Using a deep neural network for automatic colorization of greyscale images. Should describe what we exactly did...

J.L. Dorscheidt (4091981), Dawud Hage (4190696), Dennis van der Hoff (4139925), Joost Meulenbeld (4103548)

Abstract

blabla abstract

I. Introduction

In this paper several approaches to using a deep neural network to colorize grayscale images are compared. Specifically, given grayscale image, the neural network can generate all the data needed to present a color version of the same image, without input of the user.

There is a substantial amount of grayscale photographic material, from before the advent of the color camera. Color images may have a greater psychological impact on people than their grayscale counterparts. In addition, automation of the colorization process can be applied in real time to grayscale video. Specifically (infra-red) surveillance cameras often save video in grayscale format. With the automated colorization techniques it may be possible to generate real-time color video, such that a human may more quickly understand what is seen in a video, in addition to a decrease in file size. In order to do this, it is required that the algorithm can colorize an image without intervention of a human.

As will be explained below, non-machine learning techniques are available for automatic image colorization. However, the drawback of these techniques is that they require either an image comparable to the grayscale image in terms of content, or user input on what color to use for different parts of the image. While this makes things much easier than the hand-made solution with photo editing software, it still requires a human to specify these inputs, which in turn disallows a real-time solution. This makes the case for a trained machine (i.e. neural network), which uses pre-supplied training data to learn, and while actually colorizing an image does not require additional input from a human.

Identified by the types of input required for colorization, there are three approaches to make the computer convert a grayscale image to a color image. Every approach uses the texture and intensity gradients present in the image to link a part of an image to either a learned or specified color. The three approaches are as follows:

Colorize by example: Image colorization can be performed by using a color image that is related to the grayscale image, in the sense that it contains similar objects with similar colors. The patterns in the color image are compared to those in the grayscale image, directly linking colors to the grayscale patterns and finding equal patterns in the to be colorized grayscale image. For example: to colorize a grayscale image of a zebra in a Savannah, another image of a zebra in a Savannah is required. The more comparable the image, the better the result is. This method is used in [1], [2] and [3].

Colorize by user input: In this approach, the user specifies the color of different parts of the image by hand. While this is quite labor intensive, it guarantees that correct colors are used for the different segments in an image. As opposed to the *colorize by example* method, this method cannot colorize images incorrectly purely based on similarity between contrast patterns. However, if the user does not know the original colors of the grayscale image, colorization is not possible. This method is used in [4] and [5].

Colorize using a trained machine: Techniques like convolutional neural networks allow training of a machine to recognize specific patterns in an image and coupling the recognized pattern to a color. This requires supplying the machine with training images of which both grayscale and color versions are available. After the training of the machine is completed, no human is needed to colorize an image since all information needed is contained within the system. This leads to a vast decrease in time consumption during colorization. One of the biggest downsides of this method is that objects are colorized based on experience with different objects of the same class. However, not all objects can be colorized purely based on their class of object, i.e. an object of class “car” can have multiple colors, and during training multiple of these colors are shown to the machine, leading to an ambiguous mapping between object and color. The neural network method is used in [6], [7], [8] and [9].

The research presented in this paper dives into how to best colorize an image using a Convolutional Neural Network (CNN). The question as to what is actually the best colorization of an image does not have one final answer. One answer could be that the best colorization would be the one that most people would be unable to distinguish from the original color image. This immediately involves the opinion of a human, warranting for example the use of a version of the Turing test, where for both ground truth color image and colorized image are shown to a human. The human in turn needs to pick which it thinks is colorized and which is the original. Although this makes quantification of the quality possible, it is labor intensive and a direct (mathematical) link between the way the system colorizes an image and result quality is not apparent. This makes it impossible to use for any gradient-based method of training the neural network.

Another way is to mathematically define a quality measure, probably involving the ground truth color image and the colorized image, putting a number to the difference between these two. One of the most used error measures is the sum of the squared errors, calculating the difference in color per pixel, squaring every individual component and summing to obtain the final measure. This approach is natural to start with, but leads to one of the biggest unsolved problems encountered in automatic image colorization: the averaging problem.

The averaging problem arises from the following: a neural network is shown numerous examples of a certain object, of which it needs to deduce color from a grayscale image of this object. A trained neural network, in order to have the lowest possible cumulative error for all the training images, is encouraged to take some sort of mean color of all examples of the object in the training data set, which leads to the lowest mean squared error. For most objects that are linked to a certain color (such as bricks or a cloudless sky), this leads to undersaturated results, as illustrated in figure 1. For objects that are not inherently linked to a certain color, this leads to a mean that lies somewhere in the center of the color space, which in practice often leads to a generic sepia tone[9].

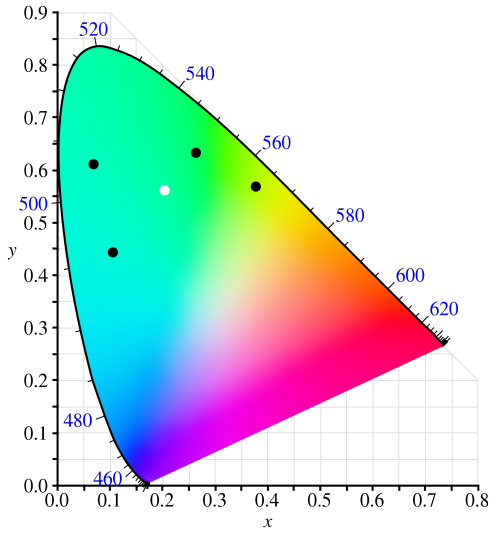


Fig. 1: The CIE 1931 color space containing all colors detectable by a human. When training a network to colorize to certain target colors of a green/blue object (black dots in image) minimizing the sum squared error, an average color will result (white dot in image), which most of the time leads to a less saturated color, i.e. more to the center of this color space. When the original colors are even more scattered, for example cars or clothes which do not inherently have a single color, a generic sepia color usually results.

In the rest of the paper the research is detailed on the problem of automatic image colorization using neural networks, starting with a literature review containing relevant other neural networks in section II. Multiple ideas are derived from these previous works. In section III these ideas in combination with new insights are described, detailing the method and different architectures to try. After that the results, a discussion of the results, a conclusion and recommendations are given.

II. Literature review

In recent years, Convolutional neural networks (CNN) have become the standard in image classification [8]. Using a CNN for image colorization is a trend that has emerged only in the past year in the machine learning community. Research from Dahl [9], Zhang et al. [10] and Cheng et al. [6] has shown promising results using CNN, however they all use very different architectures. One of the problems in the image colorization task is the maintainability of spacial information, going back from global features to a special mapping of these features in the detailed image space.

The use of max-pooling in CNN's makes the network invariant to spatial transformations. However a lot of this information is lost when the final layer of a conventional CNN is reached, which is disadvantageous for a classification and localization task. Different techniques have been proposed to regain a local mapping of the features in the final layers of the networkWelke dan?.

Another big variation lies in the loss function of the research mentioned previously. Dahl [9] and Zhang et al. [10] both use pre-trained networks for the feature mapping such as VGG16 [11], however Cheng et al. [6] completely trains the front-end module of the network from scratch. This offers more flexibility with network architecture, however a lot more computational power is required.

In this section a short summary of the related work on image colorization using CNN is shown. In addition the paper by [11] on VGG16 is explained.

A. Dahl

Dahl [9] is one of the first to use a CNN for image colorization. The network is trained in the YUV colorspace, with the advantage that Y-

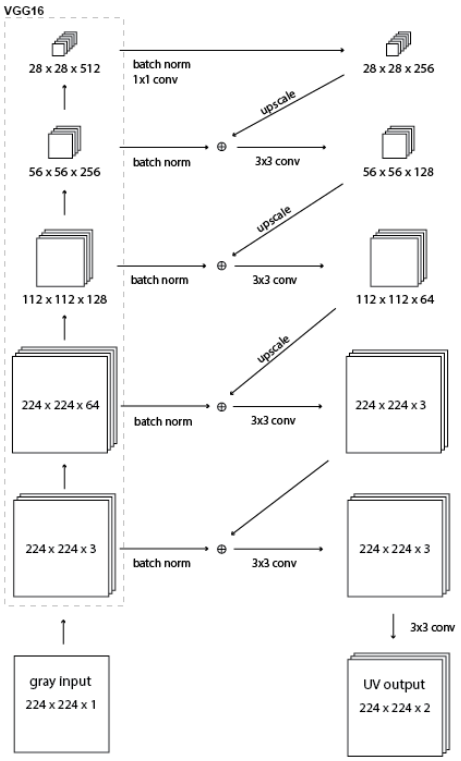


Fig. 2: Network used by Dahl [9]

channel can be used directly as the input of the network. Only the U and V channels will be computed by the network and they are concatenated with the original Y-channel, resulting in a colorized image. This way less information needs to be generated by the network. It is unclear whether the network uses exclusively information from semantic features as opposed to learning color distributions coupled to Y inputs. Zhang et al. [10] has shown in their network design that this is not the case.

The pre-trained VGG16 network is used, which already has a large variety of feature extraction. The network is trained on the ImageNet LSVRC 2012 Training Set.

Maintaining the spatial information is done with the use of partial hypercolumns, also called a residual auto-encoder [12]. The image is upscaled by a factor two after each convolutional layer in the output pipeline of the network. These upscaled feature maps are concatenated with the corresponding input layer's feature maps as can be seen in figure 2. This combination of global and more local features are then convolved using 3x3 convolutional kernels, this process is repeated until all the layers in the front-end are concatenated with the more global features.

Batch Normalization was applied after each convolutional layer block towards the concatenate layers (see section II-D). Throughout the network 3x3 convolutional kernels and rectified linear units (ReLu) [13] as non-linearities are used. The loss function consists of calculating the sum of the squared euclidean distances between the target pixel values and the output of the network. Gaussian blur is used over the target output of the image, resulting in better guidance of learning.

One of the problems encountered is that of color averaging, for images which have a large variety of color probabilities the network chooses the average of these colors. For example, cars can have a large variety of colors. The network colorizes these cars with the mean of all these colors, so in most cases cars will be colored sepia-like. It is proposed to use generative adversarial networks [14], to counter the color averaging problem. Another method to tackle this problem are variational Auto-encoders, altering a direct copy of the output, (variational) auto encoders are described by [15], [16] and [17].

B. Zhang et al.

Very promising results are obtained by Zang et al. [10], therefore a lot of the work in this paper is based on their work. A fully automatic approach is proposed that produces vibrant and realistic colorizations. Again VGG16 is used for feature extraction with some modifications on the input layers. The max-pooling operation is replaced by strides, resulting in less loss of spatial information.

In the output pipeline of the network, dilated convolutions [18] are applied, creating an exponential increasing depth of field with the same number of convolutions compared to conventional convolutional kernels (see figure 3).

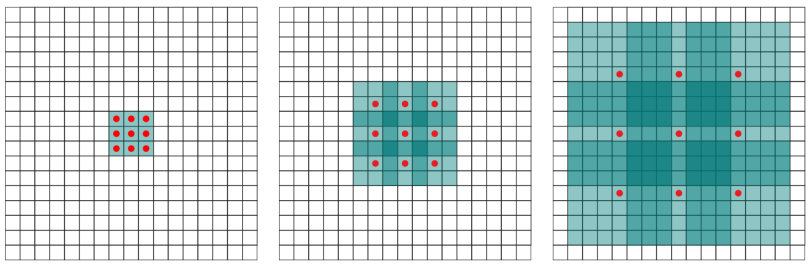


Fig. 3: Illustration of dilated convolutions. Figure c shows a receptive field of 15 after only 3 convolutions. Normal convolutions would have a receptive field of 7 after 3 convolutions [18].

In addition, the problem is posed as a classification problem, where each output pixel is classified as a certain color. This creates the opportunity of implementing class-rebalancing. Categorical cross-entropy is applied on the final softmax output layer of the network [19]. The final loss consists out of the categorical cross-entropy loss multiplied with a value function based on the probability of the target color given in equation 1, where H and W are the pixel width and height respectively, Z is the target vector, $\widetilde{Z}$ is the output vector, and v is the value function. Colors which are common in the dataset get a low loss, and colors which are rare in the dataset get a high loss, causing the network to 'steer' towards more saturated colors instead of sepia.

$$L(\widetilde{Z}, Z) = -\frac{1}{HW}\sum_{h,w} v(Z_{h,w}) \sum_{q} Z_{h,w,q} log(\widetilde{Z}_{h,w,q}) \tag{1}$$

The colorspace used is CIELab, The L (luminosity) layer is the network input. The CIELab colorspace is discretized into 313 colorbins. Target probabilities per pixel are generated by applying a Gaussian blur on the k-nearest neighbor colorbins. The output of the network consists out of the distribution of probabilities over the colorbins. In order to generate a final choice of colorbin from this output vector, an annealed mean operation is applied on the output probabilities. The temperature T in

the annealed mean operation determines whether the mode or the mean of the final distribution is taken as final color. An illustration of the effect of the annealed mean temperature T can be seen in 4 This operation is described in more detail in section III.

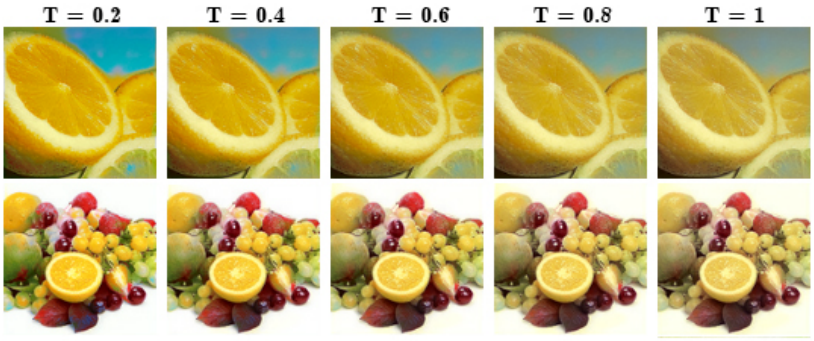


Fig. 4: Effect of applying the annealed mean operation on the color output probabilities of the network.

Various performance measures were applied by Zhang et al, the most interesting one being a 'Turing test' where participants needed to choose between a fake colorization of the image and the true ground colors of the images. The results of Zhang et. al fooled participants in 20.4 % of the cases.

C. Iizuka et al.

Recently the paper by Iizuka et al. [20] is published where they proposed a network that simultaneously classifies and colorizes the images. Their network shows very promising results. The classification is done to extract the global features of the image, i.e. if the image was taken indoors or outdoors. That is done by a CNN combined with a fully connected network. Those global features are then combined with the local features, computed by a CNN that has shared weights with the CNN used for classification, in a fusion layer.

The fused information is put through a colorization network which consist out of two upscale layers and four convolutional layers, resulting in two output feature maps containing the color channels of the image, that is combined with the input to produce the colorized image. Throughout the network 3x3 convolutional kernels are used, where the convolutional layers are placed in sets of two to increase the receptive field [11]. To maintain the spatial information strides are used on selected convolutional layers, just as is done by Zhang et al. [10].

Their novelty is the supervised classification fused with the extracted local features. The thought behind it is that the network will use the classified information to preselect the color palette to colorize the images, i.e. the network will not use green and blue when coloring an image that is taken inside. Interestingly to note is that this is the only colorization network described here that does not uses a pre-trained network, but is trained from random initialized weights where batch normalization layers are used to accelerate the learning. The update method used is ADADELTA [21], such that the learning rate is adapted automatically. The network is trained in different colorspaces, namely RGB, YUV and CIEL*a*b* where they conclude that the colorspace with the most perceptually reasonable results is the globally normalized CIELab colorspace (CIEL*a*b*).

No solution is proposed for the averaging problem, therefore this network is still struggling with those objects giving them sepia like colors.

D. Batch normalization

The networks described here all use batch normalization layers, this type of layer is first described in [22]. During training of deep networks, the combination of weights and biases can make the nonlinearities act in the saturated regime causing problems with vanishing gradient. To overcome this, the weights need to be initialized carefully and small learning rates are used resulting in slow convergence. Normalizing the output of intermediate layers over each batch, results in lower sensitivities of the weights and biases of consecutive layers with regard to the previous layer. This results in faster training due to increased learning rates, the weights initialization can be done less carefully and the need for regularization (i.e. dropout) is reduced [22].

III. Method

This section covers the different methods used to get an answer to the question of which CNN setup is best in colorization of images. Firstly, the chosen dataset will be explained. In addition the input size and the different colorspaces used will be explained. Next the different architectures used will be explained. Features extraction, training parameters, loss functions, and general architectures of the chosen networks will be evaluated. Finally the test setup will be explained.

A. Dataset and Input

Large training and validation datasets are required for training and validation of the CNNs. Some restrictions on the datasets had to be made due to limitations in computational power, resulting in a selection of images based on a certain category; fruit and landscape images.

The landscape dataset is used as a low-level test case, images of landscapes offer less distinct features coupled to a mapping of the colors, e.g. most landscapes are green at the bottom and blue at the top. The high-level test case consists of the fruit dataset. Fruit is a category which has distinct colors corresponding to distinct features. Therefore, in order to color fruit correctly, the network has to learn to link colors to distinct features in the input, straining the networks requirements. Also many fruit classes are linked to ambiguous colors, i.e. green or red apple. Taking fruit as the dataset is therefore an excellent method to check whether the network is able to counter the 'averaging' problem also mentioned before.

The datasets are generated using the popular image website Flickr[1]. Using their freely available API, a program was made that retrieved images in the required resolution and kept track of images retrieved, to avoid duplicates. The datasets retrieved had to be checked on incorrect images. To solve this problem a web application was made that enabled us to check images on defects. The checked datasets are collected in batches and stored in Python Numpy arrays as input for the network. This resulted in 2 datasets, which are summarized in table I.

TABLE I: Datasets used for training and validation of the various networks

Dataset	# Training Images	# Validation images
Fruit	6000	1000
Landscape	34000	5000

The network input are 128x128 pixel grayscale images, which are propagated through the network in batches. Using batches reduces computation time since better use is made of the parallelization of modern computer architectures. Most importantly the gradients calculated using batches are a better estimate of the gradient for the entire training set, thus leading to a more stable gradient descent [22]. Making the batch size too large makes the search to the global minimum less stochastic, thus making it converge to a local optimum more quickly, and in general converging less quickly.

The network is trained to be able to colorize an image, but the way it outputs the colorized image is dependent on the color space used during training. There are several options available:

RGB This widely used color space specifies an intensity for the channels red, blue and green. The biggest drawback of using RGB is that the color is not separated from the luminosity. In this way, the network needs not only to output a hue and saturation of a color, but also the luminosity itself. This makes it a much tougher challenge to output an image that resembles a colorized version of the grayscale image. A visualization of the RGB space is given in figure 5a.

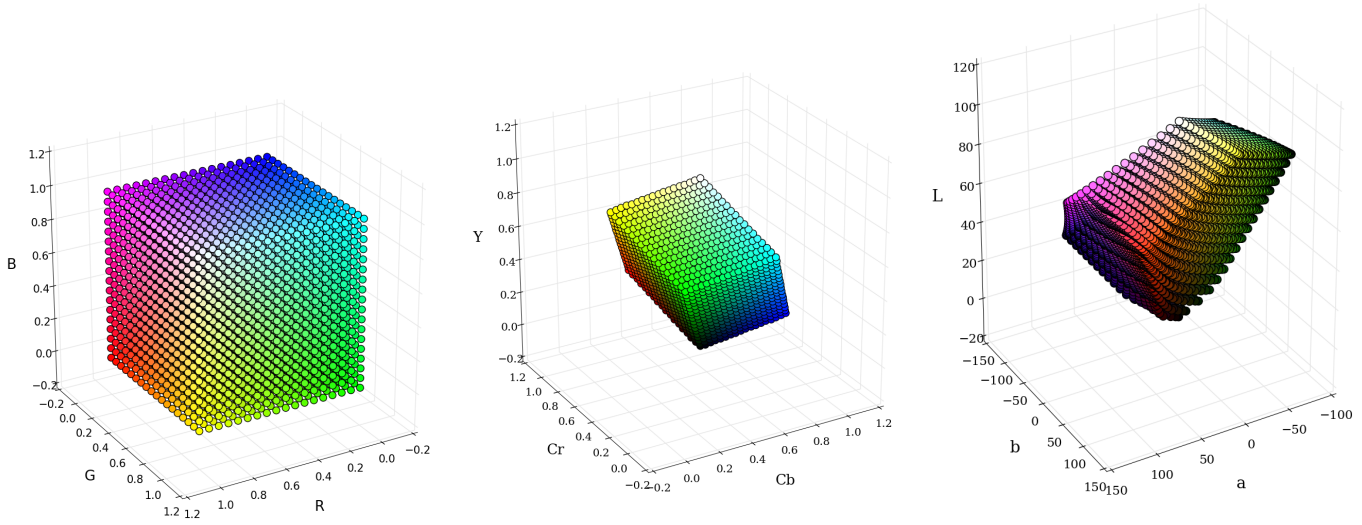
HSV Specifying the hue, saturation and value, uncouples the luminosity (value) from the color (hue and saturation). Furthermore decoupling the saturation could allow specifically tackling the sepia and saturation problem as described in section I. However, the color space is periodic in the hue axis. From the numerical perspective, this leads to an ambiguous error specification, since for one difference between target and network output color, two directions of improvement are equally valid, thus a very complicated loss function is required, where the network needs to learn this circular property of the hue axis. This colorspace is used for training on the compact network architecture, however the network did not perform as good as when it was trained on either YCbCr or CIELab. Therefore the HSV colorspace was omitted.

YCbCr The Y channel contains the luminosity of the image, while Cb and Cr layers are the chroma blue and chroma red layers respectively. While providing a separate luminosity channel, it was found that for different Y values, a given Cb and Cr combination does not specify the same color. This makes the error in the color specification dependent on the luminosity of the image. A visualization of the color space can be found in 5b.

CIELab Similarly to the YCbCr color space, the L channel specifies the luminosity, the a and b layers the color. In contrast with the YCbCr space, a given a and b value specify a color, while the luminosity only determines how light the color is. However, for a single L layer not all a and b value translate to colors that can be represented in the RGB color space (as displayed by computer monitors). The CIELab color space is visualized in figure 5c.

To summarize, the HSV, YCbCr and CIELab color spaces all have the possibility of using one channel as input to the neural network, while requiring only two outputs of the network that instead of three with the RGB color space. Combined with the input, these two outputs allow reconstructing a color image. According to Iizuka, the CIELab color space has been found to give the most correct results for a colorization network[20]. Considering the disadvantages of the HSV color space, the YCbCr and CIELab color spaces are compared during the present research.

[1] https://www.flickr.com/



(a) The RGB colorspace

(b) The RGB colorspace represented in the YCbCr colorspace

(c) The RGB colorspace represented in the CIELab colorspace

Fig. 5: The different colorspaces used in the different networks

B. Convolutional neural network properties

Convolutional neural networks contain a vast amount of properties that determine the behaviour of the network. Structure of the loss function and the model architecture being the most important properties. To compare different architectures, it is important to keep the hyperparameters the same as much as possible, resulting in a fair comparison between CNNs. The choices made in the experiment setup which are affiliated with the CNN properties will be explained here.

A few parameters are held constant over all the test cases. The parts of the network (or the complete network) that are initialized without pre-trained weights are randomly initialized. This is done using a Glorot Uniform distribution [23]. This weight initialization method samples from a uniform distribution with its variance scaled depending on the ingoing and outgoing data.

The intermediate layers of all the network consist of convolutional layers. All convolutional layers use ReLu nonlinearities [13], except for the final output layers, they are different per architecture. Kernel sizes in the convolutional layers of the network are of size 3x3. This kernel size is based upon VGG16's [11] kernel size, as is used in the other reference networks [9][10] [22]. Stride in the convolutional layers is 1x1 by default. However, in the CNNs using dilation in the output pipeline, stride is used as a way to downsample the resolution of the image, comparable to the function of max pooling, but retaining more spatial properties of the input image [18]. Padding is required when using convolutional layers due to the fact that border information of the image is lost when convolving. This property is set to keep the output resolution the same as the input resolution. Zero valued padding is used when conventional convolutions operations are performed. Symmetric valued padding is used when dilated convolutions are used, again the amount of padding is chosen to keep the output dimensions the same.

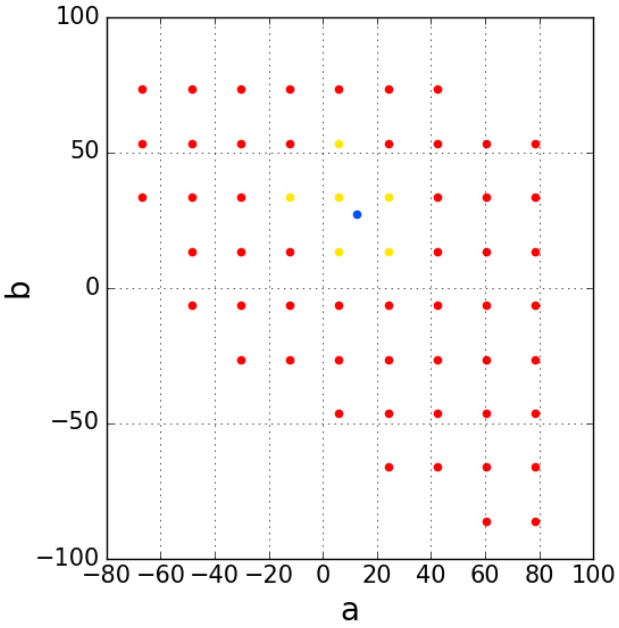


Fig. 6: K nearest neighbors

1) Feature Extraction: To recognize certain objects in grayscale images, object dependent features are extracted. This is done in the first part of the convolutional neural network, also known as the input pipeline. To extract features convolutional layers are used. Each convolutional layer convolves the input of the layer with a kernel to a set of feature maps. Max-pooling

is often used to make feature extraction spatially invariant to input transformations [11]. The kernels in the incoming pipeline of the CNN can be chosen to be initialed from scratch, or to be initialed with pre-trained weights. For example VGG16 [11] has shown excellent results in image classification, and therefore already has a large amount of feature extraction available. In this paper a comparison is made between using architectures which are completely trained from scratch, and architectures which use the input pipeline of the VGG16 network to initialize the convolutional kernels.

2) reconstruction: In the input pipeline of the network, the image is reduced to a set of features, containing decreased amounts of spatial information. To be able to colorize the image, localization of these features is required. For reconstruction, various methods are used to retrieve the original image resolution.

First of all, the feature maps can be upscaled using linear upscaling. However, to retrieve the original image not only the features but also the localization of the features needs to be done. There are multiple ways of retrieving the spatial information of the image [1] [10]. The methods tested in these paper consist of residual auto-encoders and dilated convolutions [18].

The use of a residual auto-encoder has been demonstrated by [9]. After the bottleneck of the CNN each layer is upscaled and concatenated with the parallel layer in the input pipeline of the network. This way global features are merged with the localization of those features. This process is repeated after each upscale layer until the output of the network is reached, as can be seen in figure 2.

Another technique used is the use of dilated convolutions [18]. Zhang et al. use strides in the incoming pipeline of the CNN, which causes less loss of spatial information. The bottleneck of the network consists of 28x28 feature maps. The output pipeline of the network is initialized by two layers of dilated convolutions, where each layer consists of 3 consecutive dilated convolutions. Using dilated convolutions has the effect that the input pipeline can compress the image to a larger size leaving more spatial information in the bottleneck of the image. The receptive field of dilate

3) Color Generation: In the final layers of the network, the original image color layers have to be reconstructed. Two different methods are used throughout the paper: construction using two feature maps and classification.

The construction using two feature maps is a direct result of retrieving the original image resolution through the reconstruction of the image. As a final layer, a convolutional layer is used that maps to two feature maps, which represent the two color layers that are finally used to create a colorized image.

As an aid to the colorization process, a Gaussian Blur is used on the color layers of the original image. Colorization does not require precise pixel by pixel colorization, because colors in images mostly appear in sets of pixels. When blurring, this enables the network to more easily converge towards a solution by reducing the noise in the color of the pixel set. Note that blurring the color layers does not reduce the image fidelity. This is due to the fact that the luminosity layer contains the details and contrast of the image, which is subsequently merged with the output color layers of the network. This makes the colorization problem into a regression problem, where a continuous function is sought which maps a grayscale input to the two values that determine color in the output.

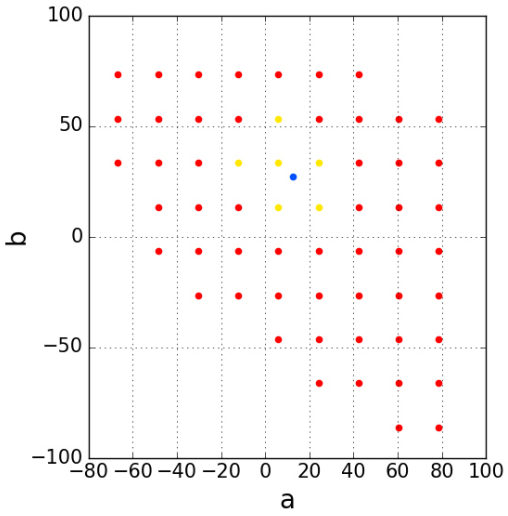


Fig. 7: An illustration of the K-nearest neighbour algorithm. The red dots represent the centers of the colorbins in the discretized colorspace. The blue dot represents the ground truth ab value of the target pixel. The yellow dots correspond the the K-nearest neighbour colorbins. The class probabilities of the targetvector are generated by applying a gaussian blur on the distances to the K-nearest neigbour colorbins.

The other method used is a classification approach to colorization: for a given input image, every pixel in the output image falls within one of a few predefined color classes. For the classification networks, the colorspace CIElab is used. The ab colorspace is discretized into n classes, each corresponding to a discretized bin in the original ab colorspace. Classification applies a loss function per pixel in the image, so each pixel has a corresponding target vector. Instead of generating one-hot vectors per pixel, probability distributions are generated, assigning probabilities to multiple colorbins. This way the network can make use of the similarity of different colors in the ab colorspace. Using a target vector which has a distribution over the classes instead of a one-hot vector, helps to steer the network to the correct colors. In addition the use of a color distribution more closely resembles the nature of the colorization problem, since not every object is linked to a single color, but rather a distribution of possible colors.

Each target vector is generated by applying a Gaussian blur over the k-nearest neighbor colorbins of the exact ab value, as can be seen in figure 7. The result of these operations over one image is a set of $128 \cdot 128 = 16384$ target vectors, each consisting of a probability distribution over the n colorbins. The final colorized version of the image is created by processing

the probability distribution per pixel. Implementing an annealed mean to pick the best color from the probability distribution allows setting the degree with which to pick either the mode or the mean of the distribution [10]. The annealed mean operation can be seen in figure 2, where **z** is the output probability vector of one pixel, T is the annealed mean temperature, and q is the total number of colorbins. Both the two feature maps and classification approach are tried during the present research.

$$f_T(\mathbf{z}) = \frac{exp(\mathbf{z}/T)}{\sum_q exp(\mathbf{z}_q/T)} \tag{2}$$

4) Loss Function: During the present research, several loss functions were used, depending on the color generation method used. The two feature maps represent the color space layers of the to be reconstructed image, depending on the selected colorspace. The loss function is then defined as the sum of the squared distances between the output of the network and the original image color layers. This can be seen in equation 3, where p is the prediction of the network and t is the target. The distances are calculated per pixel and per color channel.

$$L = \sum_{pixels} \sum_{channels} (p - t)^2 \tag{3}$$

To enable the network to use more saturated colors, color rebalancing may be applied to the loss function. This color rebalancing penalizes the the network for selecting desaturated colors, by exponentially increasing the loss function when converging towards desaturated solutions. However, it was found that this does not produce desired results, probably due to the fact that it makes no use of the actual probability of the colors, thus leading to a crude approximation of the histogram. For the two feature maps output, the standard sum squared error as in equation 3 is used.

For classification, categorical cross-entropy is used. This is done between the predicted color classification of the network per pixel and the target classification per pixel of color layers of the image. The categorical cross-entropy function is defined in equation 4, where $t_{i,j}$ is the target classification, where i specifies the pixel, and j its respective probability distribution of the classified color. $p_{i,j}$ is the predicted classification, where i depicts the predicted pixel. j its estimated probability distribution of the classified color, by the network.

$$L_i = -\sum t_{i,j} \log(p_{i,j}) \tag{4}$$

Due to the fact that in the training set the lower a and b values are highly represented, after training a bias will result towards picking lower a and b values during colorization, thus leading to undersaturated results. Class rebalancing is added to assist the network in coping with this existing skewness of the training data set. This is done by generating a histogram of the discrete colors in the dataset, using this information to balance the loss function with respect to each color. This histogram is then fused with a uniform distribution to deal with the fact that some of the colors in the histogram are so underrepresented that they have too big of an influence on the color rebalancing. To keep the color rebalancing within reasonable bounds, its expected value is kept equal to one. The class rebalancing factor is given by equation 5. The complete loss function of the cross-entropy is given by equation 1, while the mean is also taken over the batch.

$$V = ((1 - \lambda) \cdot H + \lambda)^{-1} \tag{5}$$

C. Training method

For convolutional neural networks training, stochastic gradient decent (SGD), sometimes together with momentum, is commonly used for updating the weights and biases [20][11]. The used hyper-parameters, especially the learning rate, requires careful tuning when using SGD. Often a scheduled learning rate, that is monotonically decreasing depending on the epoch, results in the best results. Since the focus of the present research lies not on what training method parameters are required, ADADELTA was used to automatically set the learning rate for every parameter[21]. Furthermore, Nesterov momentum was applied to lower the probability of getting stuck in a local minimum[24].

D. Final choice of model architectures

A major part in creating a successful neural network is finding a suitable network architecture. For image classification convolutional neural networks are widely used with success [8], [25], [11]. For our purpose a main feature of the convolutional network is that the image has to be reconstructed again, to retrieve spatial information.

In total a set of three convolutional network architectures are used. An architecture based upon Dahl [9], a pre-trained VGG16 [11] architecture this is actually Dahl, not sure about the actual amount of layers used for the reconstruction and a classification architecture [10]. The architectures of the network are split up in a feature extraction part and a reconstruction part, which will be expanded upon in the following sections.

Feature extraction

Dahl

This convolutional network is based upon the architecture used by Dahl [9]. It contains several convolutional layers, which use a 3x3 kernel throughout the network. After a set of convolutions a batch normalization is done followed by a max pool layer. Batch normalization is added such that in the reconstruction of the image the concatenated layers are in the same order of magnitude. The architecture was modified to fit the input dimensions. The network is an untrained network, having Glorit uniform distributed [23] initialized weights, meant to be trained simultaneously with the rest of the network. At the bottleneck of the architecture, the resolution of the feature maps are reduced to 16x16 pixels.

VGG16 A substantial amount of pre-trained networks are available, trained on the ImageNet classification database. The architecture used is based upon VGG16 [11], which uses a 3x3 kernel throughout the network. This network has a proven architecture, and can be obtained with pre-trained weights. Modifications on the network where required to fit the input dimensions. VGG16 is used in classification of RGB images, while our network only needs one input, a grayscale image. The pre-trained weights of the three input maps where averaged to accept a single grayscale input image. The VGG16 architecture features several convolutional layers followed by max pooling. Batch normalization was added before every max pool layer, such that in the later concatenation of the layers the values are in the same order of magnitude. At the bottleneck of the architecture the feature maps have a resolution of 16x16 pixels.

Classification The classification architecture shares much similarities with the VGG16 architecture. The main difference can be found in the final layers of the network. This is due to the fact that classification is used rather than direct reconstruction of the wanted color layers. This classification is subsequently used to colorize the image.

Reconstruction

For reconstruction of the image, linear up-scaling is used. The reconstruction begins after the bottleneck of the convolutional network is reached, where the resolution of the feature maps is 16x16 pixels. The up-scaled information is concatenated with the convolutional layer before the bottleneck that matches the up-scaled layer resolution. Then a convolutional layer is used for feature extraction of the concatenated layer. A batch normalization is applied and the processes is repeated until the original image resolution is retrieved. For both the VGG16 and Dahl based architecture a final output layer is used with a 2 feature map output, which match to the corresponding colour output layers.

For the classification the image is reconstructed to its original resolution. However, first a convolutional layer using a 1x1 kernel is used that has the same amount of feature maps as the required number of possible color classification bins. Then, these feature maps, representing discretized colours,

A detailed representation with of the various architectures is given in figure (XXX)

E. Loss function

IV. Results

In section III, the different architectures that are tested are described. The results

V. Discussion

VI. Conclusion

VII. Recommendations

probability comparison on object, picking the best joint probability for a region of the image instead of [illegible]

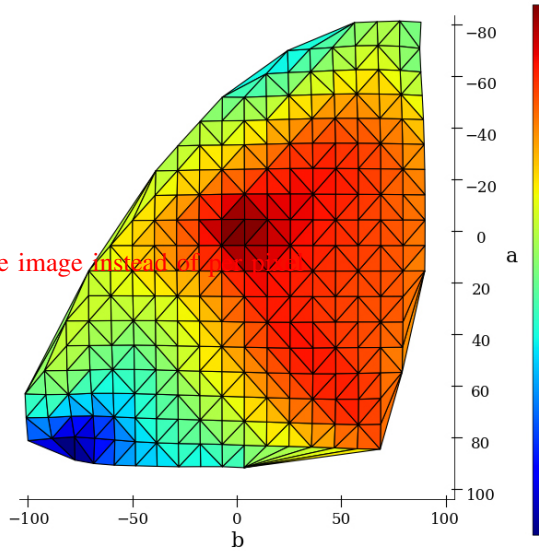


Fig. 8: The histogram of the total fruit data

TABLE II: Parameters of the compared architectures.

	compact	**compact classifier**	**compact classifier dilated**	**compact classifier dilated + concat**	**vgg16 compact + classifier**
dataset parameters					
Color space					
Fruit dataset	Yes	Yes	Yes	Yes	Yes
Landscape dataset	Yes	No	No	No	No
Hyperparameter					
Training method	ADADELTA with momentum	ADADELTA with momentum	ADADELTA with momentum	ADADELTA with momentum	ADADELTA with momentum
Epochs trained					
Batch size					
Architecture properties					
Front end module	trained from scratch	trained from scratch	trained from scratch	trained from scratch	VGG16 front end
Max pool	Yes	Yes	No	No	Yes
Strides	No	No	Yes	Yes	No
dilation	No	No	Yes	Yes	No
Concatenation	Yes	Yes	No	Yes	Yes
Kernel size	3x3	3x3	3x3	3x3	3x3
Activation function	ReLu	ReLu	ReLu	ReLu	ReLu
Batch norm	Yes	Yes	Yes	Yes	Yes
Classification properties					
Annealed mean temperature					
K-nearest neighbour	-	10	10	10	10
K-nearest neighbour sigma	-	5	5	5	5

REFERENCES

[1] G. Charpiat, M. Hofmann, and B. Schlkopf, *Automatic image colorization via multimodal predictions*, pp. 126–139. Springer, 2008.

[2] R. K. Gupta, A. Y.-S. Chia, D. Rajan, E. S. Ng, and H. Zhiyong, "Image colorization using similar images," in *Proceedings of the 20th ACM international conference on Multimedia*, pp. 369–378, ACM.

[3] Y. Zheng and E. A. Essock, "A local-coloring method for night-vision colorization utilizing image analysis and fusion," *Information Fusion*, vol. 9, no. 2, pp. 186–199, 2008.

[4] T. Horiuchi, "Colorization algorithm using probabilistic relaxation," *Image and Vision Computing*, vol. 22, no. 3, pp. 197–202, 2004.

[5] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," in *ACM Transactions on Graphics (TOG)*, vol. 23, pp. 689–694, ACM.

[6] Z. Cheng, Q. Yang, and B. Sheng, "Deep colorization," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 415–423.

[7] H. Ho and V. Ramesh, "Automatic image colorization,"

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105.

[9] R. Dahl, "Automatic colorization." http://tinyclouds.org/colorize/, 2016. [Visited on: 04-03-2016].

[10] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," *arXiv preprint arXiv:1603.08511*, 2016.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[12] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 447–456, 2015.

[13] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, 2010.

[14] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[15] K. Gregor, I. Danihelka, A. Graves, and D. Wierstra, "Draw: A recurrent neural network for image generation," *arXiv preprint arXiv:1502.04623*, 2015.

[16] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[17] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning." Book in preparation for MIT Press, 2016.

[18] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*.

[19] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of operations research*, vol. 134, no. 1, pp. 19–67, 2005.

[20] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification," *ACM Transactions on Graphics (Proc. of SIGGRAPH 2016)*, vol. 35, no. 4, 2016.

[21] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[22] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[23] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics*, 2010.

[24] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning.," *ICML (3)*, vol. 28, pp. 1139–1147, 2013.

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9.