	conv3_branch1	56	256	2	0	29	589.8	Yes
sum3	-	res3	56	256	-	-	69	0	-
conv4	top	conv4_branch2a	56	512	1	1	69	1179.6	-
	top	conv4_branch2b	56	512	1	1	85	2359.2	Yes
	bottom	conv4_branch1	56	512	1	0	77	1179.6	Yes
sum4	-	res4	56	512	1	0	93	0	-
conv5	-	conv5_1	56	256	1	0	93	1179.6	-
	-	conv5_2	56	512	1	1	109	2359.2	-
	-	conv5_3	56	512	1	0	117	1179.6	Yes
pred	-	pred_313	56	313	1	0	117	160.2	-

Residual network architecture. R spatial resolution of output, O number of channels in output, S layer stride, D layer dilation, ERF effective receptive field with regards to data layer, P number of learnt parameters in thousands rounded off, BN&S is followed by a sequence of batch normalization and scaling layers

The overall network architecture that we propose can be seen in Figure 6.3, and a detailed overview of layers is found.

In the original ResNet model, the shortcut connections are simply identity operations with no modification of the per-block input x. In our case, due to the necessity of down-sampling the input early on, the shortcut connections perform a single conv operation.

we look at several challenges that this task presents and how we choose to solve them. First, we discuss the various possible approaches to the problem in terms of overall design. Then, we take a more detailed look at individual components.

When considering the usage of CNN for this task, a natural question that may arise is whether it would be possible to use transfer learning as described in Other networks that deal with related problems exist, such as ResNet mentioned in designed for image classification. It is indeed true that many traits of the networks are shared, particularly the low level image feature extraction layers, and thus it may be possible to successfully fine-tune using at least the initial layers of a network trained for a task like image classification.

However, the first problem appears from the fact that all publicly available classification networks are trained on colored images and their performance on grayscale images is generally much lower. This is not a fundamental architectural problem, the network can still produce features from grayscale images, but the features are likely to turn out to be inefficient for the colorization task.

Another problem comes from the inner representation learned by the other networks. Since the color data is expected to be present in the input image, it is unlikely that networks trained for classification would be able to generate it based on grayscale image, if we were to train e.g. an extension on top of the hyper-column image representation of the VGG network architecture.

Larsson train a fully connected model while using modified VGG-16 convolution layers, fine-tuned on grayscale image classification, but only choose to predict a single pixel’s color at a time. During experiments, we attempted training a network on top of up to 9 initial layers of VGG-16 with frozen weights (disabled updates) that performed dense prediction, however, we conclude that training networks for colorization from scratch to be the best course of action, as it resulted in better outcomes.

Overall approach

In this task, our goal is to take a grayscale image, a single channel of image data, and transform it into a standard RGB image, an image with three channels of data.

We pose the problem as learning a mapping C such that R = C(G), where R ∈ RH ×W ×3 represents an image with RGB channels and G ∈ RH ×W represents an image with grayscale values only. To learn this mapping, we use CNN based models. To simplify training and reduce memory requirements to tractable amounts, we downsample the input images to 224×224 resolution and output a 56×56 resolution colorization, which is scaled up to match the grayscale input dimensions. Despite the radical decrease in resolution, it should be possible to learn a good looking colorization, as evident from the work of Zhang et al. [32], even if some finer detail is lost.

The input's total number of pixels is 50176 and output's total number is 3136. To upsample the colorized result, we use a conventional spline interpolation algorithm.

We can observe several properties of the formulation that influence how we approach the task.

Color space

First, we notice that as much information in the grayscale image ought to remain unchan- ged in the final, colored image, and that it should act as a passthrough channel. Therefore, color information would preferably be separate channels added to the grayscale image to obtain the result.

This effect is easily achievable by working in the CIE Lab color space. The Lab color space was deliberately designed to more closely match human perception of an image, compared to more standard RGB color space, and to contain a channel with a measure of luminance maximally decorrelated from chrominance.

An image in the Lab color space therefore consists of one channel for achromatic luminance (L) and two color channels (ab, sometimes noted as αβ). The a channel controls hues between green and red, while the b channel has control over hues between blue and yellow.channel-wise decomposition of the Lab color space.

By convention, the values range from 0 to 100 for the L channel, and from -128 to 127 for a and b channels, though not every combination of these values maps into the visible sRGB color gamut, especially the marginal values of a and b, and for that reason, we only consider values in the range of -110 to 110.

Still, in this range there are combinations which do not map back into the sRGB space naturally, in which case the conversion values are clamped into the conventional RBG space of ⟨ 0, 255 ⟩.

In this color space, our grayscale input image is seamlessly represented by the lumi- nance channel, leaving us with the task of estimating the chrominance channels a and b. HSL and HCL could alternatively be used, as they implement a similar concept of separating the lightness channel.

Color channels estimation

When estimating the a and b channels, there are several possible approaches to choose from. The choice can significantly affect the resulting visual colorization, training time or final CNN performance.

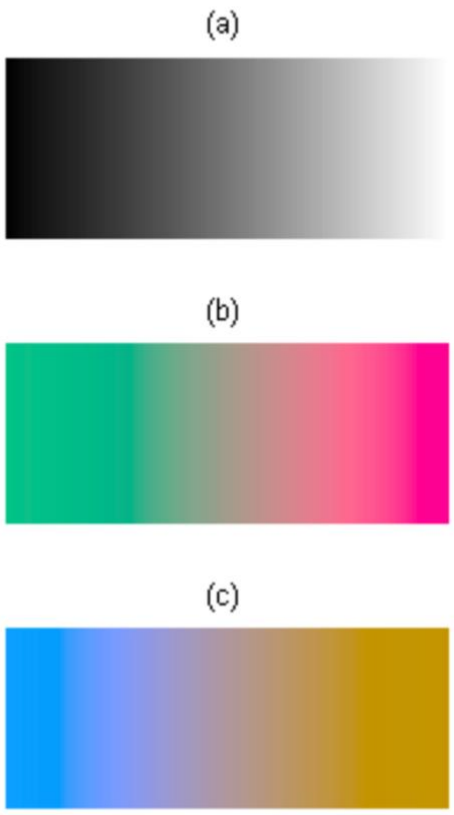
A straightforward option would be to estimate the values of a and b for each pixel directly. The advantage of this approach is the simplicity, both in terms of reasoning and implementation. It provides the option of using a simple loss functions, such as L2 Euclidean norm, which speeds up training.

Disadvantages are more subtle. One disadvantage is that this approach leaves no room for the notion of probability distribution over possible results, possibly resulting in jarring transitions between colors. For example, when a CNN predicts a pixel to have red color, there is no method of retrieving other possible colorizations or the level of confidence in the prediction.

Additionally, this approach does not handle multimodality well. If an object can take on a range of different ab values, the optimal solution to the Euclidean loss will tend to average out towards zero, producing desaturated colors. This approach is used by Iizuka, in conjunction with Euclidean norm in ab space as loss function.

A completely different approach is to pose the task as a classification problem, which is a well-studied area in the domain of CNNs, essentially turning the problem into estimating a color histogram for every pixel in the target image. We use the notion of color histogram in this context, but it is equivalent to confidence probability distribution predicted by thenetwork for every pixel, since the final layer uses a softmax activation function, which normalizes the outputs to form a proper distribution.

First, we define a set of canonical colors as classes to predict. Larsson et al.



propose sampling the ab space (or in their case, the hue/chroma space) based on evenly spaced Gaussian quantiles and selects 1024 values around the origin, but due to this sampling, this method may heavily favor more desaturated colors thanks to over-representation. Similarly to Zhang et al., we quantize the ab grid space into evenly spaced bins, taking a point every 10 units of the grid and remove any points that do not map to colors inside of the sRGB gamut. This leaves us with 313 proper ab pairs which can be used.

To convert a given value from ab to a color histogram, we find k-nearest-neighbor ($k = 5$) values in this quantized set and create a convex combination that closely mat- ches the original value (proportionate to the distances from the neighbors). Furthermore, this ”sharp” histogram is smoothed with a Gaussian kernel ($\sigma = 5$) to speed up CNN training convergence. This process is very similar to color quantization. This method works better than e.g. 1-hot encoding of the closest point on the grid, because it simulates the computation performed by CNN (it is harder to learn sharp peaks than smoothed distributions).

The CNN’s function is then to estimate color histogram for every pixel of the output image, essentially classifying each pixel’s color. In our case, that means estimating a matrix of the shape $56 \times 56 \times 313$. The cost function then calculates loss based on the this prediction and colors of the ground truth image encoded by the scheme described above.

Afterwards, there are multiple methods of converting the predicted color histogram back into an ab pair in order to obtain the result.

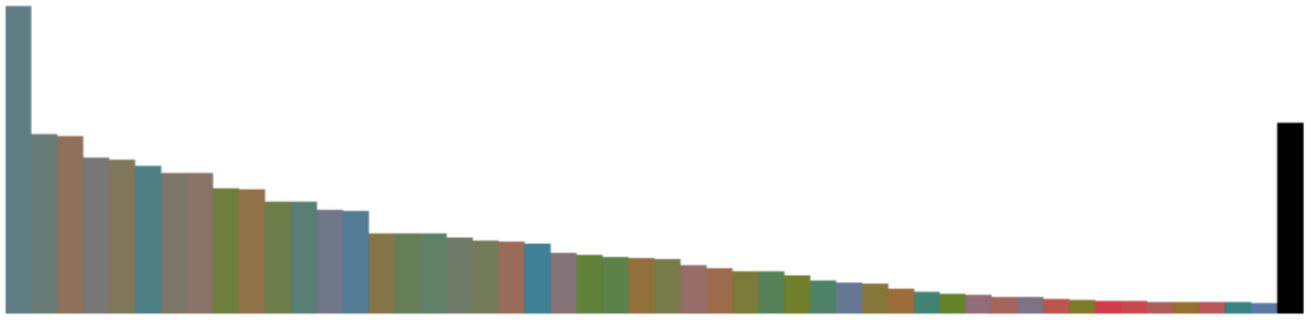
1. **Mode - treating the histogram as 1-hot encoding and assigning the pixel the color of the bin with highest prediction confidence.**
2. **Expectation - summing over the bins’ ab values, weighted by the histogram value**
3. **Sampling - drawing a random sample from the estimated probability distribution.**

Loss function

Regardless of the output color representation, we require a loss function to measure pre- diction errors and obtain meaningful gradients to update the network’s weights.

Since the color histograms are closely related (by construction) to probability distribution, we use the Kullback–Leibler divergence function as a natural distance function of two probability distributions.

Additionally, we enhance the loss function by using a prior-boosting re weighting factor, a technique called class rebalancing. The distribution of ab values in most images is strongly biased towards certain ab values, due to the presence of backgrounds such as blue sky, or green grass which take up significant portions of images. shows the measured distribution of pixels in ab space, gathered from all images in the training set. Notice that the number of pixels using more desaturated shades of blue and green is significantly higher than more vibrant colors.



$$v(P_{h,w}) = w_{q^\star},\, q^\star = \arg\max_q P_{h,w,q}$$

$$w \approx \frac{1}{(1-\lambda)r + \frac{\lambda}{313}}$$

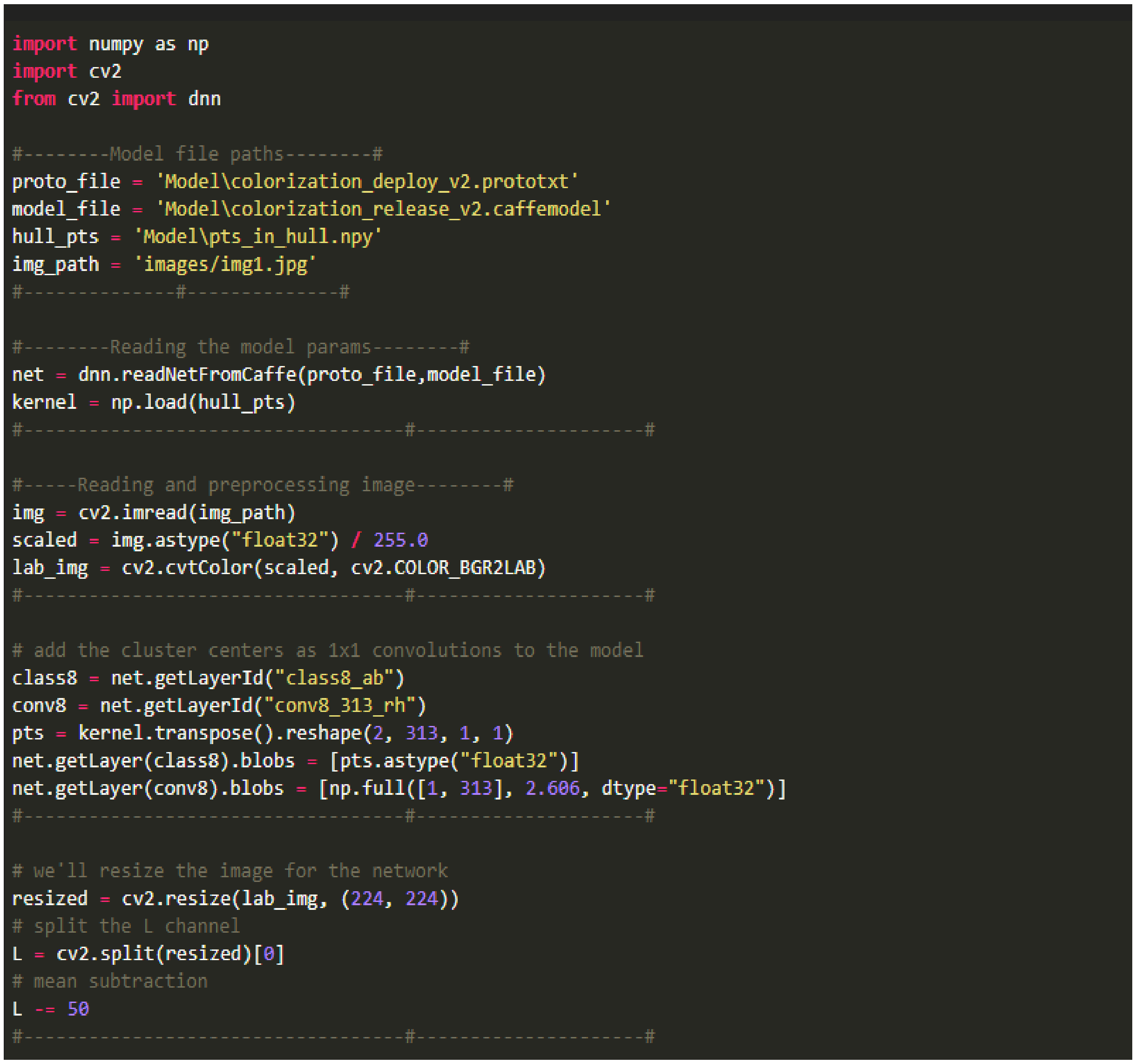
$$L(\hat{P}, P) = -\sum_{h,w} v(P_{h,w}) \sum_{q=1}^{313} (\log(\hat{P}_{h,w,q}) - \log(P_{h,w,q}))$$

CODING AND TESTING

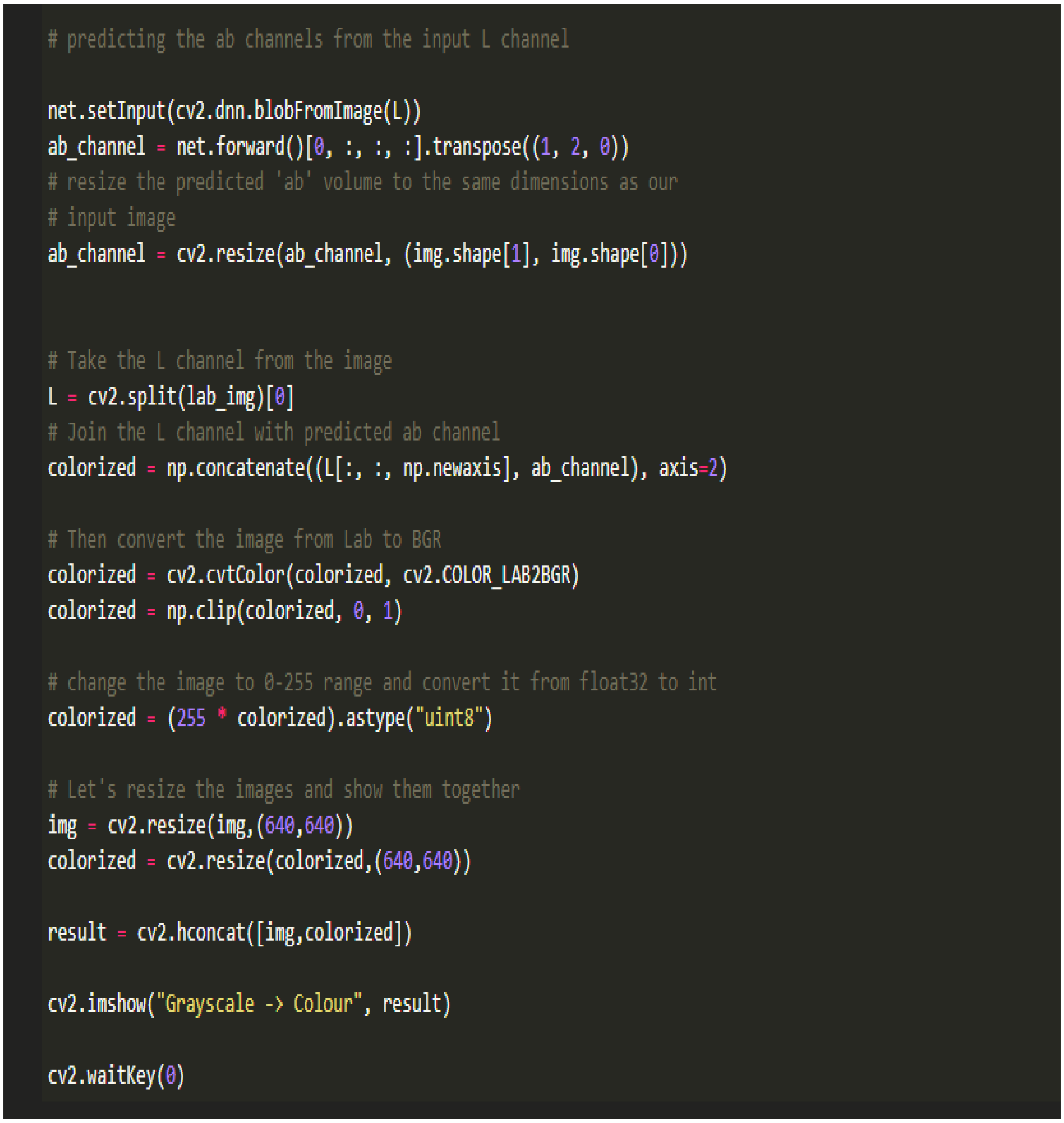
Steps:

1. Load the model and the **convolution/kernel points**
2. Read and preprocess the image
3. Generate model predictions using the **L channel** from our input image
4. Use the output -> **ab channel** to create a resulting image

Implementation:



```python
import numpy as np
import cv2
from cv2 import dnn

#--------Model file paths--------#
proto_file = 'Model\colorization_deploy_v2.prototxt'
model_file = 'Model\colorization_release_v2.caffemodel'
hull_pts = 'Model\pts_in_hull.npy'
img_path = 'images/img1.jpg'
#-------------#-------------#

#--------Reading the model params--------#
net = dnn.readNetFromCaffe(proto_file,model_file)
kernel = np.load(hull_pts)
#-----------------------------------#--------------------#

#-----Reading and preprocessing image--------#
img = cv2.imread(img_path)
scaled = img.astype("float32") / 255.0
lab_img = cv2.cvtColor(scaled, cv2.COLOR_BGR2LAB)
#-----------------------------------#--------------------#

# add the cluster centers as 1x1 convolutions to the model
class8 = net.getLayerId("class8_ab")
conv8 = net.getLayerId("conv8_313_rh")
pts = kernel.transpose().reshape(2, 313, 1, 1)
net.getLayer(class8).blobs = [pts.astype("float32")]
net.getLayer(conv8).blobs = [np.full([1, 313], 2.606, dtype="float32")]
#-----------------------------------#--------------------#

# we'll resize the image for the network
resized = cv2.resize(lab_img, (224, 224))
# split the L channel
L = cv2.split(resized)[0]
# mean subtraction
L -= 50
#-----------------------------------#--------------------#
```



```python
# predicting the ab channels from the input L channel

net.setInput(cv2.dnn.blobFromImage(L))
ab_channel = net.forward()[0, :, :, :].transpose((1, 2, 0))
# resize the predicted 'ab' volume to the same dimensions as our
# input image
ab_channel = cv2.resize(ab_channel, (img.shape[1], img.shape[0]))



# Take the L channel from the image
L = cv2.split(lab_img)[0]
# Join the L channel with predicted ab channel
colorized = np.concatenate((L[:, :, np.newaxis], ab_channel), axis=2)

# Then convert the image from Lab to BGR
colorized = cv2.cvtColor(colorized, cv2.COLOR_LAB2BGR)
colorized = np.clip(colorized, 0, 1)

# change the image to 0-255 range and convert it from float32 to int
colorized = (255 * colorized).astype("uint8")

# Let's resize the images and show them together
img = cv2.resize(img,(640,640))
colorized = cv2.resize(colorized,(640,640))

result = cv2.hconcat([img,colorized])

cv2.imshow("Grayscale -> Colour", result)

cv2.waitKey(0)
```

RESULT AND DISCUSSION

In this chapter, we first compare the trained variants quantifiably, with image error me- trics compared to the ground truth colorization. Then, we present colorized results of the different variants of the network architectures and discuss these results. We set our results side by side with automatic color transfer methods. Finally, we propose some possible further improvements of the obtained results with post processing algorithms.

To obtain the colorized images with the same resolution as input images, we use the spline interpolation method implemented in the SciPy Python library [41] in all variants. The input size used for the network prediction used are 224 × 224 (56 × 56 output). The L channel is passed through from the input image to the output image unchanged and this transfer is done with the highest resolution available (i.e. after upsampling of the color channels - 256 × 256) to preserve as much quality of the grayscale input as possible.

$$\mathrm{PA}(P_{ab}, \hat{P}_{ab}) = \frac{1}{I}\sum_{i=1}^{I} \frac{\sum_{n=1}^{N}(|P_{ab}^{i,n} - P_{ab}^{i,n,neigh}| < thresh) \cdot (|\hat{P}_{ab}^{i,n} - \hat{P}_{ab}^{i,n,neigh}| < thresh)}{\sum_{n=1}^{N}(|P_{ab}^{i,n} - P_{ab}^{i,n,neigh}| < thresh)}$$

where I is the number of images and N is the number of pixels in an image, neigh contains pixels in valid 4-neighborhood of a pixel and thresh is a user-defined value con- trolling the leniency of the metric. This metric calculates the ratio of pixels in the colorized image that have a consistently colored neighborhood (all neighboring pixels have ab values within thresh of the investigated pixel) compared to the number of consistently colored pixels in the ground truth image. A pixel is only counted towards the error value if the same pixel position has consistently colored neighborhood in the ground truth image, but does not in the colorized image.

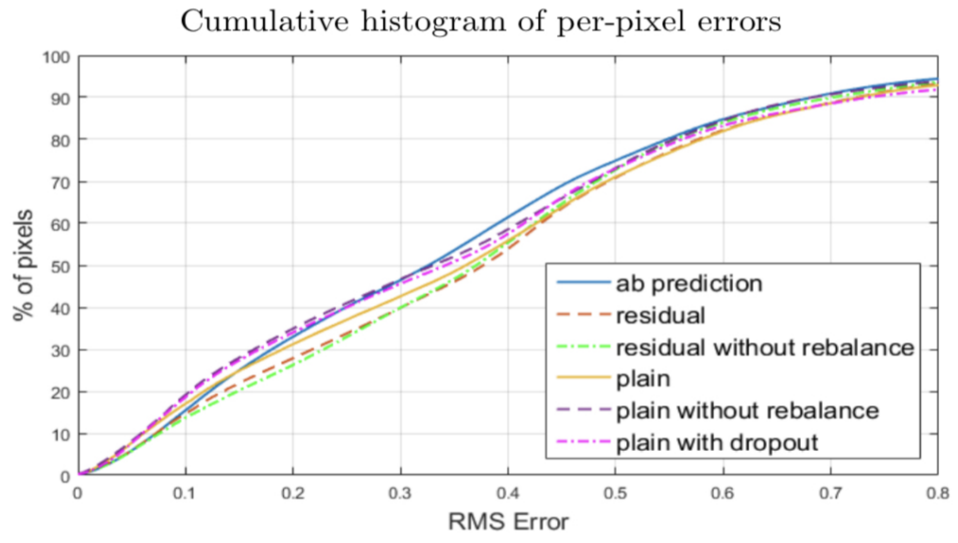
We include this metric because when comparing to ground truth images, the RMSE and PSNR metrics will penalize reasonable, but incorrect color choices for many objects (e.g. blue skirt instead of red skirt) more than obvious colorization artifacts. PA decouples the color values of ground truth and the result, instead focusing on color differences between neighboring pixels only.

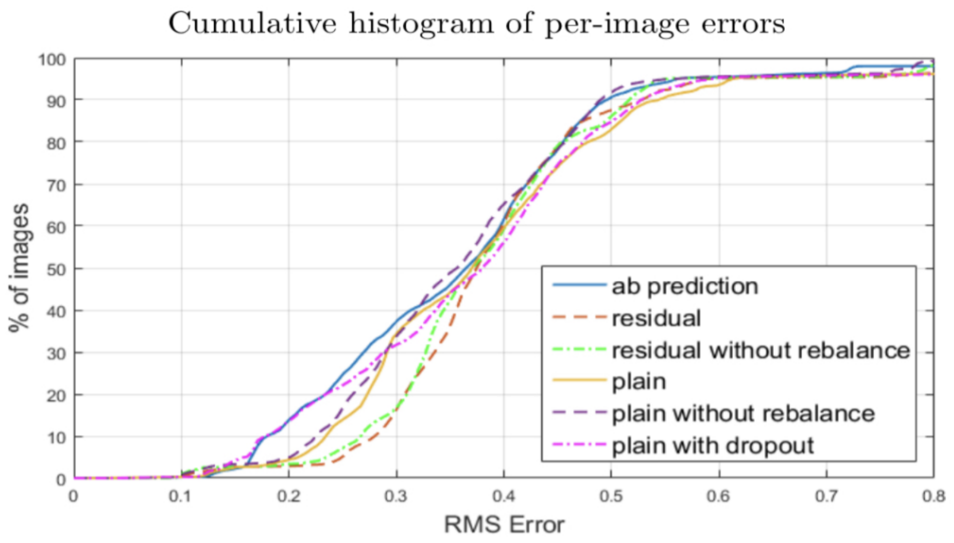
we provide cumulative histograms of per-pixel RMS errors for all model variants, as well as cumulative histograms of per-image RMS errors for all model variants. These figures better demonstrate the distribution of error magnitudes for each variant, showing which models achieve higher percentages of pixels (images) with low errors. Higher per-image errors generally mean either bad spacial artifacts or incorrectly (but perhaps plausibly) colored large sections of the image, such as uniformly colored backgrounds.

Interpreting these results is not straight-forward, perhaps with the exception of the PA metric. In the PA column of Table 8.1, we can clearly see that all classification variants produce significantly more artifacts compared to simple ab prediction. Rather than the output format, we attribute this to the loss function used during training, which, incidentally, is why the L2 ab CNN variant performs quite well in the other two metrics.

Visual comparison of variants

It is worth noting that while the above metrics provide a quantifiable result, they fail to capture the semantic intricacies of particular colorizations, and therefore, it is necessary to analyze the results quantitatively too. This is rather impossible to do rigorously, and conclusions can be considered subjective or influenced by personal tastes. When comparing the models, we try to provide this analysis using as many objectively observable features as possible, but it is inevitable that some perceptions will be affected by opinions. Some researchers choose to offload this problem by carrying out a two-alternative survey of real vs. fake test, showing each image for a brief moment to the participants, however, it would be highly impractical when comparing multiple variants.





For the purposes of comparison, we processed the testing episode described in Chapter 4 and an extra episode of the same Pictures, which has not been newly hand-recolored (specifically, we choose the 25th episode, called ”O vodnick ́em kolovr ́atku”), in hopes of observing generalization properties of each variant. We refer to this extra episode as Here we present several examples of colorized images to demonstrate the basic proper-ties of the colorization implemented by the network variants, as well as multiple examples which show commonly seen shortcomings of the colorization method. Full results, inclu- ding episode sequences turned into video files, are available on the attached CD described in Appendix A. More hand-picked results can be found in Appendix B.

In some cases, our models create colorizations that are semantically plausible and while it would be far-fetched to say that it is hard to distinguish the results from hand-made colorization, they are far from trivial, exhibiting signs of semantic understanding of the scene. These images usually feature characters or objects that are uniformly colored in the training dataset.

Not all variants perform consistently on the whole testing set, in fact, it is common for each variant to perform well in some areas, while being less successful in others, even though there are examples where none of the variants produce particularly good looking results. Observing these differences can be quite interesting, comparisons of all variants on selected examples.

Forms of failure

To properly asses strengths and weaknesses of each variant, we first describe commonly seen forms of failure. Some failures only cause some parts of the image to become im- plausible and others can, in some cases, cause the whole image to be perceived as badly colorized. While some objects may still be colored naturally, other parts of the image contain problematic areas.

The failures are most commonly observable on backgrounds, as they tend to have limited (often none at all) texture information, span large portions of the image and appear colored with a number of different colors in the training set, with limited cues for correct colorization (e.g. a background appears pink or blue for wall or sky respectively, with no indication of whether the current scene happens indoors).

Observed forms of failure appearing in most variant’s results, include the following cases.

- **inconsistent coloring** - objects contain spatially inconsistent colors, such as color shifts or artifacts
- **edge pollution** - inconsistent color around the edges of the image
- **color bleeding** - object’s color boundary spills over its intended edges
- **color deficiency** - colors appear too desaturated or too uniform across the entire image

Some of the variants also show specific failures, such as sensitivity to luma fading, producing extremely different colorizations when the same scene is fading in or out. Many images combine more than one of these failures, but usually, one is most noticeable.

Comparison to color transfer methods

To fully understand the performance of our models, we attempt to compare our models with current color transfer methods.

Comparing algorithm performance can be quite unfair however, since most color trans- fer methods were designed to work on natural images and do not specifically deal with the difficulties outlined in Section 4.2. Nevertheless, we offer comparison of our fully au- tomated method to two selected algorithms of Welsh et al. and Deshpande et al. which are also used for comparison by Zhang et al. and Larsson et al.and their source codes are available online.

for three image examples colored by the different methods. On all tested images, our method visually outperforms color transferring algorithms, but the plausibility of the colorizations created by the color transferring methods is highly dependent on the chosen references. Every method produces results with artifacts, but the color transferring methods create much less visually consistent colorizations.

Possible refinements

After obtaining the colorization results, given the unique properties of the dataset, there are improvements that can be made in order to make the colorization look more akin to ground truth's style. One option would be to produce scribble images by treating the colorization produced by the CNNs as the manual input required by other, traditionally human assisted algorithms, described in Section 3.1. This, in turn, would make the algo- rithms fully automatic, all the while preserving the various desirable properties that they have otherwise been designed to have. We briefly experimented with this idea, for exam- ple, by sampling single-pixel scribbles in areas of locally maximal classification confidence (spatial peaks in predicted color histograms), but found that no commonly used scribble algorithms work well on such input. Instead, we propose the following methods.

Segmentation with flood fill

The first refinement that we experiment with is one that attempts to make the colorization more uniform, given that the ground truth images are colored very uniformly (e.g. no blurring around edges or soft transitions of colors) and contain limited textures and large regions that should be colored evenly.

After observing that edges are very clearly defined, we can do this by automatically

segmenting the grayscale input image into disjoint areas using a seeded 4-neighborhood

region growing mean-shift segmentation algorithm with empirically selected threshold value (in particular it was 3 for the normalized L range of [0, 1], which worked consistently 51

well across the whole set, though an inferred threshold value could improve the result further, as in some images smaller or larger values provide more sensible segmentations). For thresholding to be more effective on images with varying luminance (mostly fading in/out of scenes), the L channel is normalized into a constant range on every image.

Applicability in video

So far, we have only described results concerning colorization on single images. However, since the original data comes from video sequences, it comes naturally that the ultimate goal should be to colorize a full image sequence plausibly.

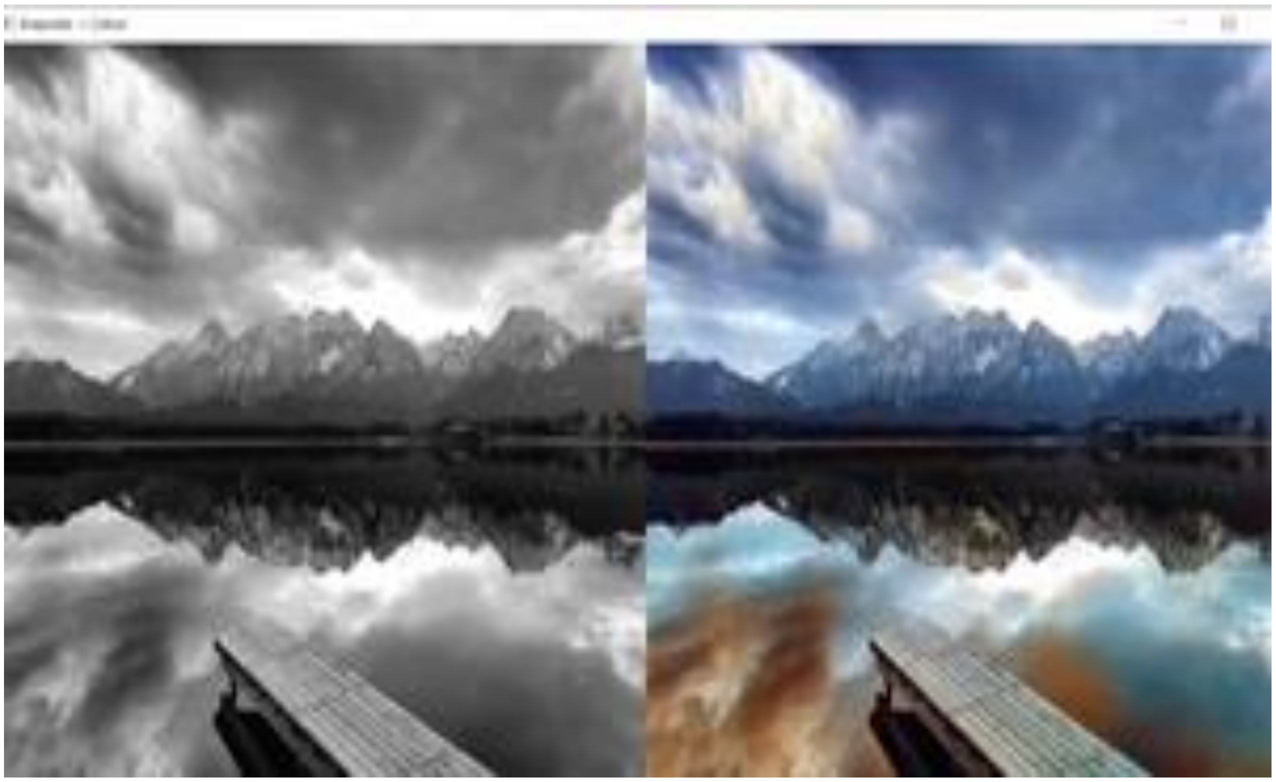
Unfortunately, many problems only manifest when the images are put into a sequence - such as the fact that most objects tend to change color with changing scale or object rotation (translation is affected to a lesser degree, due to translational invariance of con- volutional layers combined with using small filter sizes). This effect can also be observed with occlusion, although it is not as severe. These problems imply that the models have not been able to fully learn rotation or scale invariance.

Another problem that emerges is that one object's proximity can change the coloring of an unrelated object. This effect is shown in Figure 8.9, where two subsequent frames are shown. Notice that the only thing changing between the frames is the position of the doves, but this inadvertently affects the colorization of the clock face, making it inconsistent.

This is not limited to cases where occlusion happens, and it is most commonly seen on backgrounds.

These problems are mostly prevalent among all model variants, with no clear impro- vements or deteriorations between the variants. It leads to strange looking sequences, with color of objects changing as they get closer or further away from the camera or are rotated slightly.

OUTPUT EXPECTATION



CONCLUSION

Image Colorization had always been a tedious task, and much attempts have been done to automate this task. In this review paper, we have seen different methods of image Colorization that have been modelled and their results and drawbacks. We have seen approaches that are semi-automatics well as the current models of fully automatic image colorisation using convolutional neural networks and classifiers. We have also seen that the best results for automatic image Colorization so far, has been obtained by using Multistage convolutional neural networks along with classifiers.

We presented a method of fully automatic colorization of unique grayscale Pictures images combining state-of-the-art CNN techniques. Using the right loss function and color repre- sentation, we have shown that the method is capable of producing a plausible and vibrant colorization of certain parts of individual images even when applied to a moderately sized dataset that has properties which make it harder to colorize than natural images, but does not perform as well when applied to video sequences.

In doing so, we visually and quantifiably compared several variants of CNN design, which differed in loss functions, architectures and regularization methods. It is clear that the models we used have a hard time learning colorization of large uniform regions such as background sky or walls but fare better when smaller objects and characters are present.

We also proposed two methods of improving the generated results which greatly increase the visual resemblance of generated colorization to the ground truth images.
One novel contribution is using and comparing a model inspired by residual CNNs for the task of colorization and showing that despite the smaller ERF and fewer parameters, it can generate results that are comparable or even surpass plain convolutional neural networks in generalization to unseen data.

REFERENCES

- Victor Osma-Ruiz, Juan I. Godino-Llorente,Nicolas Saenz-Lechon, and Pedro Gomez-Vilda,An improved watershed algorithm based on efficient computation of shortest paths, Pattern Recogn., 40(3), 10781090(2007.

- R.C. Gonzalez, R.E. Woods, Digital Image Processing, Addison Wesley Publishing, Reading, MA, 1987.

- Vivek George Jacob, Sumana Gupta, "Colorisation of Gray scale images and videos using a semiautomatic approach" 2009 16th IEEE International Conference on Image Processing (ICIP).

- Matthias Limmer, Hendrik P. A. Lensch,"Infrared Colorisation Using Deep Convolutional Neural Networks", 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA).

- R.C. Gonzalez, R.E. Woods, Digital Image Processing, Addison Wesley Publishing, Reading, MA, 1987.

- T. Welsh, M. Ashikhmin, K. Mueller "Transferring color to greyscale images", ACM SIGGRAPH 2002 Conference Proceedings, pp. 277-280, (2002).

- T. Horiuchi, S. Hirano, "Colorization algorithm for grayscale image by propagating seed pixels", Proceedings of IEEE International Conference on Pattern Recognition, pp. 457-460, (2003).

- Liron Yatziv ,Guillermo Sapiro, "Fast image and videoColorisation using chrominance blending", IEEE Transactions on Image Processing, vol. 15, no. 5, pp. 1120-1129, May 2006.

- K. Simonyan, A. Zisserman, Very deep convolutional networks large scale image recognition., 2014.