Bag of Visual Words

Computer Vision Project 2

Authors:

Neil de la Fuente

Daniel Vidal

27/01/2023

Introduction

The Bag-of-Visual-Words (BoVW) is a method for image classification that is commonly used in computer vision and image processing. The basic idea behind BoVW is to represent an image as a histogram of visual words. A visual word is a group of similar feature descriptors of an image, such as SIFT descriptors. The histogram represents the frequency of occurrence of each visual word in the image. The histograms of images about different objects are supposed to have very different shapes.

The BoVW process consists of several steps. First, a set of local feature descriptors, such as SIFT, are extracted from each image. These descriptors are then quantized into visual words using a clustering algorithm, we used k-means, even though we also checked how it works with Gaussian Mixture Model but it was very inefficient, computationally speaking. The resulting visual words were then used to create a histogram of visual words for the image. This histogram represents the frequency of the words in the image and can be used for image classification.

In our project, we used the BoVW method to classify images into one of the 10 classes: ['person', 'cat', 'bird', 'horse', 'aeroplane', 'bicycle', 'sofa', 'train', 'chair', 'bottle']. The dataset that we used to train and test our classifier consisted of images of these classes. So we used a combination of feature extraction, quantization, and classification techniques to create our final classifier.

Dataset:

The dataset we used comes from the PASCAL Visual Object Classes challenge page and contains thousands of images of persons, cats, birds, airplanes, bicycles, bottles and many objects; it also have an xml file for each image that describes its content and gives the label of the objects inside each image. The dataset provides much more things and information but the data we are interested in is the original images and their corresponding xml files to load the images and link them with their labels.

Loading and Scrubbing the data:

In order to make this project successful, first of all, we had to open the data, which turned out to be a whole journey, we tried several things but finally, we found a couple of key tools, Posixpath and Element tree for reading the xml files. We were working in Google Colabs in a jupyter notebook so we loaded the data in our Drives so we can upload the images and the xml files in the notebook. We use the xml files to know the image name of each image and the label of the objects contained in the images.

We got to open it, but the data was so unbalanced, there were thousands of persons and just 163 horses, so we normalized that and worked using just 1630 images, 163 per class. We divided the 1630 images with their own labels into a train and a test set with an 80-20 rate.

Then we created a dictionary for train images and another one for test images, In each of them, the key would be the image identifier (i.e. ‘image_002’) and the value is a list where we would store data about that image, i.e. the array of the image, the label, and also descriptors(which, later on, will be created).

Our Bag of Visual Words algorithm:

Calculating the descriptors:

For computing the descriptors of each image we used SIFT, Scale Invariant Feature Transform helps locate the ‘keypoints ‘ of the image, describing the ‘keypoints’ themselves and their surroundings. These ‘keypoints’ are scale & rotation invariants that can be used for our goal in BoVW.

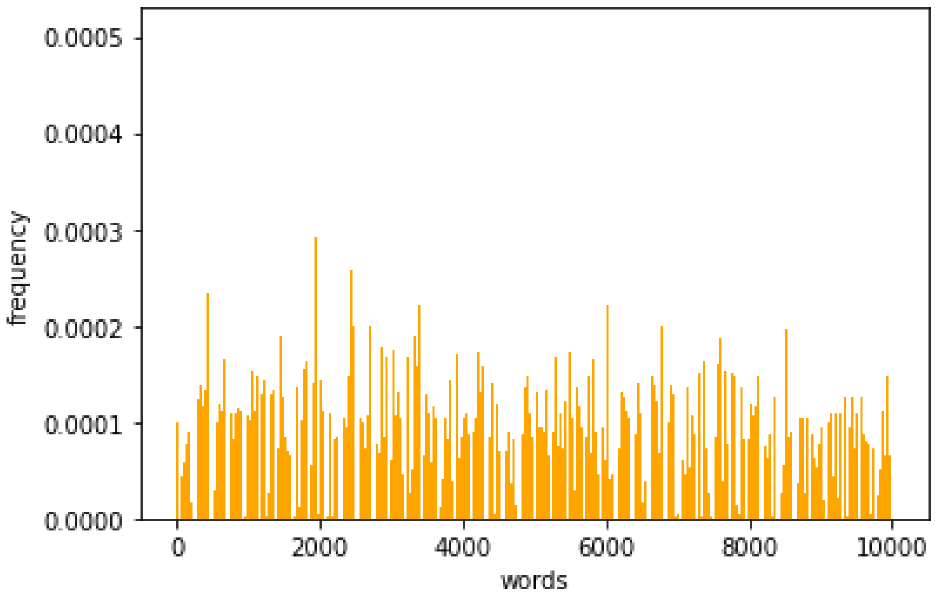
So we first passed the images to Grayscale and then, using SIFT as explained, we got the descriptors for both training and test images and appended them to the train_images and test_images dictionaries.

Finally, we had to concatenate all the descriptors that had the same class, resulting in a train descriptors dictionary with labels as keys and all the descriptors of the given class as values.

Calculate the Vocabulary

For the generation of the vocabulary we decided to make a big Vocabulary using directly all the descriptors of the training that consists of 10 thousand words in order to have no repeated words in different classes by calculating a mean histogram per class and concatenating them. We used Mini Batch Kmeans to generate the words, an algorithm that is much faster than Kmeans and was useful to see if our code was working well without waiting too long for the execution.

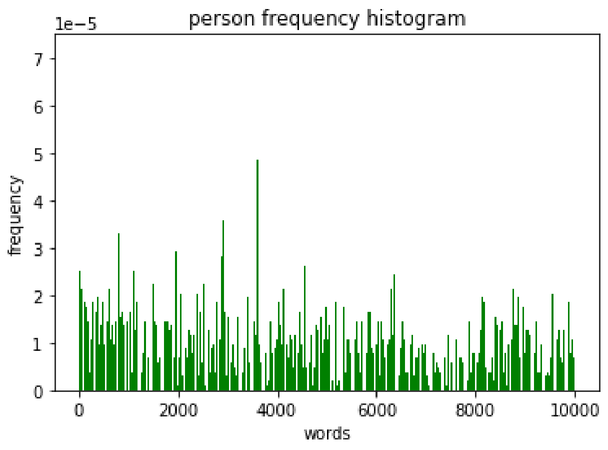
With this Vocabulary of 10 thousand words we made a normalized histogram diving the histogram by the number of descriptors in the whole training set just for visualization about how the frequency of all the words looks like:

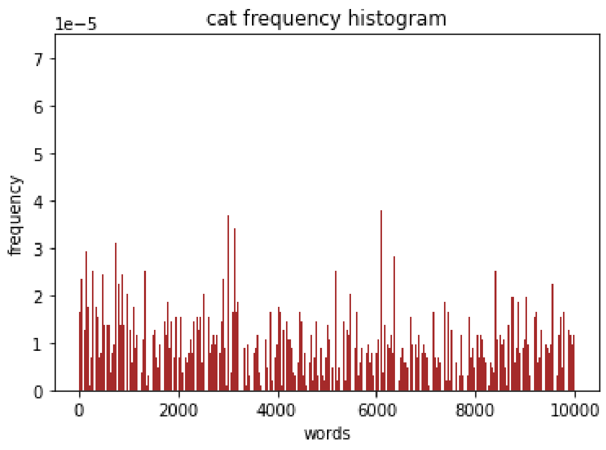


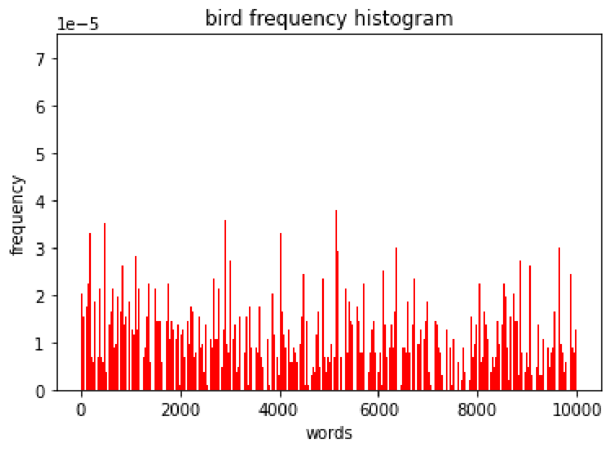
Generate the Histogram of each class

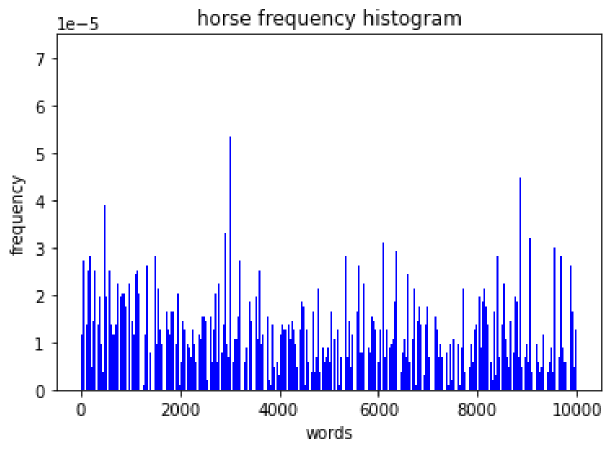
To generate the histogram of each class we used all the descriptors of each class to predict the words of each class using the mini batch kmeans so we have a word per each descriptor in the training set. Then, we count how many times each word appears to calculate the histogram and we normalize them by dividing by the number of total descriptors in the whole training set.

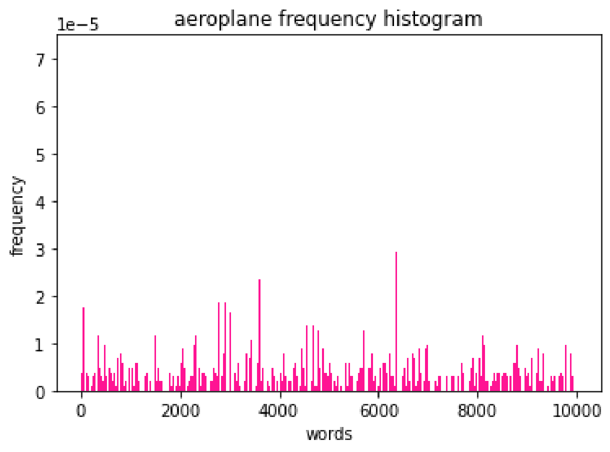
The Histograms that we obtained were the following:

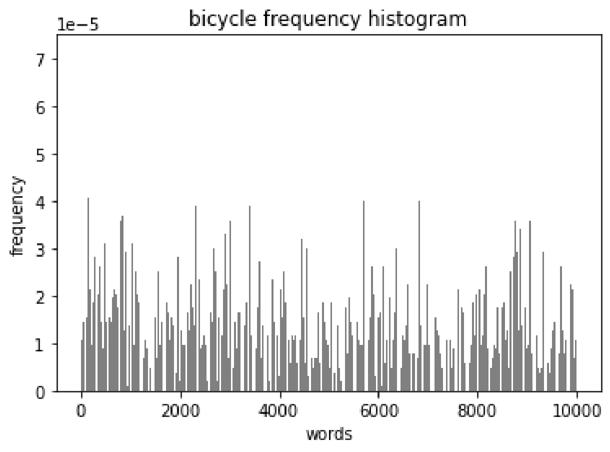


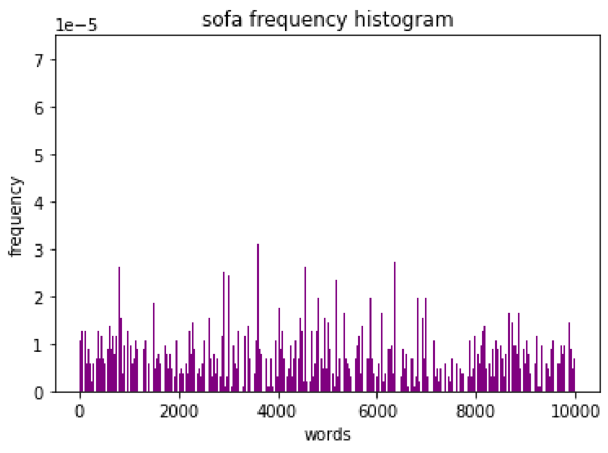


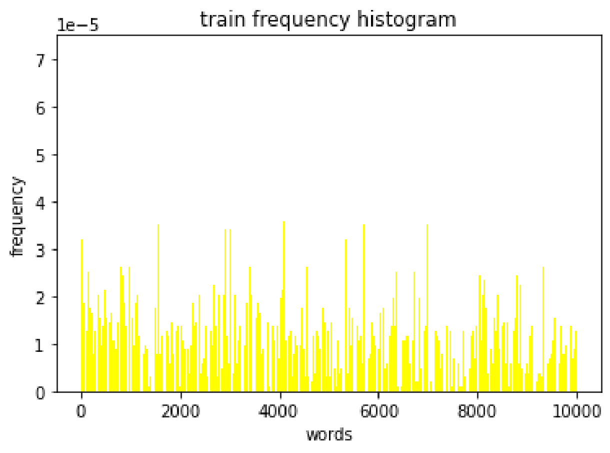

bird frequency histogram
1e−5
frequency
words

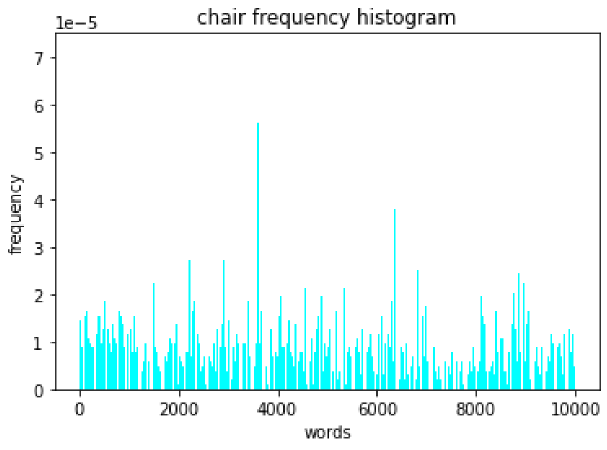

horse frequency histogram
1e−5
frequency
words

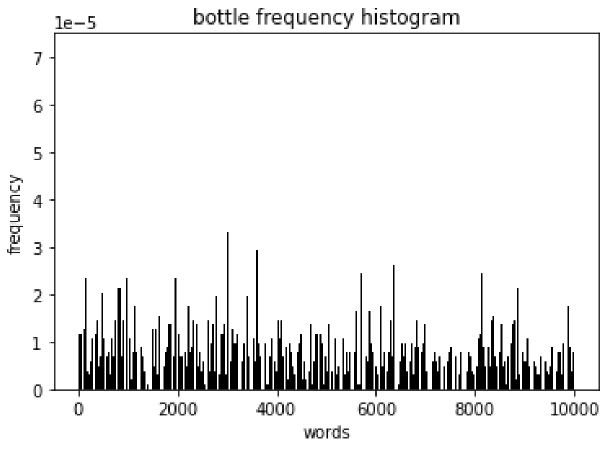

aeroplane frequency histogram
1e−5
frequency
words


bicycle frequency histogram
1e−5
frequency
words


sofa frequency histogram
1e−5
frequency
words


train frequency histogram
1e−5
frequency
words





Testing and Classification

For the classification we have to calculate the histogram of each test image and compare ist histogram with all histograms of the classes and the resulting class will be the one in the histogram that is more similar to the normalized histogram of the test image. The normalization for the test histograms was done dividing the histogram by the total number of descriptors in that image. The comparison can be done using multiple different metrics such as euclidean distance, cosine similarity, histogram intersection, manhattan_distance and many other metrics. The one that we chose was cosine similarity as it used to provide the best results in accuracy.

The performance of the algorithm was measured using a confusion matrix, precision, recall, F-1 score and so on. The results we obtained were the following:

```
[[12  1  3  4  0  0  2  0  2  3]
 [ 0 19  2  3  0  2  4  0  0  6]
 [ 3  7  8  0  3  3  9  3  0  2]
 [ 0  0  0  9  1  6  5  5  2  1]
 [ 1  0  2  1 10  2  6  2  3  2]
 [ 2  3  3  2  2 15  2  2  3  2]
 [ 0  5  6  0  5  1 10  1  3  2]
 [ 2  2  1  2  3  2  7  3  4  3]
 [ 1  4  2  2  5  6  4  3  8  4]
 [ 2  3  0  1  0  1  4  1  2 16]]

This is the full report:
              precision    recall  f1-score   support

   aeroplane       0.52      0.44      0.48        27
     bicycle       0.43      0.53      0.48        36
        bird       0.30      0.21      0.25        38
      bottle       0.38      0.31      0.34        29
         cat       0.34      0.34      0.34        29
       chair       0.39      0.42      0.41        36
       horse       0.19      0.30      0.23        33
      person       0.15      0.10      0.12        29
        sofa       0.30      0.21      0.24        39
       train       0.39      0.53      0.45        30

    accuracy                           0.34       326
   macro avg       0.34      0.34      0.33       326
weighted avg       0.34      0.34      0.33       326
```

Conclusions

In conclusion, the use of Bag of Visual Words (BoVW) techniques for image classification is a powerful and widely used method in computer vision. The BoVW method is based on the idea of representing images as a histogram of visual words, which are obtained by clustering local features extracted from the images, as we explained.

In this written work, we have explored the use of BoVW technique to classify images. We have implemented the method step by step, starting with the extraction of local features from the images, then clustering the features to obtain the visual vocabulary, and finally creating a histogram of visual words for each image and using it as a feature vector for image classification.

The knowledge and experience gained from working with BoVW techniques has been very pleasant. The method is simple, yet powerful, and it has proven to be robust and reliable for image classification tasks.

Additionally, BoVW technique can be combined with other image processing techniques such as feature extraction, dimensionality reduction, or machine learning algorithms, such as SVM, to improve the performance of the classification task.

In summary, BoVW techniques are a valuable tool for image classification and a good starting point for working on computer vision projects. It's a simple yet powerful method that can be used to obtain good results in a variety of applications.

Experience

This project was a very challenging experience for us. We have never had to face a dataset that contained many Gigabytes of images and information and algorithms that require a high computation time so we had to organize our code very well and think very carefully our ideas so we don’t waste too much time executing useless code. But, we both are so grateful to have done this project because, even if our algorithm does not have a very high accuracy , it gives us a very useful experience, knowledge about computer vision and programming skills that will be useful for sure in our Artificial Intelligence degree and our career path.

Also we want to say that accuracy is not all for us, the experience has been much valuable than that, also we wanted to keep loyal to the initial Bag of Visual Words approach, from the beginning and up until the end, we did not want to use classifiers such as SVM´s (which has already been applied by us in the Machine Learning subject project), we decided to keep it real and make histogram comparison to classify, which maybe is not as accurate but it is certainly comforting, and we are truly proud of our job, which has been entirely done by us, function by function and line by line.