**Colorizing black and white pictures (without user interactions)**

Fekete-fehér képek színezése (felhasználói interakció nélkül)

## *Team - balit\_learning :*

## *Gurubi Barnabás - DXEXVR,*

## *Mátyás Gergely - IL21NI,*

## *Horváth Ákos - DKILK6*

**Abstract/Kivonat**

We approach the problem of colorizing black and white (gray scale) images. The colour of pixel is highly depends on the features of its neighbours, hence we use Convolutional Neural Networks (CNNs). Our work contains an Auto-Encoder type network which is not pre-trained. The model itself is fairly big, contains a huge amount of parameters therefore the training takes considerably long time. We train on 70 thousand images approximately. Our system gets these images in gray scale format and outputs the colour channels. Unlike the training, the generation of the coloured images from they pale version is rapid enough, especially compared to the training time. We propose our final solution to be automatic, so no user imput should be required. This problem contains huge uncertainty, due to its difficulty, for this reason our results are not perfect, nevertheless they are definitely worth to be considered.

A kitűzött probléma a fekete-fehér képek színezése. Egy képet alkotó pixelek színe nagymértékben függ a szomszédos pixelek tulajdonságától, ebből kifolyólag a megoldás során Konvolúciós Neurális Hálókat (CNNs) használunk. A munkánk egy Auto-Encoder típusú hálót dolgoz ki, ami nem tartalmaz előzőleg betanított részt. Maga a modell meglehetősen nagy, rendkívül sok paramétert tartalmaz, ezáltal a tanítása igen hosszú időt vesz igénybe. Hozzávetőlegesen 70 ezer képen tanítjuk a modellt. A rendszer fekete-fehér formában kapja a képeket bemenetként, és kimenetként visszaadja a szín csatornákat. A tanítással ellentétben a színes képek generálása az eredeti színtelen verzióból kellően gyors, főleg a tanuláshoz képest. A végső megoldás automata, vagyis nem igényel felhasználói beavatkozást, segítséget. Maga a probléma sok bizonytalanságot hordoz magában a nehézségéből kifolyólag, ennek következtében az eredmények nem tökéletesek, ennek ellenére mindenképpen hasznosnak találjuk a fontolóra vételüket.

**Introduction**

Lets take a look at the problem. Given a gray scale image, the prediction of the colours seems to be a rather fair or even a bit too complex task, taking the possible variates into consideration. Although observing the image more carefully, we can start to rely on our experiences since we know the original (or common) colour of lots of things. For example the sky during a bright day is blue, or trees used to be green (of course in certain conditions). Needless to say that many objects can have more than one possible colours, like clothes or other artifical entities. Our goal was to give a plausible prediction. Albeit these predictions often imperfect and incomplete, they have creditable parts also.

To give a comparison of how complex the problem is let’s examine how they colorize black and white frames during the colorization of a movie with the classic methods: The artist tipically begins by dividing the picture into regions and than assigning a color to each region. This method is called segmentation method and it is a very time consuming process that can take hours to finish. In contrast our network takes only a few seconds to colorize an image (ofc if we have already trained our network before) which is obviously not as good as if it had been colorized by an artist .

During our work we used a PC with 16 GB memory and a GTX 1070. This GPU is without question one of the strongest in the market that’s available and affordable for the common users nowadays. Even though we had this relatively strong setup the training still lasted a long time (see in detail at the part where we discuss the teaching process).

**Related work**

**Methods/Network architecture**

**Implementation**

**Getting the images and preprocessing them**

Since our goal was to colorize any pictures we needed a dataset that includes pictures from different themes. Because of the high complexity we also needed the dataset to be large. Although we read in many articles that they used around a million images, we were limited by our resources. We decided to use a dataset of ~70k 256x256 sized images, which we downloaded from from the site: <https://pixabay.com/> using a script:

**import** re

**import** requests

**from** bs4 **import** BeautifulSoup

**from** PIL **import** Image

**from** resizeimage **import** resizeimage

**from** io **import** BytesIO

**for** i **in** range(0, 5000) :

site **=** 'https://pixabay.com/hu/photos/?&pagi=' **+** str(i)

response **=** requests.get(site)

soup **=** BeautifulSoup(response.text, 'html.parser')

img\_tags **=** soup.find\_all('img')

urls **=** [img['src'] **for** img **in** img\_tags]

**for** url **in** urls:

filename **=** re.search(r'/([\w\_-]+[.](jpg|gif|png))$', url)

**if** ((**not** '.gif' **in** filename.group(1)) **and** (**not** '.png' **in** filename.group(1))):

**with** open(filename.group(1), 'wb') **as** file:

**if** 'http' **not** **in** url:

url **=** '{}{}'.format(site, url)

response **=** requests.get(url)

img **=** Image.open(BytesIO(response.content))

img **=** resizeimage.resize\_cover(img, [256, 256])

img.save(file, "JPEG")

print(i)

**Loading and formatting pictures**

So after we downloaded the database, we needed to load it to our program and format them. As we read in a lot of articles for this problem it is better if we change the usual RGB channels to Lab representation. (where L is for lightness, a and b are for the color spectrums green–red and blue–yellow) This way the L channel provides us the black and white pictures which represent the input and the network has to predict the a and b values. Also it is better for the network if the values are between 0 and 1 so we 'normalized' them. (we know that L is between 0-100 and a,b are between -128 - 128 )

**Teaching**

**Evaluation**

**Testing**

**Future plans/ Summary**

**References**