Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Instytut Radioelektroniki i Technik Multimedialnych

Praca dyplomowa

na studiach: **Studium Podyplomowe**
Głębokie Sieci Neuronowe – Zastosowanie w Mediach Cyfrowych

Multimodalna koloryzacja obrazów w skali szarości
z wykorzystaniem głębokich sieci neuronowych.

mgr Tomasz Ostaszewicz

opiekun pracy dyplomowej

mgr inż. Rafał Protasiuk

Warszawa 2020

Multimodalna koloryzacja obrazów w skali szarości z wykorzystaniem głębokich sieci neuronowych

Streszczenie. Niniejsza praca dyplomowa podejmuje problematykę multimodalnej koloryzacji obrazów z wykorzystaniem głębokich sieci neuronowych. Celem zaprezentowanego modelu jest koloryzacja obrazów w skali szarości na różne sposoby w zależności od kolorów z obrazów referencyjnych. Zaprezentowany model pokazuje, że multimodalna koloryzacja jest możliwa nawet wtedy, gdy w zbiorze uczącym nie ma przykładów tych samych zdjęć o różnej koloryzacji. Model do koloryzacji stanowi połączenie architektury Resnet, U-net i algorytmu Patch Match. Bazową przestrzenią kolorów użytą do uczenia modelu jest CIE Lab. W celu uzyskania szybszego procesu trenowania algorytm Patch Match został zaimplementowany bezpośrednio w środowisku CUDA. Wyniki z modelu koloryzacji na zbiorze zawierającym zdjęcia kwiatów przedstawią naturalne wyglądające zdjęcia kwiatów pokolorowane zgodnie z kolorami z obrazów referencyjnych. Analiza rezultatów modelu pokazała również słabości modelu przy koloryzacji zdjęć zawierających wiele drobnych kwiatów oraz niektórych zdjęć zawierających bukiety róż i tulipanów. Porównanie wyników z innymi publicznie dostępnymi modelami do koloryzacji pokazało, że zaproponowany model na zbiorze kwiatów generuje najbardziej naturalnie wyglądające koloryzacje wśród wszystkich porównywanych metod.

Słowa kluczowe: Głębokie sieci neuronowe, koloryzacji, obrazy w skali szarości, wielomodalna koloryzacja obrazów w skali szarości, Resnet, U-net, Patch, Match, przestrzeń kolorów CIE Lab

Multimodal colorization of greyscale images using deep neural networks

Abstract. This thesis presents deep neural network model for multimodal colorization of greyscale images. The aim of presented model is to colorize greyscale images in different ways using colors from supplied reference images. This model clearly shows that training multimodal colorization model is possible without using different colorizations of the same grey scale image in training process. Colorization model is constructed using combination of Resnet, U-net deep neural network architectures and Patch Match algorithm. The base color space for model training is CIE Lab. Use of custom implementation of Patch Match algorithm directly on CUDA platform allowed to speed up training time. Constructed Colorization neural network produces naturally looking colorization results on flowers data sets. Analysis of result of the model also showed model weaknesses when colorizing images with many fine grained flowers and for some pictures of buckets of roses and tulips. Comparison with other publicly available colorization models showed that the proposed model produces the most naturally colorizations on flowers data set among compared methods.

Keywords: deep neural networks, colorization, greyscale images, multimodal colorization of greyscale images, Resnet, U-net, Patch Match, CIE Lab color space

Table of contents

1 Introduction

Grayscale image colorization is a task that takes as input greyscale image and outputs colorized images. Is it is expected that colorization will be natural looking. Approaches to image colorization before deep learning involved various different methods like for example:

- Kernel Hilbert Spaces in “Image and Video Colorization Using Vector-Valued Reproducing Kernel Hilbert Spaces” [1]
- Probabilistic in paper “Colorization algorithm using probabilistic relaxation” [2]
- Statistical in “Sparse natural image statistics and their applications to colorization and compression” [3]
- Support Vector Machines in “Large-scale image colorization based on divide-and-conquer support vector machines” [4]

Some of traditional approaches took slightly different approach. In order to improve colorization results they use additional hints to model colorization. Examples are presented in papers:

- Using human color cues in “Colorization using Optimization” [5]
- Using reference color images in “Colorization by Example” [6]

In 2012 deep learning revolution in image processing begun. Deep learning models outperformed traditional approaches in image processing. This revolution also impacted image colorization subfield. New approaches to image colorization using deep learning models involve:

- Convolutional Neural Network with Residual Blocks in “Deep Depth Colorization” [7]
- Unet-like architecture in “Coloring with limited data: Few-shot colorization via memory augmented networks” [8]
- Generative Adversarial Networks in “Chromagan: Adversarial picture colorization with semantic class distribution” [9]

Analogously to traditional colorization methods there are also deep learning approaches that use some form of hints about color from other images or from human made inputs. Examples are:

- Using human cues in “Scribbler: Controlling deep image synthesis with sketch and color” [10]
- Using reference color images “Deep exemplar-based colorization” [11]

Colorization without additional cues about color usually implicitly assume that greyscale images will be colorized in one way only. That is given image in greyscale the result of the method, that is colorized image, will always be the same for input image. In practice often one greyscale image can be colorized in multiple ways such that each colorization is naturally looking, e.g. human cloths, car, birds, flowers can be colorized in many ways. Colorization approach that assumes multiple outcomes for one given image is called multimodal colorization. The aim of multimodal approach is to obtain multiple natural colorizations for the same gray scale image.

The aim of this thesis is to investigate one of the models in area of multimodal colorization methods. Model considered in this thesis is presented in scientific paper: “Deep Exemplar-based Colorization” [11] (link: https://arxiv.org/pdf/1807.06587.pdf) by Mingming He , Dongdong Chen , Jing Liao , Pedro V. Sander and Lu Yuan. In simple terms the model for colorization of grey scale images uses additional color image. This color image serves as a source of reference colors. Details of the model architecture and implementation are presented in the following chapters.

2 Theoretical aspects of proposed model

Proposed model consists of three parts:

2.1 Similarity model which is based on Resnet34 architecture.
2.2 Patch Match algorithm.
2.3 Colorization model which is based on U-net like architecture.

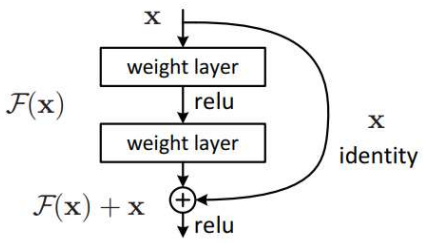
These two models use standard deep neural technics like Batch Normalization, ReLU activation function and skip connections. These technics accelerate model training and improve final metrics of the trained models. As for optimization AdamW optimizer was used. AdamW optimizer differs from Adam optimizer in that weight decay is applied only during weight update and not in loss function (when gradients are calculated). AdamW improves convergence vs. Adam

Described above architectures are quite standard and well known in Deep Learning field.

ReLu activation function:

$$ReLU(x) = \begin{cases} 0 & if\ x < 0 \\ x & if\ x \geq 0 \end{cases}$$

Resnet34 building block skip connection:



Source of image: [12]

Loss function used in Adam optimizer has a form with regularization:

$$L(f(x_t, \omega_t), y_t) + \frac{1}{2}\lambda\omega_t^2$$

Thus calculated gradient has form:

$$\mathrm{g_t} = \nabla_{w_t} L(f(x_t, \omega_t), y_t) + \lambda\omega_t$$

This gradient is used to calculate exponentially weighted averages m_t of previous gradients $g_t, g_{t-1,} \dots, g_1$ and exponentially weighted averages v_t of squared gradients $g_t^2, g_{t-1}^2 \dots, g_1^2$. Then update rule is

$$\omega_{t+1} = \omega_t - LR_{t+1}\left(\frac{m_t}{\sqrt{v_t + \epsilon}} + \lambda\omega_t\right)$$

where ϵ is small constant to avoid division by zero in case v_t is zero.

In AdamW loss function is assumed to be without regularization part.

$$L(f(x_t, \omega_t), y_t)$$

So calculated gradient has a form

$$\hat{g}_t = \nabla_{w_t} L(f(x_t, \omega_t), y_t)$$

These gradients $\hat{g}_t, \hat{g}_{t-1} \dots, \hat{g}_1$and squared gradients $\hat{g}_t^2, \hat{g}_{t-1}^2 \dots, \hat{g}_1^2$ (that is without regularization part) are used to calculated exponentially weighted averages, respectively: $\widehat{m}_t$ and $\hat{v}_t$. Then update rule of AdamW optimizer is similar to Adam optimizer (but with differently calculated gradients).

$$\omega_{t+1} = \omega_t - LR_{t+1}\left(\frac{\widehat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} + \lambda\omega_t\right)$$

3 Proposed model

3.1 General schema

Model takes as input a pair of images:

1. Target image (denoted as T) – the image that model aims to colorize
2. Reference image (denoted as R) – image that serves as color reference to colorization

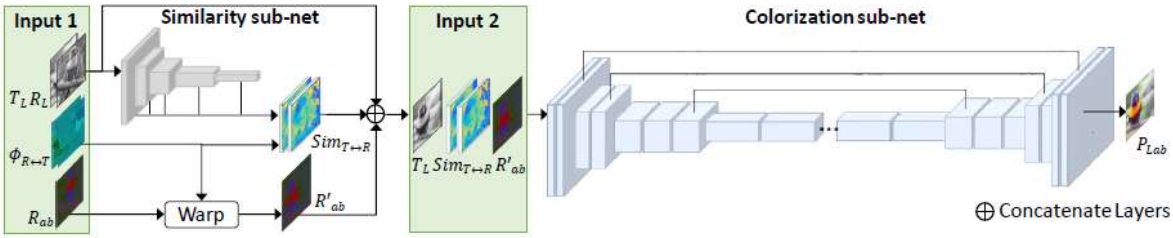
Subindex at target or reference image denotes channels associated with image e.g.:

T_{RGB} – target image in RGB color space,

R_L – reference image with L channel form CIE Lab color space

T_{RGB} and R_{RGB} images before training are converted to CIE Lab scale: T_{Lab} and R_{Lab}. Then gray scale images T_L and R_L are fed into similarity network. For each image 5 sets of feature maps are extracted (exactly at the end of layers conv1, layer1, layer2, layer3, layer5 denoted at similarity network figure). These feature maps from T_L and R_L are passed into Patch Match algorithm which returns cosine similarity distance mapping between each patch of features from T_L and its closest correspondence patch from feature map of R_L(patch is constructed by taking 3x3 subset from Height x Width dimensions and all channels from feature map). These mappings are denoted as $PMD_{T\to R}$ (stands for Patch Match Distance) for similarity distance between T_L and R_L feature maps and $PMD_{R\to L}$ for similarity distance between R_L and T_L feature maps (please note that maps $PMD_{R\to L}$ and $PMD_{L\to R}$ are not inverse of each other). With each closest patch we can associate corresponding index of that patch, that is we have index values $PMI_{R\to L}$ and $PMI_{L\to R}$ Having those indexes we can use them to warp color spaces. For training T_{ab} is doubly warped using indexes PMI from similarity distance calculation that is $PMI_{T\to R\to T} := PMI_{T\to R}(PMI_{R\to T})$. So as a result we obtain warped T_{ab} image: $T'_{ab} = T_{ab}(PMI_{T\to R\to T})$. During inference this mapping is replaced with warping R_{ab} image that is: $R'_{ab} = R_{ab}(PMI_{T\to R})$ which uses warped R_{ab} colors instead of warped target T_{ab} colors. Color similarity distances $PMD_{T\to R}$ and $PMD_{T\to R\to T}$ concatenated together are denoted as $Sim_{T\leftrightarrow R}$ in the picture below.

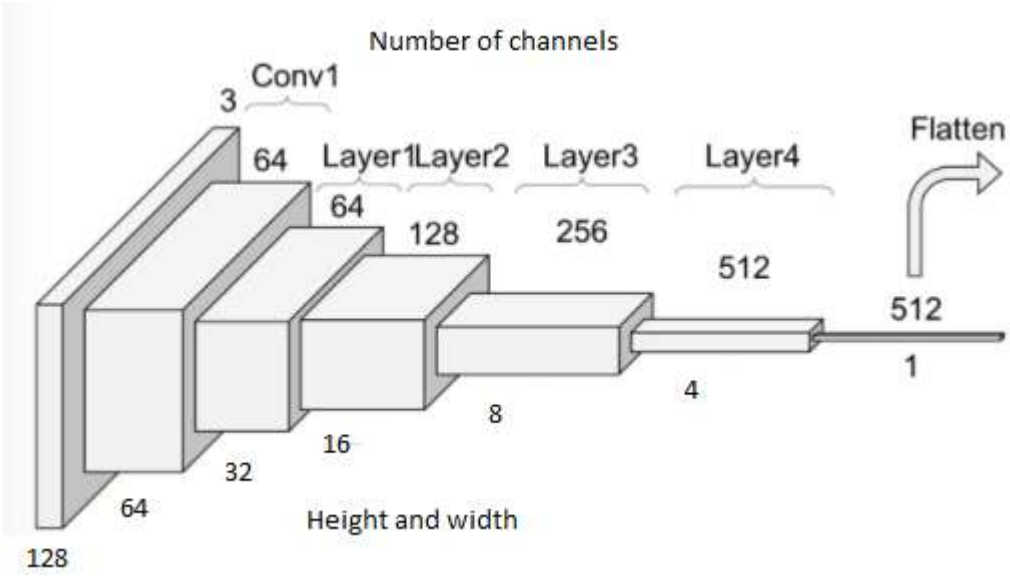
Constructed in this way data is inputed into colorization network, which has U-net like architecture. The whole system look like in the image below.



Source of the image: [11]

3.2 Similarity network schema

Similarity network is based on Resnet 34 architecture. The schema of similarity network is presented in a picture below.



Source of the image: https://medium.com/@paul.m.carter1992/fast-deployment-of-an-image-classification-web-app-with-fast-ai-14d0bac48ae7

Pytorch's computation graph for similarity network is presented in attached below file (graph is too large to be displayed on single A4 page).

similairty_ciel_model_torchviz_schema.p

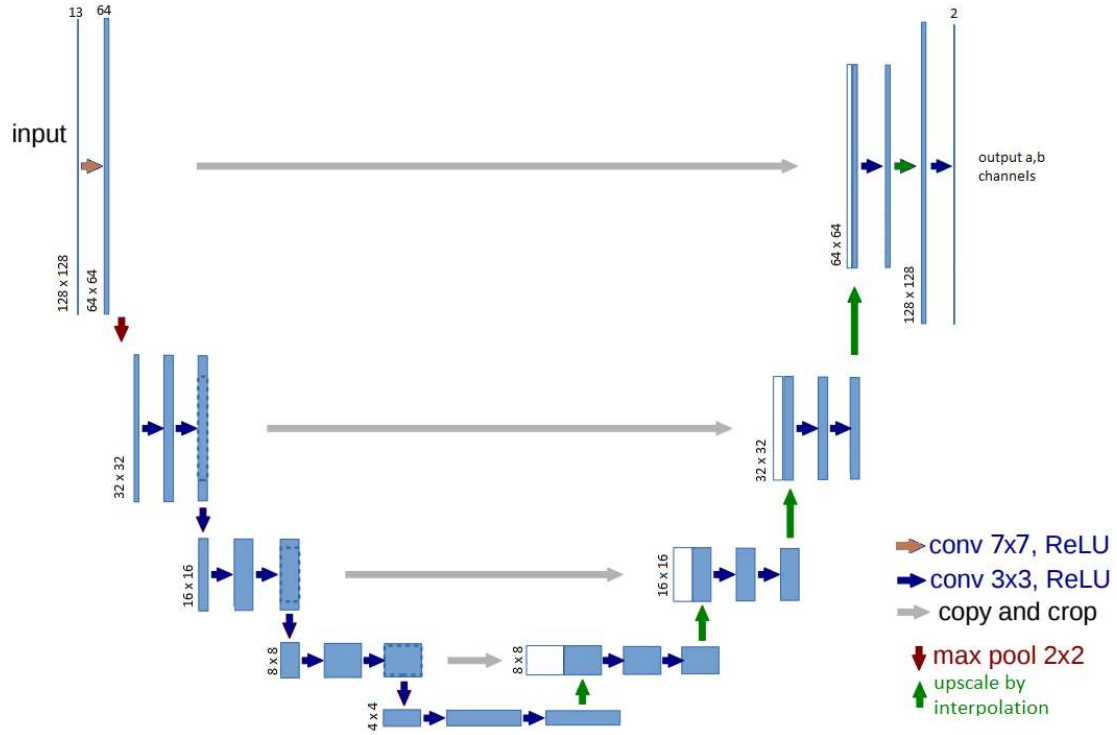
Similarity network is used in two ways:

1. Similarity CIE L network that is used to calculate features from T_L and R_L images.
2. Similarity CIE RGB network that is used to calculate perception loss part in colorization network.

Both similarity networks before training colorization network were finetuned on training data set. Finetuning started from Resnet 34 with weight pretrained on ImageNet data set. For finetuning only all model head parameters and Batchnorm parameters (including Batchnorm layer in model body) were unfrozen. Loss used during finetuning was standard cross-entropy loss (in training data set each image had assigned category label).

3.3 Colorization network schema

Colorization network has Unet-like architecture. The following picture presents colorization network schema.



Source of above image: [12]

Pytorch's computation graph for colorization network is presented in attached below file (graph is too large to be displayed on single A4 page).

colorization_model
_torchviz_schema.pc

Colorization network loss has two parts: chrominance loss and perception loss

$$L(f_{colorization}(x), T_{ab}) = L_{chrom}(f_{colorization}(x_{T\to R\to T}), T_{ab}) + \alpha L_{perc}(f_{similarityRGB}(convertToRGB(f_{colorization}(x_{T\to R}), T_{RGB}))$$

1. Chrominance loss:

$$L_{chrom}(f_{colorization}(x_{T\to R\to T}), T_{ab})$$

is a loss Mean Squared Error between predicted ab colors and true colors T_{ab}

2. Perception loss:

$$\alpha L_{perc}(f_{similarityRGB}(convertToRGB(f_{colorization}(x_{T\to R}), convertToRGB(T_{ab}))$$

Is a Mean Squared Error between features of layer 4 of Colorization RGB model. This loss is used to make sure that predicted images and target images have the close high-level features that is colorization is naturally looking. Also this part forces model to use reference colors as inputs. This is important for inference stage where T_{ab} is not available that is only R_{ab} is used as color source.

4 Model implementation

4.1 Description of training and validation data sets

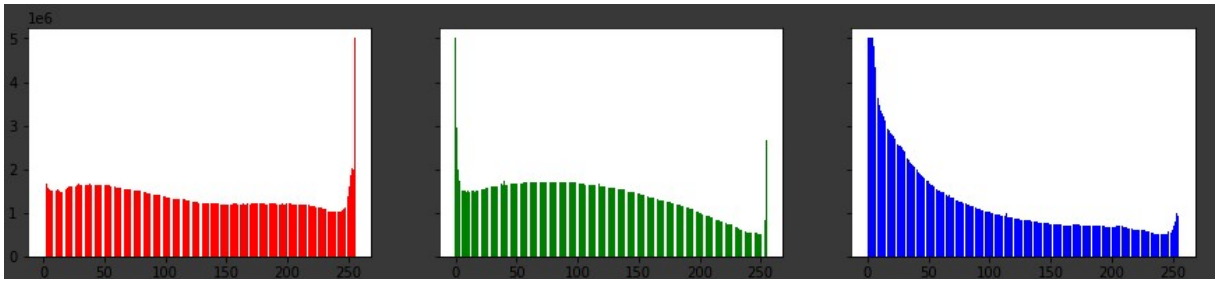
Data set for model training and validation was downloaded from Kaggle site:
https://www.kaggle.com/alxmamaev/flowers-recognition

The data set contains 4326 color images of flowers grouped in a 5 classes. Before training 13 images were removed due to PIL library having some problems with reading those images. Next data set was separated into training and validation part. The following table present details of obtained data sets.

Classes	Training set	Validation set
daisy	686	81
dandelion	956	94
rose	708	74
sunflower	663	69
tulip	887	95
	3900	**413**

Sample images from the data set are presented below:

Color histogram for Red, Green and Blue colors calculated on training set is presented below:



4.2 Input image transformations

For Similarity RGB model the following data transformations were used:

a) Image resize to 128+32
b) Random augmentation transformations: random horizontal flip, random rotation random affine transformation, random resized crop to 128 pixels
c) Normalization using RGB mean and standard deviation from Image Net data set: mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]

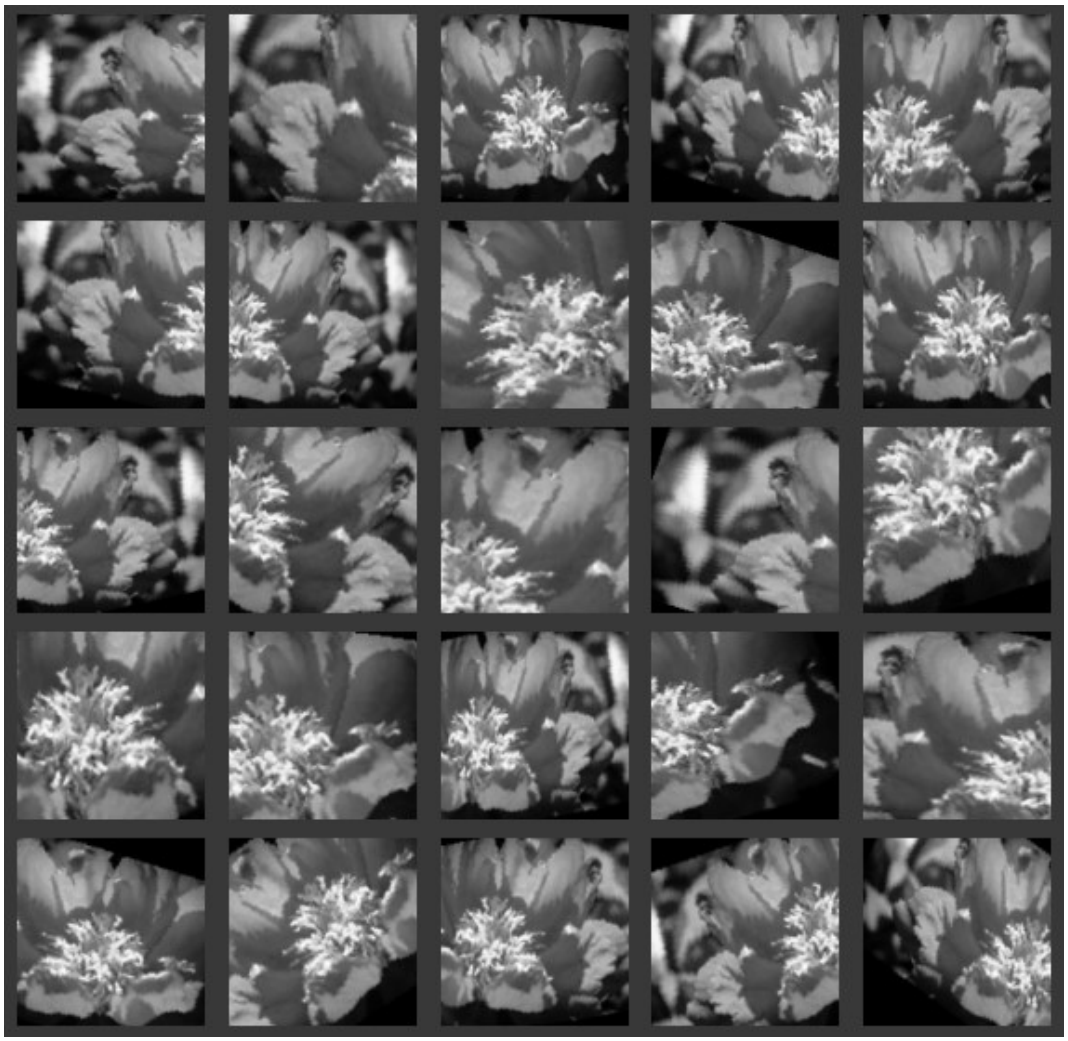
For Similarity CIE L model the following data transformations were used:

a) Image resize to 128+32
b) Random augmentation transformations: random horizontal flip, random rotation random affine transformation, random resized crop to 128 pixels
c) Normalization using L mean and standard deviation from training data set: mean=45.4391 and std=26.2266

Results of data augmentations for CIE L model for one sample image are presented below.

Sample image:

Sample augments of the same image:



Transformations for Colorization model were the same as used during Similarity CIE L model training. This is due to the fact that input to colorization model are feature maps from Similarity CIE L model (transformed with Patch Match algorithm – as described in model architecture section).

4.3 Patch Match algorithm implementation

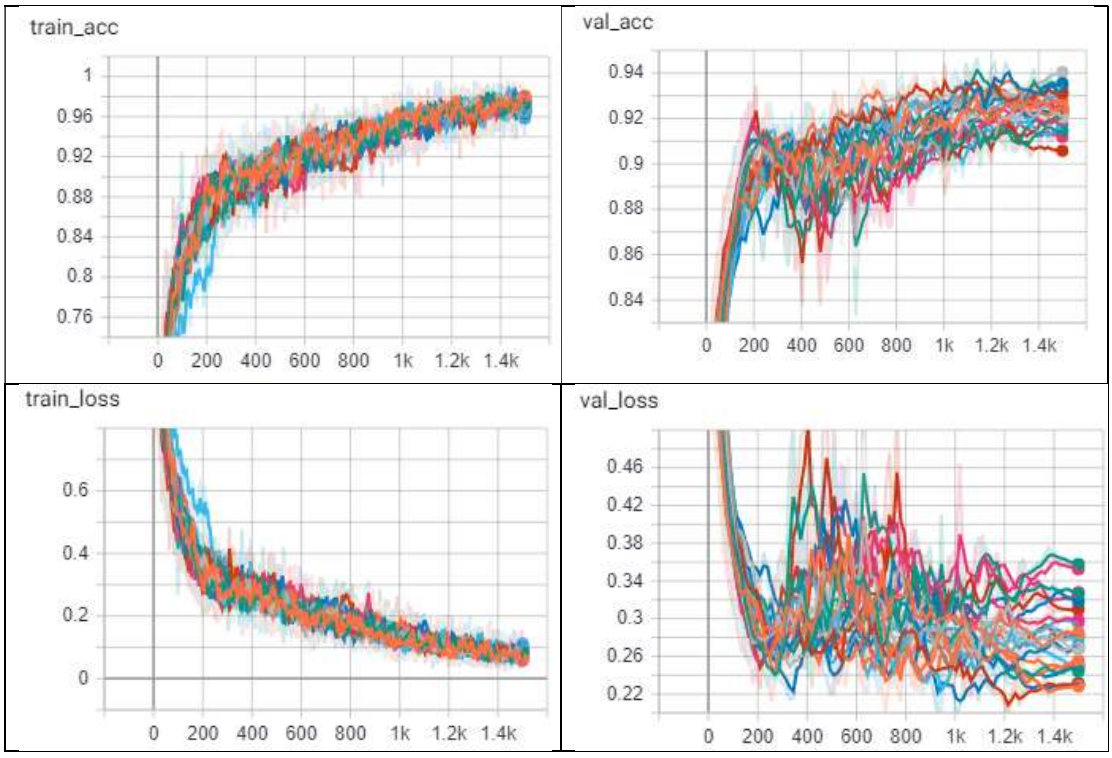
Patch Match algorithm involves multiple distance calculations between patches selected from target and reference images. Let's see how. Both batch tensors of target and reference images have shapes [BS, C, H, W], where BS-batch size, C – number of channels, H-height, W-width. Then from target and reference tensors, patches 3x3 (of HxW) are selected. Patch tensors have shape [BS, C, 3, 3]. Then cosine distance between two patches (from target and reference image) is calculated (for each pair of images). As a result distance vector of shape [BS] is obtained. This patch distance calculation is performed thousands of times during one Patch Match algorithm run. Current tensor and neural network toolbox in Pytorch library does not have easy to use function for Patch Match algorithm (like for example torch.nn.Conv2d class for 2d Convolution which also works on patches from input tensor). Implementation of Patch Match algorithm using basic tensor operations in Pytorch resulted in efficient code, which runs

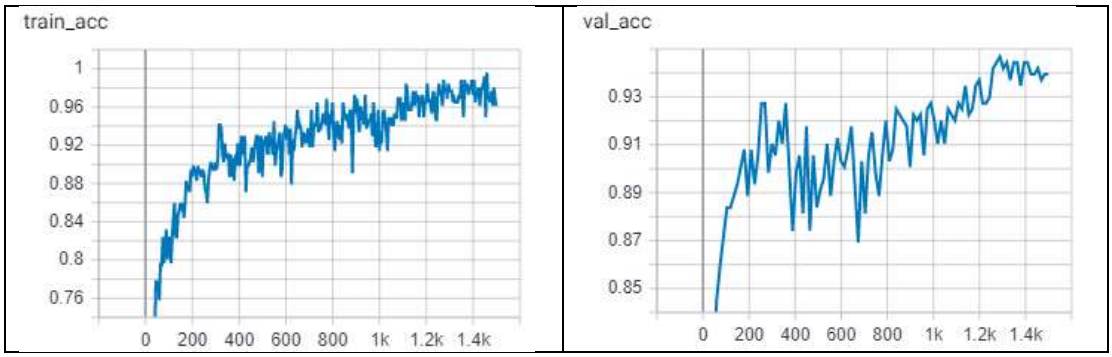
each distance calculation between patches in sequential order. To give more precise numbers, for example for target and reference tensors of shape [96, 64, 64, 64] Patch Match algorithm implemented in Pytorch runs 5 minutes and 12 seconds. This is a time needed to prepare input batch for Colorization network using “Pytorch” implementation. To correct this inefficiency Patch Match algorithm was directly implemented in CUDA C++ programming language. Patch Match algorithm CUDA implementation runs in 12 seconds, which means 26 times speed-up. Due to Patch Match algorithm implementation in CUDA C++ training of Colorization Network in 53 hours was possible.

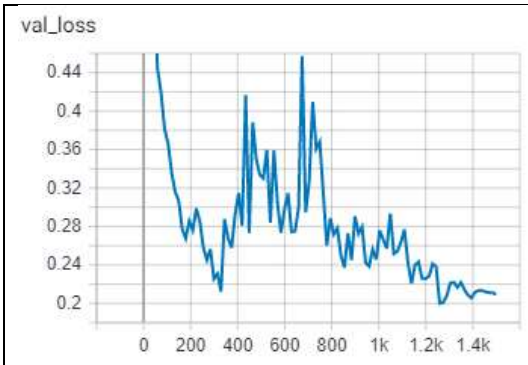
5 Results and conclusions

5.1 Results of training RGB and CIE Lab models similarity models.

Below are presented graphs of finetuning RGB similarity model for different training runs. Different training runs were varied by random choice of hyper parameters: learning rate for the model body parameters, learning rate for the model head parameters, application or not of weight decay for batch norm parameters. The model with best validation accuracy was chosen. Similarly CIE L similarity model was trained. The best obtained validation accuracy were: 95% for RGB similarity model, 89% for CIE L similarity model.



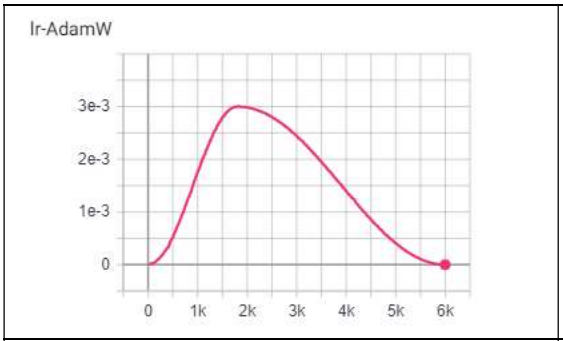




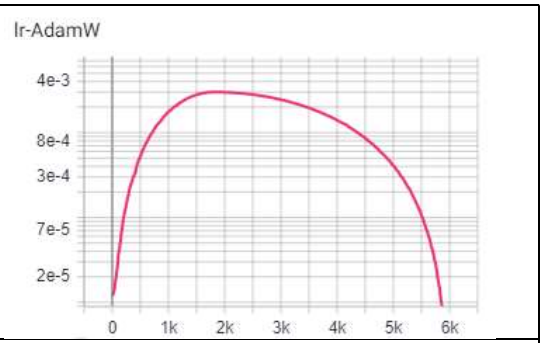
5.2 Training Colorization model without Patch Match algorithm

Colorization model without Patch Match algorithm did not use results of Patch Match algorithm. Input to colorization model has shape [batch size, 13, height, width], where 13 means number of channels. Channel number 0 represents grayscale image represented as L channel from CIE Lab color space. Channels 1 to 12 are results of Patch Match algorithm. Patch Match results were replaced with zero tensor.

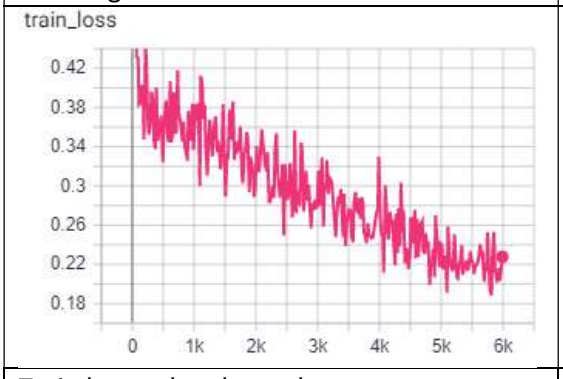
Training was performed on 150 epoch using single V100 GPU on Google Colab. It took approximately 3 hours to train. Below are graphs from training process:



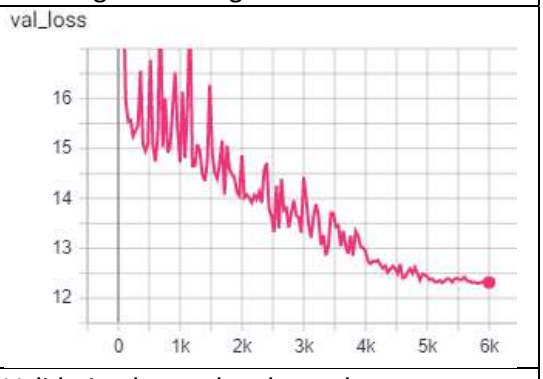
Learning rate vs batch number



Learning rate on log scale vs batch number

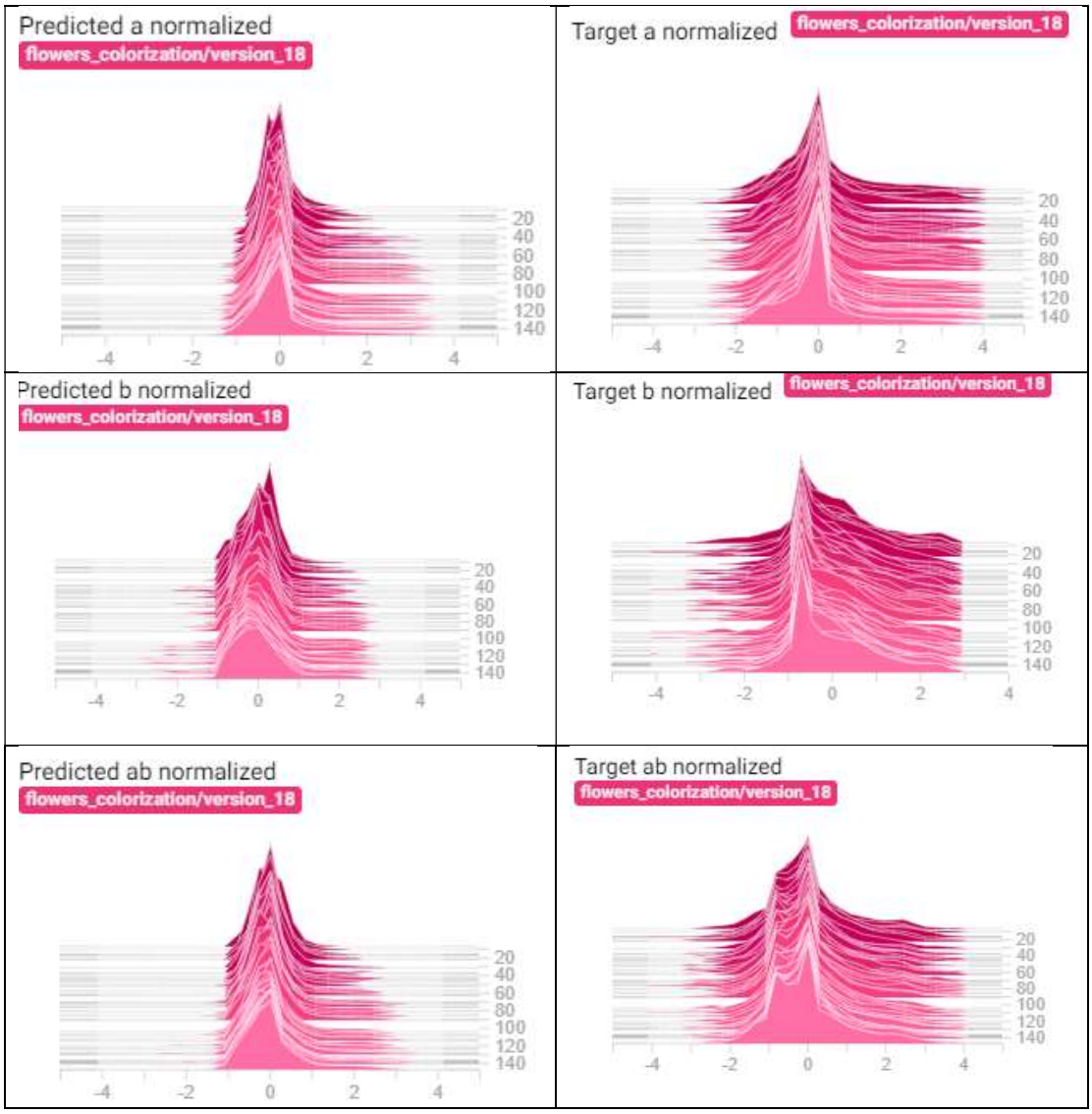


Train loss vs batch number



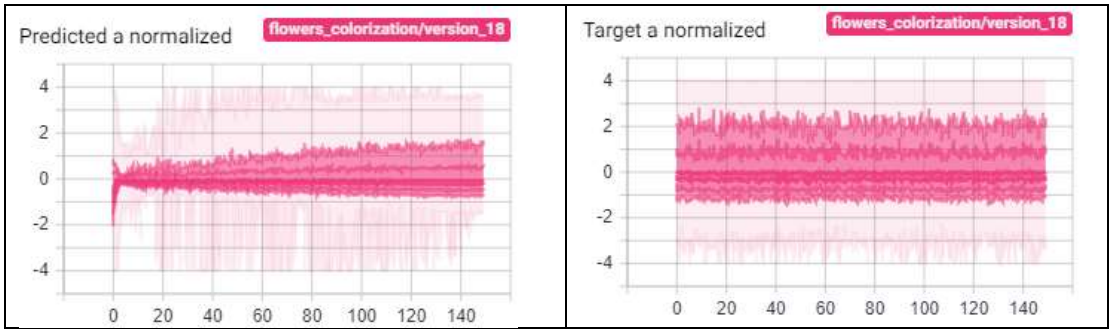
Validation loss vs batch number

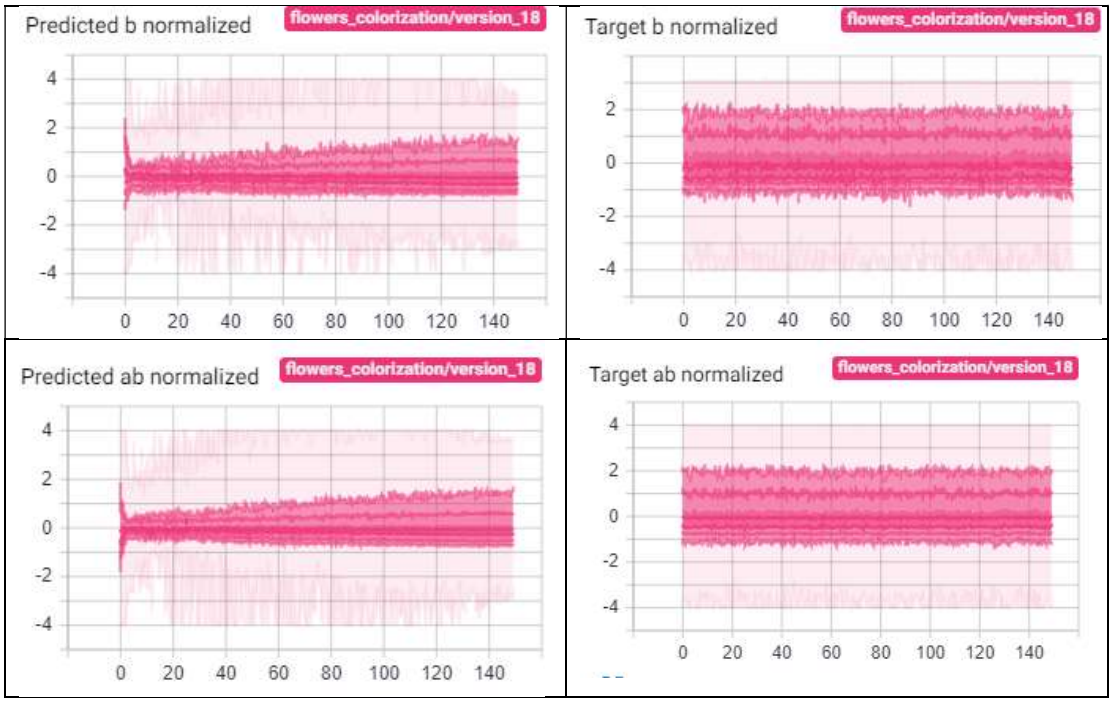
Below are presented distributions of normalized a and b channels for predicted and target images for subsequent epoch from training process. Distributions were generated on batch level. The x axis represent normalized a and b channel values and y axis represent epochs



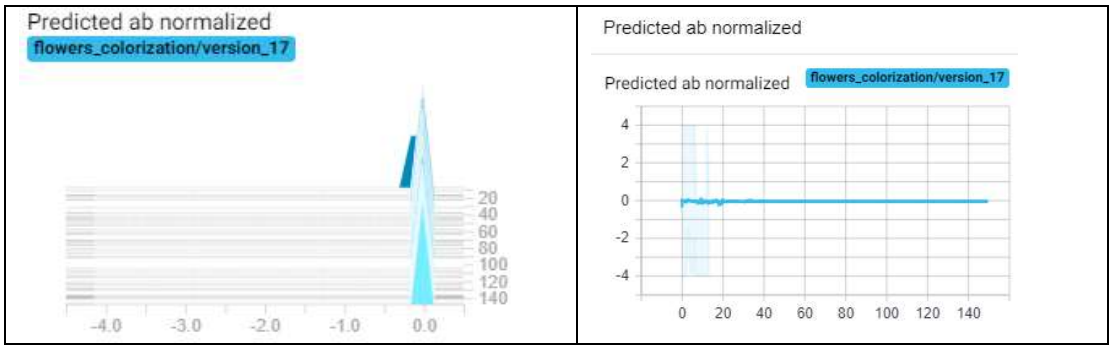
Visually predicted values seam to resemble target values. In both images clear spike at 0 point is visible. Predicted b channel has shape more similar to normal distribution whereas on target images is has clearly visible spike at around -0.8.

Let's see predicted and target distributions from another perspective. In images below x axis is epoch number and y axis is normalized a and b channel value. Each distribution is generated on batch level (for predicted and target images).





As seen above both normalized a and b channels for predicted images with subsequent epoch got more similar to target distributions. Target distributions as expected are stable thorough epochs. Each distribution is generated on batch level (for predicted and target images). Is worth noting initial shrinkage of predicted distributions at the beginning of the training. This presents rapid change of weights. Randomly set weights adjust to perform specific roles (e.g. form specific filters that detect edges, colors, shapes etc.). This rapid change was done at low learning rate levels beginning from 1e-5. This a concept of so called learning rate warm-up. Other training experiments with larger initial learning rate showed that model weights changes too much that training process didn't result in model convergence. Example of such instable training with too high initial learning rate is presented below. As seen predicted values are grouped around 0. Training through 150 epoch did not result in finding optimal minimum.



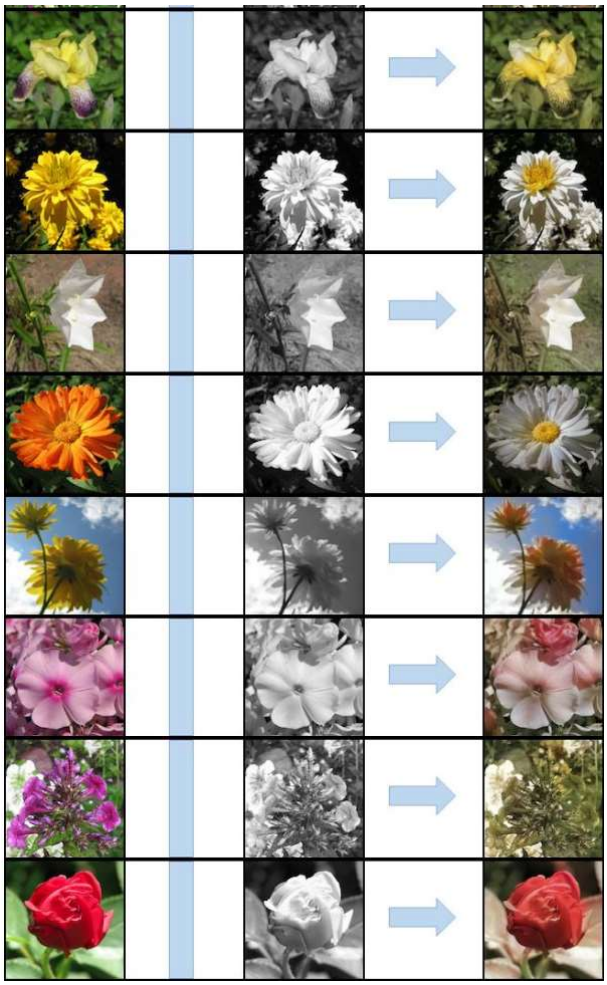
Test images grand truth:

Predictions of model after each fifth epoch are presented below.

First epochs	Subsequent epochs
	

Results in form of animated gif are presented below. Each slide presents prediction after subsequent 5 epoch. The first row are images predicted by model (for epochs 0, 5, 10, … 145). The second row displays target images.

Other sample results from test set are presented below (images were randomly selected). In first column are original images, in second column grey scale images, in third, outcome of model colorization.

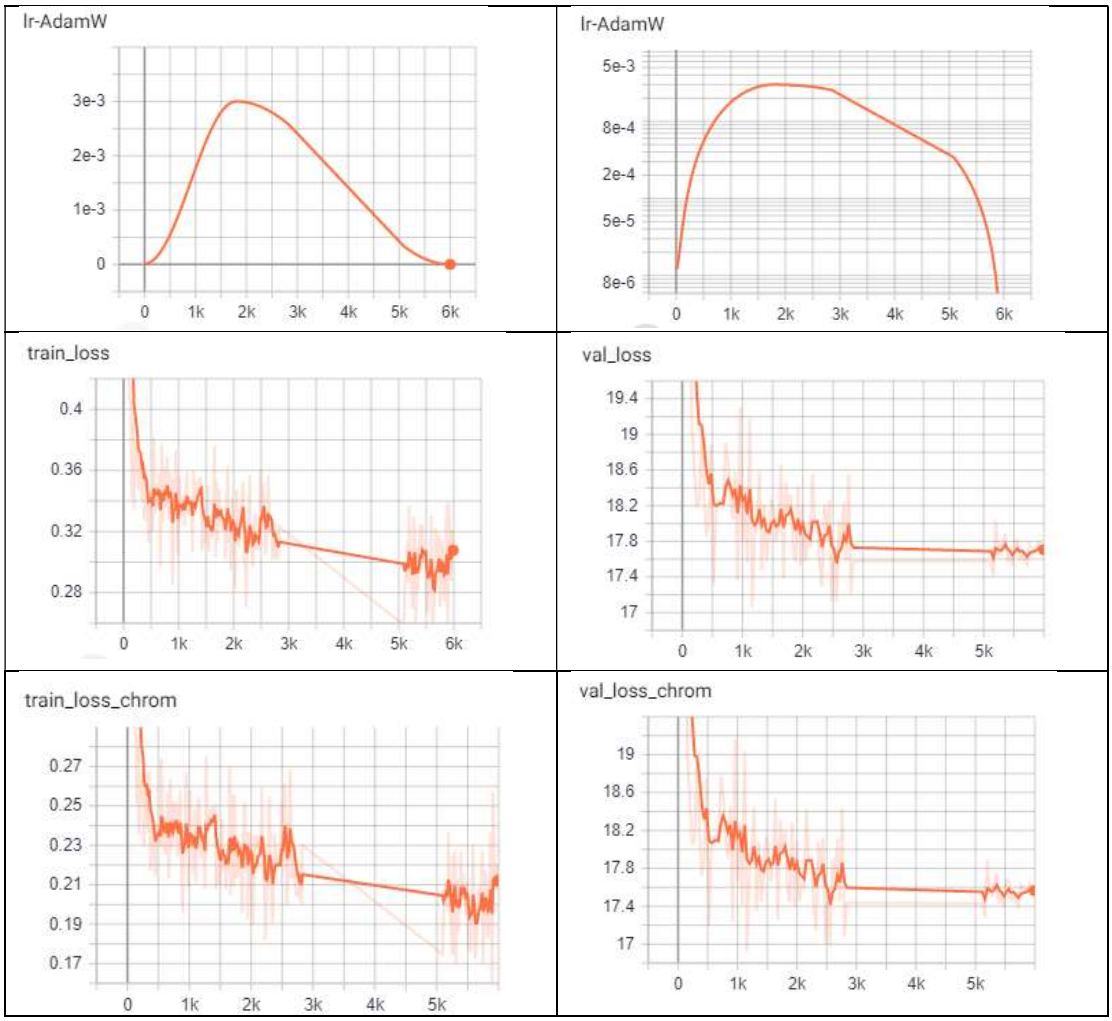


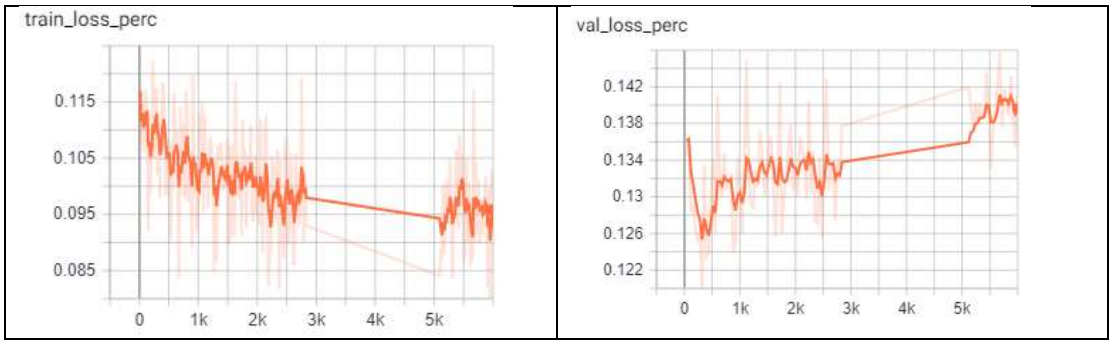
5.3 Training Colorization model with Patch Match algorithm

Colorization model with Patch Match algorithm was performed on 150 epochs. Please note that this means there were 50% of total number of batches had chrominance batches and 50% had perception batches, whereas during training colorization model without Patch Match algorithm had 100% chrominance batches.

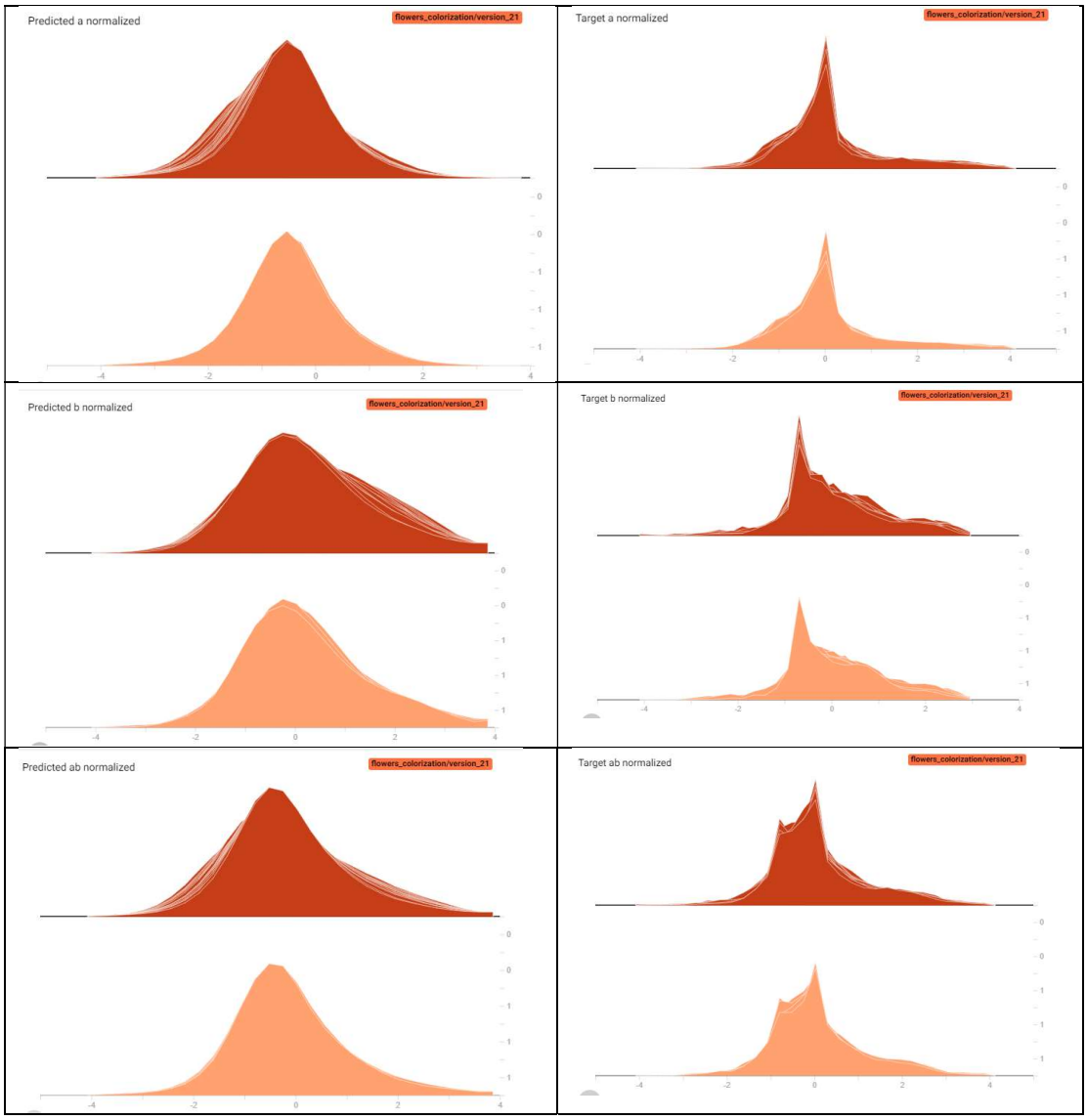
Training was performed on 150 epoch using single V100 GPU on Google Colab. It took approximately 53 hours to train.

Below is presented learning rate (in normal and log scales), training and validations losses (including chrominance and perception parts). Unfortunately values between 70 and 127 of training process were not saved due to Colab session restart.

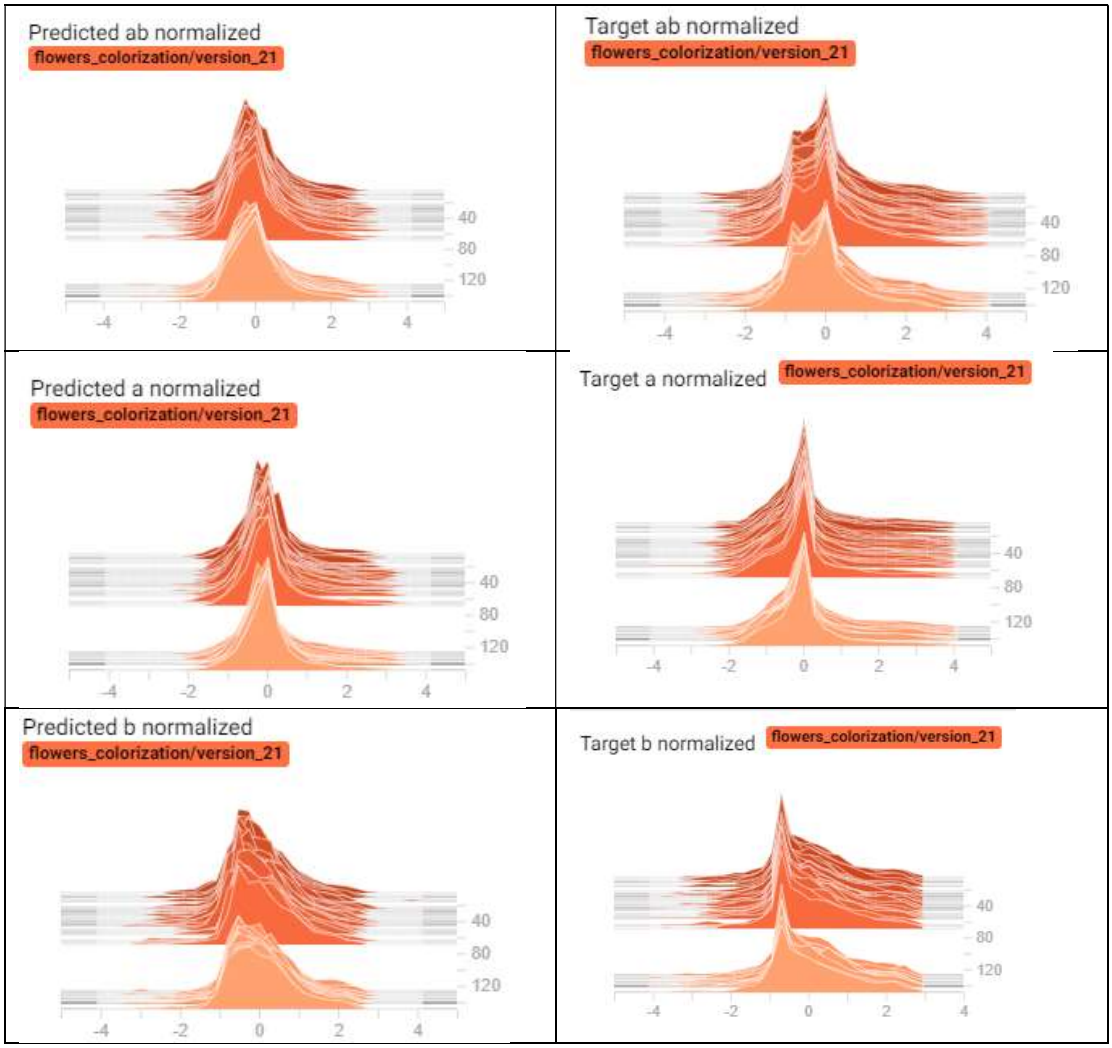




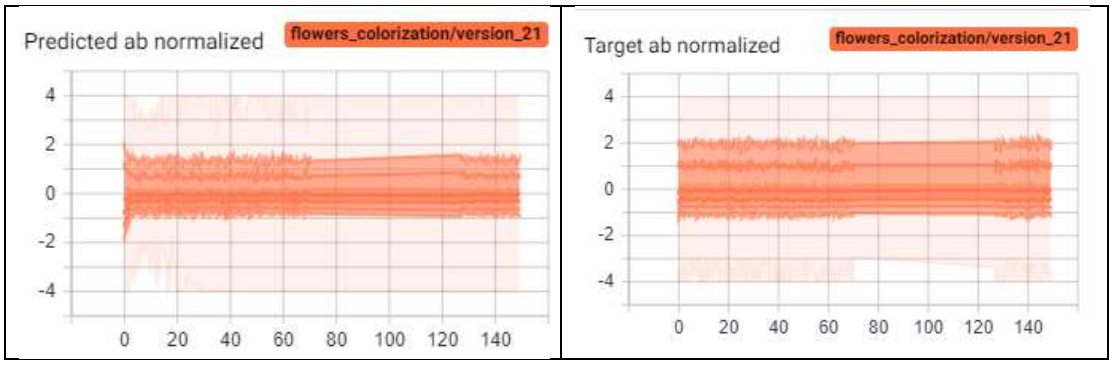
Below are graphs from just of the beginning of training process where network results are random (because of randomly initialized weights). There is a clearly visible misalignment between predicted and target CIE a,b channels distributions.

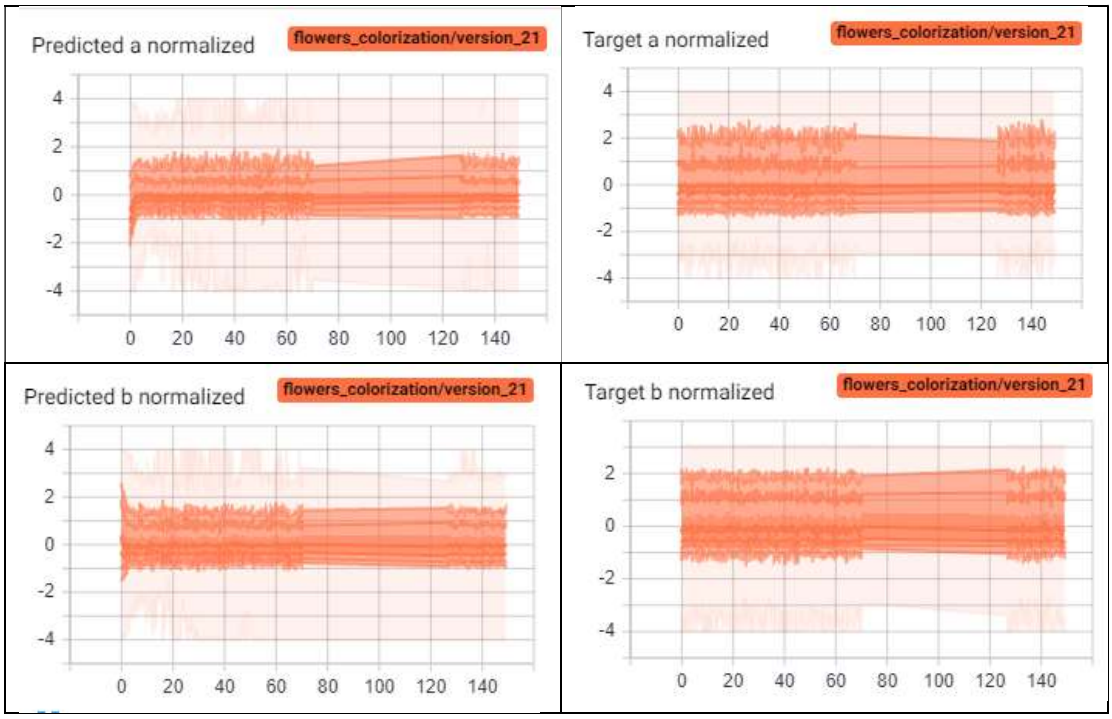


Distributions of predicted and target CIE a,b colors for the whole training process are presented below. It is worth to compare them to just beginning of the training (see plots above). It is visible that distributions of predictions aligned with target distributions, but it is not 100% alignment.

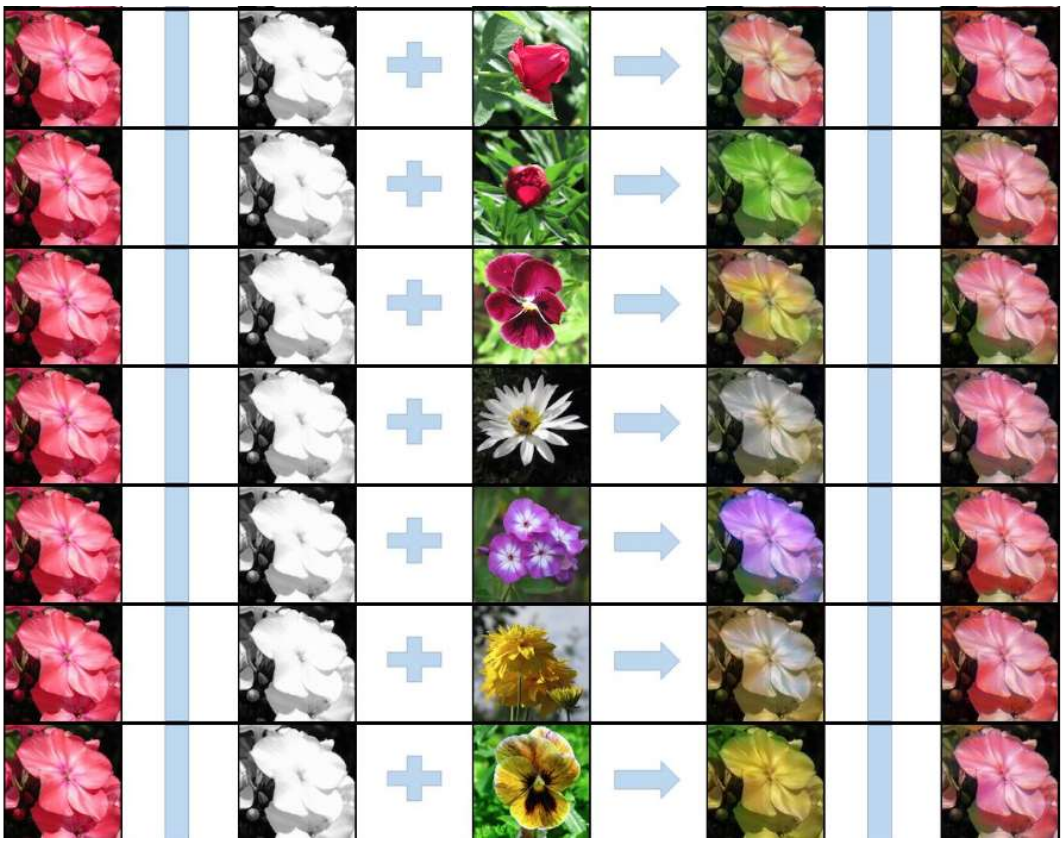


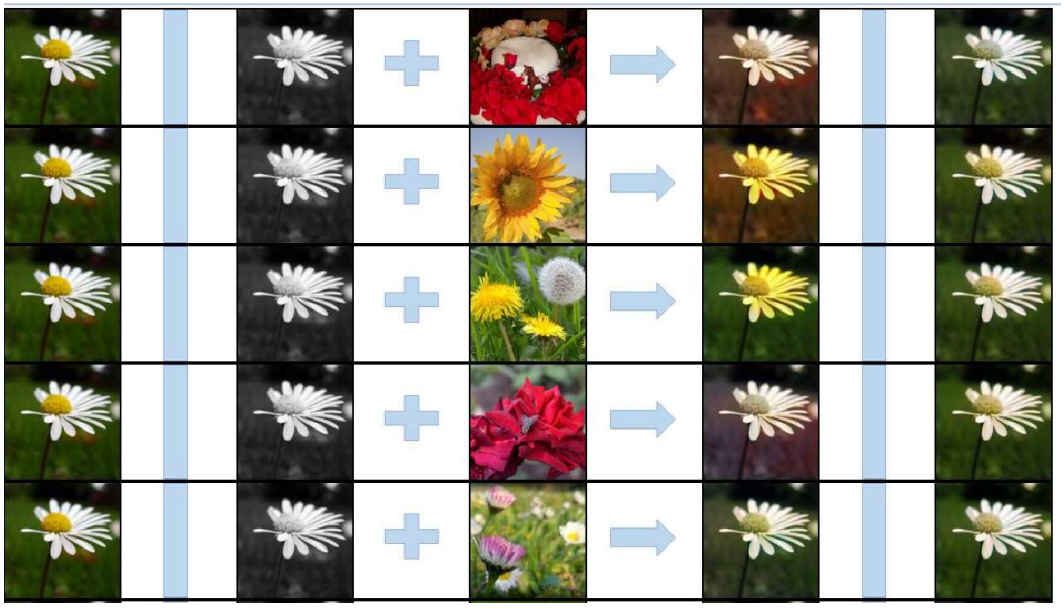
Below distribution of a, b colors is presented in another view (x axis represents number of epochs).





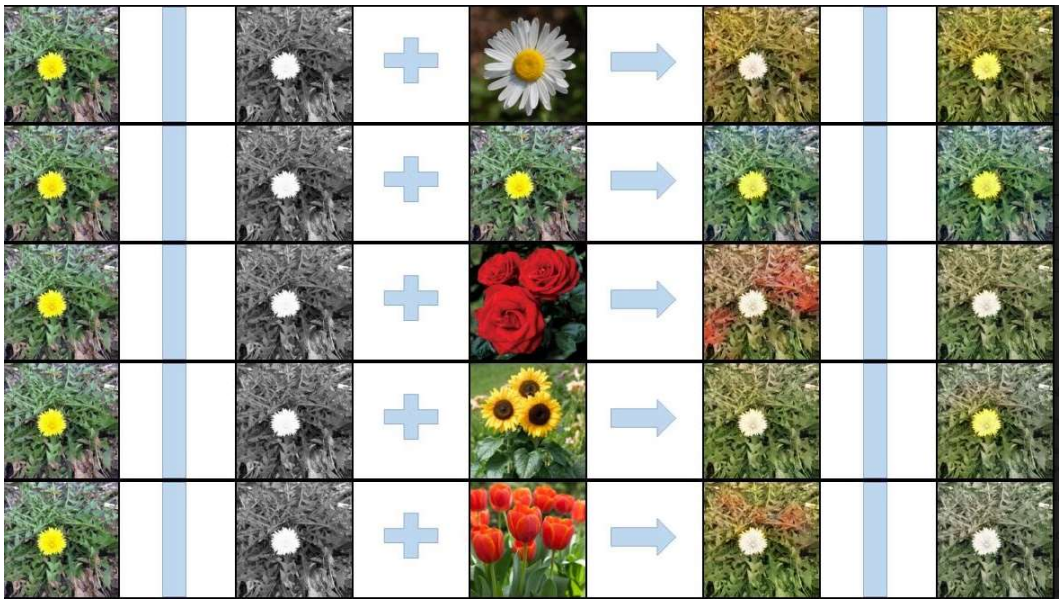
The results of trained model are presented in image below Description of columns: first column – target image, second column – greyscale target image, third – reference image, forth column – colorization by model, fifth column – colorization using also colors from target image.



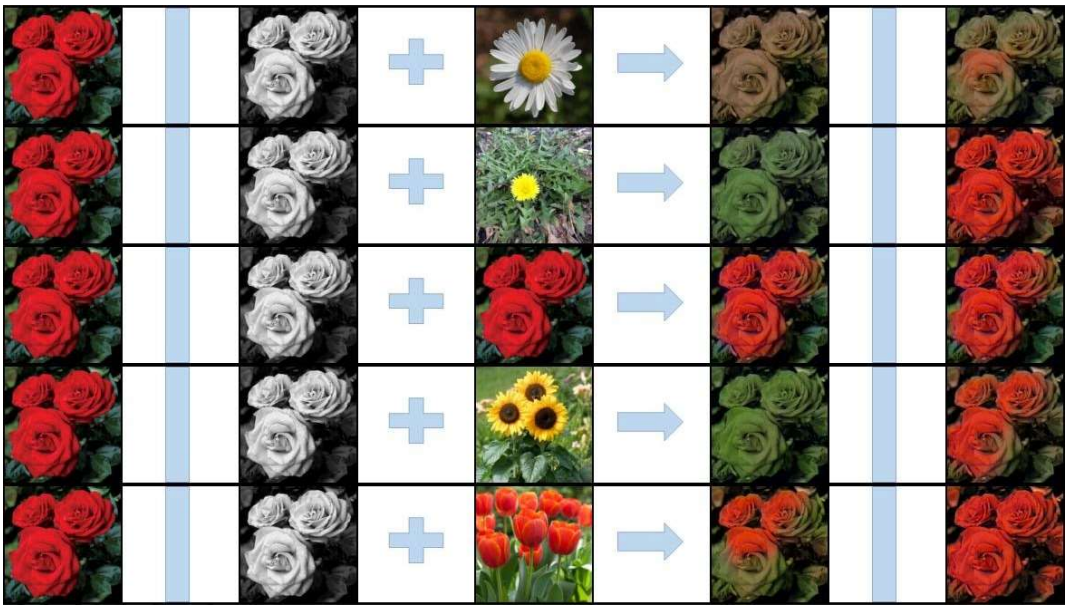


Some fail cases:

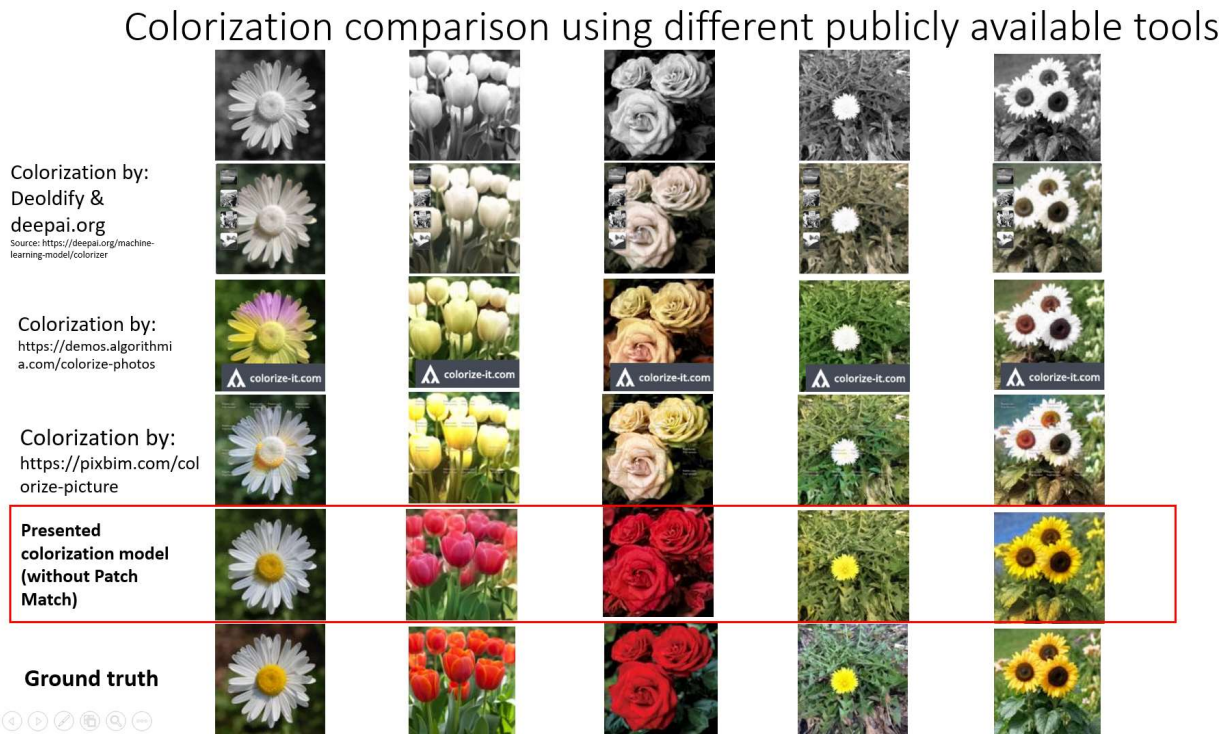
Fail cases appear when image has many fine grain flowers or leaves. It seem that model cannot clearly decide how to approach colorization for that kind of images:



Collection of roses appears to be problematic. From semantic point of view it seems that model treats roses like leaves (thus tends to color roses using green color from background leaves).



Results of the colorization model (version without Patch Match algorithm) were compared with other publicly available colorization methods. The results obtained from Colorization model (trained without Patch Match) visually are the most naturally looking among compared methods. Please note that Colorization model was trained only on flowers data set, whereas other methods are more general colorization methods thus they do not necessary perform as well on flowers data set.



5.4 Conclusions

This thesis presents model for multimodal colorization problem. Presented model clearly shows that using deep learning models multimodal colorization is possible without human clues. For images with single large flower in the center model works remarkably producing naturally looking colorizations. Weak points of the model are visible when there are many fine grained flowers in the image. In those situations model cannot clearly assign colors to fine grained flowers and conditional colorization results in lack of color diversity. There were also observed some problems with colorization of buckets of roses and tulips. From semantic point of view it seams that model confuses petals of roses with green leaves thus some of the colorization of roses results in green color. Comparison with other publicly available colorization models showed that the proposed model produces the most naturally colorizations on flowers data set among compared methods.

Bibliography

[1] Minh Ha Quang, Sung Ha Kang and Triet M. Le, "Image and Video Colorization Using Vector-Valued Reproducing Kernel Hilbert Spaces," *Journal of Mathematical Imaging and Vision,* vol. 37, p. 49–65, 12 02 2010.

[2] T. Horiuchi, "Colorization algorithm using probabilistic relaxation," *Image and Vision Computing,* vol. 22, pp. 197-202, 01 03 2004.

[3] Alexander Balinsky and Nassir Mohammad, "Sparse natural image statistics and their applications to colorization and compression," in *2010 IEEE International Conference on Image Processing*, Hong Kong, China, 2010.

[4] Bo-Wei Chen, Xinyu He, Wen Ji, Seungmin Rho and Sun-Yuan Kung , "Large-scale image colorization based on divide-and-conquer support vector machines," *The Journal of Supercomputing,* p. 942–2961, 02 04 2015.

[5] A. Levin, D. Lischinski and Y. Weiss, "Colorization using Optimization," in *ACM SIGGRAPH 2004 Papers*, New York, USA, 2004.

[6] R. Irony, D. Cohen-Or and D. Lischinski, Colorization by Example, Eurographics Symposium on Rendering: The Eurographics Association, 2005.

[7] F. M. Carlucci, P. Russo and B. Caputo, "Deep Depth Colorization," Rome, Italy, 2018.

[8] S. Yoo, H. Bahng, J. Lee, J. Chang and J. Choo, "Coloring with limited data: Few-shot colorization via memory augmented networks," in *IEEE Conference on Computer Vision and Pattern*, 2019.

[9] P. Vitoria, , L. Raad and C. Ballester, "Chromagan: Adversarial picture colorization with semantic class distribution," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020.

[10] P. Sangkloy, J. Lu, C. Fang, F. Yu and J. Hays, "Scribbler: Controlling deep image synthesis with sketch and color," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[11] M. He, D. Chen, J. Liao, P. V. Sander and L. Yuan, "Deep exemplar-based colorization," *ACM Transactions on Graphics,* vol. 37, no. 4, p. 47, 2018.

[12] O. Ronneberger, P. Fischer and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," Freiburg, Germany, 2015.

[13] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2015.

[14] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," 2017.