Deep Learning

Coloring grayscale images

Lior Danon
Tal Ladijensky

Contents

Introduction

The problem we are investigating is coloring black and white photos. It is a difficult problem because we will have to multiply the inputs shape by 3 in the output layer.
This problem is interesting mainly from 2 reasons.
1. black and white photos have been with us since the early 19th century, color photography became more common only from the mid-20th century. Most historical photos that we know and cherish are black and white photos like the liberation of Jerusalem on the 6-day war.
Coloring old photos will give us better understanding and help everyone learn better their history.

2. People today posses many photos, it is in their phones, in their computers, in their photos albums and we can even paste photos on wood Those images of us, our leaders, our loved ones, the nature, etc. can be damaged. We all know at least one blur or noisy image we wish to repair and our project intends to solve this problem.

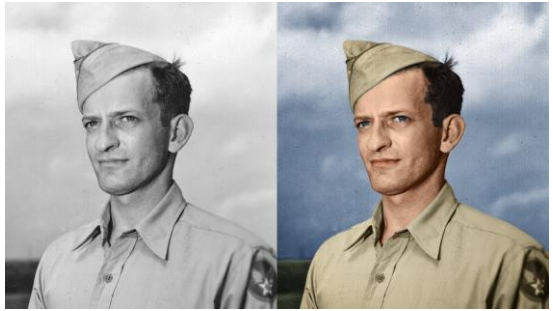
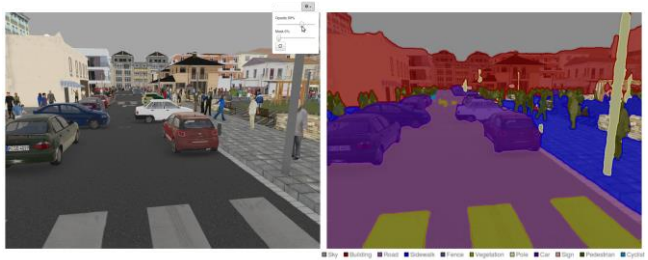


Image Segmentation

Image segmentation is the process of partitioning a digital image into multiple objects in order to classify the objects. The idea is to find each object and detect it. Image segmentation works by dividing the image to segments and make use of the important segments for processing the image. The usage of image segmentation will give us more meaningful results. It has multiple applications and we will use it for coloring the same object because of it's object detection



CNN Applications

Convolutional Neural Networks can help solve many problems such as:

- Pattern classification
- Categorizations
- Function approximation
- Optimization
- Control on the inputs and the outputs

In the field of deep learning, CNN are found to give the most accurate results in solving real world problems such as computer vision and natural language processing.

How to solve the problem using DL

Deep learning can help us to solve the problem.

Neural networks are a proven tool for image recognition and image processing.

Convolutional Neural Networks identifies objects in images.

It can search for content in the images and act on the content it has found.

It can analyze the shape of an object and by doing this it would extract the wanting features we need.

In a CNN layer each convolution window focuses on one part of the image. For example, in our case of animal images, one group could identify the head, another the legs, another the tail etc.

The group tried to solve the problem by using a deep learning network which dependents on convolutional layers.

We examined the published papers on this subject and created networks influenced by the academic work created in the subject.

Our dataset is approximately 25,000 pictures which will be a combination of different animal photos found online which were shuffled and mixed to 2 groups.

80% of the data will be subjected to the train folder.

20% of the data will be subjected to the test folder.

Datasets

We used Kaggle (https://www.kaggle.com/) to find our images datasets.

Our dataset is the official “Scene classification” of a wide range of natural scenes.

Architecture

The difficulty of the problem is that the output layer is three times the size of the initial input layer.

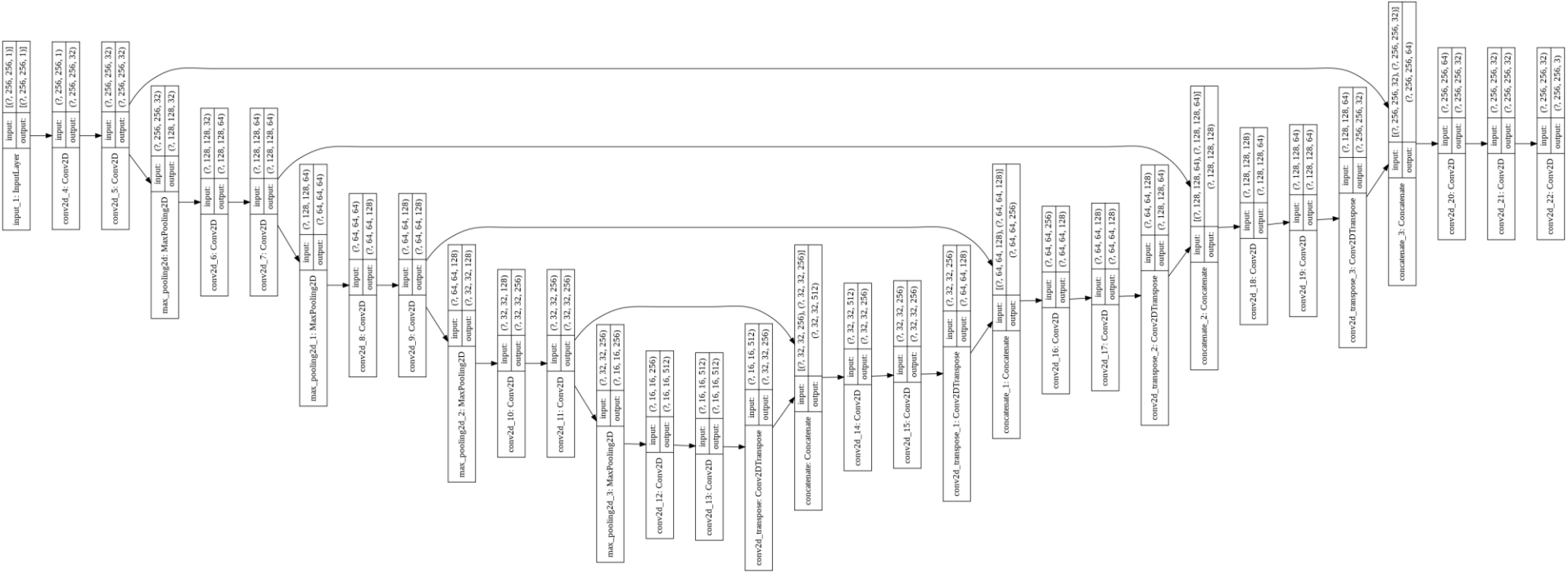
In order to solve this problem we have implanted our U-net architecture.

This U shaped architecture presented the best results in 10 different solutions we have tried.

The architecture of out own U-net:

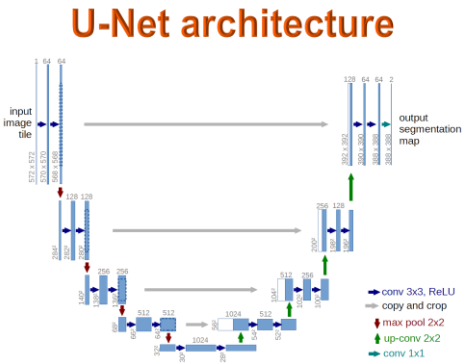
```
image_size = (256, 256)  # power of two recommended for downsampling

inputs = Input(image_size + (1,))
conv1 = Conv2D(32, (3, 3), activation='relu', padding='same')(inputs)
conv1 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv1)
pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
conv2 = Conv2D(64, (3, 3), activation='relu', padding='same')(pool1)
conv2 = Conv2D(64, (3, 3), activation='relu', padding='same')(conv2)
pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
conv3 = Conv2D(128, (3, 3), activation='relu', padding='same')(pool2)
conv3 = Conv2D(128, (3, 3), activation='relu', padding='same')(conv3)
pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
conv4 = Conv2D(256, (3, 3), activation='relu', padding='same')(pool3)
conv4 = Conv2D(256, (3, 3), activation='relu', padding='same')(conv4)
pool4 = MaxPooling2D(pool_size=(2, 2))(conv4)
conv5 = Conv2D(512, (3, 3), activation='relu', padding='same')(pool4)
conv5 = Conv2D(512, (3, 3), activation='relu', padding='same')(conv5)

up6 = concatenate([Conv2DTranspose(256, (2, 2), strides=(2, 2), padding='same')(conv5), conv4], axis=3)
conv6 = Conv2D(256, (3, 3), activation='relu', padding='same')(up6)
conv6 = Conv2D(256, (3, 3), activation='relu', padding='same')(conv6)
up7 = concatenate([Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(conv6), conv3], axis=3)
conv7 = Conv2D(128, (3, 3), activation='relu', padding='same')(up7)
conv7 = Conv2D(128, (3, 3), activation='relu', padding='same')(conv7)
up8 = concatenate([Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(conv7), conv2], axis=3)
conv8 = Conv2D(64, (3, 3), activation='relu', padding='same')(up8)
conv8 = Conv2D(64, (3, 3), activation='relu', padding='same')(conv8)
up9 = concatenate([Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same')(conv8), conv1], axis=3)
conv9 = Conv2D(32, (3, 3), activation='relu', padding='same')(up9)
conv9 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv9)
conv10 = Conv2D(3, (3, 3), activation='relu', padding='same')(conv9)
```

Visualization of our model: total trainable params: 7,760,355



U-Net:

U-Net is a network which segmentizes ‘small’ images.
It solves the problem of patch classification.
The innovation behind U-Net is the idea of encoder-decoder architecture is and the skip connection completes the loss of data created by the pooling layers.
The encoder part is the left part of the image and the decoder is the right side of the image.
The network is called “U-Net” because of their U shape.
Today, many architectures are inspired by it such as our network.



Activation Function-ReLU:

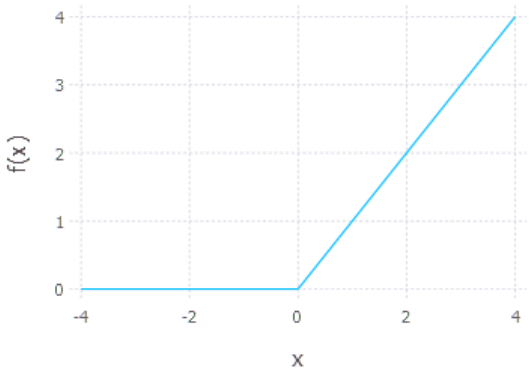
ReLU is the most used activation function today.

The function is nonlinear and can be easily backpropagated.

If the input is negative it will change to 0 and this neuron will not be activated.

It makes this function efficient and good for computation.

$$f(x) = x^{+} = \max(0, x)$$



Cost function-MSE:

MSE- the **mean square error** estimator measures the average of the square of the errors.

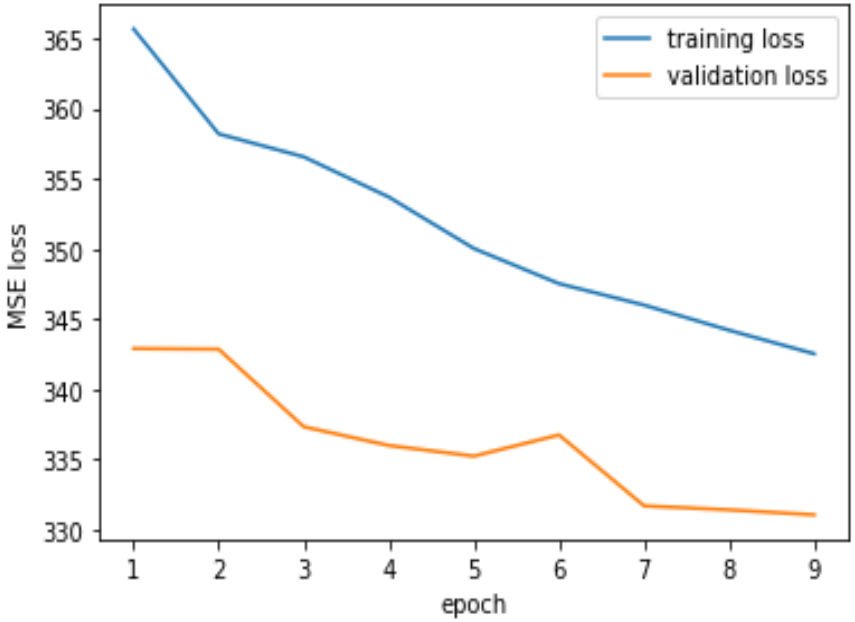
MSE is a non-negative function where values closer to zero are better.

MSE incorporates both the variance of the estimator and its bias.

Here we define the function as:

$$MSE = \frac{1}{n}\sum_{k=1}^{n}(\widetilde{Y}_i - Y_i)^2$$

Results (mathematics)

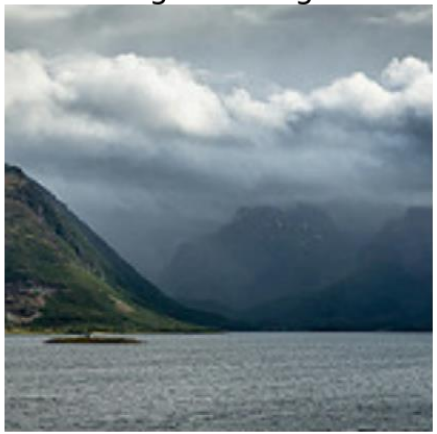


Results (visually)

Input Image

Model Prediction

Original Image



Results (visually)

Input Image

Model Prediction

Original Image

Results (visually)

Input Image

Model Prediction

Original Image

Conclusion

In this presentation, we tried to achieve the coloring of black and white images.

We used the innovation of U-net in order to create an end to end solution which does not involve post-preprocessing.

It seems that our results brings ‘life’ to grayscale images but further work is necessary.

As of today we did not improve known models and improvement is must.