Assignment A04

ENSE 885AY Application of Deep Learning in Computer Vision

Scene Recognition with Bag-of-Words

Student Name: Behnam Moradi

Introduction

This is the fourth assignment of our wonderful course on **deep learning applications in computer vision** at the University of Regina. Students are required to come up with unique and accurate algorithm for the following functions:

1. Tiny images representation and nearest neighbor classifier (accuracy of about 18-25%).
2. Bag of SIFT representation and nearest neighbor classifier (accuracy of about 50-60%).
3. Bag of SIFT representation and linear SVM classifier (accuracy of about 60-70%).

Dataset

In this assignment, we will be using a dataset including the followings. Inside the project directory, there is a folder called “data” which includes two sub-directories named “train” and ‘test’. They will be used for training and testing our algorithms.

Tiny images representation and nearest neighbor classifier

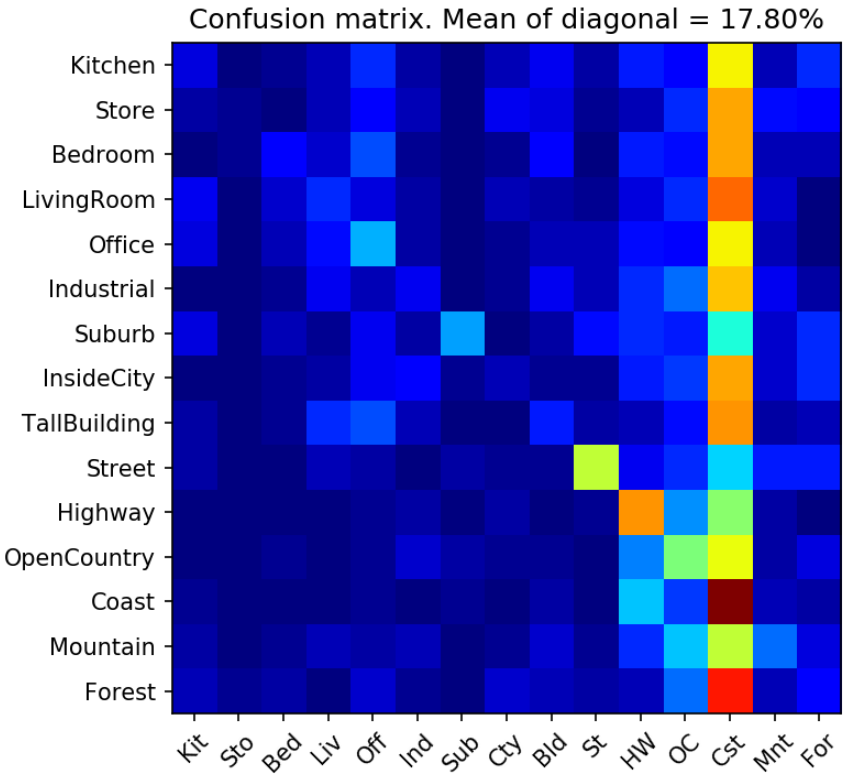
The idea behind tiny image representation is very simple. We need to get an original image and reduce its size to let’s say 16x16 pixel. In the “student_code.py”, there is a function called `get_tiny_images(image_paths):` This function is implemented in a way to receive the dataset path and return a numpy array including in which every row has 256 elements which is a (16x16) image. We will be reducing both train and test images to 16x16 pixel images.

Our first image classifier is the K-nearest neighbor data classifier KNN. We will be using 1-NN in our implementation. The idea behind KNN is quite simple. Let's say all our images are reduced to 16x16. It means every image is simply a 256 dimensional vector. For every test image, we use the following algorithm:

1. Calculate the distance between the test image and all training images
2. Find the minimum distance and it correspondence image
3. Assign this label to the test image

As you can see, KNN is not learning anything during the process and that is why KNN for image classification is not accurate at all.

Here is the confusion matrix of using KNN. The accuracy is around 18% which is very low.



Bag of SIFT representation and nearest neighbor classifier

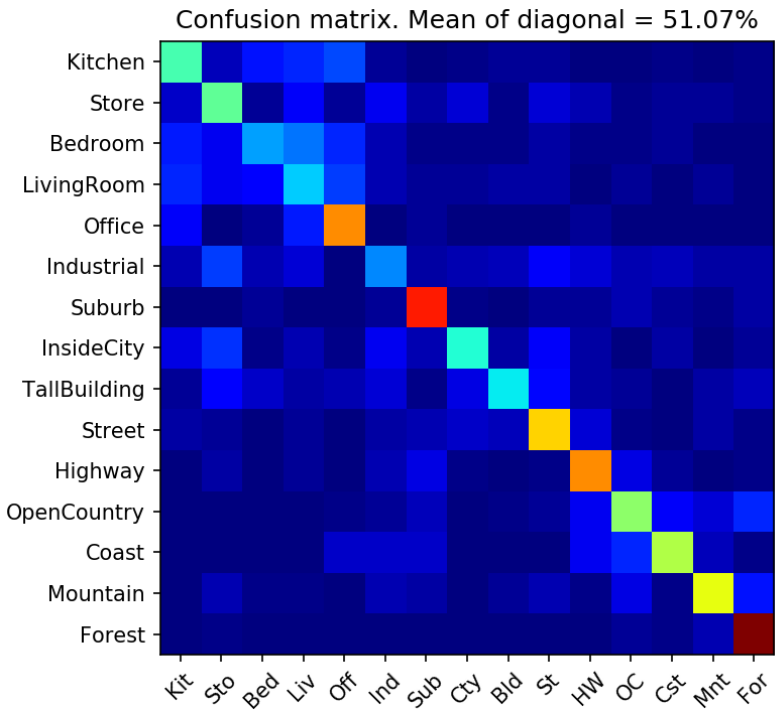
In this section, we use another image representation form called BAG of SIFT. Check the following algorithm:

1. Creating a vocabulary of visual words for each image inside the training dataset. Simply we can do sampling to select a huge number of local feature for each image and then we can cluster them using KNN. Attn: Number of clusters=vocabulary size
2. Calculating the center of clusters and each center will be a word inside our vocabulary
3. Creating histogram for each testing image
4. Normalizing the histograms
5. Consider 50 words for each vocabulary
6. Consider 220 features for each image

The aforementioned steps are implemented inside the "student_code.py" in the following functions:

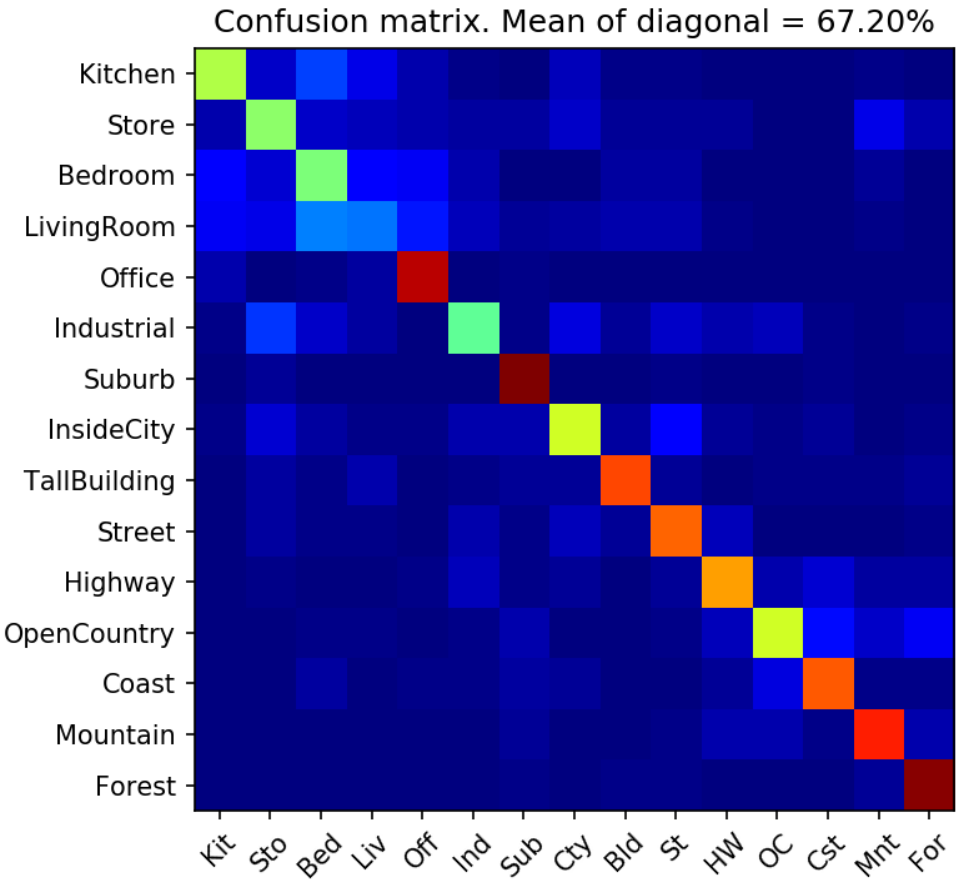
1. `build_vocabulary(image_paths, vocab_size)`
2. `get_bags_of_sifts(image_paths, vocab_filename)`

I used step_size of 20 and randomly I selected 20 features for each image to build a vocabulary. Then KNN algorithm is applied to the data. Accuracy: 51.07 %



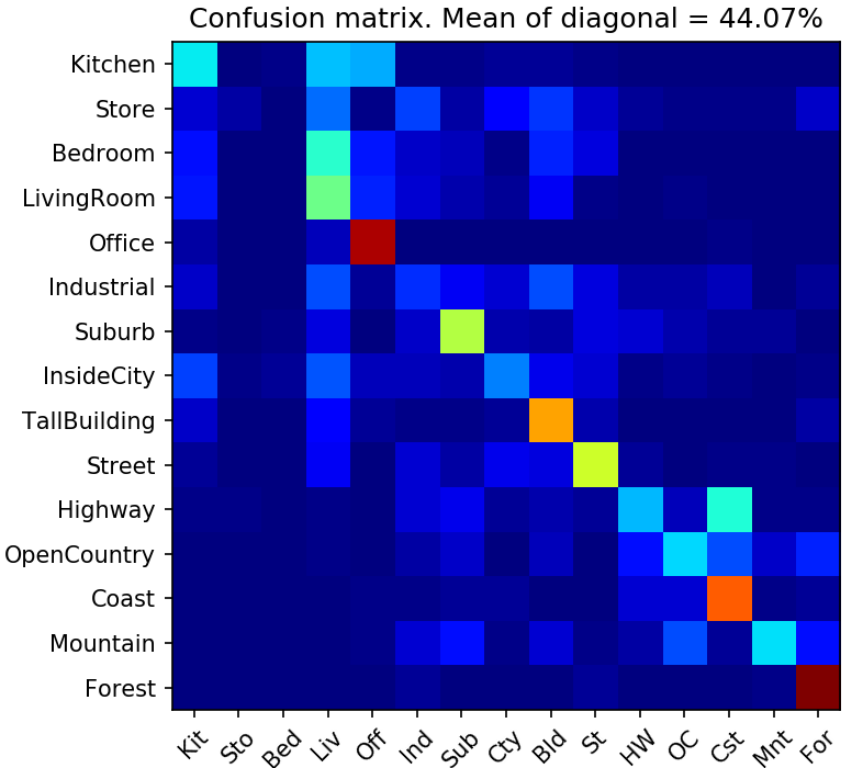
Bag of SIFT representation and linear SVM classifier

In this section the SVM will be used to train 15 models for each sub-dataset. We will be testing each model against the others and select the one which has the highest score.It is important to mention that I will not be implementing the SVM itself. I will be using its implemented function in “sklearn.svm”.
Vacab_size = 200 →Best Accuracy: 67.2%

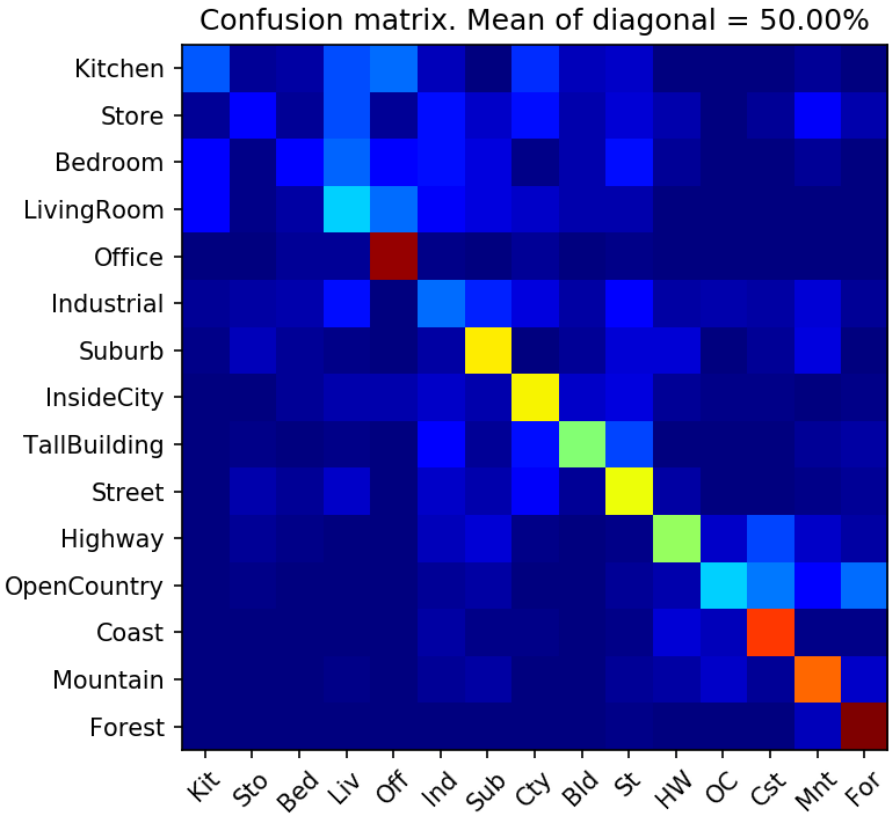


Experimental Design: Experiment with many different vocabulary sizes

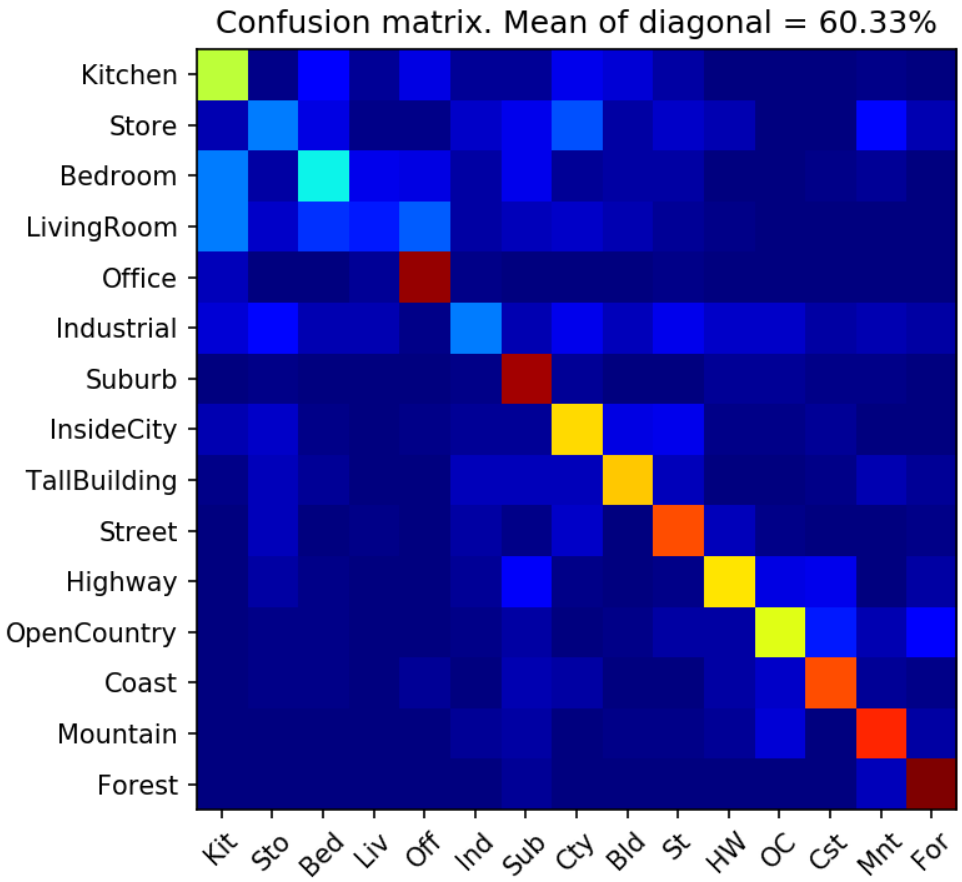
Vacab_size = 10 →Accuracy: 44%



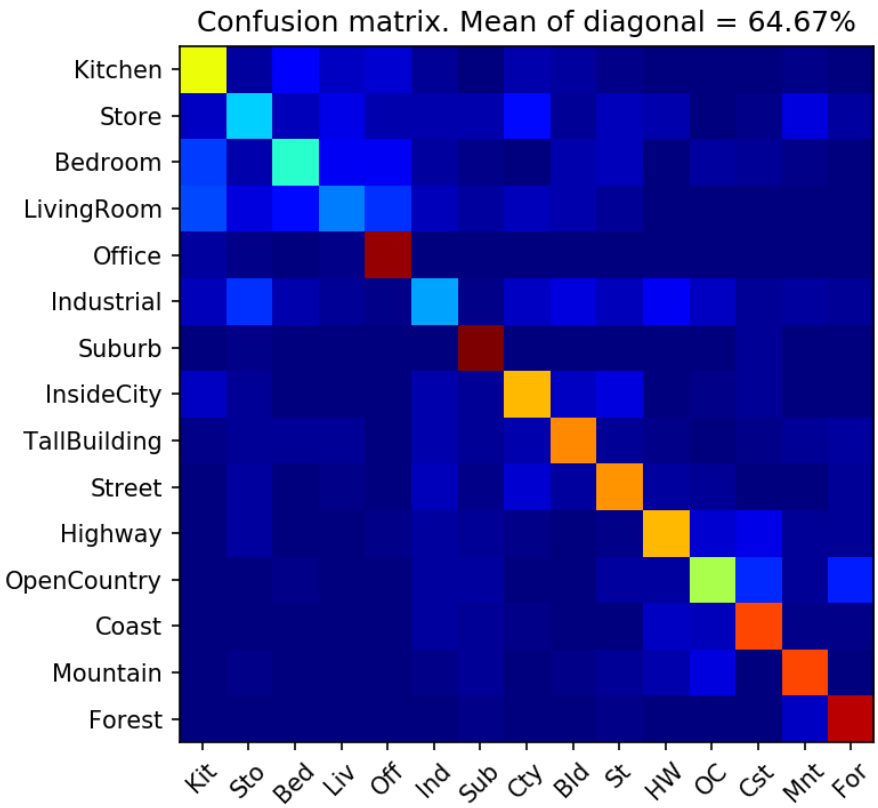
Vacab_size = 20 →Accuracy: 50%



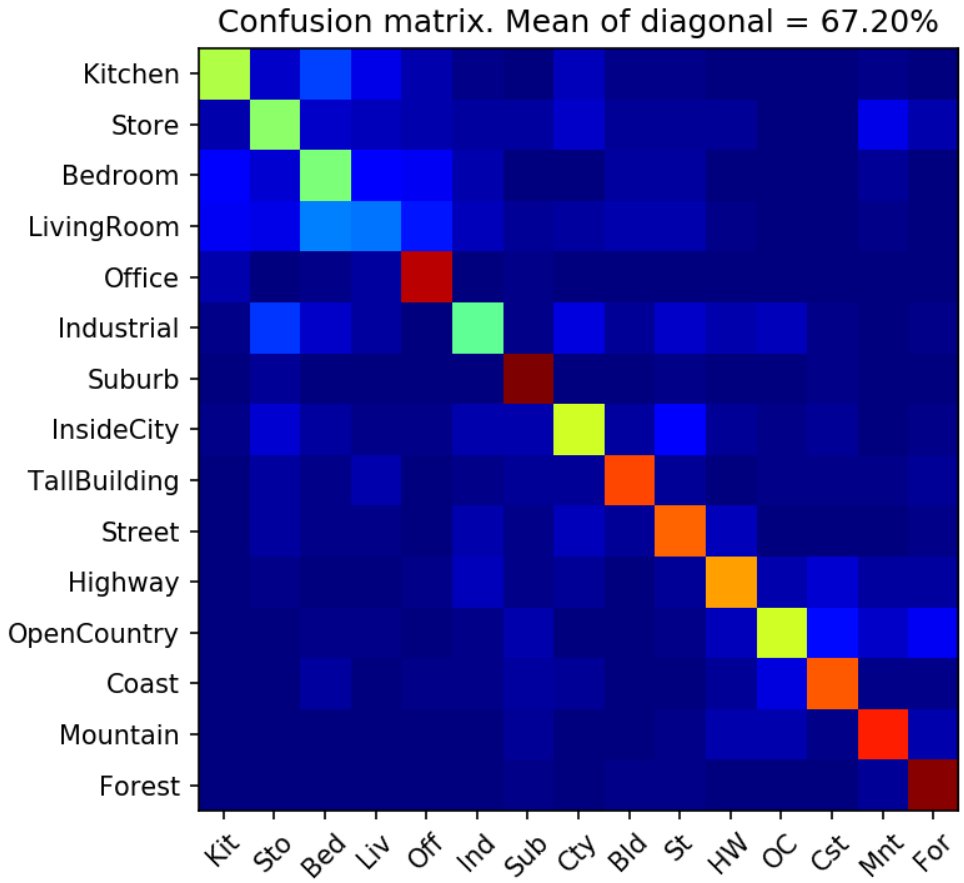
Vacab_size = 50 →Accuracy: 60.3%



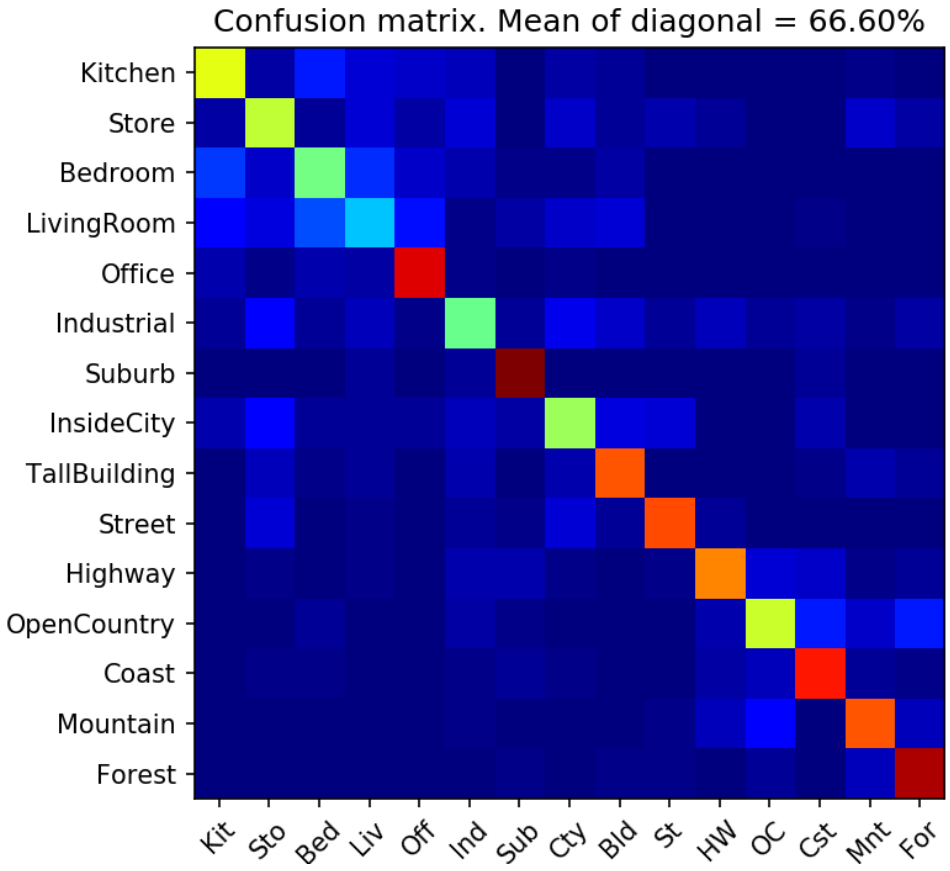
Vacab_size = 100 →Accuracy: 64.0%



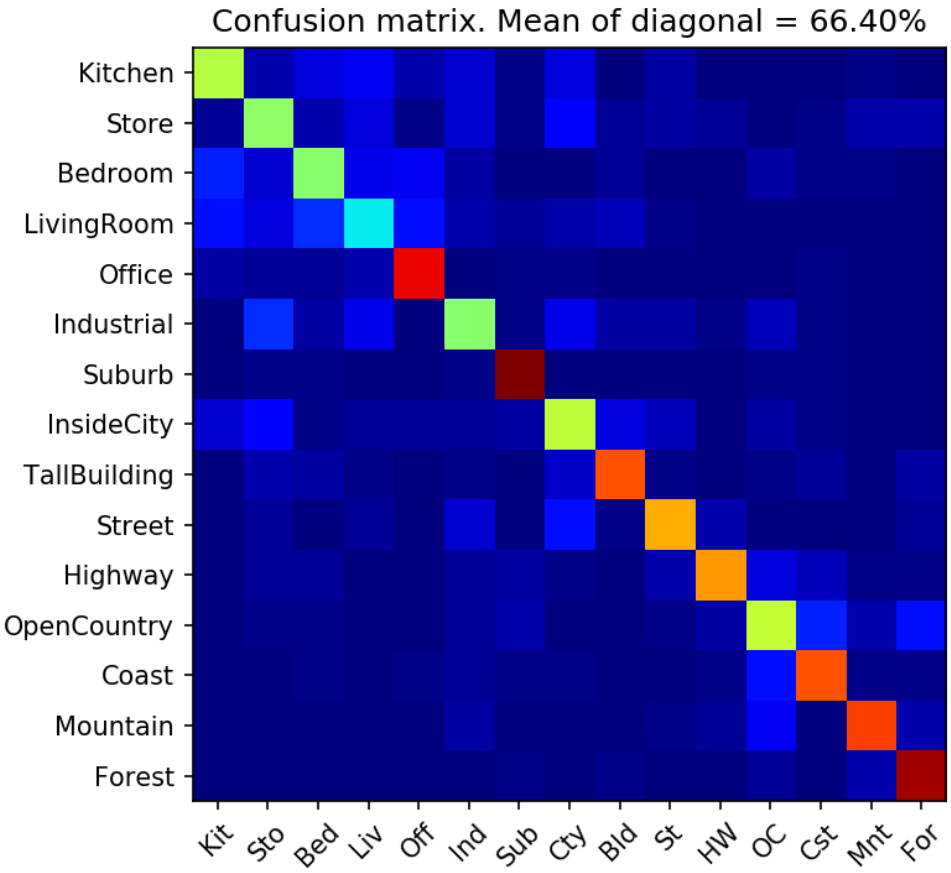
Vacab_size = 200 →Accuracy: 67.2%



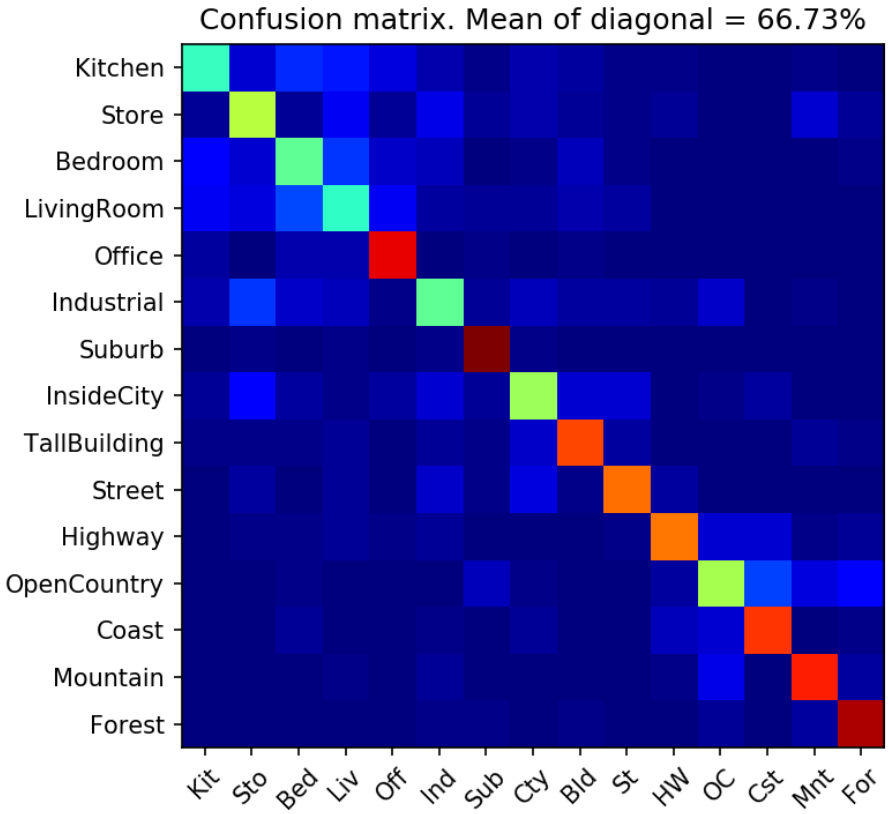
Vacab_size = 400 →Accuracy: 66.6%



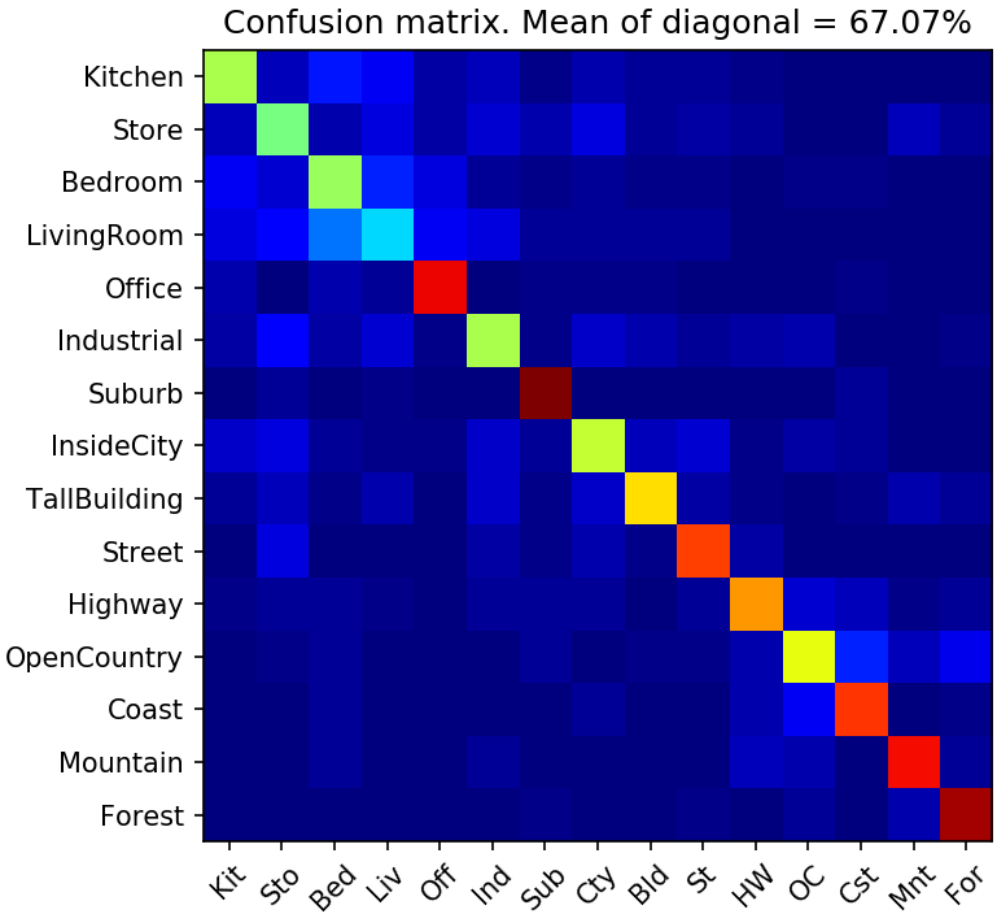
Vacab_size = 600 →Accuracy: 66.4%



Vacab_size = 800 →Accuracy: 66.73%



Vacab_size = 1000 →Accuracy: 66.73%



Experimental Design: Cross Validation

I created a cross validation plan to find the best hyper parameters associated with the best accuracy. I have performed 10 runs with 100 random training images for each run. According to my results:
Average performance: 49%
Standard Deviation: 4.14

References

[1]: S. Lazebnik, C. Schmid and J. Ponce, "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories," *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, New York, NY, USA, 2006, pp. 2169-2178, doi: 10.1109/CVPR.2006.68.
[2]: K. Chatfield, V. Lempitsky, A. Vedaldi, A. Zisserman The devil is in the details: an evaluation of recent feature encoding methods British Machine Vision Conference, 2011
[3]: Large image datasets: A pyrrhic win for computer vision?, anonymous authors, OpenReview Preprint, 2020.

Thank you!
Student Name: Behnam Moradi
Student Email: bmn891@uregina.ca