Image Colorization

Srinivas Tunuguntla (stunuguntla@wisc.edu)
Adithya Bhat (bhat5@wisc.edu)

Introduction

We intend to build a system to solve the problem of artificially colorizing grayscale images, in a completely automated mechanism. We plan to use supervised Machine Learning techniques to build a model that learns local and global features corresponding to each pixel, and the colors that correspond to those features, from the training set. Our objective over the course of this project is to build the best image colorization system that we can, and to this end we will both reuse / re-implement ideas we find interesting, and perform experiments to help us make design choices.

Related Work

Image colorization has historically been an area of interest, but most successful techniques traditionally involved user-input in the form of 'color seeds', or similar reference images. However, the resurgence of Neural Networks has resulted in significant gains in completely automated colorization systems too, especially in the last couple of years.

Fully Automated Colorization

1. ML Techniques with image features created via pre-processing. This corresponds to standard ML applications, where feature engineering plays the major role.
2. Convolutional Neural Networks are the current state of the art for automated colorization. The features themselves are learnt via the Neural Network's mechanism, as opposed to created via pre-processing.
   This is the segment we plan to work on.

User-assisted Colorization

1. Scribble based Techniques
   A user manually provides color scribbles (also known as seeds or hints) that give a reference color for various parts of the image. The system learns to extend the colors to rest of the image via image segmentation, etc, but the colors themselves are not learnt.
2. Reference Image Techniques
   Image features are used to find similar regions in the grayscale image and the reference image, and color transfer takes place between the two images. However, finding suitable reference images is vital to this method.

Our Project

In our project, we intend to build the best colorization system that we can, given time, experience, and computational power limitations. For this, we will re-implement / reuse existing work, as well try to mix and match learnings from different sources.

Datasets

We plan to use two different datasets to train our model: CIFAR-100 and ImageNet. We are currently using CIFAR-100 which has 100 classes of images each containing 600 images of size 32x32. We use 500 images from each class as the training set and the rest 100 images from each class as the testing set. The small size of the images results in faster convergence of models and hence allows us to experiment with various techniques.
We plan to use the ImageNet dataset to train the final model. ImageNet contains 21,338 categories with a total of 14,197,122 images of various sizes. ImageNet is larger in scale and diversity than the other image classification datasets.

Frameworks

Working in Python seemed to be the best option for our project, due to to maximal support for ML. All the below have libraries for Python.

- Caffe
  - Very commonly used.
  - Has a Model Zoo - repository of trained models with weights, which can be used out of box.
  - Con - has many dependencies, and is harder to get started with.
- Lasagne
  - Lightweight library to build and train neural networks in Theano.
  - Only works on Theano.
  - More flexibility and transparency, but more complicated for beginners.
- Keras
  - Easiest to get started with.
  - Can work with TensorFlow or Theano.
  - Abstracts away a bunch of details (trade-off : control vs ease of use)
- We intend to go with Keras, due to ease of use, and change if we have a pressing need to.

Color Space Models

- RGB, CMY - luma (image intensity), and chroma (color information) are tightly coupled
- YUV
    - Y - Luminance
    - U,V - Chrominance
    - Reduces bandwidth for chrominance components, taking human perception into account.
- HSV - Hue Saturation Value - cylindrical system
    - H : 0-360 degrees
    - S : 0-100%
    - L/B : 0-100%
- CIE Lab, CIE Luv
    - L : 0-100
    - a,b : -100 to +100, or, -128 to 127
- We are currently using the CIE Lab color space, as euclidean distance between points in this color space apparently correspond better to visually perceived differences.
- That said, we might experiment with other types, for a given error function.

Approach

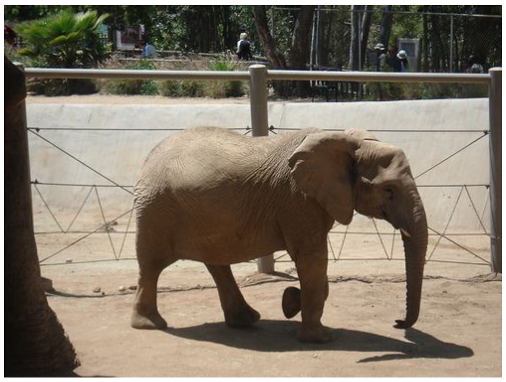
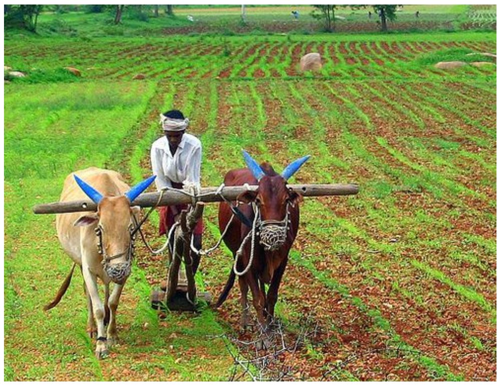
Our approach is based on deep convolutional neural networks. We derive a set of global features that act as a prior to indicate the type of the image and to decide the macro colors of the image. For example, if the global features indicate that it is an outdoor image, the model will be biased to adding colors of sky, grass, water etc. We use a pre-trained neural network that is trained for an image classification task to obtain this information. Since the network is trained for a classification task, the last few layers capture the global context of the image and hence can be used as global features. We will use the VGG-16 network, and experiment with others, in this project.

We then derive a set of local features on the image using a convolutional neural network. We are still experimenting with the architecture of this network. The current model has 3 convolutional layers, each with 3x3 kernels and 1x1 strides. The first layer has 64 filters while the second and third layers have 128 filters each. We use batch normalization and max pooling to regularize the parameters at each layer. All the activation units are rectified linear units.

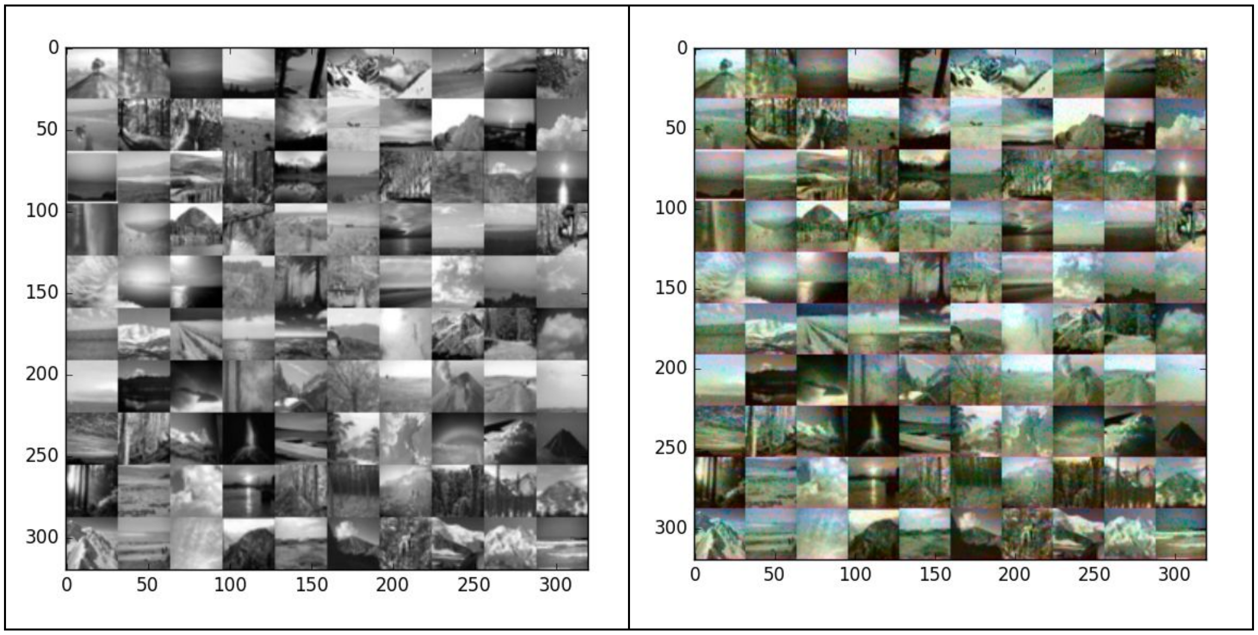
We then combine the global and the local features obtained as described above and feed them to the final colorization network. This network consists of two convolutional layers each with 64 filters of size 3x3 and stride 1x1. The final layer is densely connected with ‘tanh’ activation units. The final layer acts as the output and predicts both the color components a, b simultaneously.

Current Results

Converted the VGG-16 caffe model to Keras for Transfer Learning.
The hope is that the features learnt in this model will turn out to be relevant to colorization too.

	
(u'n02504458', u'African_elephant', 0.98831379)	(u'n03967562', u'plow', 0.74705422)
(u'n02504013', u'Indian_elephant', 0.011305372)	(u'n03868242', u'oxcart', 0.1379173)
(u'n01871265', u'tusker', 0.00030840875)	(u'n02403003', u'ox', 0.10923596)

As an initial model, we built a network that has two convolutional layers followed by a dense layer. The first convolutional layer has 64 filters while the second layer has 128 filters. Each filter is of size 3x3 and stride 1x1. The network takes as input a grayscale image and returns the color components as output. Note that we are have not yet incorporated the global features discussed above and the network is only 2 layers deep. A sample of test grayscale images and their corresponding outputs from this model are shown below.



Possible Experiments

- Classification vs Regression
- Attempt to Re-use various models for image classification / other related areas
    - VGGNet vs ResNet, etc
    - Need to handle the fact that input will be a grayscale image somehow.
- Various Error Functions
    - CNN based colorization systems typically produce washed-out or sepia toned colors for objects that could have multiple valid colors.
    - When the model sees similar objects with almost identical features, but different colors, it is penalized depending upon the loss function, and hence tends to predict the average of the values it has seen.
    - Hence, the loss function used also plays a major role.
- Various color space models.
- Hypercolumns vs different networks for local features and global features.
- Activation functions for the final layer.

Issues

One of the issues with deep neural networks is that they take very long times for training. To train networks on millions of images with high resolution, clusters of CPUs or GPUs are generally used. Since we do not have access to such resources, we plan to restrict our datasets to a reasonable size with a limited set of categories. Another potential solution is to initialize the network using weights from networks that are trained for other tasks (similar to the transfer learning approach to derive global features described above) and then start the training process.