

Scene Recognition using computer vision techniques

Athithya Panchapakeshan Rajeswari
Graduate Student

Sunil Baliganahalli Narayana Murthy
Graduate student

Abstract

Scene recognition is a very interesting computer vision problem which has a very diverse set of applications like robotics and surveillance [1]. Scene recognition tasks has seen tremendous improvements recently due to extensive work on Convolutional Neural Networks(CNN) [ref]. One of the traditional and widely used technique to tackle Scene Recognition is the Bag of Visual Words(BoVW) approach which tries to segment the given scene into small image patches and then classify it. In this project we propose multiple solutions to a 15 categories scene recognition problem based on BoVW and CNN.

Introduction

The image's scene is defined as the physical setting of the environment where the image is captured. Images / photos are contain multiple objects like house, animals, people etc which collectively compose a scene like bedroom (composed of bed, pillow etc), street (cars, people, road etc), forest, highway etc. The usual computer vision challenges like lighting, scale, occlusion arise here as well which makes it difficult to get really good classification accuracies. Deciding the scene category of an Image can be quite difficult because of intrinsic image ambiguities. For example, there are lots of similarities between a office scene and a bedroom scene, both scene could have common objects like chairs, tables and couches.

Literature survey

A big part of the existing work is based on Bag of Visual Words [3]. Authors in [4] use SIFT features to extract key descriptors from the image on which they then apply the Bag of Visual Words model to create a vocabulary of visual words. Later, the training images are converted to visual word histograms which is then fed into a linear kernel Support Vector Machine(SVM) for classification. As an extension to the previous approach [5], a Gaussian Mixture model is applied to the k-means visual words cluster and then fed into the SVM classifier. Another large part of the existing literature is based on Neural Networks, Authors in [6] use a convolutional neural network to train and classify images into scenes. Authors in [6]

claim to have trained their neural network on a dataset containing 7 million images with 207 distinct categories. Similar approaches have been discussed in [8] that use a convolutional neural network to train and classify scenes.

Problem definition

Humans are able to categorize or recognize the image without any efforts, despite the fact that images vary in size, viewing angle, lighting, translated, rotated etc. The data is growing every second, it's humanly impossible to categorize all of the data (text, image etc). We need to exploit the power of computers to automatically recognize the image and extract information. However it's a challenge to makes the machine understand and identify images. The main goal of this project is to propose and implement a scene recognition engine that identifies images across 15 different categories. Also keeping in mind that the constructed model generalizes well to recognize unseen images.

Proposed Techniques

In this section we briefly discuss the proposed techniques for scene recognition. We use Bag of visual words or Bag of features (BoF as ML users call it) using K-Means clustering, Bag of visual words using Gaussian Mixture model and convolutional neural networks. The first two approaches use classical machine learning methods which takes in a set of features and cluster them into groups. So feature extraction is critical aspect for the first two techniques to work.

Feature Extraction

The feature engineering is critical to constructing a quality model. For the images, the features are extracted using SIFT (Scale invariant feature transform).



Fig-1. Original Image of bedroom



Fig-2. Extracted SURF features

The reason of choosing SIFT as a feature extractor is because SIFT features are invariant to scale, translate, and rotate. However, SIFT is computationally intensive, so as an alternative we

used another technique called SURF (Speeded up robust features). SURF is a local feature detector and descriptor. It can be used as feature extraction in object recognition, 3D reconstruction, image registration etc. SURF is claimed to be several times faster and robust than the original SIFT implementation [9]. The above image fig-2 shows the SURF features extracted.

Bag of Visual words using K-means

In this approach multi-scale features are extracted from the images across all 15 categories. Each feature are described using 128 dimension value and key points. Each feature describes a visual blob of the image also called visual word. These features are then fed into K-means to produce k-clusters, where k can be 400, 1000, etc. The k-cluster center are now used as a visual vocabulary and models the distribution of each cluster as a new codebook representation. K-means model codebook does a hard-assignment of feature to a particular cluster based on nearest neighbour cluster.

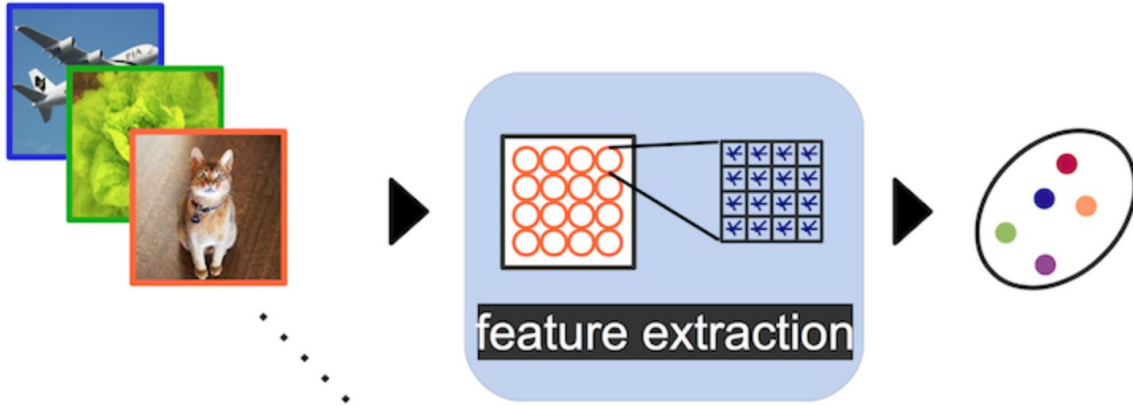


Fig-3. Learning Visual vocabulary construction

Once we have the visual vocabulary, we construct a discriminative model using supervised learning techniques like Linear SVM, probabilistic models(pLSA) etc. From a set of training images, SURF features are extracted and categorized into k-clusters.

K-Means clustering

Given a set of features $\mathbf{F} = \langle F_1, F_2, \dots, F_n \rangle$, where each feature is 128-dimension, k-means goal is to cluster them into k-clusters, $\mathbf{K} = \langle C_1, C_2, \dots, C_k \rangle$. So as to reduce the sum of squared distances within cluster centers.

$$\arg \min_{\mathbf{S}} \sum_{i=1}^k \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$

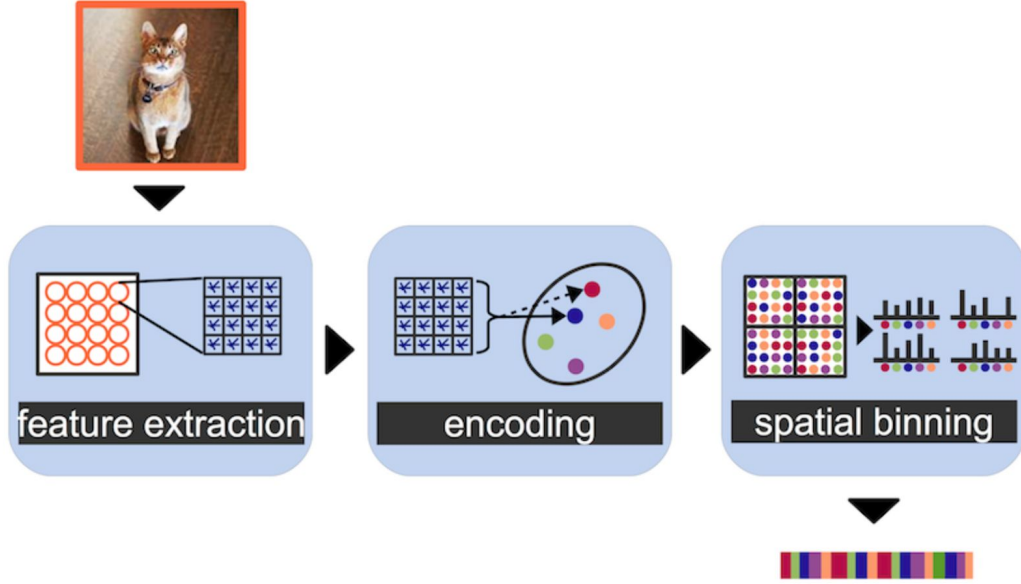


Fig-4. Classifying a scene

The cluster label frequencies are then summarized into k-bin histograms. These histograms and known scene category are then fed into support vector machine(SVM) with linear kernel to crunch numbers and create a model for classification. To classify an unseen image, features are extracted and clustered. These clusters are encoded in histogram and fed into SVM to recognize the scene label.

Bag of Visual words using Gaussian Mixture models

As explained in the previous technique, each image when fed into the SIFT/SURF algorithm, will give out a set of descriptors each of which is 128 in length. Let n represent the number of training images that we have. Let c represent the number of scene categories and Let d_i represent the set of descriptors for an image i . Therefore, the total number of descriptors that we have is:

$$D = \sum_i \sum_j d_{ij}$$

D is then fed into a k-means clustering algorithm. The number of clusters (a parameter) in the k-means algorithm will represent our visual word vocabulary size. Let k represent our visual vocabulary size. Each of the visual word cluster is then fit with a Gaussian Mixture Model having m components.

Each image i , from the training set, is again fed into the SIFT algorithm to get d_i descriptors which is scored againsts the k Gaussian Mixture Models (having m components). Therefore each image will now be represented as a vector with k scores. The Gaussian Mixture Model scores of the k^{th} cluster for the j^{th} descriptor in the i^{th} image are calculated as follows:

$$GMM_k(d_{ij}) = \sum_m gauss_k(d_{ij})$$

Where $gauss_k$ represents the weighted log probabilistic score calculated using the GMM parameters of the k^{th} cluster. The descriptor scores are then summed up together to obtain a k length vector for each image. These $n - k$ length training vectors are then fed into a Support Vector Machine with a linearized kernel.

To test an image, we first obtain its GMM scores and then feed it into the trained SVM which gives us an output category.

Convolutional Neural networks

In the previous approaches, feature selection is critical to building an accurate model. Hand picking the features from millions of images can be painful. Instead we can learn these features using Deep neural network. This section describes the two different approaches taken to construct model for scene recognition using Convolutional neural networks (CNN).

Theano and Lasagne

Lasagne is a deep learning library for python. It closely functions with theano which is a numerical and symbolic computation library which offers functionalities similar to Tensor Flow. A more detailed overview of Lasagne and Theano is given in these [10,11] articles. Eben Olsen [12] has created a very good tutorial on Lasagne and Theano which explains how to instantiate and create convolutional neural networks easily using these frameworks.

As opposed to traditional Neural Networks, Convolutional Neural Networks are designed to have lesser neuron connections without compromising on the effectiveness of training. In fact, Convolutional Neural Networks perform better than traditional neural networks when it comes to image classification tasks [6]. One of the main challenges of neural network training is that, we require a lot of computational power that we lack in this project.

We tried several convolutional network architectures with a series of convolution and pooling layers in this project. The simplest being, a convolutional neural network with a simple input transformation layer followed by a 64×64 convolution layer which convolves the input layer the resulting output is then fed to an activation layer and then subsequently to a 64×15 dense layer (a layer that connects every neuron to every other neuron in the previous layer). The output layer then consists of 15 nodes corresponding to 15 categories. Because of the simplicity of the network we got an accuracy of mere 14% on a trained model.

We also tried to train bigger network architectures in this project. The architectures were created but the training could never be completed because of computational and memory bottlenecks. A workaround to this problem is to use a pre trained neural network model. The next approach talks about using a pre trained model for our scene classification problem.

Tensor Flow and Inception model

In this previous approach we tried to build neural network from scratch with limited hidden layers. This model didn't yield good accuracy due to limited number of layers and training CNN with small subset of data. The key factor to create a good model using CNN are large number of hidden layers(convolution, max and spatial pooling, output) and training with tons of images. Since it's time consuming to create such a model from scratch we turned to using pretrained models.

Inception v3 is a Google's pre trained model on ImageNet - Large Visual Recognition Challenge using the data from 2012 using tensor flow. This model is trained on millions of images across 1000 of categories. Since tensor flow runs on both GPU and CPU it's much more faster than other approaches. The below figure shows the architecture of the inception, which contains various layers like Avg Pool, Max Pool, Dropout and Fully connected layers. Convolutional layers acts a image filtering kernels, Avg Pool and MaxPool downsize the image by performing subsampling, Dropout layer avoid any overfitting by selectively dropping some of the input and this acts a regularization parameter. Finally softmax layer gives the soft-assignment of the image classification across different categories.

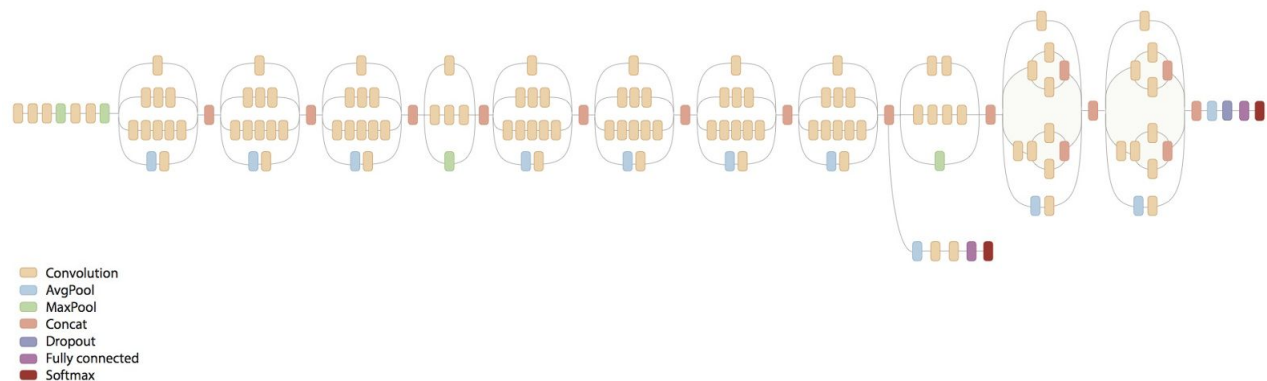


Fig-5. CNN architecture [13]

The pre trained model is trained on our image dataset and labels to recognize our scene. The way this information is imparted to the inception is through "*Transfer learning*". This has been done for object detection, zero-shot learning, image captioning, video analysis and multitudes of other applications. Fine-tuning the model is very important to get accurate model. Some of the parameters that can be fine tuned are number of convolutional layers, learning rate - which determines how to learn, number of iterations to loop before converging etc.

Contributions

Athithyaa

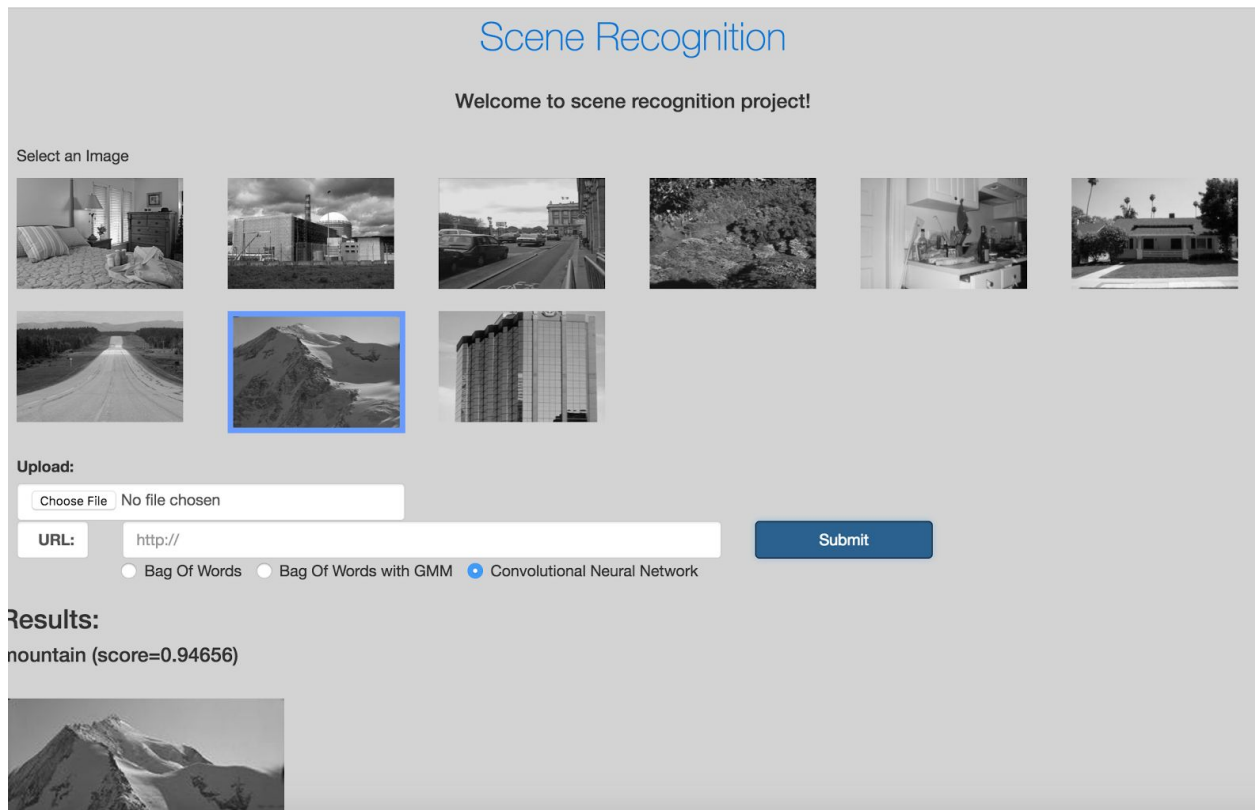
1. Implemented an improvised scheme for Bag of Visual Words model using Gaussian Mixture Models.
2. Brainstormed the initial Bag of Words approach based on a theoretical BOVW paper.
3. Implemented multiple Convolutional Neural Network Architectures using Lasagne and Theano.
4. Tried GPU optimizations for Theano and Lasagne.
5. Implemented a customized scoring scheme for the Bag of Visual Words model using Gaussian Mixture Models.
6. Minor changes to existing user interface to run our models.
7. Created visualizations and confusion matrix for Gaussian Mixture Models.

Sunil

1. Implemented Scene recognition using Bag of Visual words using K-Means to obtain a baseline of 38%.
2. Image preprocessing like resizing images to a fixed size and RGB to grayscale conversion before feeding them into CNN.
3. Feature extraction from images using SIFT and SURF detectors using openCV framework.
4. Fine tuning of parameters in K-Means (cluster size, batch size etc), SVM (kernel methods, penalty parameter etc) to improve the recognition rate to 63.08%.
5. Analyzing the effect of number of clusters on the accuracy of the BoW model.
6. Developed Visualization charts like histograms, confusion matrix and impact of number of clusters on accuracy of the model and data.
7. Implemented scene recognition using Convolutional neural network in Tensor Flow & Fine tuning the learning rate to obtain an accuracy of 89%.
8. Developed a web application for user interaction using Django and bower. Integrated with tensor flow scene recognition.

User Interface

Run the Django server from tensor flow virtual environment and navigate to the URL
<http://localhost:4000/recognition/>



The screenshot shows a web application titled "Scene Recognition" with a subtitle "Welcome to scene recognition project!". Below the title, there is a section labeled "Select an Image" displaying a grid of nine grayscale images. The images include a bedroom, an industrial facility, a city street, a rocky landscape, a kitchen, a house, a road, a snow-capped mountain (which is highlighted with a blue border), and a tall building. Below the image grid, there is an "Upload:" section with a file selection button labeled "Choose File" and the text "No file chosen". To the right of the file selection is a "URL:" label and a text input field containing "http://". Below the URL field are three radio buttons for model selection: "Bag Of Words", "Bag Of Words with GMM", and "Convolutional Neural Network" (which is selected). A blue "Submit" button is located to the right of the radio buttons. Below the "Submit" button, the "Results:" section displays the text "mountain (score=0.94656)" and a small thumbnail image of the selected mountain scene.

Fig. User interface for Scene recognition.

Infographics and Results

In this section we discuss the experimental results of the proposed techniques. Accuracy of the models, parameters that were used to tune the models etc.

Dataset

The original dataset “Tiny Image dataset” consists of 80 million images across 1000 categories with ~ 250GB size. In the following pages, we have some infographics explaining the results of our models visually. Most of the graphics given below are self explanatory.

We took only a subset of Tiny image dataset with contains ~8000 training images and ~3000 test images across 15 different categories like bedroom, forest, inside city etc.



Fig-6 Tiny Image dataset poster [2]

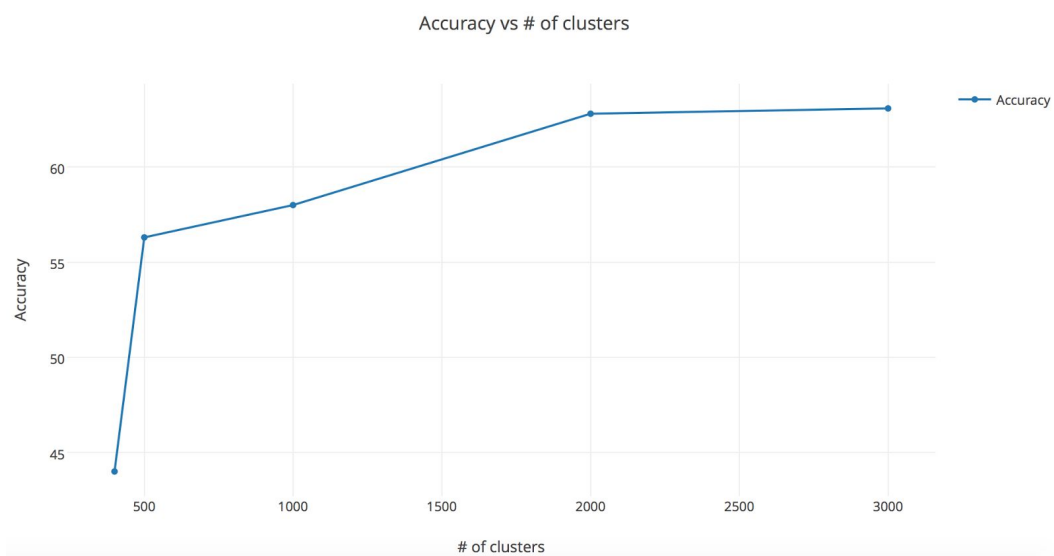
Accuracy

The below table summarizes the accuracy of different models and techniques we have tried.

Method	Accuracy
F ₀ :Bag of Visual words using SIFT	~39%
F ₀ + Normalization of histogram	~41.2%
Bag of visual words using SURF (descriptors of 64 dimension with orientation)	~44%
Bag of Visual words using Gaussian Mixture model with a custom scoring scheme	~53.00%
F ₁ : Bag of visual words using SURF (descriptors of 128 dim without orientation) + Normalization	~56.3%
Bag of Visual words using Gaussian Mixture model	~56.48%
F ₁ + 1000 clusters	~58%
F ₁ + 2000 clusters	~62.8%
F ₁ + 3000 clusters	~63.08%
ConvNets with inception tensorflow	~89.4%

Fine Tuning

Adjusting the number of clusters.



Given below is a category wise histogram representation based on our visual words vocabulary size 3000:

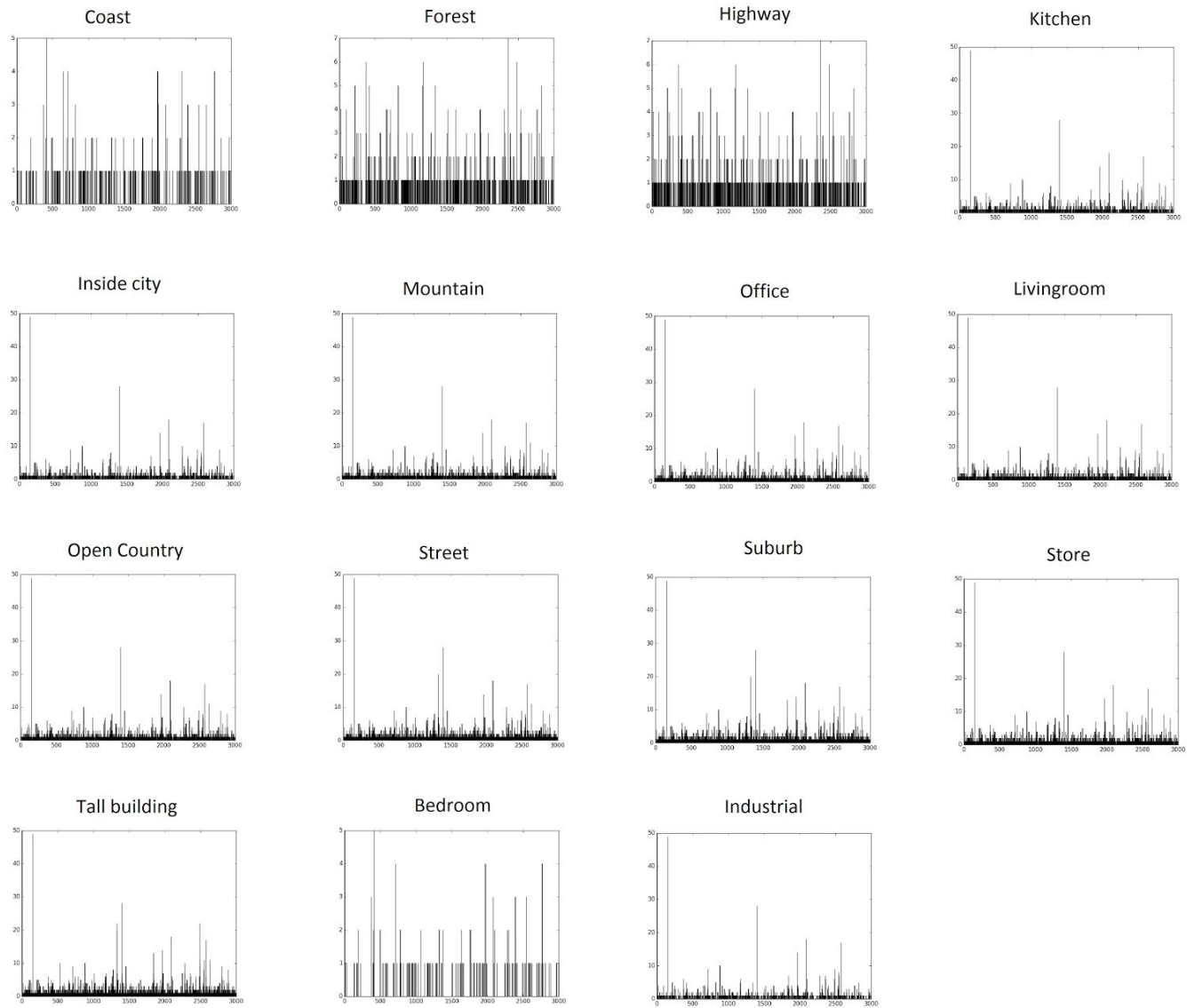


Fig-7. Histogram visualization across different categories

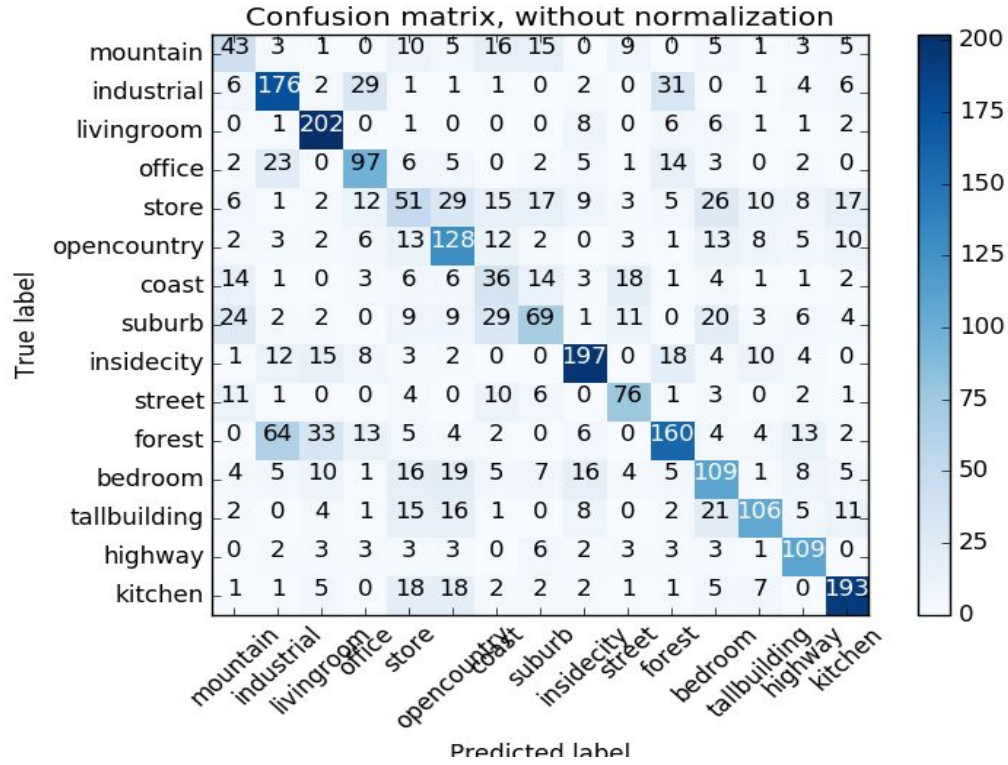


Fig-8. Confusion matrix for BoW with 1000 clusters

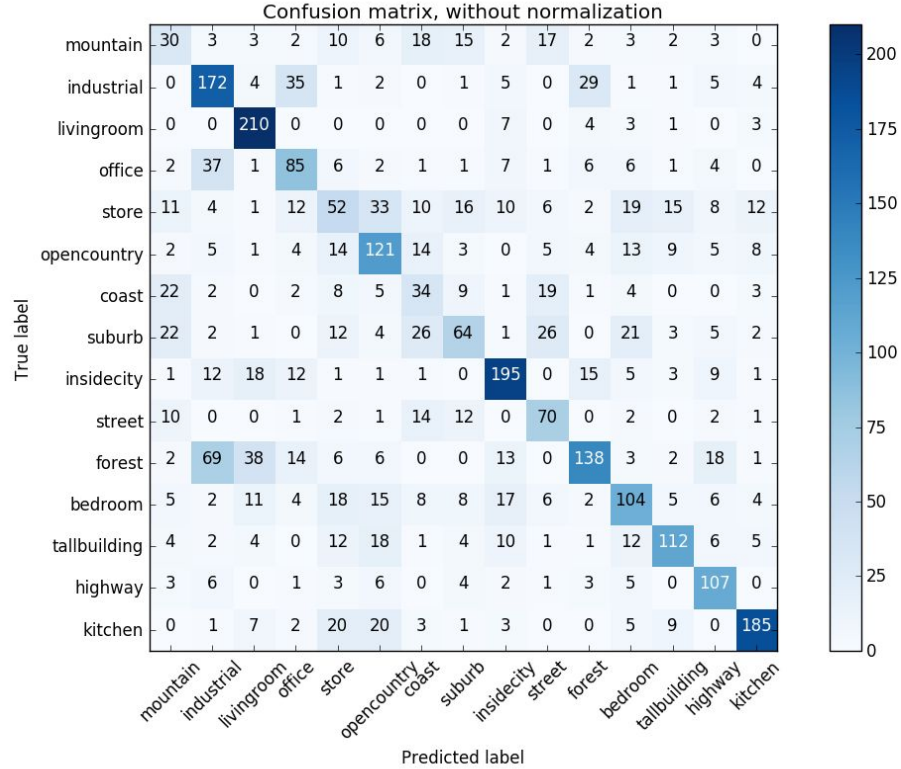


Fig-9. Confusion matrix for BoW with 400 clusters

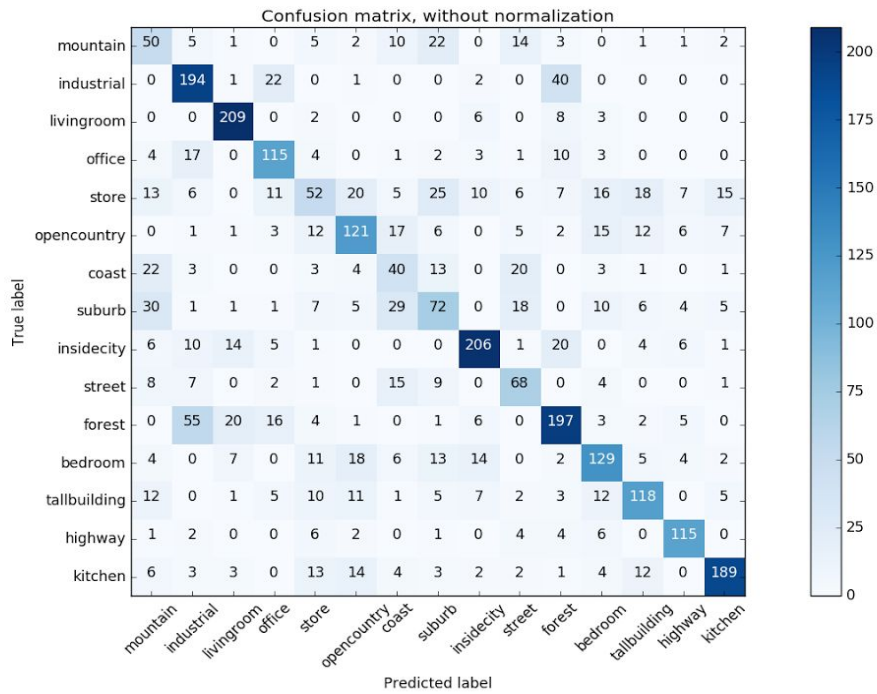


Fig-10. Confusion matrix for BoW with 2000 clusters

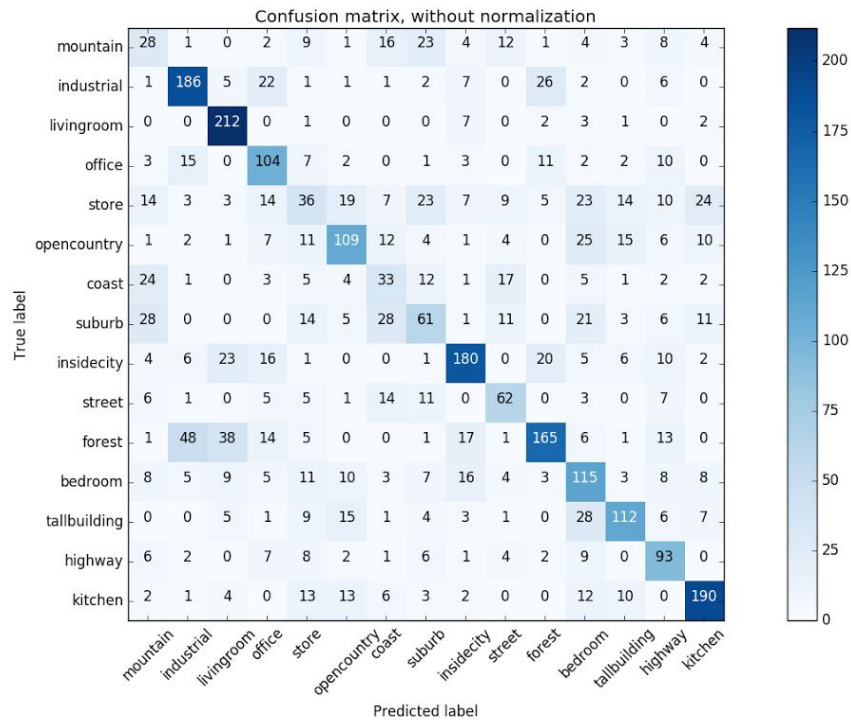


Fig-11. Confusion matrix for BoW with GMM

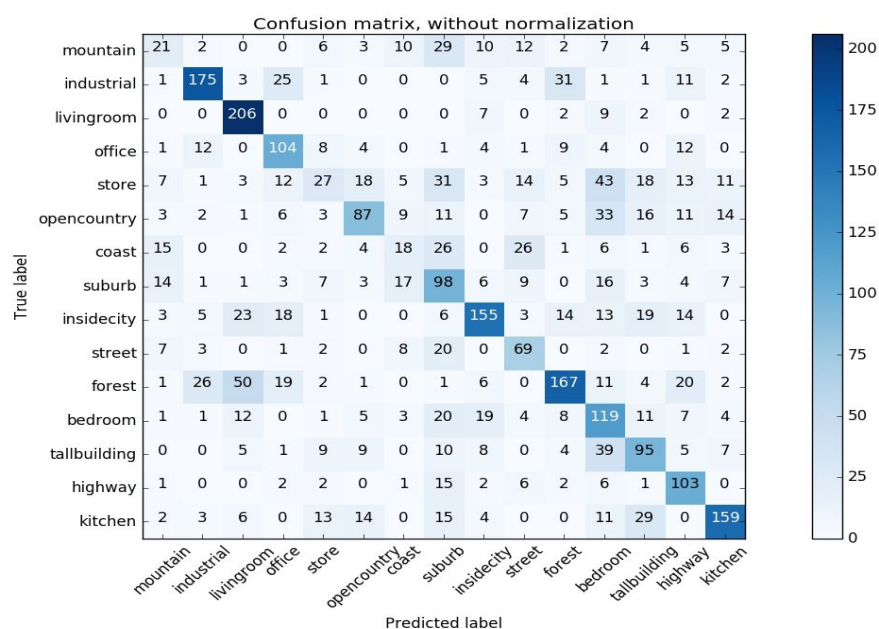


Fig-12. Confusion matrix for BoW with GMM with custom scoring

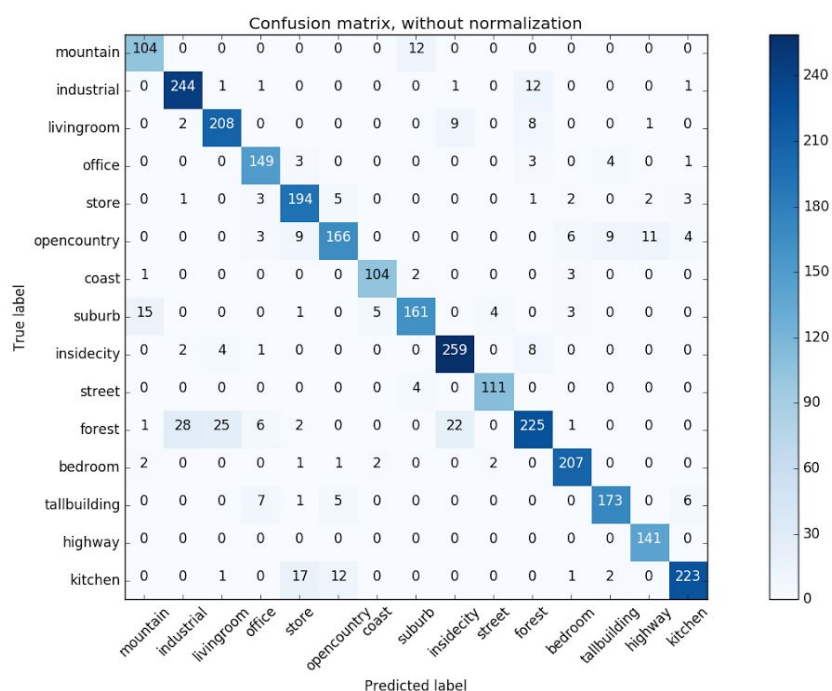


Fig-13. Confusion matrix for CNN using tensor flow

Conclusion

We tried several approaches to our scene recognition problem in hand out of which the approach with Convolutional Neural Networks seems to give the best accuracy of 89 %. This was expected since, Convolutional Neural Networks provide an abstraction over feature engineering. Much of the feature engineering process is taken care of the neural network itself therefore, there was not a lot of computer vision based feature engineering that happened for this approach. On the other hand, though our Bag of Visual Words model seems to give accuracies on the lower end, we employed a couple of computer vision based feature selection techniques like extracting SIFT & SURF feature and applying image clustering. Therefore, our project is a mixture of what we learnt in the class and also what's available outside the class.

Future work

As future work for BoW model, we can include features like color histogram, spatial pyramid matching etc along with SIFT/SURF features. In the BoW model, it's evident that images like forest were commonly misclassified. Color histogram of forest will have more green component than red, blue and thus can be used to infer some information about the scene. For convolutional neural networks, we can try with very deep neural networks and Residual deep neural network. According to this paper [14], microsoft ResNet (residual networks) are able to achieve an accuracy of 95%.

Instructions to run code

Please refer to the scene recognition github Wiki page for the detailed instructions on how to run the code. Github: [Scene recognition Wiki](#)

What we learnt?

- Image processing and manipulations techniques using OpenCV.
- Different techniques to extract local descriptor and features from the Images using SIFT(scale invariant feature transformation) and SURF(speeded up robust features).
- Constructing visual word vocabulary using K-Means and Gaussian Mixture models.
- Object recognition using bag of visual words
- Using Theano, Lasagne and Tensor Flow to create and train Convolutional Neural Network

Advice to next batch of students

Choose an interesting and a less challenging topic to start with. Then, as the project takes some shape incorporate more challenging ways to tackle the problem your project tries to solve. Small incremental steps to solve the problem will give you better results and organization. Once you feel confident with your project, explore and try out different techniques which are not taught in class.

References

- [1] [Scene Recognition using Bag-of-Words](#) by Sarthak Ahuja and Anchita Goel
- [2] [Tiny Image Dataset](#)
- [3] <http://ijsetr.org/wp-content/uploads/2016/01/IJSETR-VOL-5-ISSUE-1-64-68.pdf>
- [4] Gang, Liu, and Wang Xiaochi. "Improved Bags-of-Words Algorithm for Scene Recognition." Physics Procedia 24 (2012): 1255-1261.
- [5] Fernando, Basura, et al. "Supervised learning of Gaussian mixture models for visual vocabulary generation." Pattern Recognition 45.2 (2012): 897-907.
- [6] Zhou, Bolei, et al. "Learning deep features for scene recognition using places database." Advances in neural information processing systems. 2014.
- [7] [Computer vision project github Repository](#) by Athithya and Sunil
- [8] Russakovsky, Olga, et al. "Imagenet large scale visual recognition challenge." International Journal of Computer Vision 115.3 (2015): 211-252.
- [9] Wikipedia: https://en.wikipedia.org/wiki/Speeded_up_robust_features
- [10] <http://deeplearning.net/software/theano/>
- [11] <https://lasagne.readthedocs.io/en/latest/>
- [12] <https://github.com/ebenolson/pydata2015>
- [13] [Google research blog](#)
- [14] [Identity Mappings in Deep Residual Networks](#) by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun Microsoft Research