University of Macedonia

Applied Computer Science

Assignment for BoVW(Bag of Visual Words)

Konstantinos Pasvantis

November 24, 2022

Contents

List of Figures

1 Introduction

Bag of visual words is a widely known algorithm for image classification. Given a dataset of images, this algorithm finds the image features, the patterns that we observe in every image. Image features consists of keypoints and descriptors, two concepts that we will disscuss later on this assignment.

The goal is simple: to implement the bag of visual words algorithm with different paratmeters in a dataset of my choice, to find the features of every image and classify them using histograms. The word parameters here should not lead to confusion. First we have to implement it with the help of the most common feature extractor, namely SIFT, and then with the ORB extractor. As we will discuss later, we need to use a clustering algorithm to create vocabularies in order to classify the images. In this assignment we must use two diferent clustering algorithms, the first being MiniBatch K-means and the second being a clustering algorithm of our choice. Furthermore, we have to use Bag of Visual Words (we will refer to it as BovW from now on) algorithm with the above 'parameters' two times. First of all in a splitted dataset with a proportion 80/20 for our train/test data, and then in a splitted dataset with a proportion 60/40 for train/test data.

In the first section of this assignment we will briefly explain how BovW algorithm works.. After the short explanation, we will see the results for the algorithm when used in our dataset, with the help of what we recorded in our excel file. The classifiers that we used in this assignment are Decision Trees with max depth 7, k-Nearest Neighbors, Gaussian Naive-Bayes and Support Vector Machines

The dataset we will use consists of 825 different shoe images designed by Adidas, Nike, or Converse. You can see the dataset here. The way we are dealing with this dataset is to download it, and then making a new folder, namely ShoesNO as used in code, that consists of all images without the train or test folders, i.e. a new folder with subfolders of every Adidas, Nike or Converse shoe. We then use the split-folder library to, well, obviously, split the folder into train and test data two times, once for 80/20 and once for 60/40 proportion for train/test.

2 Bag of Visual Words

We already mentioned that this algorithm, with the help of a feature extractor, finds image features that consists of keypoints and descriptors. The keypoints of an image are the main points found by the extractor to describe the image. The way a feature extractor finds the keypoints of an image is by processing the image pixel by pixel and searching for things that differentiate each picture from another, like edges, corners, or regions that differ in brightness and color from surrounded regions. It is important to notice here that keypoints do not change for an image, if it is rotated or scaled. For example, the keypoints for a motorcycle could be the wheels, the handlebars, or even the saddle. For a human, keypoints may be the nose, the mouth or the eyes. The descriptors of the images are the descriptions of the keypoints that are extracted. When the feauture extractors find the keypoints and descriptors for every image, we create vocabularies, i.e. visual words, for every image given. Using the features and the descriptors we found, we proceed to create clusters of the features that have similar descriptors. After this, every image is expressed as a frequency histogram of the features we previously found. The histograms for every image helps the training of a machine learning model, in order to match an image with a given set of features and descriptors.

3 Sift Models

In this section we project the results we had when we used the SIFT feature extractor.

3.1 Decision Trees

As we can see from the metrics above, where we are using the MiniBatch K-means clustering algorithm, the metric scores for train data are close to 70%, with the metric scores of the 60/40 proportion of our splitted dataset having better results in the train set. Nevertheless, when we try to predict the class of test images, we can observe that metric scores for the 80/20 proportion are actually better, but still lower than 48%.

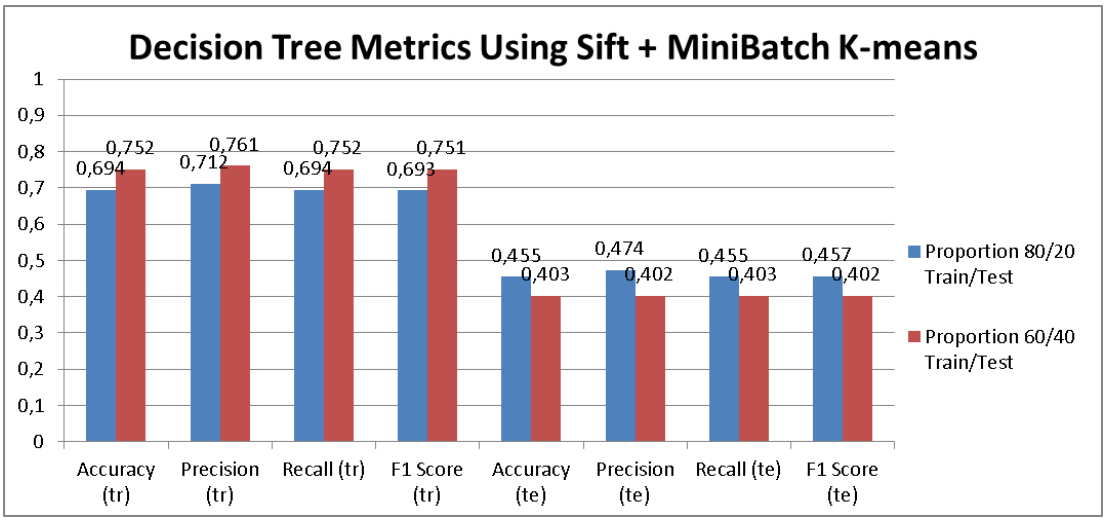

Figure 1: Metric Scores for Decission Trees using SIFT

3.2 k- Nearest Neighbors

This time, metric scores for both train and test data, are better when we splitted the data in proportions of 80/20. The thing that is worth noticing, is that while metrics for train data are lower than metric scores for train when we used the Decission Trees classifier, metric scores for test set are better than those of Decission Trees.

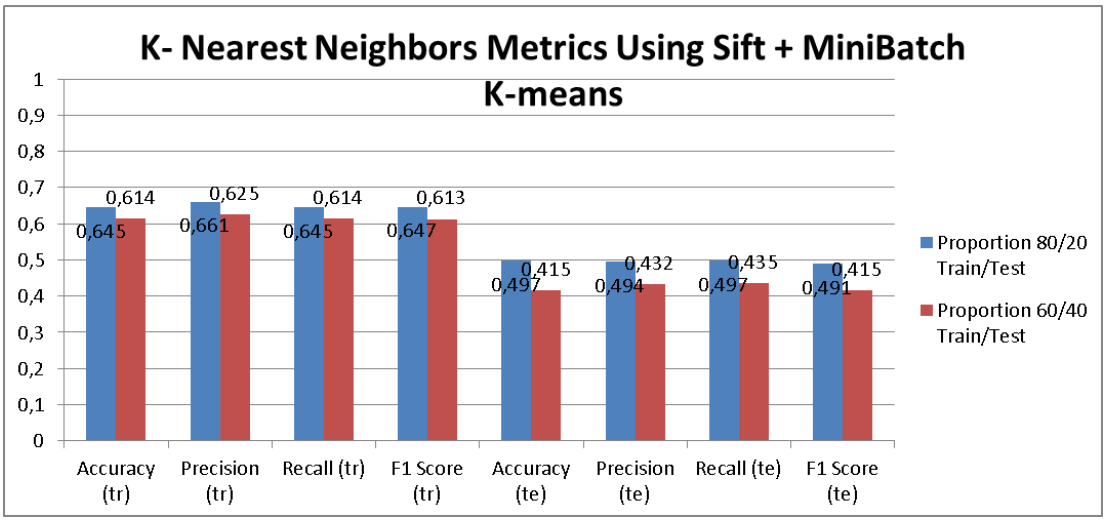

Figure 2: Metric Scores for k-Nearest Neighbors using SIFT

3.3 Gaussian Naive Bayes

The metric scores for test data are better than the previous two classifiers, with the metric scores for 80/20 proportions even exceeding 50%, but the results for train data are way lower than the other models with SIFT and MiniBatch K-Means.

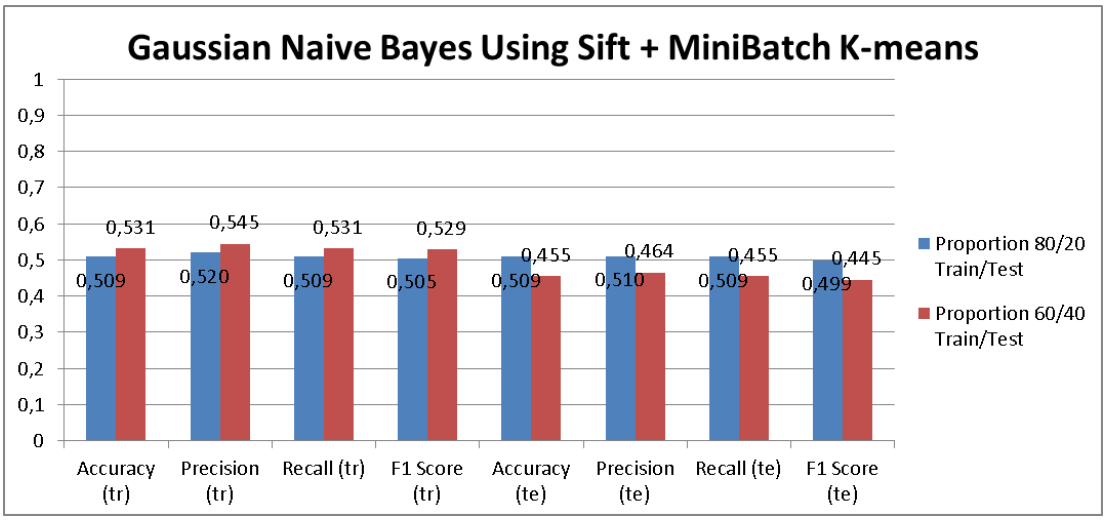

Figure 3: Metric Scores for Gaussian Naive Bayes using SIFT

3.4 Support Vector Machines

We can say without doubt that we have the best model so far. Here we see all metric scores, no matter which data (train or test) we are using, or proportions, are higher than any other model so far, except the precision for train set when compared to the same score of Decission Trees .Although, we obsereve once more that metric scores for train set are better than these of test set.

3.5 Different Clustering Algorithm

We must note that when we tried to use a different method for clustering, google colab ran out of RAM memory every time. The only time this didnt happen was using DBScan, but then the code didnt stop running when we tried to cluster.

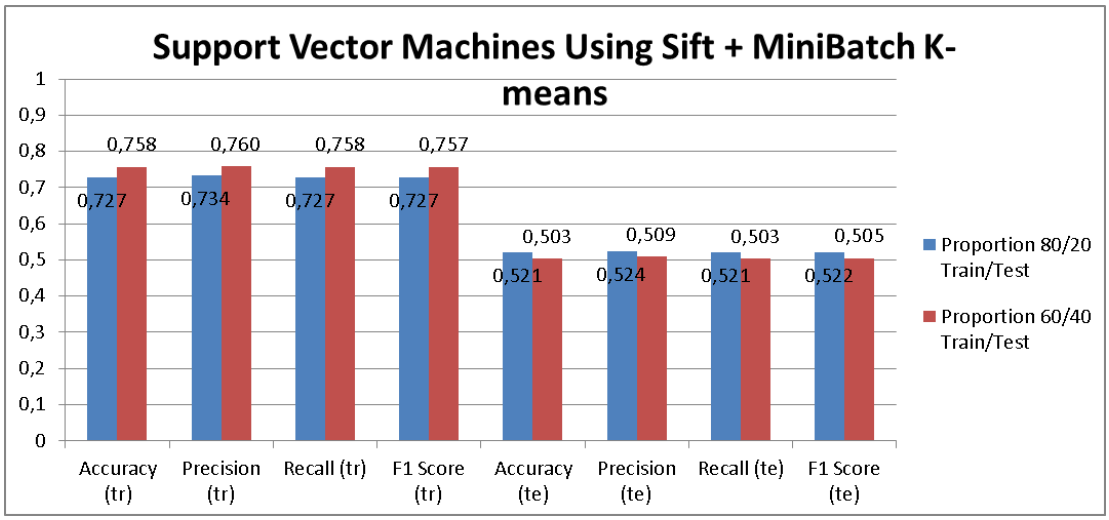

Figure 4: Metric Scores for Support Vector Machines using SIFT

4 ORB Models

In this section we present the results we had when we used the ORB feature extractor.

4.1 Decision Trees

We notice that the metric scores of the test set are again lower than these of train set. Moreover, Decision Trees seem to make better predictions when fed with a 60% of whole dataset train set. When compared to the Decision Trees of the previus section, when we used the SIFT feature extractor, we can tell that metric scores are lower.

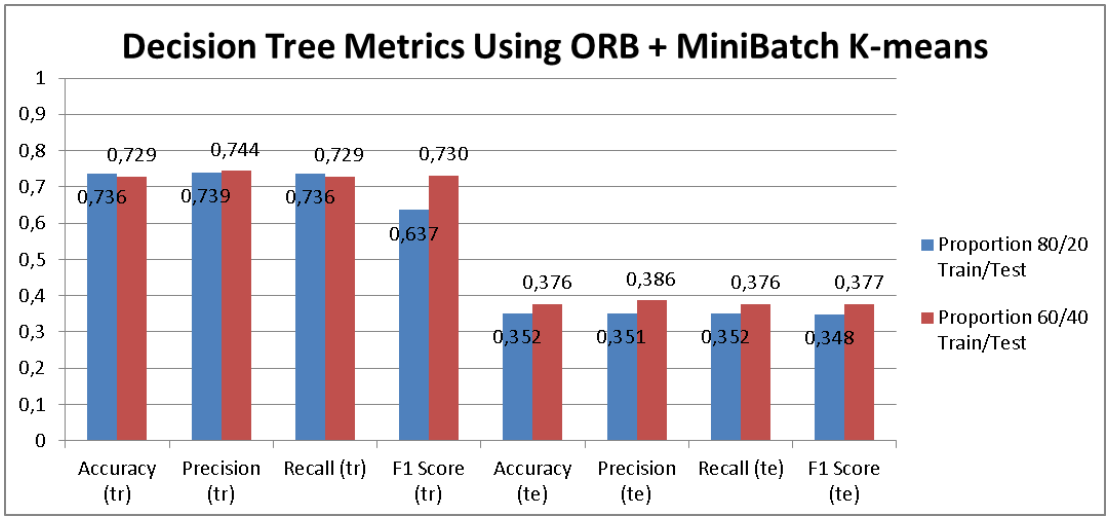

Figure 5: Metric Scores for Decision Trees using ORB

4.2 k-Nearest Neihgbors

We can compare this Classifier with this of Decision Trees by saying the same things we told in the SIFT section. Meaning that once again the metric scores for train data is worse than these of Decision Trees, and that metric scores for test data are better than these of Decision Trees. Also, while SIFT extractor worked better than ORB using Decision Trees for both proportions of the splitted dataset, we can notice here that ORB extractor predicted more accurate the classes of the 60/40 spliited dataset into train/test.

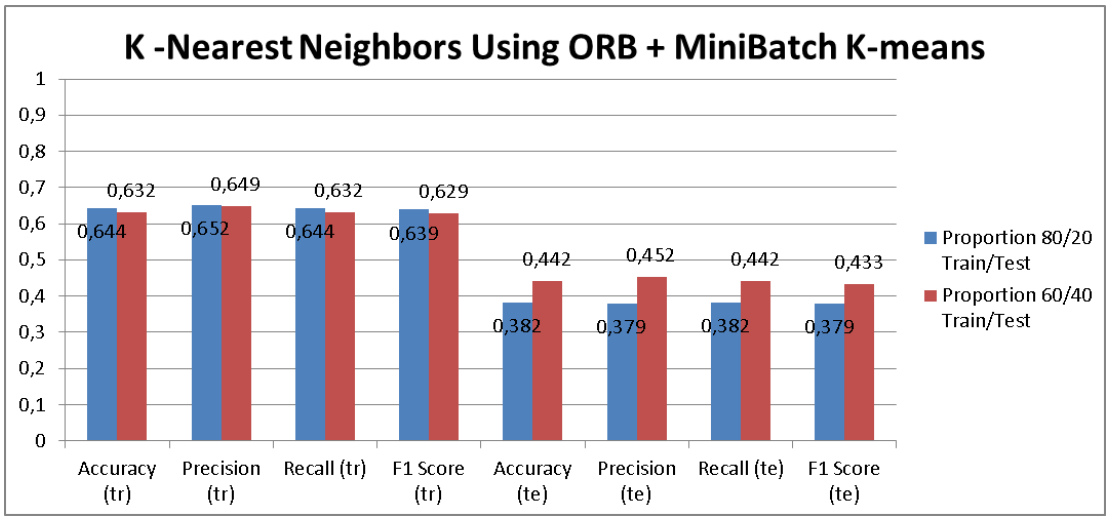

Figure 6: Metric Scores for k-Nearest Neighbors using ORB

4.3 Gaussian Naive Bayes

While the predictions for test set are better in the 80/20 proportion than this of k-Nearest Neighbors using ORB, they are worse in this 60/40 proportion. When compared to the Gaussian Naive Bayes where we used SIFT, we can tell that every metric score is worse for test set. When compared to the previous classifiers using ORB, we can tell that metric scores for test set is better than this of k-Nearest Neighbors and these of Decision Trees when we spliited the data in 80/20. For the proportion 60/40, metric scored for test set are better than these of Decision Trees using ORB, but worse than these of k-Nearest Neighbors.

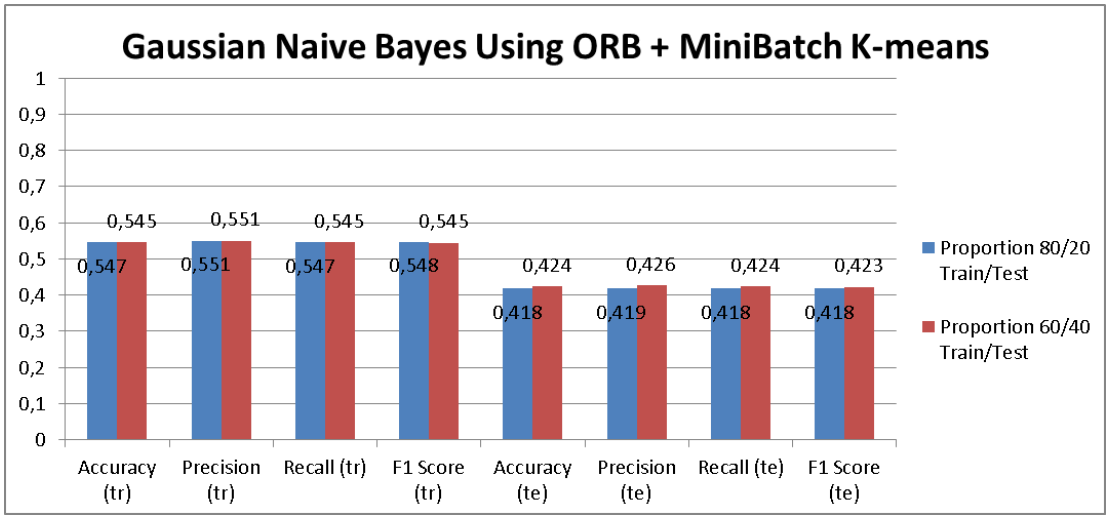

Figure 7: Metric Scores for Gaussian Naive Bayes using ORB

4.4 Support Vector Machines

First of all, when we splitted the dataset into 80/20 train/test, we can see that this model predicted with higher percentage than all of the previous classifiers with ORB. But in the 60/40 split the metric scores are worse when compared to the previous classsifiers,except these of Decision Trees. When compared to the SVM classifier using SIFT, metric scores are worse in all perspectives.

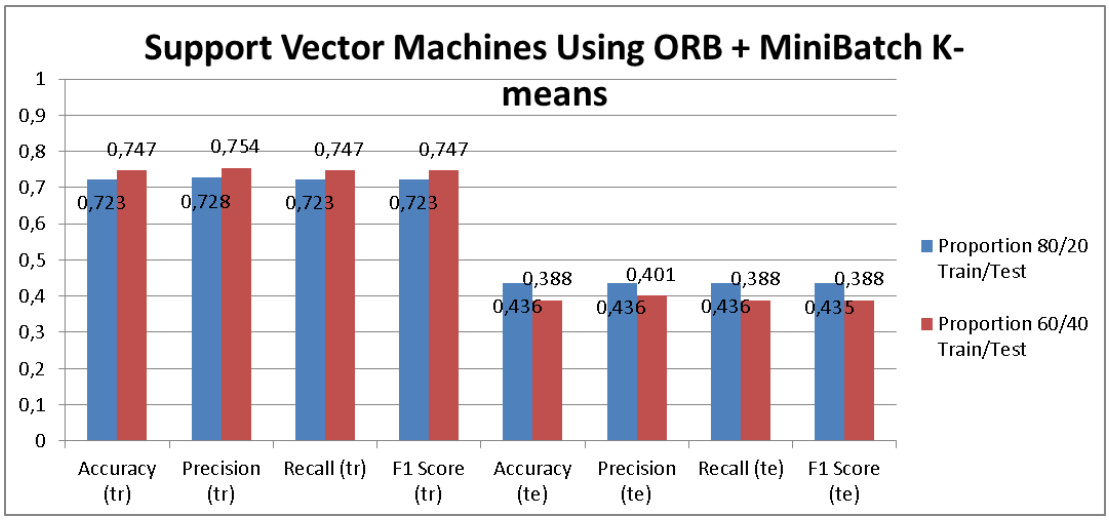

Figure 8: Metric Scores for Support Vector Machines using ORB

4.5 Different Clustering Algorithm

Once again, when we tried to use a different clustering algorithm from Mini Batch k-Means, google colab had the same problem. Ram memory was not enough for this dataset.

5 Conclusion

In this assignment we used two feature extractors to apply the bag of visual words algorithm in our dataset. Apparently, the goal was to use a different method than MiniBatch k-Means, but as we said, we could not program this properly, casuing google Colab to run out of RAM memory. To train every classifier we used, namely Decision Trees, k-Nearest Neighbors, Gaussian Naive Bayes and Support Vector Machines, we splitted the original dataset with 825 images of shoes two times. At first we splitted it into a 80/20 proportion of train/test set, and then into 60/40 proportion of train/test set. In the previous sections we commented the results of every classifier and every extractor but now we wilL make some general conclusions.

To begin with, most of the classifiers using the ORB feature extractor worked much poorly than those that was used after the feature extraction using the SIFT extractor. The only exception was the metric scores for test set of k-Nearest Neighbors, when the dataset was splitted into 60/40 proportions of train/test. But generally speaking, we can say with certainty that SIFT was a better feature extractor for our dataset.

The best classifier out of these four we used was of course Support Vector Machines. Anyway, it is the only classifier whose metric scores for test set are above 50%. If we want to be more specific, to talk about the best model when we used the ORB feature extractor is more difficult. For example, when we splitted the dataset in 80/20 train/test, the model that makes the best predictions was once again support vector machines, but when we splitted the dataset into 60/40 train/test, the best model was k-Nearest Neighbors.