Grayscale Portrait Colorization with Neural Networks

Szürkeárnyalatos Portré Színezés Neurális Hálókkal

Varjas István Péter, Vitanov George

Abstract—English and Hungarian

Englis: We chose grayscale image colorization as our task. To simplify the problem, we decided to focus on colorizing human faces. We used an IMDB dataset of portraits, about 240 000 images as our training and testing data. To better suit our needs, we converted all images into LAB colorspace, and rescaled them to 128x128 pixels. We used Unet architecture as our neural network. The first five layers make up the encoder, consisting of 2D convolutional layers going from (128x128x1) to (8x8x512). Our decoder consisted of up sampling, 2D convolution, and concatenate layers, resulting in our (128x128x2) 2 color channel output. We used the concatenate layers, to connect up the encoder layers directly with the corresponding decoder layers. Both in the encoder and decoder we used ReLu activation, and batch normalization between each layer of neurons, and MAE then later on SSIM as loss functions. With this structure, we achieved decent results after 20 epochs on our database, after which we did not find any improvement. Our network currently is capable of accurately recognising human faces, and coloring them realistically, including facial characteristics, such as lips, eyebrows, and beards. The network can also distinguish between different hair colors, although sometimes mistaking gray hair for bright blond. We also achieved the realistic colorizations of different ethnicities, for example we can colorize afro Americans and Europeans accurately in the same image too. The shortcoming of our network comes in colorizing the background of the portraits. We attributed this to the very diverse artificial backgrounds contained in our dataset, thus not representing any pattern which the network could learn.

Hungarian: Feladatunknak szürkeárnyalatos képek kiszínezését választottuk. A probléma egyszerűsítése végett azt tűztük ki célul, hogy emberi portrék élethű kiszinezését valósítsuk meg. Adatbázisként egy IMDB portrékból összeválogatott adathalmazt használtunk, melyből 240 000 használható képet sikerült kiszűrnünk. A jobb kezelhetőség érdekében a tanító és teszt adatainkat LAB színtérbe konvertáltuk, és átméreteztük 128x128 pixel méretűre. A feladat megvalósításához a Unet nevű neurális háló struktúrát alkalmaztuk. Encoderként 2D konvolúciós rétegeket használtunk (128x128x1) pixeles bemenettől egészen (8x8x512) konvolúcóig elmenve 5 lépésben. Dekóderként upsampling, 2D konvolúciós, és összekapcsoló rétegeket használtunk, melyek (128x128x2) kétféle szín réteg kimenetet adnak. Mind az enkóderben, mind a dekóderben ReLu aktivációs függvényt használtunk, és batch normalizációt rétegenként. A költségfüggvény kezdetben átlagos abszolút hiba volt, majd 24 epoch után a képek minőségének értékelésére kifejlesztett Structural Similarity (SSIM) metódust alkalmaztuk. Ezzel a struktúrával kielégítő eredményeket értünk el 20 tanítási epoch után, melyet követően már nem javult a háló teljesítménye. A hálónk pontosan felismeri az emberi arcokat, és élethűen kiszínezi a bőrfelületeket, az ajkakat, a szemöldököket, és szakállat. Továbbá a háló meg tudja különböztetni az eltérő hajszíneket, habár az ősz hajat néha világos szőkének véli. Ezen kívül különböző etnikumból származó embereket is pontosan tud kiszínezni, például afro-amerikaiakat és európaiakkal egy képen belül is valósághűen színezünk. A neurális hálózatunk gyengesége a portrék hátterének kiszínezésében rejlik. A hiba forrását az adatbázisunkban található rengeteg különböző mesterséges háttérnek tulajdonítjuk, mivel így nincs semmilyen minta a hátterekben melyre a hálónk rá tudna tanulni.

Index Terms—Neural Network, GAN, General Adversary Networks, Image Colorization, Portrait

1 Introduction

THE abstracts already summarised what we have done, so first lets rewiev what this paper contains:

- Previous solutions
- The structure of our Neural Network
- Execution:
  - Data collection
  - Training
  - Evaluaton, and hyperparameter optimalization
  - Testing
- Possible Improvements
- Summary

2 Previous Solutions

We found numerous approaches and solutions to the grayscale image colorization problem, but none were specific to portraits, thats why we decided to build one. Bellow are some solutions wich we examined:

- Colorful Image Colorization
- Deep Colorization
- Image-to-Image Translation with Conditional Adversarial Networks
- Image Colorization with Generative Adversarial Networks
- Image Colorization Using a Deep Convolutional Neural Network
- A learning-based approach for automatic image and video colorization
- Convolutional Neural Network based Image Colorization using OpenCV

Some approaches use colour labels, preset by humans on the image, to apply the selected colour automatically to the segmented area /cite. We wanted a more autonomous solution without interaction so we decided against this.

An other similar approach is, to interpret the colorization problem first as a segmentation problem, then segment colorization. /cite We wanted a simpler solution, with the segmentation and colorization not separated as tasks. There are also solutions which use general pretrained networks on image classification, coloring, and segmentation problems, but we chose a more specific problem to leverage its special characteristics, and using this approach would have eliminated this advantage.

We adopted two approaches from the listed solutions: we used a convolutional Neural Network for colorization, and experimented with further training it, as a generator network in a GAN architecture.

3 The structure of our neural network

We based our colorized image generating network on the Unet architecture. This neural network arhitecture consists of an encoder part and a decoder part. During the encoder part we use 2D convolutional layers. During the decoder, we use 2D up sampling with concatenate layers, and convolution after each concatanate. The concatenate layers join each encoder layer to the corresponding decoder layer, to later apply a 2D convolution to them. We used ReLu as activation in each layer, and used batch normalization after each layer. We setteled on Adam as an optimizer, and sigmoid as the output layer activation function. After the last convolutional layer, we used one more concatanate layer, to join the grayscale data, to our two predicted color layers.

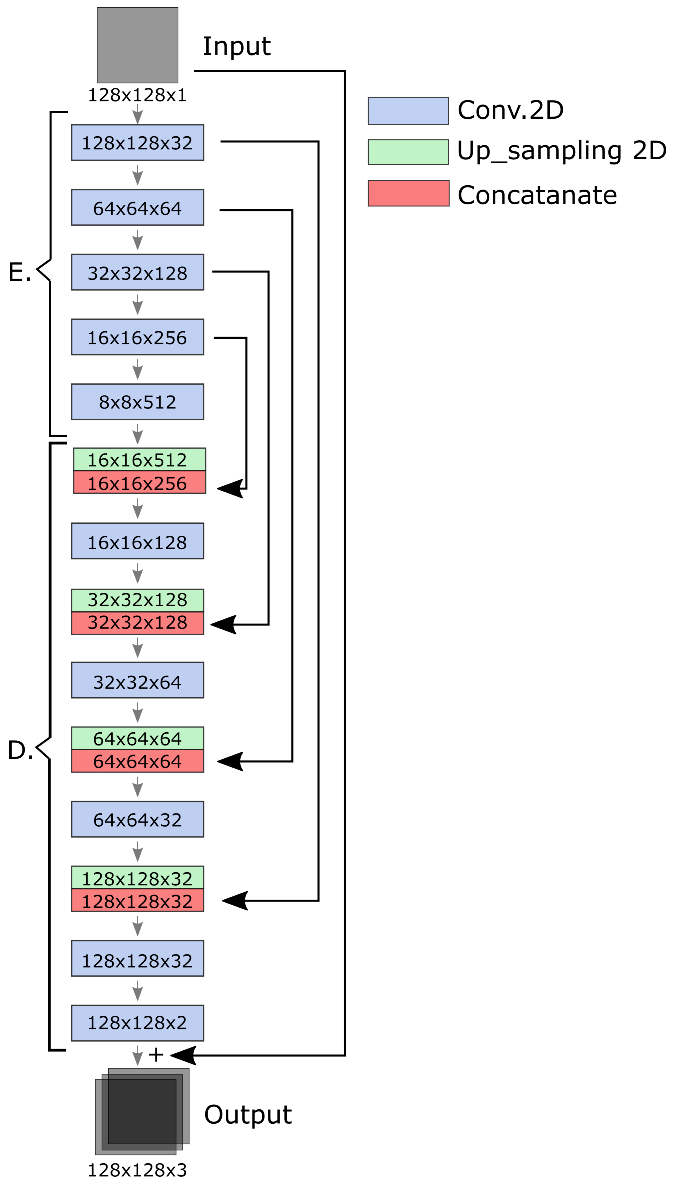

Figure 1: Unet Neural Network structure

S

4 Execution

4.1 Data Collection

We used an IMDB dataset of portraits, about 240 000 images as our training and testing data. To better suit our needs, we converted all images into LAB colorspace, cropped and rescaled them to 128x128 pixels. The LAB colorspace consist of 3 channels, one grayscale channel, and two colour scale channels. This simplifies the needed neural network, because we can give the grayscale channel to the network as the input, and expect only the two colour channels as the output. For fast data processing, but also optimised storage of our dataset, we stored our data in a hdf5 format using the h5py python library. We stored the LAB pixel values of the images in int formats instead of floats to save space, and did the required normalization on our data, when loaded by the generator function. We scaled our input data to 0-1 scale, because we used ReLu as activation in our neural network, and it would have altogether ignored negative input values. Our dataset also contained faulty images, and grayscale ones, which we had to filter out. As a result of this, from around 500 000 images, we managed to use 240 000 as our dataset.

Figure 2: Faulty images reduced our dataset

4.2 Training

We started training our network by running test manually with different parameters, and applying the settings that produced the best results. First we started using Mean Absolute Error as out loss function. This produced good results, but after a while our network did not improve further.

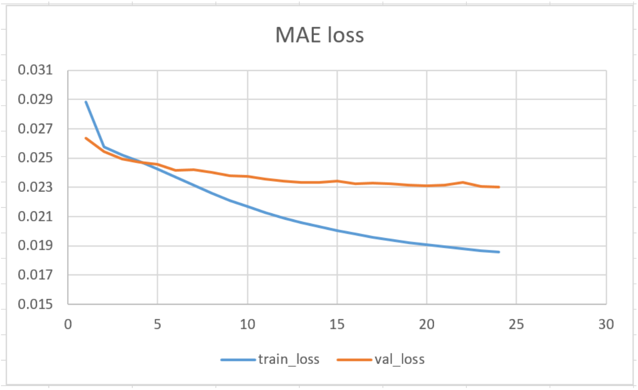

Figure 3: MAE loss function learning rates

To further improve our neural network, we decided to switch loss functions between training epochs, and continue training with the updated loss function. We used SSIM as the updated loss function. /citeSSIM

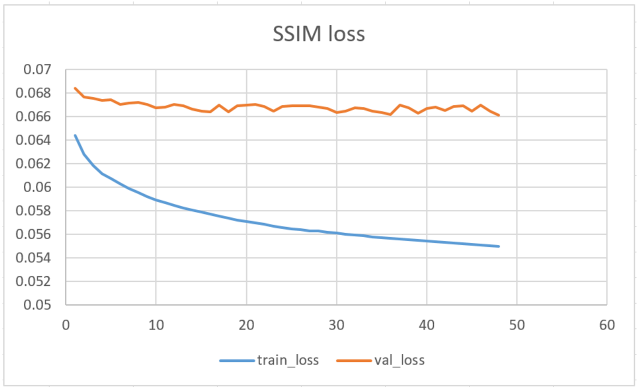

Figure 4: SSIM loss function learning rates

4.3 Evaluation and hyper parameter optimization

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.
[2] https://arxiv.org/abs/1511.08861