COLORIZATION OF GRAYSCALE IMAGES

Team Name: SkyNet

Area: Computer Generated Art

Team Members

1. 14CE10007 Anukul Kumar
2. 14EC10002 Aditya Sinha
3. 14EC10050 Saurabh Dash
4. 14EC30021 Oindrila Saha
5. 14EC30037 Tejas Nitin Lad
6. 14EC30039 Vineet Jain
7. 14EC30044 Preetham K.S.
8. 14MA20029 Ramit Pahwa
9. 14MA20053 Harsh Bajaj
10. 14MT10033 Sahil Chaddha
11. 16CS1FX01 Gabriel Werner

Guided by:

Prof. Pabitra Mitra

Mr. Anirban Santara

Mr. Biswajit Paria

A report presented as final submission for Term Project in the course of Machine Learning (CS60050) undertaken in Autumn 2016.

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

1 Problem Statement

Given a grayscale image as input, find the plausible color version of the photograph.
The problem to be solved is basically a fully automatic approach to colorization without any human interference.

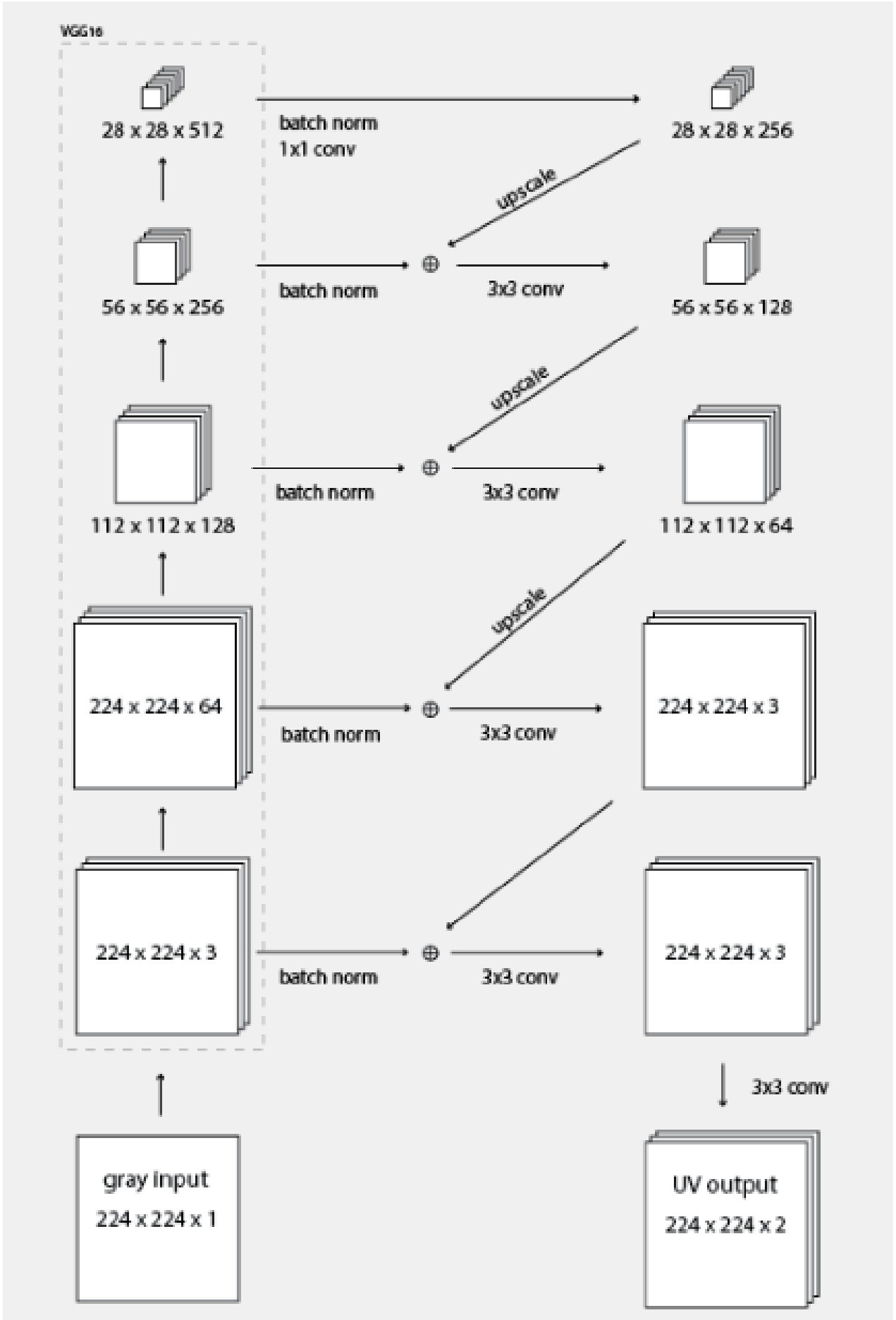
2 Architecture

The VGG-16 convolutional network is a standard architecture used for object detection in images using CNN. However, in our PS, we need to not only detect objects in the grayscale image, but also to associate U and V values with it for colorization. For this, we remove the final class labels of the VGG-16 output (truncated VGG-16). Instead, we have used additional architecture, wherein we have first passed the grayscale image through VGG-16. Then, using the highest layer, we have performed batch normalisation to infer some color, which is upscaled and merged with the batch normalised color information from the next highest layer. Proceeding like this, we have worked our way to the bottom of VGG-16 architecture and obtained the UV output. From there, it's only a matter of linear transform to convert orthonormal basis YUV to RGB color output.

3 Detailed Explanation

In CNN classification models), more info can be extracted than just the final classification. Intermediate layers in classification models can provide useful color information.

Concept of a hypercolumn in CNN is used. A hypercolumn for a pixel in the input image is a vector of all the activations above that pixel. The way this has been implemented was by forwarding an image through the VGG network and then extracting a few layers (specifically the tensors before each of the first 4 max-pooling operations), upscaling them to the original image size, and concatinating them all together.


VGG16
28 x 28 x 512
batch norm
1x1 conv
28 x 28 x 256
upscale
56 x 56 x 256
batch norm
3x3 conv
56 x 56 x 128
upscale
112 x 112 x 128
batch norm
3x3 conv
112 x 112 x 64
upscale
224 x 224 x 64
batch norm
3x3 conv
224 x 224 x 3
224 x 224 x 3
batch norm
3x3 conv
224 x 224 x 3
3x3 conv
gray input
224 x 224 x 1
UV output
224 x 224 x 2

The resulting hypercolumn tensor has tons of information about what's in that image. Using this information I should be able to color the image.

The residual encoder model (shown on page 2) has been used. Its almost an autoencoder but from black and white to color, with residual connections. The model forwards a grayscale image through VGG16 and then using the highest layer infers some color information. Then it upscales the color guess and adds in information from the next highest layer, and so on working down to the bottom of the VGG16 until there is 224 x 224 x 3 tensor. We used residual connections to add in information as it works its way down the VGG16.

4 Change in output images with iterations

Intermediate outputs were obtained while training the model.

The input to the model is the left-side grayscale image. The output is the middle image. The right image is the true color.

1. Iteration = 1000

2. Iteration = 6000

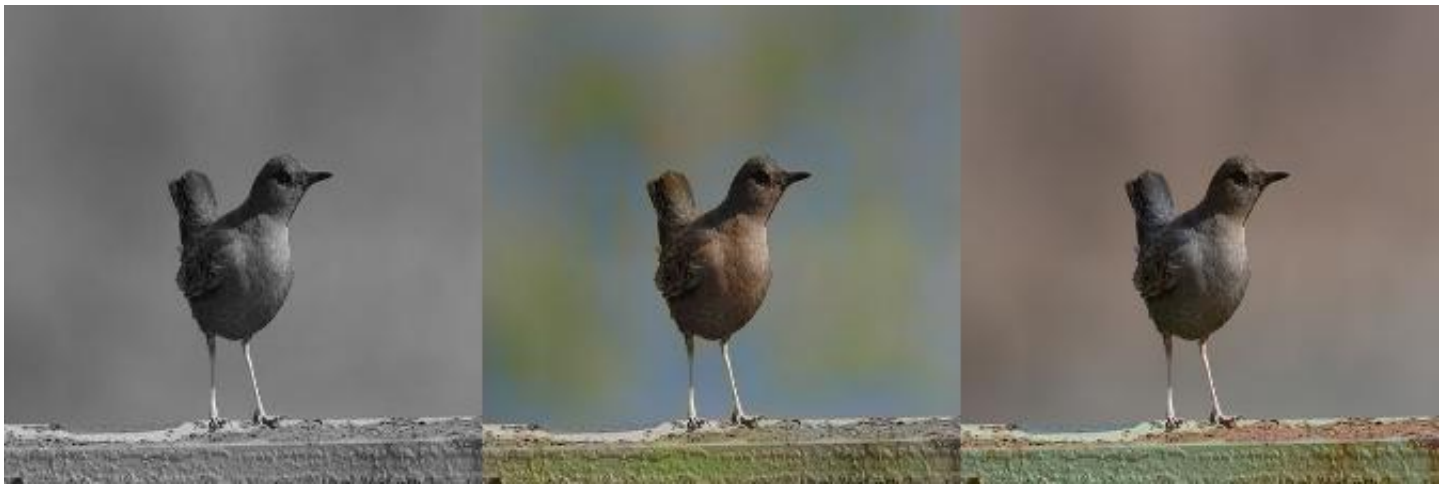
3. Iteration = 22000

4. Iteration = 38000

Ws we can see the model is able to identify features more sharply and add more colors to the image (it learns more number of colors) as it goes through successive iterations(as number of iterations increase).

5 Results

The input to the model is the left-side grayscale image. The output is the middle image. The right image is the true color which the model never gets to see



6 Relevant Links

Team: https://github.com/orgs/cs60050/teams/skynet
Repository: https://github.com/cs60050/SkyNet
Website: https://cs60050.github.io/SkyNet/website/