Michael Chan

RBE 549

Get Tiny Images:

When writing this function, all that was really needed was to resize the image from a roughly 200x200 picture to a 16x16 image. Then I reshape the 16x16 image into a 256x1 vector. This is essentially the feature vector for the image. You then run this for every single image in the set. So 1500 iterations.

Nearest Neighbors Classify:

For this function, I made the decision to just use the L2 distance as my distance. I took every single feature vector in my test set, and compared it to every single image in my training set. I then sort that matrix to find the smallest distance vector in the matrix. I then take the smallest distance vector and add it to my predicted categories.

Build Vocabulary:

For build vocabulary, for each image I detected SURF features of the image and then extracted HOG features from the top 300 points. I then made the design decision that rather than take all of the detected features, I would take a random permutation of features from each group and then add that to an overall matrix. I do this for each image, and then run k-means clustering on the whole overall matrix. This returns my vocab which is a nxd matrix, which is the number of clusters I wanted by the feature descriptor length.

Bag of Words:

For bag of words, I started off doing the same thing as I did in build vocab. I detected surf features and then extract HOG features around the strongest 300 points. I then compute nearest neighbors for the feature to each of the features of the vocab. This finds the distance of the feature to the centroids found by kmeans. I then find the index of the smallest distance as that will give me the closest centroid the feature is to. I then add that index to a histogram of length vocab. Then I normalize the histogram and append it to the image features.

SVM classfier:

For each category, I compare the category to the training labels using strcmp. This returns a binary representation of matches that I call matchInd. I then use fitclinear on train image feats and matchInd to find the curve that best fits it. Then since fitclinear outputs Beta and Bias, I just input those fields into the equation W*X+B where W is the Beta and B is the bias. This outputs the confidence that I then append into a scores

Feature + Classifier	Percentage
Tiny Images + NN	20.1%
Bag Of Words + NN	50.3%
Bag Of Words + SVM	45.4%

Confusion Matrix for Bag of Words + NN

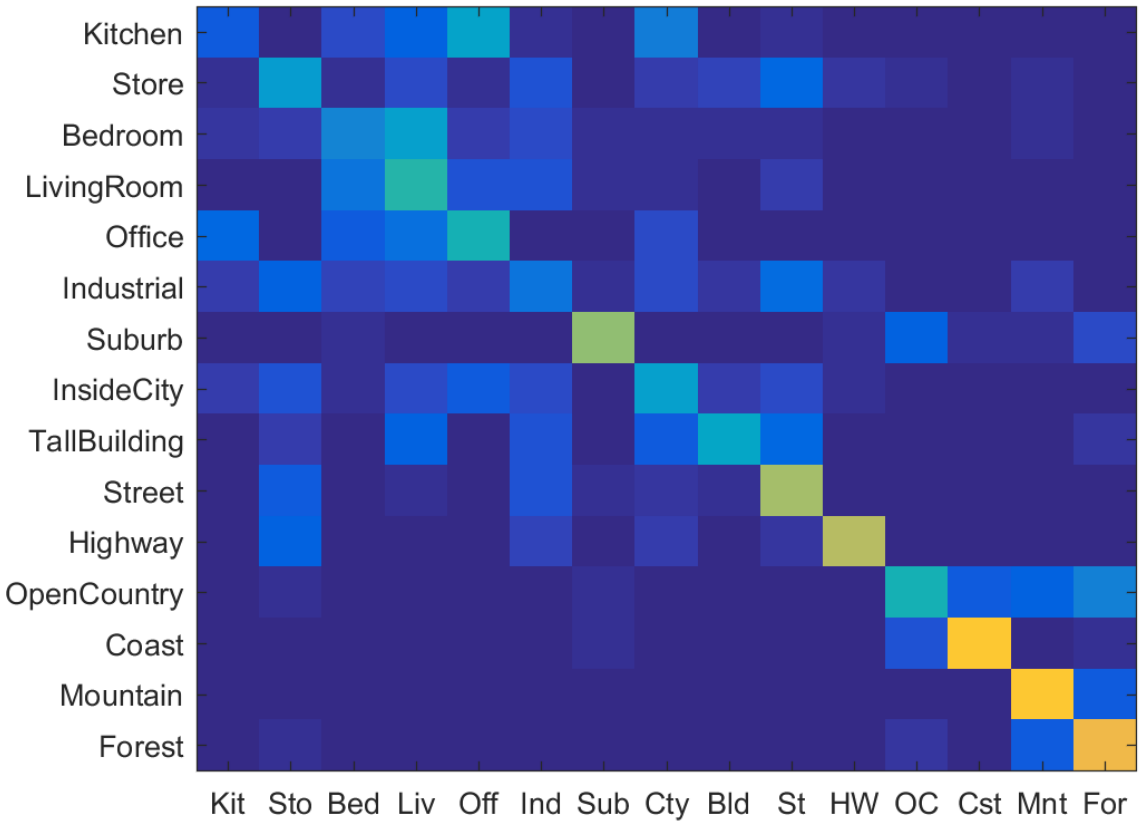
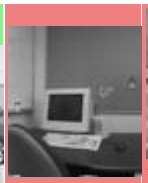
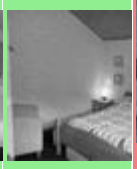
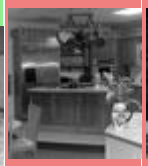
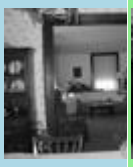
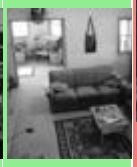
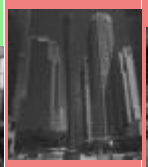
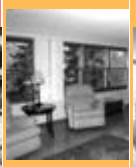
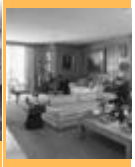
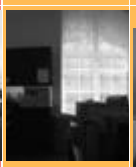
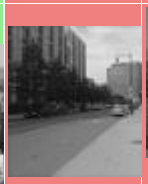
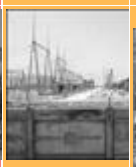
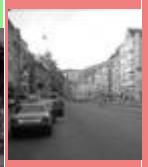
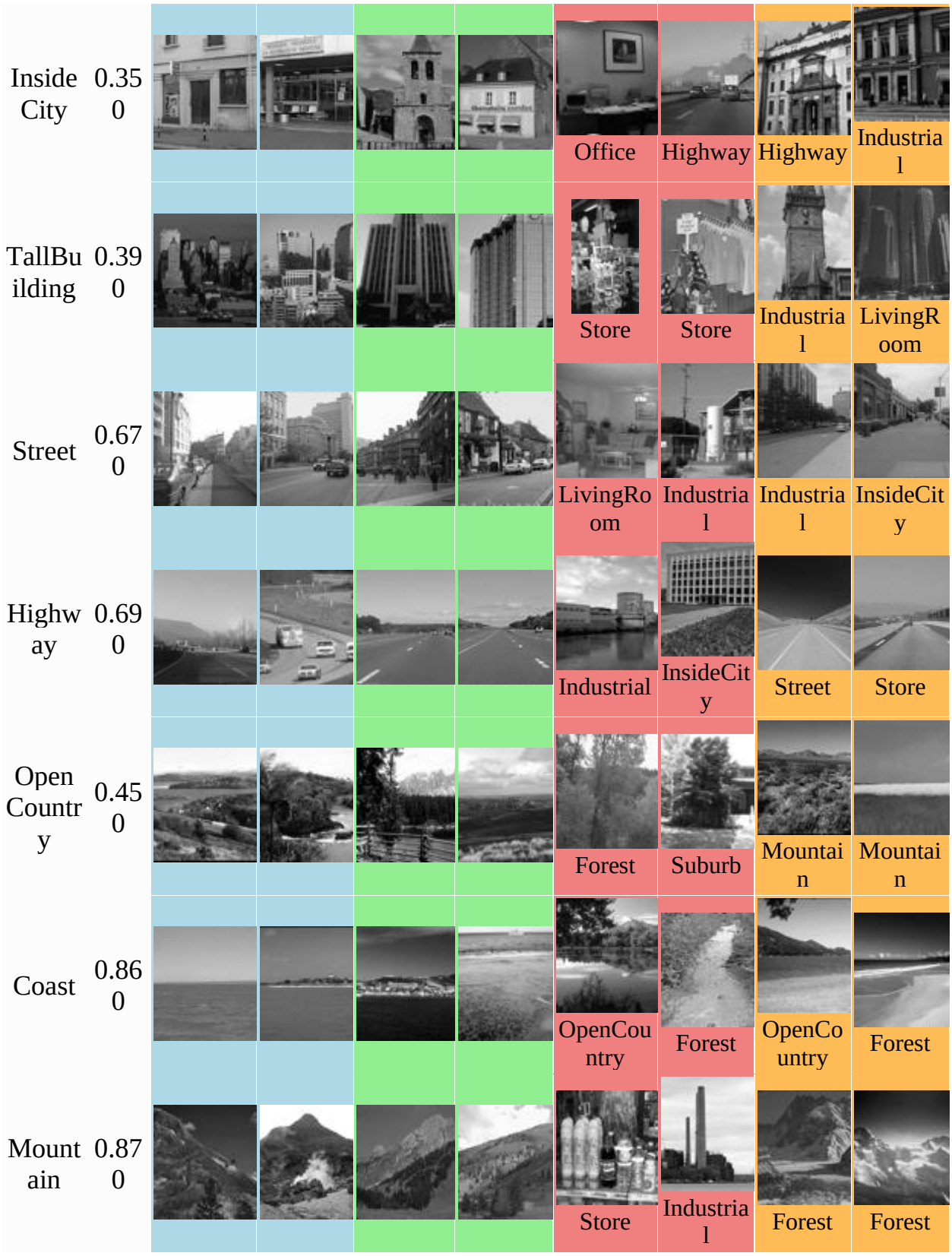


Table of Classifiers

Category name	Accuracy	Sample training images	Sample true positives	False positives with true label	False negatives with wrong predicted label

Kitchen	0.120					 Office	 InsideCity	 InsideCity	 LivingRoom
Store	0.340					 InsideCity	 Street	 Street	 Street
Bedroom	0.260					 Kitchen	 Suburb	 Store	 Industrial
Living Room	0.470					 TallBuilding	 Store	 Industrial	 Bedroom
Office	0.450					 InsideCity	 Industrial	 Bedroom	 Bedroom
Industrial	0.200					 Street	 TallBuilding	 InsideCity	 Suburb
Suburb	0.630					 Street	 Bedroom	 OpenCountry	 OpenCountry


Inside City
0.350
Office
Highway
Highway
Industrial
TallBuilding
0.390
Store
Store
Industrial
LivingRoom
Street
0.670
LivingRoom
Industrial
Industrial
InsideCity
Highway
0.690
Industrial
InsideCity
Street
Store
Open Country
0.450
Forest
Suburb
Mountain
Mountain
Coast
0.860
OpenCountry
Forest
OpenCountry
Forest
Mountain
0.870
Store
Industrial
Forest
Forest

Category name	Accuracy	Sample training images	Sample true positives	False positives with true label	False negatives with wrong predicted label
Forest	0.800			Bedroom Suburb	Mountain Mountain