Scene Recognition

Omar Ahmed , 201300604

Mohamed Ashraf , 201302677

Mohamed Hussien , 201300116

Ahmed Medhat , 201303779

Remote sensing project
Fall 2015

INTRODUCTION

One profound character of making us humans is our recognition . humans recognize the world at a single glance. Recognizing the category environment around takes tens of milliseconds from human brain. Moreover, it is our capacity to learn more and more and save what we learn of places and environments, so we can recall them whenever we want to. That is achieved by continuously sampling the world several times per second, reaching an exposure to millions of images within a year [1]. The goal of this project is Scene recognition which is a task of computer vision, that is one of many tries humans do in their journey to make artificial system that reaches the human ability in scene recognition.

Research/Background

Scene recognition can be achieved by:

1- finding a convenient image representation

- Bag of words model (BoW)

2- A powerful classification tool

- Support Vector Machine (SVM)

Bag of words model (BoW):

it is considered the state of the arts method in image classification , and it is widely used for scene recognition. it treats the features of image as words. To represent an image using Bag of words model, we treat the image as if it a document that contains words. to achieve the representation, we follow these steps. first is feature detection, then comes feature description and last thing is codebook generation.

- Feature description:

After detecting the features. the image is divided into patches. the goal here is to represent each patch as a numerical vector (descriptors). the descriptor has to be robust which means not to be affected by rotation, scaling, illumination to a certain extent. One of the most famous descriptors is **SIFT*** (Scale-invariant Feature transform). Sift converts each patch to 128-dimensional vector. After doing SIFT, the image now is a collection of 128-D vectors.

* for further understanding of SIFT please check this link: (http://aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction/)

- codebook generation:

  At this point we need to map each vector-represented patch to a vocab-words. a vocab-word can be considered as a representative of several similar patches. this can be done by **k-means clustering** of all the descriptors(128-d vectors), where the centroids of the clusters are the vocab-words, and the number of clusters is the size of vocab-words.

Now, the image can be represented as a histogram of the vocab-words. in other words, the image becomes an array of size equals to the the vocab-words size. the vocab-words become the new features that the image is going to be classified according to.

Support Vector Machine (SVM):

In machine learning, support vector machines are supervised learning models used for classification and regression analysis. Given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other. In the case of support vector machines, a data point is viewed as a N-dimensional vector, and we want to know whether we can separate such points with a (N-1)-dimensional hyperplane. This is called a linear classifier. There are many hyperplanes that might classify the data. One reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. So we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized [4].

Dataset

in this project we are using a data set called " the 15 class scene". This set contains 15 categories of images. The 15 scene categories are office, kitchen, living room, bedroom, store, industrial, tall building, inside cite, street, highway, coast, open country, mountain, forest, and suburb. Images in the dataset are about 250*300 resolution. This dataset contains a wide range of outdoor and indoor scene environments [2]. The code trains and tests on 100 images from each category (i.e. 1500 training examples total and 1500 test cases total).

PROCEDURE

In the implementation of SIFT and SVM, we used a library called **VLFeat**. VLFeat is a cross-platform open source collection of vision algorithms with a special focus on visual features (for instance SIFT and HOG) and clustering (k-means, hierarchical k-means, agglomerative information bottleneck) [3].

Step 0: Setup parameters, vlfeat, category list, and image paths.

- Setting up vlfeat library
- Identifying the categories
- Identifying the image paths
- Setting the number of training examples per category to use
    - we trained on 1500 images and tested on 1500 images. 100 images per category

Step 1: Represent each image with the appropriate feature

- Build the vocabulary (visual words) that will be used as features later using

build_vocabulary.m, it will get the SIFT descriptor of these images first, then use kmeans with k specified as the vocabulary size
    - we used 400 vocab words
- If the code has been run before and there is an existing vocabulary (vocab.mat), the code doesn't regenerate vocabulary.
- Get the features for each image to be used later in SIFT using get_bags_of_sifts.m

Step 2: Classify each test image by training and using the appropriate classifier

- Train the SVM from the training features done in the previous step (which is a histogram of visual words for each image), then apply the trained weights on the test set and put the labels in the predicted_categories vector to compare it later with the test labels and get the accuracy
- train a one vs all linear SVM classifier, and apply each classifier of the 15 to each image and the category is assigned based on the highest result
- 5 fold cross validation is applied to get the best regularization parameter (lambda) and avoid overfitting
- Take a sample from the negatives to prevent it's over representation (1:3 positives to negatives), with the negatives taken randomly from all the negatives

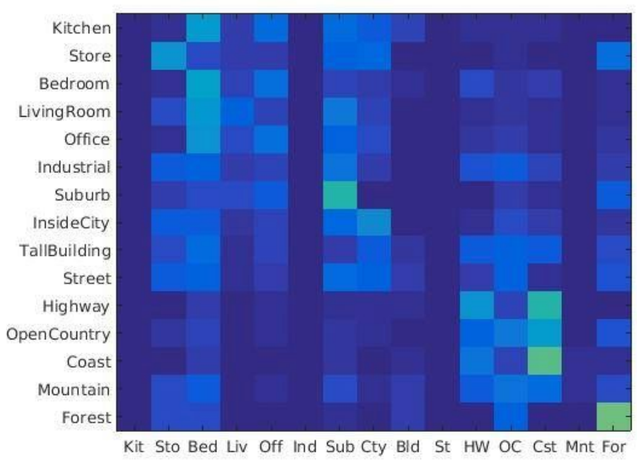
Step 3: Build a confusion matrix and score the recognition system

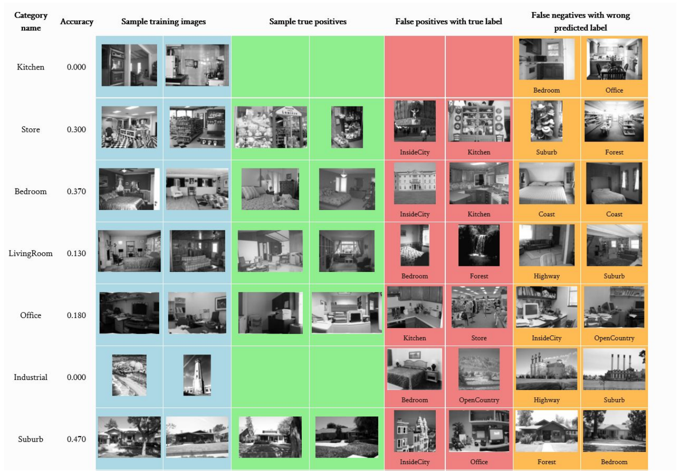
you can find further documentation in the code.

RESULTS

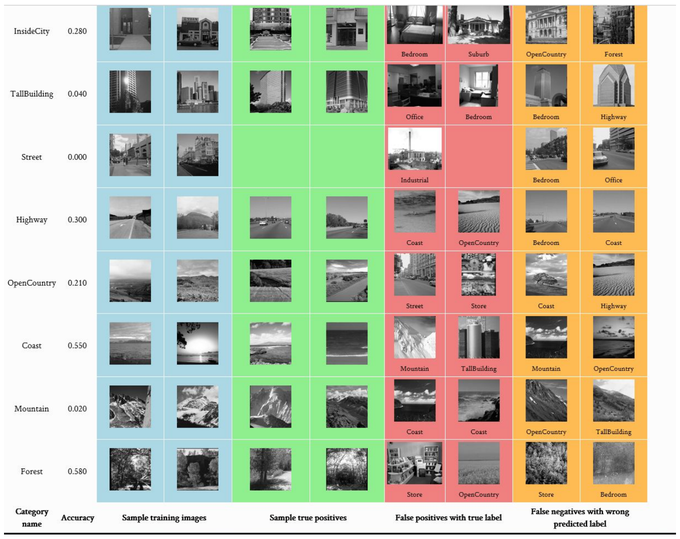
Confusion Matrix:

Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class





Category name	Accuracy	Sample training images		Sample true positives		False positives with true label		False negatives with wrong predicted label	
Kitchen	0.000							Bedroom	Office
Store	0.300					InsideCity	Kitchen	Suburb	Forest
Bedroom	0.370					InsideCity	Kitchen	Coast	Coast
LivingRoom	0.130					Bedroom	Forest	Highway	Suburb
Office	0.180					Kitchen	Store	InsideCity	OpenCountry
Industrial	0.000					Bedroom	OpenCountry	Highway	Suburb
Suburb	0.470					InsideCity	Office	Forest	Bedroom



Category name	Accuracy	Sample training images		Sample true positives		False positives with true label		False negatives with wrong predicted label	
InsideCity	0.280					Bedroom	Suburb	OpenCountry	Forest
TallBuilding	0.040					Office	Bedroom	Bedroom	Highway
Street	0.000					Industrial		Bedroom	Office
Highway	0.300					Coast	OpenCountry	Bedroom	Coast
OpenCountry	0.210					Street	Store	Coast	Highway
Coast	0.550					Mountain	TallBuilding	Mountain	OpenCountry
Mountain	0.020					Coast	Coast	OpenCountry	TallBuilding
Forest	0.580					Store	OpenCountry	Store	Bedroom

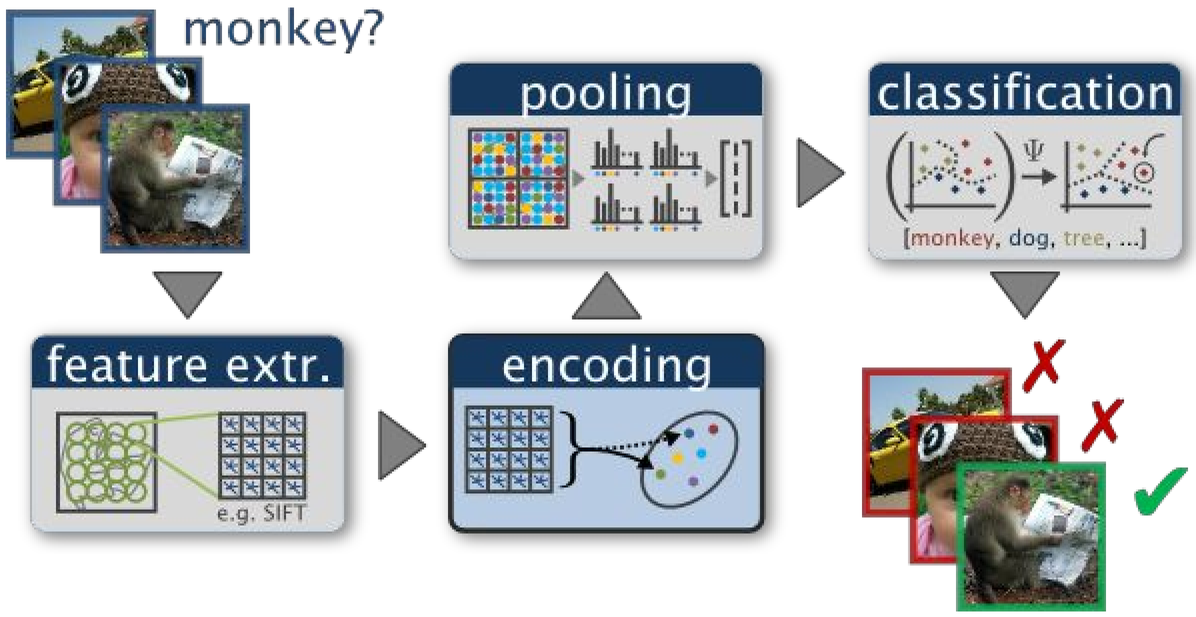
At the beginning we got accuracy that ranges between 7-11%.

But after tailoring the parameters more than once, we achieved 22%. However the aspired accuracy from this algorithm is 60-70%

you can find the results presented in a html webpage in this directory: ~code\results_webpage\index.html

CONCLUSION

in conclusion, what is aspired in this project is to make machines recognize places. this recognition can be achieved by breaking down the problem in firstly finding a way to represent image and then classify the image after representation into one of the 15 categories of the dataset that we used. the representation of the image was achieved by using SIFT and bag of words and the classification was achieved by using support vector machine.



REFERENCES

1. Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. "Learning Deep Features for Scene Recognition Using Places Database." (n.d.): n. pag. Web
2. http://qixianbiao.github.io/Scene.html
3. Vedaldi, A., & Fulkerson, B. (2010). VLFeat: An open and portable library of computer vision algorithms. Proceedings of the International Conference on Multimedia - MM '10, 1469-1472.
4. https://en.wikipedia.org/wiki/Support_vector_machine
5. http://cs.brown.edu/courses/cs143/proj3/

Google Drive link for the code:

https://drive.google.com/file/d/0B1LeeQZdQE_9a2pXNExGajdoZDQ/view?usp=sharing