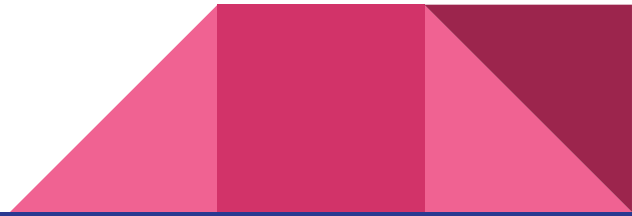
Automatic Image Colorization

Team 22

Problem Statement

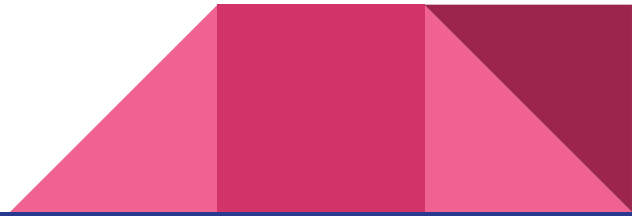
Image colorization is a difficult problem that often requires user input to be successful. Our algorithm automatically colorize black and white images of nature without direct user input.

Image colorization is the process of adding color to grayscale or sepia images, usually with the intent to modernize them. By hand, this can be a time consuming and repetitive process, and thus would be useful to automate.



Challenges

Colorization is fundamentally an ill-posed problem - two objects with different colors can appear the same on grayscale film. Because of this, image colorization algorithms commonly require user input, either in the form of color annotations or a reference image.

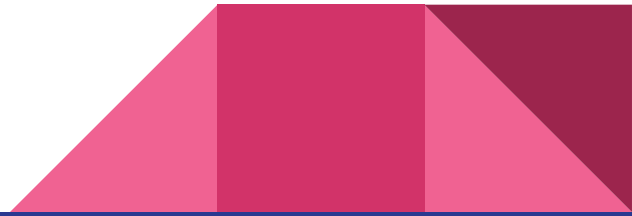


Overview

In order to define the problem, we represent images in the YUV colour space, rather than the RGB colour space.

In this colour space, Y represents luminance and U and V represent chrominance.

The input to our algorithm is the Y channel of a single image, and the output is the predicted U and V channels for the image.



Advantage of YUV over RGB

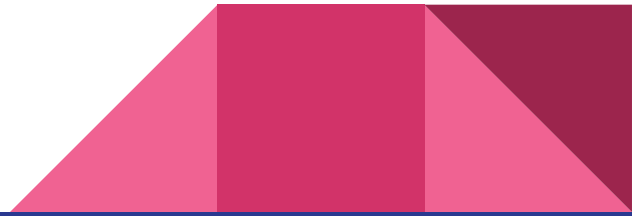
We already have the Y channel as input , we only need to predict the U and V channels .

YUV channel is closer to human perception, so the results obtained will be better when perceived .

Dataset Information

At present manually selected images of landscape have been taken for training and testing.

We have used 30 images for training and 10 images for validation and testing.



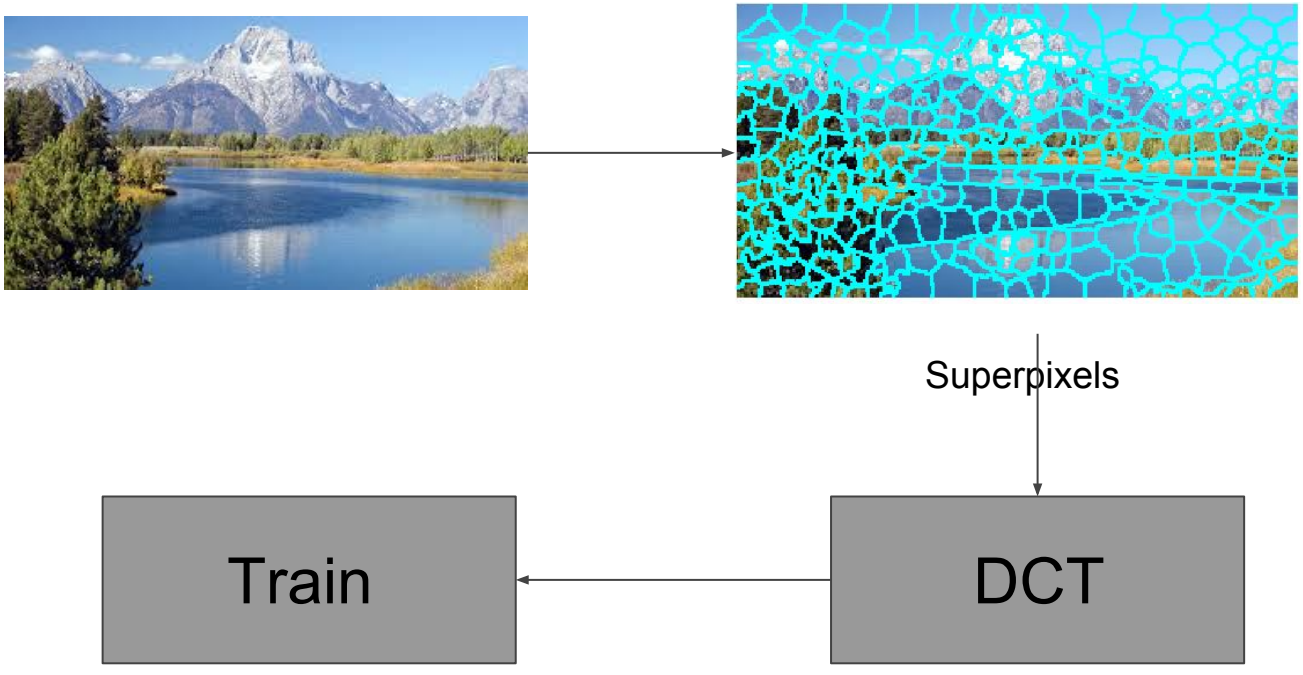
Black Swan

Initially dataset had random images selected of sceneries downloaded from flickr using image tags ‘mountains’ and ‘landscape’ .

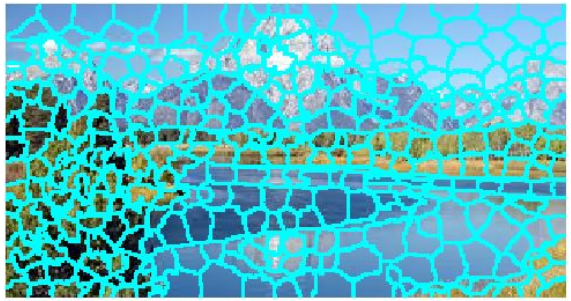
But results obtained were very poor, similar to black swan problem.

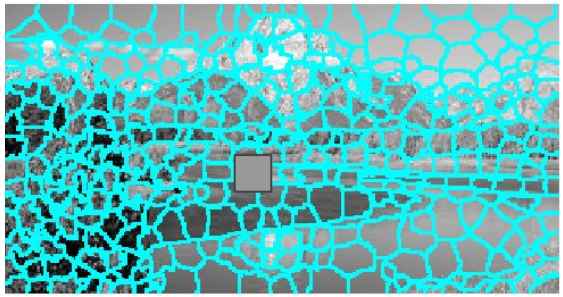
Images then were manually sorted to make sure that testing image had a similar mountain/scenery with training images.

Algorithms used



Feature extraction





Window around each superpixel

DCT

Dct of each window

Expected value

YUV channel

Mean over each superpixel.

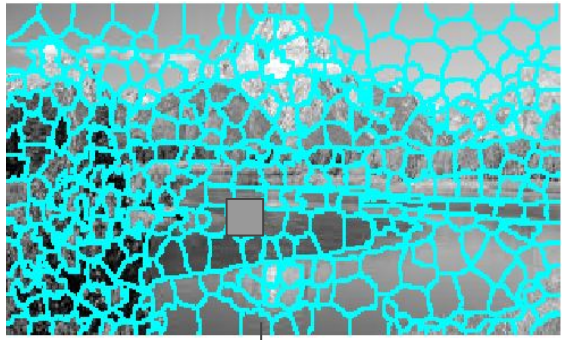
Image Training

We use two regressions - one for U space an done for V space.

The data matrix remains same in both cases, while the expected values change.

The trained model is then used to predict the color .

Image Testing



predict

DCT

Image Testing

We do the same procedure for the grey-scale images and get the Y vector respectively for each image.

Then we apply the model created while training and with its help we determine the corresponding U_mean and V_mean values.

Now for every pixel we use its Y value and together with the U_mean and V_mean we determine the value for the pixel.

Results

We have tried two training methods, support vector regression and kernel ridge regression .

Results for different window sizes, for different method was compared .

Number of superpixels were set approximately to total pixels/pixels in one window.

Error Parameter

We have taken sum of mean square error over each channel to be our error parameter .

For each pixel, the sum of square error over each of the three channels is taken and total sum for all pixels is then divided by the number of pixels.

Percentage error is then found by dividing by maximum error .

Results

Window size 16x16 using SVR

Results

Window size 16x16 using KRR

Results

Window size 14x14 using SVR

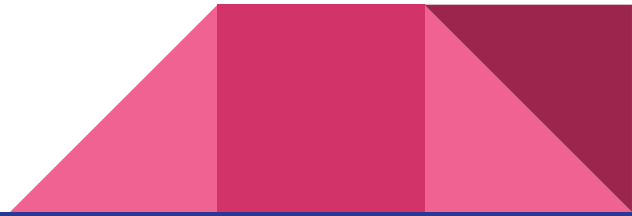
Results

Window size 14x14 using KRR

Results

The following results are obtained by using 12x12 window size and kernel ridge regression.

Window size smaller than this couldn’t be tried because of memory issues .



Results

Window size 12x12 using KRR

Results

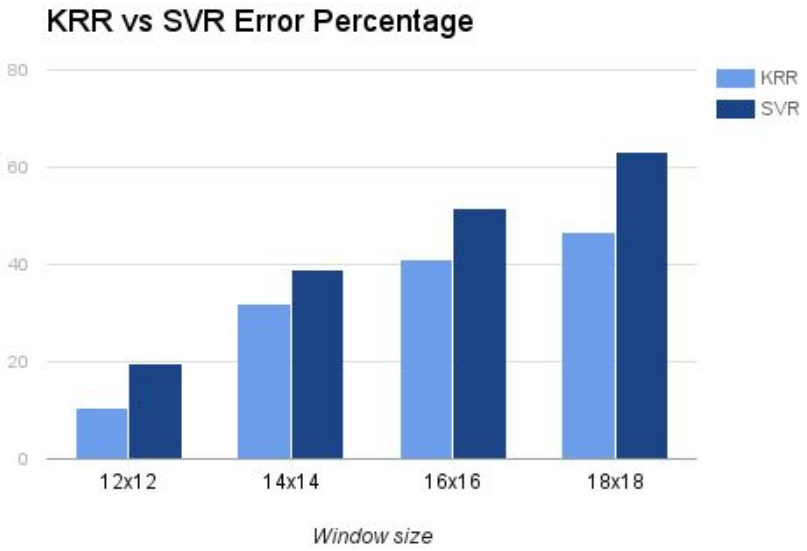
Window size 12x12 using KRR

Results

Window size 12x12 using KRR

Results

Comparing % error for different window size and training methods.



Thank you

Member 1 : Raj Manvar

Member 2 : Aagam Shah

Member 3 : Sanket Shah

Team Name : The Fafda Gang