Notes from Ruichi:

* Why do they think the features can be learned by feature?
* Force model to learn relationship between features and colors
* Learning representations for automatic colorization
* Understand input and output, high level understanding of the task
  + Know method, not equations
  + 1 or 2 pages of slides for next week
* Semantic segmentation is difficult and probably not used in our paper
* UNDERSTAND THE TECHNICAL PROCESS
* Use google to understand literature and the fundamental ideas
  + Why do they use Lab, Hue/Chroma instead of RGB?
* The neurons actually producing the histograms
* KL divergence as loss
* Investigate if R = 1 really is a good way to do it?
* Kernel, NOT SLIDING WINDOW
  + Pooling, convolutional…
  + Why is pooling/blurring helpful?
    - Eliminates noise, eliminating details and capturing the abstract
* Deep learning gets more refining permits generalization ability
* Forming a hypercolumn comes from information from different scales
  + Histogram is the output of the CNN, and color is the output of the histogram
* **Big Question: do they consider contextual pixels or just the pixel itself (one pixel or a box of pixels)**
* Why is everything Red? Possibly reclassify default
* Possibly implement semantic level stuff
* Increase window size beyond 1 x 1 “pixels”

Is it a distribution or is it one hot vector?

10/13

We presented our presentation to Ruichi today. We discussed the Hue, Saturation, and Brightness scheme referenced in the paper in order to try to understand it. He wanted us to come back next week with something more explainable. Segmentation of image is given input, you label each pixel of image. Figure out how their input and output happens as well as where their inference happens. Goals for next week is to find out what the 313 value corresponds to in the input dimension, and why it seems like red is the default color.

10/20

We found the article corresponding to the code we are currently using for colorization, titled “Colorful Image Colorization”. One of the naïve issues with the paper is that there is too much rebalancing. In our test results, we are now seeing default colors to red and blue, instead of only getting red. We hypothesize that the reasoning for this is because red, blue, and green are the default colors because those are the sharpest hues in an RGB system. Despite changing our article, we should maintain our course towards bridging semantic-level and pixel-level colorization. The consistency in the images are not captured. Possibly implement a scene classifier (urban scene, nature scene, etc). Also look into using a web crawler to compile images. Transfer RGB to AB. Use the function in the paper for the Python code.

10/28

We have implemented a scene classifier that still needs some work. Possibly implement a sign classifier as well. The problem is that we don’t want to have too many classes. Find out what the program handles worst, and then we’ll have a classifier for the 10 most problematic scenes. Then we’ll train a linear classifier as a naïve scene classifier, including basic averaging solutions. Another possibility is color distribution. Next time, we should be able to read more in depth into the associated paper and develop a scene classifier.

11/4

We ran into some road blocks with using an SVM and bulk image data to make a scene classifier. We should build an approximate scene classifier rather than a full classifer. By using our image data, we should get a single base color from each of our classes by finding the average color at each point. Then, we related it to our CNN. In this way, we can train the scene classifier and add it to the end of the CNN pipeline. If the weighted average/imaging doesn’t work, look at the documentation’s method. We need to focus on figuring out the ab and LAB color models as opposed to the traditional rgb color model.

11/11

Our current training data has a slight bias. Ruichi helped us fix the svm so that our training data has almost no loss. We need a way to access/calculate the predicted distribution. We can start building slides on the entire story: 1-2 minutes on the colorization problem and related works (Stanford paper and reference images, the paper whose code we used), 2 minutes showing examples of the original algorithm output and discuss the troubleshooting (red-blue-green default overfitting) [good: we don’t do any ridiculous guessing, bad: we lose the sharpness of the color], 2 minutes explaining our motivation of constraining these failures by making naïve scene classifiers. And 1 minute elaborating our goals for the remaining half the semester, which is learning how to merge the color using the training data.

11/18

In our presentation, we did not address why rgb is favored (because the current code favors extremes rather than averages). With our new data, our loss is 36%. We need to quickly implement the naïve method, and if it does not work, move on. Two possible naïve implementations: we can either get one color per image (get the sum of all images in the training set resulting on **one value/color**), or one color per pixel (for one category we get the output as a **256x256 matrix**). Be sure to **resize** consistently as well (recommended 256x256). To merge, make sure the color space is **consistent**, and then find a mean of the code’s output and our output. By **understanding** their merging method, we can see if we can develop reasonable merging based on averaging. The major challenge is how we incorporate our color. Be sure that we do not classify our scene incorrectly. Calculate the **MSE** (mean square error); ideally this error is small. This ensures that we do not “eyeball” it, but rather have a more objective evaluation.