

Image Classification using Bag of Visual Words

Kranthi Kiran Chinnakotla
Computer Science ITCS-6156 student
University of North Carolina at Charlotte

Prasanna Kumar Rajendran
Computer Science ITCS-6156 student
University of North Carolina at Charlotte

Thomas Neil Kattampallil
Computer Science ITCS-6156 student
University of North Carolina at Charlotte

Siddharth Shankar
Computer Science ITCS-6156 student
University of North Carolina at Charlotte

I. INTRODUCTION

This paper explores the problem of Image Classification, specifically using the 'Bag of Visual Words' concept. In recent years the amount of image data that is being generated has increased exponentially. Applications like Instagram, Facebook Images and Snapchat have millions of users, all generating several million images every day. There are around 27,800 images uploaded every minute to Instagram, which represents data that can be harnessed. In order to better categorize, manage and obtain useful data from these images, we need to be able to classify the image data to recognize the objects, people or locations that the image contains.

In this paper we will be focusing on the problem of converting the problem of Image Classification, reducing it to a Bag of Words classification problem, which is a well explored area of Machine Learning with many approaches to solve it. This would allow us to apply multiple text classification algorithms and check the accuracy of prediction.

II. DATA PREPARATION

A. Selecting a Data Set

For the purposes of our Image Classification system, we needed to get several labelled test images, stored as their respective categories. In order to do this, we used the Caltech101 dataset. Caltech 101 is a popularly used dataset for image recognition that has 101 categories of objects, such as Airplanes, Motorcycles, Chairs, Eagles, etc. The images consist of scenes, consisting of a foreground containing the object as labelled, and a background which consists of ambient background elements. Since there are a variety of categories, some which may be visually similar and others that are visually distinct, it is a suitable dataset to measure classification accuracy between classes.

Other candidates for possible data sets included the Caltech6, rejected for being too sparse, and the Caltech 256, which was

not picked as the complexity and size of data set was exponentially larger.

B. Limitations of Data Set

The number of images in each of the category folders of the Caltech101 dataset range from 40 to 400. On average most of the data sets have a 100 images. This presents a problem because of the bias in learning that the unequal number of images might introduce into the model. Additionally, the number of categories in the data set is quite large, which is why we chose to work with a subset of the data set, 30 categories with 90 images each, out of which 60 images were put into the train set and 30 images were put into the testing set.

In several text classification problems, the number of documents in the training set, i.e. the number of rows in the bag of words, is often in the magnitude of several thousand whereas the training data set in this experiment consists of a total of 1800 images. If the number of images in each category in the train data set was increased, it would greatly increase the size of the training data set and boost the accuracy of the classifiers.

Additionally, the sizes and resolutions of the images in the data set are inconsistent, leading to an inconsistent number of features extracted from each of the images. This also introduces a measure of bias in the results and accuracy.

C. Components of the Data Set that are not needed.

The images are in color, but when the feature extraction algorithms always convert to grayscale, as the interesting features are related to gradient values rather than color values. Other feature extraction algorithms that extract color data would also have an increase in accuracy, and would probably be a good direction to consider in future iterations of such systems.

III. DATA PRE-PROCESSING

Each image is first converted to grayscale and then run through a feature extraction algorithm. Both MATLAB and Python have some functions that can extract image features. MATLAB through its inbuilt Image Processing Toolbox and Python through an external library called OpenCV. The specific algorithm used for feature extraction can vary, but our implementation utilizes the SURF features extraction algorithm (Speeded Up Robust Features). This extracts the points of interest in each image. In trials we have obtained similar accuracy values with HoG features as well (Histogram of Oriented Gradients). The MATLAB function for SURF allows for a selection of an upper limit of features and key points from each image. Once the key points are extracted, the SURF algorithm is run once more with the key points as an input, in order to extract the descriptors of each key point. A descriptor is a mathematical matrix representation of the local area around the keypoint locations in the image. These descriptors are the real numerical representation of the interesting features of an image. These descriptors are put together as a collection of features. This is the starting point for constructing the bag of Visual words.

D. K-Means Clustering

Once the features are all extracted, K-Means Clustering is used to group similar features together as Visual words. This is conceptually represented by the K number of clusters that is formed after K-means clustering is run on the image feature data set.

What this algorithm does is group similar features together into a 'word', forming a vocabulary or a dictionary of image features which can be used as a reference to classify other images.

To explain this in layman terms, consider each image feature as a single piece of a jigsaw puzzle. If the feature is sufficiently interesting, just looking at the image patch on a single piece of a jigsaw puzzle should be sufficient to let you understand what the whole image of the puzzle is. If not, the presence of a few other pieces would a person deduce what the image is.

If there are several categories with similar features, such as a wheel, which could be present in an image of a car, or a truck or a talon, which could be present in the picture of an eagle or any other bird. In order to simplify common features that are present in multiple images and in multiple categories, it makes sense to group the features into one 'Visual Word'.

E. Translating the images into the Vocabulary

Going back to the features in each image, we can identify which of the K clusters the feature corresponds to. For each image we can create a histogram of K bins, and increment the

value of the Kth bin every time an image feature from that image corresponds to the Kth cluster in the K-means cluster. In this way, we convert the image from a collection of features to a set of words with their corresponding features. Stacking these vectors results in a Bag- of -Visual Words.

IV. IMAGE CLASSIFICATION

The next step in our project is to classify the pictorial scenes. We have 30 categories and 60 images in each category, totaling to 1800 images. The main scope of this project is to evaluate through the various classification machine learning algorithms and conclude the one which is more appropriate of the Image classification, based on the parameters of the Confusion matrix.

Algorithms under consideration for Comparison and collaboration are:

1. KNN
2. Naïve Bayes
3. Support Vector machine.

F. Training Model using KNN:

The one of the algorithms we chose to solve this scene classification problem was KNN. For making KNN, we followed two different types, 1. The first one the naïve implementation using the Euclidean distance method and 2. second one is using the build-in library, train the model using training data point and data labels.

This is built by using the Naïve system like extracting the feature from all training image and forming a vector out of it and finding the K-means cluster and picking the centroid of all the feature vector and forming the Histogram for all the images, which constitutes the bag of visual words. This is the model for classifying the test images. With the test images, we need to extract the feature and calculate the Euclidean distance and find the minimum distance of the feature from the Bag Of visual words, calculate the histogram of this distance and the index of the maximum Histogram point gives the index of Label, which is categorized.

Then, after reconstructing the Image as features and kept the track of each features till the boundary of each category and calculating the Histogram based on this final one. This helped in increasing the accuracy to 16%. The grayscale was used initially, later when images were resized to a constant value helped in increasing the overall accuracy to 17. Tested the same to the different test images. This accuracy seems to be constant.

The change of k value in the K-mean clustering algorithm was an important factor that contributed to the performance.

G. Naïve implementation of KNN classifier:

The training dataset which will be the histogram of bag of visual words, will be holding the index information of the centroids. We need to fetch the test image and compute the feature vector and find the Euclidian distance on all the centroids of the training image, then return the index of the minimum distance. This index is used to retrieve the class from the training data set. This implementation is the naïve procedure. The overall running time is quite large for this and the accuracy is about 17%.

This implementation was done using the following in-built libraries:

We used the VLFeat package in the matlab libraries for

- vl_kmeans – A library for finding the K-means.
- detectSURFFeatures – For detecting the interesting features.
- extractFeatures – Extracting the strongest features from the feature space.
- vl_alldist2 – For finding the distance between the training histogram of bag of features and test image.

The dimensions of the features extracted using the SURF algorithm is 128 by 1, as it is the extended SURF features function that is utilized in order to get better feature accuracy. The standard option of 64 by 1 size of feature may also be used to speed up the performance at a cost to accuracy.

H. Training Model using Naïve Bayes

The other algorithm we using to solve this scene classification problem is Naïve Bayes.

For Naïve Bayes, we are using the python and OpenCV. The OpenCV is used to extract the image features and Python is used for actual computation of Model and validation. Once the image is processed and features are extracted, the resulting matrix is stored in the CSV format. Here the data preprocessing is done by removing predominant zero pixel components. Now, we have interesting regions which we can converted to the Input argument file for Python.

1. In python, Numpy and Pandas libraries are used for processing the input file
2. An array of input vectors and respective labels for training the model in Scikit learn. We used Gaussian model of Naïve Bayes.
3. Then the test images are even processed the same way as the training images and classes are predicted using the model designed in the previous step.

I. Training Model using SVM:

The other algorithm we using to solve this scene classification problem is SVM.

For SVM, we are using the python and OpenCV. The OpenCV is used to extract the image features and Python is

used for actual computation of Model and validation. Once the image is processed and features are extracted, the resulting matrix is stored in the CSV format. Here the data preprocessing is done by removing predominant zero pixel components. Now, we have interesting regions which we can converted to the Input argument file for Python.

1. In python, Numpy and Pandas libraries are used for processing the input file
2. An array of input vectors and respective labels for training the model in Scikit learn.
3. Then the test images are even processed the same way as the training images and classes are predicted using the model designed in the previous step.
4. The value of C was kept as 1 and the kernel used is linear.

J. Difference in the Performance of each Classifier:

The KNN was out performing the other two classifiers for the fact that the data was not linear enough to get good accuracy in SVM. Naïve Bayes and KNN were almost neighborhood in the accuracy rating. The detail analysis of the results are performed in the later part of the report.

K. Dimensionality reduction:

As we are dealing with the images, the vector dimensionality reduction of the feature size was done using Principal Component Analysis.

Steps:

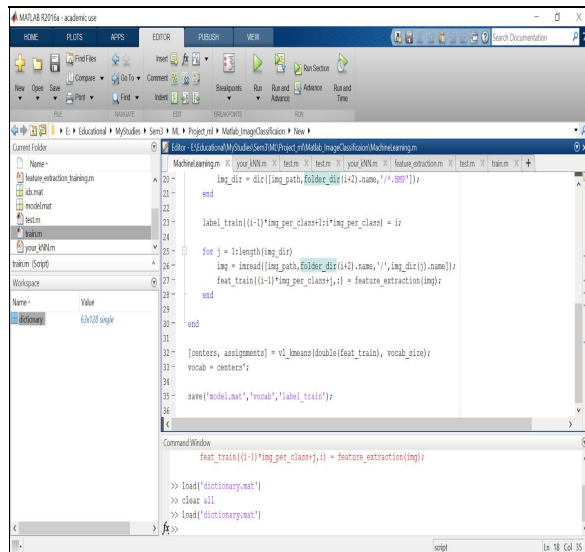
1. Resize the image to be uniform and crop the image to 15% X and Y axis.
2. Represent every image as a vector space
3. Compute the average of the vector space which will be the average vector
4. Subtract the Mean face vector from each image vector
5. Compute the covariance matrix using `cov(image)`;
6. Compute the Eigenvectors of the covariance matrix and find the eigen vector feature which will be stored as the model in this approach.
7. Now, these Eigenvectors will be fed as the training vector to the model.
8. Prediction of the testing image is on this Model.

As this is the lossy reduction, there is a depreciation in the overall accuracy of the model. However, the time taken to complete the computation reduced.

So, as the number of training set per category is less, we planned to not used PCA reduction for this data set.

L. Sample Code snippets:

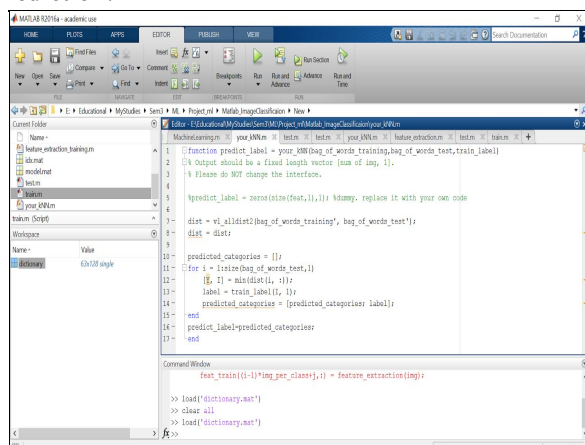
KNN Training:



```
function [train_dir, test_dir] = get_dirs(folder_dir, num_imgs)
    img_dir = dir(fullfile(folder_dir, '*.*'));
    label_train = zeros(1, num_imgs);
    for j = 1:length(img_dir)
        img = imread(fullfile(folder_dir, img_dir(j).name));
        feat_train(j,:) = feature_extraction(img);
    end
    [containers, assignments] = vl_learner(double(feat_train), vocab_size);
    vocab = containers;
    save('model.mat', 'vocab', 'label_train');
end

% Command Window
>> feat_train(i-1)*img_per_class; i = feature_extraction(img);
>> load('dictionary.mat')
>> clear all
>> load('dictionary.mat')
```

Model Prediction:



```
function predict_label = your_KNN(img, your_KNN, bag_of_words_training, bag_of_words_test, train_label)
    % Predict label for a single image (img) using the trained KNN model.
    % Please do NOT change the interface.
    % Predict label = zeros(size(img,1)); % dummy, replace it with your own code
    dist = vl_distance2(bag_of_words_training, bag_of_words_test);
    dist = dist;
    predicted_categories = [];
    for i = 1:size(bag_of_words_test,1)
        [I, J] = min(dist(i, :));
        label = train_label(I, J);
        predicted_categories = [predicted_categories label];
    end
    predict_label = predicted_categories;
end

% Command Window
>> feat_train(i-1)*img_per_class; i = feature_extraction(img);
>> load('dictionary.mat')
>> clear all
>> load('dictionary.mat')
```

Thus, we have implemented the image classification using the three Algorithm and tried to compare and collaborate the results from them. To seek the best classifier for the Scene recognition, which is our primary problem statement.

V. ALGORITHM TUNING AND FEATURE ENGINEERING:

Our main goal of the project was to try out multiple possible algorithms available for image classification and figure out the algorithm that had performed the best of all.

We had 2700 sample of images and 30 classes for all these images. So, we had to do the cross fold on the train data and generate cross validate train and cross validate test dataset from the original train data. We had the cross fold split for 4 times and measured the accuracy by applying Naïve Bayes,

SVM and KNN, we did this before we actually applied the algorithm on the test data set.

Among all the algorithms we had implemented for the image classification, the SVM with Gaussian kernel had given good accuracy, followed by KNN, SVM with linear kernel and the Naïve Bayes algorithm.

Further research led us to additional approaches. Apparently the deep learning technique would give better performance compared to any of these algorithms, but the running time for this algorithm was more. To implement the convolutional network and deep learning, we had to do the transfer learning methodology, where already trained features are applied to the dataset, instead of training the features for images from the scratch. The transfer learning or the already trained features could be extracted using an image net or google net. This approach could be implemented in a future version of the system.

We couldn't use the features extracted for the naïve Bayes algorithm, as all the images didn't have same size of features, because of the variation in the sizes of the images. We had to do some engineering on the features that we extracted. OpenCV Python library was used for this project, to extract the features we had used SIFT from openCV. The 400 features were extracted from the images using SIFT.

After the features are extracted we had followed two methods to apply the data for our algorithms,

1. Bag Of Words K-means clustering approach.
2. Extracting all the features and converting them in to 1D array and fit the model with the labels that are provided by the dataset.

The first approach was unsupervised learning approach and the second approach was supervised learning approach. We will discuss below in details about the two approaches:

Approach 1: Extract the key points and descriptors from each image (gray scale), add the descriptors to the BOWKmeans, generate clusters using all the descriptors. Create a vocabulary of words, each word is a centroid of the cluster. While creating the clusters we had changed the K Value, the accuracy had increased as we increased the K value till 1000, but it took more time to run and fit the model. Once the centroids and clusters are generated, we had applied SVM(non-linear kernel) and got the accuracy on the validation datasets and test datasets.

Approach 2: This approach was mainly for the Naïve Bayes algorithm and SVM LinearSVC algorithm. The features extracted from the openCV SIFT package was not the same for all the images and they were in a 2 dimensional array format. When we did the PCA to take the important features

only, we couldn't get much accuracy, so we had decided to consider all the features for the images. We had exported all the features in to a ".CSV" file, using "numpy" library in python. In the .CSV file for the rows (representing images) which had fewer columns (representing features), we had replaced the cells with "0" s. Imported the file using another python powerful library "pandas". Now we had equal features for all the images, for the labels which were given with the datasets are read directly. The number of labels were same for the train and test dataset.

We had repeated the same process for the test dataset, to normalize the features. Once we had the train dataset and test dataset and their respective labels, we had applied the Naïve Bayes and SVM linear SVC approach.

Conclusions and Results

For our experiments our team took 101 Caltech data which contains 101 classes each consisting of 60 images. It totals to 1800 images in the training set and the same in the testing set. After performing the three different classifiers and extracting the features in the bag of words format, the accuracy percentage is highest in the KNN followed by SVN Linear Classifier and Naive Bayes Nearest Neighbor Classifier.

To get the idea of the performance of these classifiers the experiment is done on less number of classes and less number of images. In case the number of classes are two containing 60 images each and the validation set has the same number of images as the training set then it leads to an overfitting case with greater than 87% of accuracy.

It has also been found out that if there is a data in which a specified portion of the image is priorly known which has to be taken for consideration by the algorithm then it produces quite high accuracy. On comparing with our previous results the accuracy enhances to almost double the value which we are getting in our dataset.

Looking at these empirical values, there can be a hypotheses that if the number of classes are less to categorize in comparison to the number of images then it leads to a good model and eventually a much better accuracy can be produced. Similarly if the number of images are too high compared to number of classes. For instance, Naive bayes with gaussian model was applied using the bag of words concept by our team in which there were 20 classes and approximately 52000 samples with 11000 features. The number of samples are quite high compared to the number of classes. In this scenario, the model is trained quite well which can identify true positives for more than 90 samples out of every 100 samples.

REFERENCES

The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

[1] VM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition.

Summary: This paper discusses the hybrid approach of using KNN and local SVM that preserves the distance function on the collection of neighbors

[2] Naive Bayes Image Classification: beyond Nearest Neighbors

Summary: This paper discusses that the Naïve Bayes with KNN is the powerful, non-parametric approach, this is mainly because of not using vectorization.

[3] Classification of fruits using computer vision and a multiclass support vector machine.

Summary: This paper refers to idea of using KNN with SVM on subset of image data, multi label classification.

[4] Comparing Image Classification Methods: K-Nearest-Neighbor and Support-Vector-Machines- BYUNG-SOO KIM², SILVIO SAVARESE³ Department of Electrical Engineering and Computer Science University of Michigan Ann Arbor, MI 48109-2122

In this paper, we used a general Bag of Words model in order to compare two different classification methods. Both K-Nearest-Neighbor (KNN) and Support-Vector-Machine (SVM) classification are well known and widely used

[5] In Defense of Nearest-Neighbor Based Image Classification - Oren Boiman(The Weizmann Institute of Science Rehovot, ISRAEL),Eli Shechtman(Adobe Systems Inc. & University of Washington),Michal Irani(The Weizmann Institute of Science Rehovot, ISRAEL)

In this paper they propose a trivial NN-based classifier – NBNN, (Naive-Bayes Nearest-Neighbor), which employs NNdistances in the space of the local image descriptors.

[6] Classification of Images Using Support Vector Machines - *Gidudu Anthony, * Hulley Greg and *Marwala Tshilidzi *Department of Electrical and Information Engineering, University of the Witwatersrand, Johannesburg, Private Bag X3, Wits,

2050, South Africa.

In this paper, the two approaches(One-Against-One (1A1) and One-Against-All (1AA) techniques) are evaluated in as far as their impact and implication for land cover mapping. The main finding from this research is that whereas the 1AA technique is more predisposed to yielding unclassified and mixed pixels, the resulting classification accuracy is not significantly different from 1A1 approach

[7] Some images courtesy Mathworks.com

<https://www.mathworks.com/help/vision/ug/image-classification-with-bag-of-visual-words.html>