Mid-Evaluation

Team 6/6

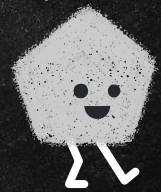
IMAGE COLOURING

Aritra Banerjee
2018102006

Jayati Narang
2018101066

Sahil Bhatt
2018111002

Srivathsan baskaran
2018101049

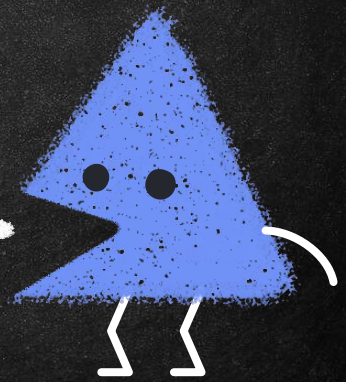


OVERVIEW

We are implementing "*Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification*".

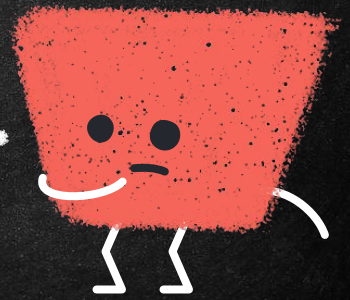
Paper link : Let there be Color!

The main objective of our project is develop a model capable of automatically colorizing black and white images.



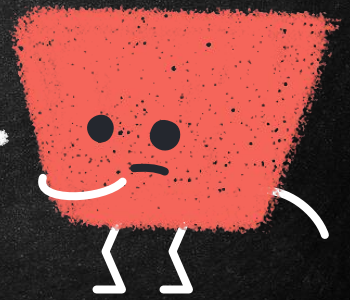
PROCEDURE

- This is a fully automated data-driven approach for colorization of grayscale images.
- In this, we use a combination of global image priors, which are extracted from the entire image, and local image features, which are computed from small image patches, to colorize an image automatically.
- Global priors provide information at an image level such as whether or not the image was taken indoors or outdoors, whether it is day or night, etc.



PROCEDURE

- Local features represent the local texture or object at a given location.
- The approach is to use CNN. The global and local features are extracted jointly and then fuse together to perform the final colorization.
- This method requires neither pre-processing nor post-processing.
- To predict the chrominance, CIE L*a*b colorspace is used.

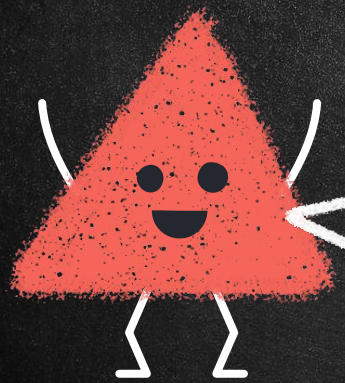


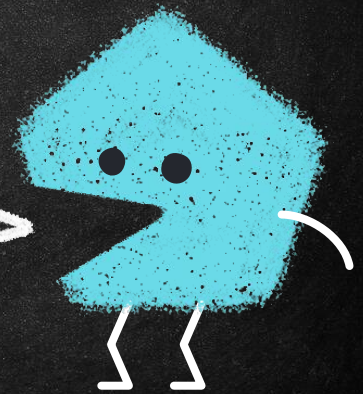
Dataset: Places365

Places365 is a dataset provided by MIT contains images from a list of 365 places. The total dataset size is roughly 21GB and it contains images organized into training and validation directories, which further contain directories named as labels for the place being shown in the images it contains.

Due to limitations wrt storage space, time and computing power, we consider only ten places for the purpose of the project.

As part of this evaluation, we have experimented with two of them - Botanical garden and Japanese garden.





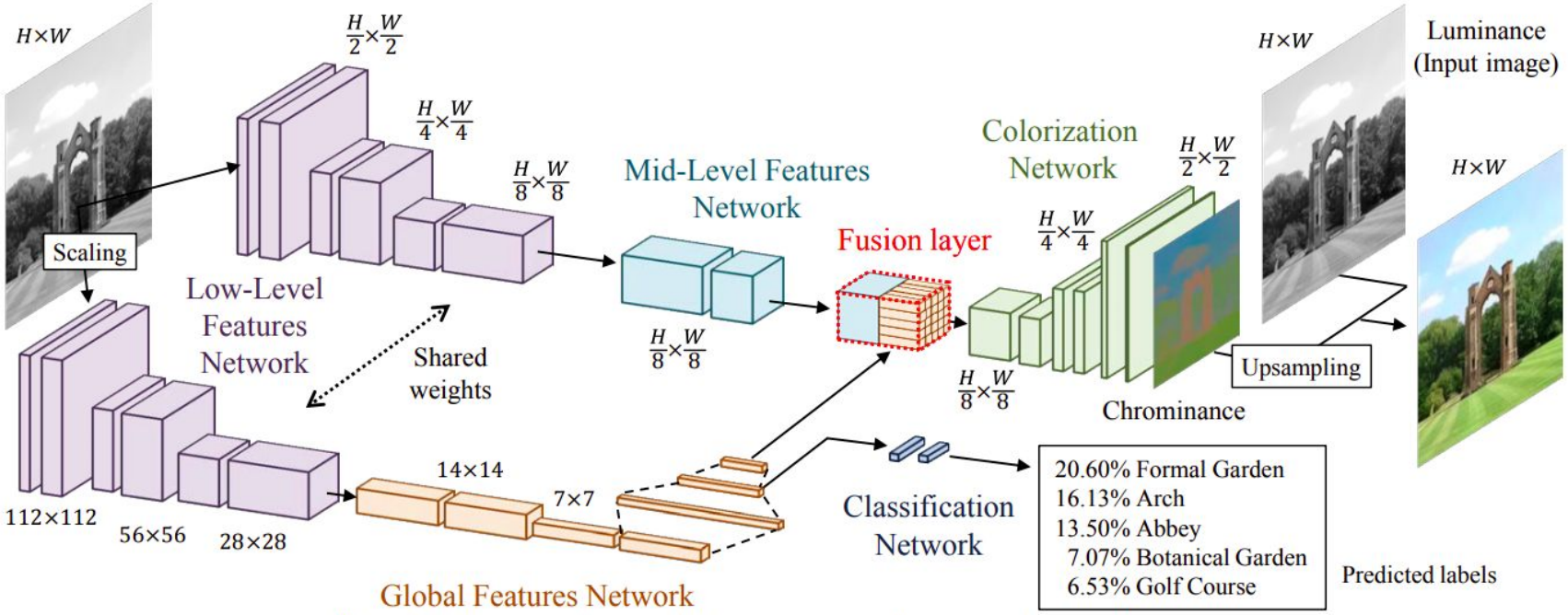


Figure 2: *Overview of our model for automatic colorization of grayscale images.*

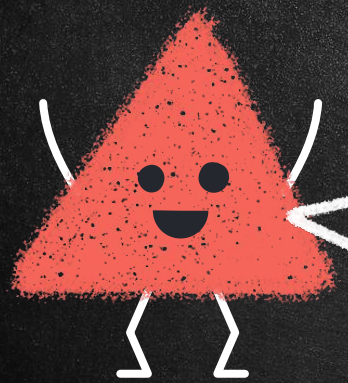
Table 1: *Architectures of the different networks used in our model. Fully-Connected (FC) layers refer to the standard neural network layers from Eq.* (1)*, convolution layers use Eq.* (2)*. The output layer consists of a convolutional layer with a Sigmoid transfer layer instead of a ReLU transfer layer. Outputs refers to the number of output channels for the output of the layer. Upsample layers consist of using the nearest neighbour approach to increase the resolution of the output by a factor of* 2*.*

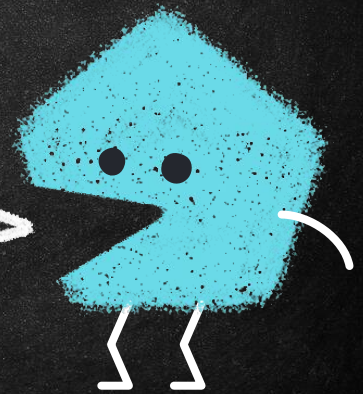
Image taken from the reference paper

GOALS ACHIEVED

We have implemented:

- Low-level feature networks
- Mid-level feature networks
- Global priors
- Fusion layer





METHODS IMPLEMENTED

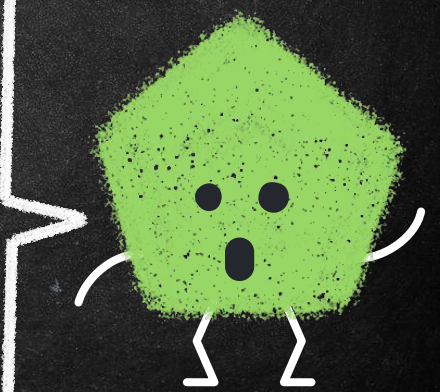
We have implemented following methods:

1. A Low-level features network:

We have created a 6-layer Convolutional Neural Network to obtain low-level features directly from the input image. The convolution filter is shared between the global features network and the mid-level features network. We have used 3x3 convolution kernels exclusively and a padding of 1 x 1.

```
class LowLevelFeatures(nn.Module):
    def __init__(self):
        super(LowLevelFeatures, self).__init__()
        self.conv1 = Conv2d(1, 64, 2)
        self.conv2 = Conv2d(64, 128, 1)
        self.conv3 = Conv2d(128, 128, 2)
        self.conv4 = Conv2d(128, 256, 1)
        self.conv5 = Conv2d(256, 256, 2)
        self.conv6 = Conv2d(256, 512, 1)

    def forward(self, x):
        out = F.relu(self.conv1(x))
        out = F.relu(self.conv2(out))
        out = F.relu(self.conv3(out))
        out = F.relu(self.conv4(out))
        out = F.relu(self.conv5(out))
        out = F.relu(self.conv6(out))
        return out
```



METHODS IMPLEMENTED

2. A Mid-level features network:

For this we have processed the low-level features further with two convolutional layers. The output is 256-channel mid-level features. The mid-level features networks are fully convolutional networks, such that the output is a scaled version of the input.

3. A Global features network:

To get a global image we have further processed the low-level features with four convolutional layers followed by three fully-connected layers resulting in a 256-dimensional vector

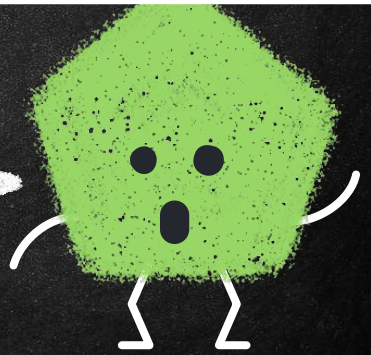
```
class MidLevelFeatures(nn.Module):
    def __init__(self):
        super(MidLevelFeatures, self).__init__()
        self.conv7 = Conv2d(512, 512, 1)
        self.conv8 = Conv2d(512, 256, 1)

    def forward(self, x):
        out = F.relu(self.conv7(x))
        out = F.relu(self.conv8(out))
        return out


class GlobalFeatures(nn.Module):
    def __init__(self):
        super(GlobalFeatures, self).__init__()
        self.conv1 = Conv2d(512, 512, 2)
        self.conv2 = Conv2d(512, 512, 1)
        self.conv3 = Conv2d(512, 512, 2)
        self.conv4 = Conv2d(512, 512, 1)

        self.fc1 = nn.Linear(7*7*512, 1024)
        self.fc2 = nn.Linear(1024, 512)
        self.fc3 = nn.Linear(512, 256)

    def forward(self, x):
        y = F.relu(self.conv1(x))
        y = F.relu(self.conv2(y))
        y = F.relu(self.conv3(y))
        y = F.relu(self.conv4(y))
        y = y.view(-1, 7*7*512)
        y = F.relu(self.fc1(y))
        y = F.relu(self.fc2(y))
        out = F.relu(self.fc3(y))
        return out
```



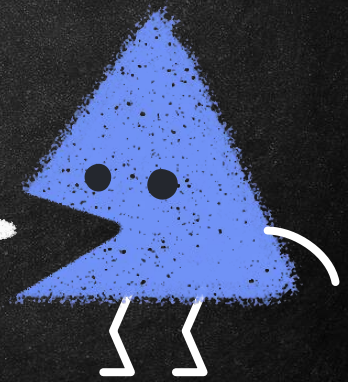
Hyperparameters

Hyperparameters:

- Number of epochs : 30
- Batch size: 32
- Learning rate (α) = 0.001
- Training set: 4096 (Botanical Garden) + 4096 (Japanese Garden)
- Validation set: 128 (Botanical Garden) + 128 (Japanese Garden)
- Test set: 96 (Botanical Garden) + 96 (Japanese Garden)

Total time taken (train,test and val): 32 minutes, 14 seconds.

You can find the best model given here.



Results

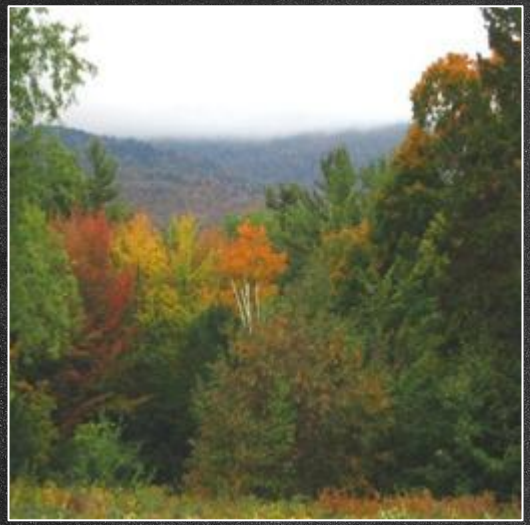
An image of a botanical garden from our test set

Colorized image obtained using our model

One can observe how it predicted the trees to be green and skies to be blue.

Note: Our dataset contains only colored images, these are converted to grayscale during training. However, the final model can be run using colorize.py file, in which the user must provide black and white images.

Results



An image of a botanical garden from our test set

Colorized image obtained using our model

Note: Our dataset contains only colored images, these are converted to black-and-white during training. However, the final model can be run using colorize.py file, in which the user must provide black and white images.

Results

A black and white image of a botanical garden randomly taken from the web

Colorized image obtained using our model

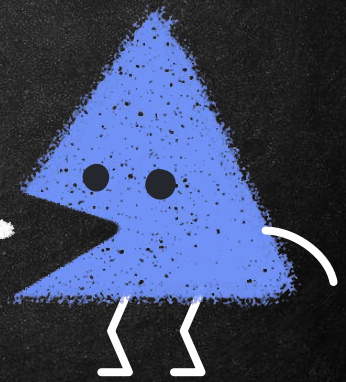
Results

A black and white image of a botanical garden randomly taken from the web

Colorized image obtained using our model

What's next ?

- Training our model on a larger set of images.
- Plotting loss graphs for training and validation (For this phase we just manually observed the losses after each epoch).
- Building the classification network.
- Trying new hyperparameter values.
- Obtaining intermediate results (if possible).
- Ablation study
- Comparison of our colorization method with our methods (if any)
- Basic web application to demonstrate the project (if time permits)



Thank you