Bringing old movies to life: colorization of B&W video sequences

Per Ljung
perlj@chalmers.se

Eric Johansson
ericjoha@student.chalmers.se

***Abstract*—We present two network architectures trained for the purpose of colorizing images. The first network is implemented from scratch and tested on minor datasets of fruits for the purpose of understanding the process of colorization. The second network is based on the ResNet-18 architecture and is trained on images from IMDb to optimize its ability to further colorize black and white movies. We illustrate the performance of the networks in terms of loss function decay and visualization of predicted images. The results show great promise, with room for improvement in future work.**

I. Introduction

In today's industry (and research) we are witnessing a rapid increase in automatizing all sorts of tasks by means of computer processing, and in particular deep learning. In this work, we look into the possibility of applying deep learning to colorize black and white images. As the history of photography has its beginning early in the 19th century, the technology has for the majority of its lifetime been restricted to black and white photos. It was not until around 1935 that color photography started becoming popular. However, it was still very expensive, so it was first during the 1950's that color photography became common in overall society. Nowadays, artists spend months of work in colorizing images from the pre-color photo era. Hence, we aim to reduce this time frame by constructing a deep neural network that takes black and white images as input, and returns corresponding colorized versions. Furthermore, we investigate the possibilities of colorizing a sequence of images, such as, e.g., a scene from a classic film shot in black and white.

The task of colorizing photos has recently shown much popularity in the machine learning community and has been studied in several works. For example, [1] aims for a simple approach by applying Self Organizing Maps, [2] uses a deep neural network including the ResNet-model pre-trained on the ImageNet dataset, and [3] utilizes the Inception-ResnetV2 architecture. For more related works, see, e.g., [4, 5, 6].

In all above mentioned papers, the authors pre-process the images by converting them from the standard RGB format into some format based on one lightness channel and two channels for colors, with the most common choice being the CIELAB color space (also referred to as *L*a*b**). In this way, the grayscale lightness channel can be used as input, and the two color channels can be returned as output. One can further combine the input lightness channel with the output color channels to create the full image and then convert it back into the standard RGB format. An important advantage of this approach is that the network only requires the prediction of two outputs, instead of the three outputs necessary for the RGB format.

Our approach throughout this work is much based on previous works. At first a network is implemented from scratch, and analyzed on a small dataset of fruits to confirm that the network is able to colorize trivial examples. For the purpose of colorzing black and white films, a large dataset of images from IMDb is used for training. For this data, we use another network architecture that combines the ResNet-18 structure in combination with an upsamling scheme, mainly based on the work in [7]. The data in both cases is pre-processed by converting it to the *L*a*b** color space.

The outline is as follows: in Section II we describe the method used in this work, mainly in terms of pre-processing of data and network architecture. Section III presents and discusses the results with regard to training and validation loss as well as visualization of validation images. At last, Section IV concludes the project and presents possible future work.

II. Method

In this section, we present the datasets used for the training of the networks, along with corresponding pre-processing, and discuss the two types of network architectures implemented.

A. Data and data pre-processing

Throughout the project, we consider two different datasets:

- The dataset *Fruit and Vegetable Image Recognition* (see [8]), containing 36 classes of fruits and vegetables scraped from Bing Image Search.
- The dataset *IMDB-WIKI* (see [9]), with images of the most popular 100,000 actors listed on IMDb website.

Both datasets consist exclusively of images in RGB format. As earlier mentioned, it is preferable to utilize *L*a*b** formatted images, and hence a large part of the pre-processing lies in the conversion between these two formats. This is primarily dealt with using handy functions from the `scikit-image` package. The decomposition of an image into its respective *L*a*b** channels is shown in Fig. 1. All images are also normalized such that the values of each channel are in the interval $[0, 1]$. Lastly we split the *L*a*b** image into two parts, the *L** channel used as input and the *a*b**-channels used as the ground truth. This is all done in a custom built PyTorch module called `DataFolder` in combination with a transformation that resizes all images to size 256×256. For the training data,

the transformation further utilizes data augmentation in the sense of random horizontal flips and random cropping.

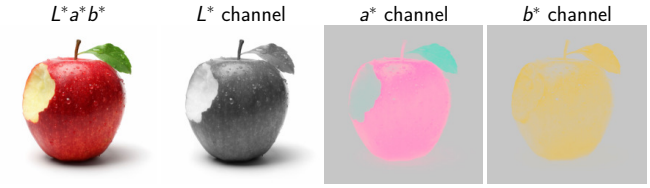

Fig. 1: Decomposition of an image into its L^*-, a^*- and b^*-channel, respectively.

B. Network architecture

Two different networks with similar architecture were created. Firstly, a network built from the ground up, referred to as FruitNet, was used to make sure that the intended architecture was capable of colorization images when training on single images and small, homogeneous datasets. Secondly, a network based on the ResNet-18 model, was used to learn the task of colorizing images from a larger and more diverse dataset.

The basic idea of the network architecture is to use a convolutional network inspired by autoencoders (see [10]) but instead of regenerating the input L^*-channel, the goal is to generate the a^*b^*-channels representing the colors corresponding to the input image. The first part of the network can be compared to the encoder part of an autoencoder and consists of a number of convolutional layers with kernel sizes 3×3 with a constant padding of 1 such that `stride=1` outputs feature maps of the same size as the input and `stride=2` halves the width and height of the input. By altering between a stride of 1 and 2 the feature maps of the network will gradually decrease in size resulting in a successively larger receptive field which in turn enables the network to detect more general features. The second part of the network can be compared to the decoder part of an autoencoder. It is similar to the first part in that it consists of stacked convolutional layers but with the difference that feature maps now gradually increase in size until they reach the same spatial dimensions as the input image. This is done by the nearest neighbor upscaling method meaning that each pixel of the input is duplicated into a `scale×scale` patch in the output where `scale` is an argument given to the upscaling function. A scaling of 2 thus results in an output with doubled width and height. The number of kernels corresponding to each convolutional layer may be chosen arbitrarily/experimentally except for the last layer where the number of output channels is set to 2 such that the last layers output can be interpreted directly as the prediction of the a^*b^*-channels corresponding to the input. An illustration of the FruitNet architecture is shown in Fig. 2. In addition to the upsampling, each convolutional layer is followed by batch normalization layer and element wise applied ReLU.

For subsequent part of the project, where a larger and more diverse dataset is considered, a similar network is implemented, where the first part (the encoder part) consists

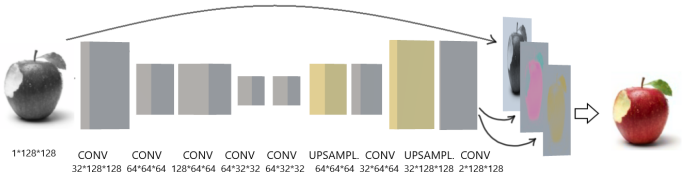

Fig. 2: Illustration of the FruitNet architecture. Each convolutional layer (except for the last one) is followed by a batch normalization layer and ReLU activation. The upsampling uses the nearest neighbor method with `scale=2`. Note that the first channel of the output is taken as the predicted a^*-channel and the second channel as the predicted b^*-channel. The output a^*- and b^*-channels are then concatenated with the input L^*-channel to construct the predicted colorized image.

of the first 4 residual blocks from the ResNet-18 model. Note that ResNet-18 is built to take images with three channels as input, while our input consists of only one channel. It is thus necessary to modify the kernels of the first layer. This is done by summing the weights of all channels resulting in kernels with the original dimensions, but with only one channel. For details regarding code and experiments, we refer to the GitHub repository of the project [11].

III. Results/Discussion

In this section, we present the results of our networks, i.e., FruitNet as well as the network based on ResNet-18 architecture, respectively. At first, we show how FruitNet manages to colorize food, starting from a single class of carrots, extending to larger datasets of multiple classes of fruits. Following this, we present the performance of the second network on the *IMDB-WIKI* dataset it has been trained on, as well as its ability to colorize black and white film classics (not included in the training or validation data). In all cases, decay of loss functions is presented along with visual comparisons.

A. Colorization of fruits

To test FruitNet's ability to colorize images, it was trained on a single class, carrots, from the *Fruits and Vegetables* dataset. This class was split into a training set of 95 images and a validation set of 9 images. An example of input image, its prediction for different epochs, together with ground truth can be seen in Fig. 3. As seen from the figure the network is able to distinguish between carrot fruits and the green blasts, while encountering minor problems with detailed colorization of the blasts. This network was further tested on a subset containing both apples and bananas to test the performance on multiple classes of fruits. The subset was split into a training set of 158 images and a validation set of 10 images. An illustration of the network performance for different epochs can be seen in Fig. 4. The figure shows how the network fails to predict the red color of an apple. The reason for this is that the data contains both green and red apples (with majority red apples). Consequently, FruitNet predicts the color of an apple as a

Fig. 3: FruitNet validation of a bunch of carrots for different epochs, compared to input and ground truth.

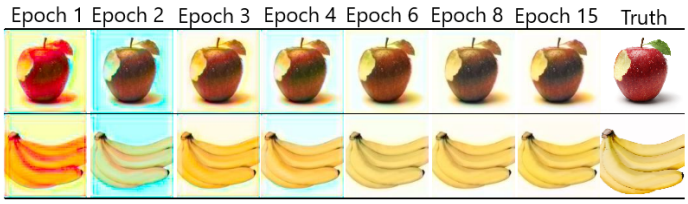

Fig. 4: FruitNet validation of an apple and a banana for different epochs, compared to ground truth.

reddish sepia tone. Another insight from Fig. 4 is the exact colorization of the banana which emphasizes the importance of feature detection. Since all bananas are yellow, and the first part of the network detects 'banana features', it can ignore the red and green colors it recalls from previous images of apples. The decay of the loss function through all epochs is plotted for validation and training data of both cases in Fig. 5.

B. Colorization of film

The second network considered, based on the ResNet-18 architecture, was trained for 15 epochs on a subset of the *IMDB-WIKI* dataset, containing 189,000 images. This was split into a training set of 188,000 images and a validation set of 1,000 images. The decay of the loss as function of number of epochs is depicted in Fig. 6. An example of how the trained network performs on the validation data is illustrated in Fig. 7, where the target data is put side-by-side to the input grayscale data and the predicted image. The figure shows how the network is good at characterizing human skin tones, while backgrounds of colors other than black or white tend to take on a brownish tone. This makes sense as in the *L*a*b** color

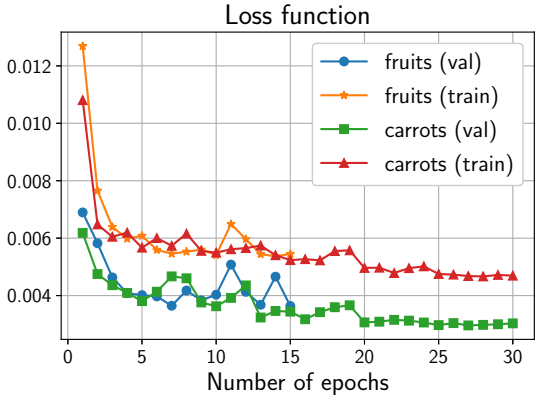

Fig. 5: Loss as function of number of epochs for FruitNet, trained on carrots and fruits, respectively.

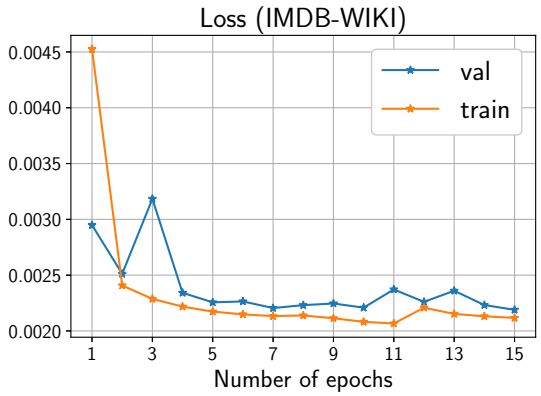

Fig. 6: Decay of loss function for the IMDB-WIKI data over 15 epochs of training.

space, a brownish tone corresponds to a combination of the center regions of the *a**- and *b**-channels, thus being the best guess in terms of mean squared error.

As a final test, the trained network was used to colorize sequences from classic black and white films. Two such examples, where three stills from the same scene were colorized, can be seen in Fig. 8-9. In Fig. 8, we see how the network performs a realistic colorization of human skin tones in both facial close-ups as well as for a single hand. The light background however take on a brownish shade, as mentioned, due to it being a qualified guess for backgrounds in general, given that the backgrounds in the training data follow a uniform color distribution. Fig. 9 further shows that the network is able to colorize multiple persons in one frame well, along with their clothes, while the overall images still contain a sepia-like tone. Positively enough, the network shows consistency in its colorization, causing very little flicker in the sequences.

In all examples, the network manages to colorize human skin tones and clothes well, while being very restricted to the sepia tone for more general objects. This makes sense, as the dataset is handpicked to perform well on humans, a choice made by the fact that humans are the majority context of classic films. A downside is that the surroundings are not particularly pleasant visually speaking. Experiments to circumvent this were performed, using a dataset called *Places* (see [12]), consisting of images of random everyday sites. Using this dataset, a network was created that performed very well in colorizing nature, such as grass, trees, sky, etc. However, the resulting network had major troubles in colorizing human skin, causing sequences from old films to be poorly colorized, with the exception of open outdoor environments.

IV. Conclusion and Future Work

In this work, we have created and trained a neural network, both from scratch and using the ResNet-18 model as basis. The network implemented from ground up was trained on subsets of a dataset of fruits and vegetables, and performed well on single colored classes, while encountered minor struggles

Fig. 7: Example how the trained network predicts the middle grayscale images. The leftmost image is the target, and the rightmost is corresponding prediction.

Fig. 8: Colorized stills from the film *Psycho (1960)*.

with multiple colored classes, such as apples. The second network was developed for the purpose of colorizing black and white movies. Human skin tones and clothes were accurately colorized, while an overlying sepia-like tone tends to dominate backgrounds and non-human objects. This due to sepia being a qualified guess in terms of mean squared error in the utilized *L*a*b** color space.

There are several possible approaches to improve the presented results. One is to utilize the *Places* dataset to train a network in colorizing general environments, and then apply transfer learning using photos of humans from the *IMDB-WIKI* dataset to fill in the human skin tone gaps. If the general sepia tone is reduced, there is consequently a larger risk of flicker in the image sequences. Thus, future work possibilities also exist in the area of reducing flicker between images. One approach here is the construct an RNN, whose previous hidden states depend on previous images, so that information is passed on throughout the sequence. Furthermore, averaging between images to construct smoother color transitions is also of interest.

Fig. 9: Colorized stills from the film *Casablanca (1942)*.

References

[1] M. Richart, J. Visca, and J. Baliosian, "Image colorization with neural networks," *2017 Workshop of Computer Vision (WVC)*, pp. 55–60, 2017.

[2] D. Mikhalina, A. Kuz'menko, K. Dergachev, and V. Shkaberin, "Image colorization," *GraphiCon'2019 Proceedings. Volume 2*, 2019.

[3] M. R. Joshi *et al.*, "Auto-colorization of historical images using deep convolutional neural networks," *Mathematics*, vol. 8, no. 12, 2020. [Online]. Available: https://www.mdpi.com/2227-7390/8/12/2258

[4] Z. Cheng, Q. Yang, and B. Sheng, "Deep colorization," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[5] D. Varga and T. Szirányi, "Fully automatic image colorization based on convolutional neural network," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2016, pp. 3691–3696.

[6] G. Larsson, M. Maire, and G. Shakhnarovich, "Colorization as a proxy task for visual understanding," in *CVPR*, 2017.

[7] L. Melas-Kyriazi. Image colorization with convolutional neural networks. [Online]. Available: https://lukemelas.github.io/image-colorization.html

[8] K. Seth, "Fruit and vegetable image recognition, version 6," https://www.kaggle.com/kritikseth/fruit-and-vegetable-image-recognition, accessed: 2021-10-20.

[9] R. Rothe, R. Timofte, and L. V. Gool, "Deep expectation of real and apparent age from a single image without facial landmarks," *International Journal of Computer Vision*, vol. 126, no. 2-4, pp. 144–157, 2018.

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel distributed processing: explorations in the microstructure of cognition*, 1986.

[11] E. Johansson and P. Ljung, "Bringing old movies to life: colorization of b&w video sequences," https://github.com/LjungPer/deep-learning-project, 2021.

[12] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Advances in Neural Information Processing Systems*, Dec. 2014.