2nd Assignment

Computer Vision - Classification

Machine Learning and Natural Language Processing

Christakakis Panagiotis

ID: aid23004

Postgraduate Studies Program in "Artificial Intelligence and Data Analytics", University Of Macedonia

Date: 20/11/2022

Computer Vision - Classification

Contents

Figures

Pictures

1 Introduction

In this assignment we'll try to implement the technique Bag of Visual Words (BoVW) for image classification.

The dataset we chose contains images of various animals and its ultimate goal is to best classify them in the right mammal category. The total available images are 830 and animal classes are the following:

1. Bear
2. Cat
3. Dog
4. Elephant
5. Lion
6. Goat
7. Horse
8. Lion
9. Tiger
10. Wolf

The idea of Bag of Words was first used for document classification. Occurrences of words are counted and represented in a vocabulary histogram. Building on this technique, nowadays in computer vision a bag of visual words represents a collection of local features for each category we want to classify. These features can be extracted from the image dataset with various detection algorithms and afterwards using a clustering technique they're represented in a histogram which constitutes the final image dictionary. Finally, our model classifier tries to guess in which category every test set image should belong to.

In our experiments we'll try to apply those techniques in order to test our model's performance scores. We'll start with the famous feature extraction algorithm named SIFT and then we'll move on to BRISK. Equally, for clustering we'll use k-Means and then whichever algorithm fits our problem. Moving into the classifiers, we'll test several models using Classification Trees, k-NN, Naïve Bayes and Support Vector Machine. Lastly, all the above combinations will run for 80% - 20% and 60% - 40% train to test ratio.

Dataset is available on Kaggle.

2 Methods

For implementing our methods we'll use python and appropriate libraries. Some of them are numpy, sklearn, cv2, os, splitfolder etc.

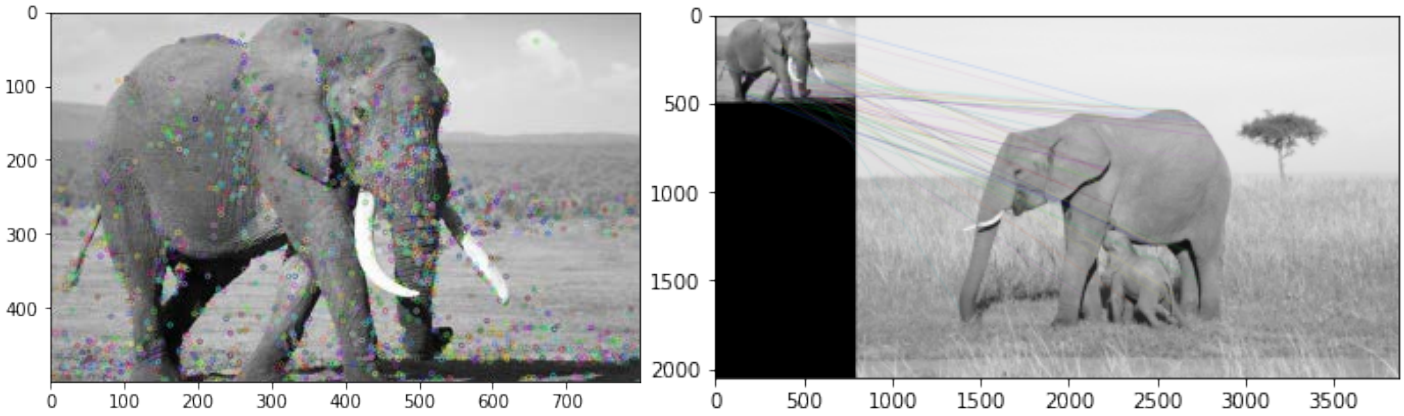
First of all, we download and unzip the dataset from Kaggle. The unzipped folder contains 10 separate folders with images for each mammal class. In order to run our experiments, we have to split the image dataset into train and test. With the help of *splitfolders* library we easily get two separate train and test folders.

For our first experiment our train to test ratio will be 80% - 20%. Each image must be transformed to a smaller one before our feature extraction algorithm starts. The visual dictionary will be created using SIFT. What this algorithm does is that it extracts point-like features and describe them without being affected by orientation, scale, illumination and different viewpoint. This invariance of many transformations helps SIFT key points to be detected easier on new images which represent the same object.

In order to select a point, the algorithm follows a chain transformation of the given images, for example blurring, downsampling and changing its exposure. SIFT uses downsampling to create a set of images with different scale and combined with blurring to reduce noise, the local points are extracted without being scale-dependent. Next, the algorithm tries to enhance the selected features using a technique called Difference of Gaussians, which creates new pictures by subtracting every image from the previous image on the same scale. On this new set of images, the following step is to find the important keypoints that can be used for feature matching while removing the low contrast ones. This effort to find the best points is not limited to the first image but to all its available scales too, by comparing neighboring pixels.

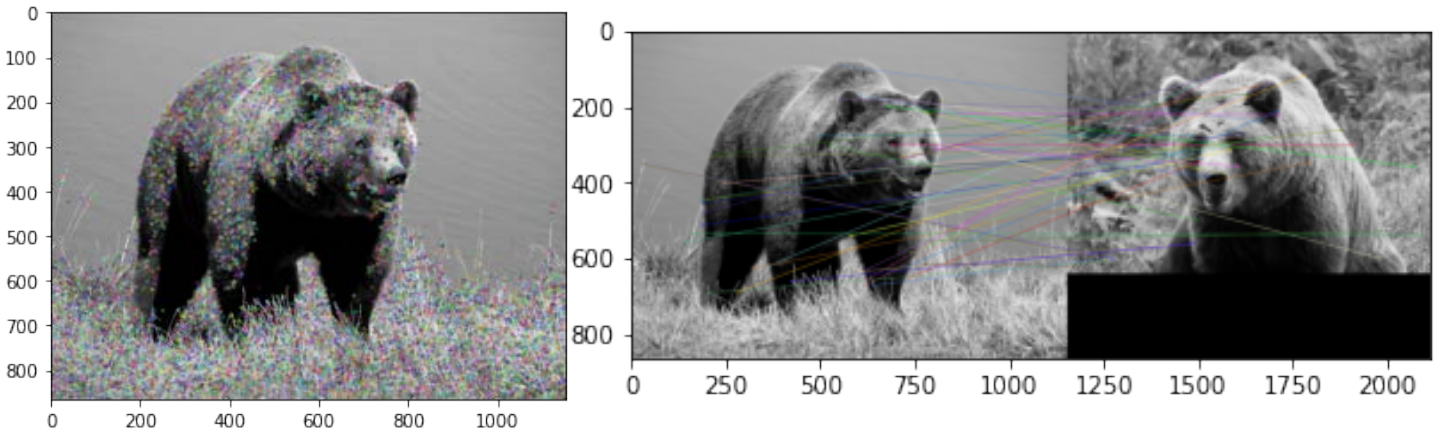
At this stage, each keypoint is assigned to a specific orientation to represent its direction, while its calculated magnitude represents the intensity of the pixel it belongs. A histogram is now created given those two values. The bin at which we see the peak will be the orientation for the point. Lastly, SIFT uses the neighboring pixels of each point to generate a unique fingerprint **descriptor**. Programmatically what SIFT returns is a list whose fist element is also a list that holds all local features from all images without an order and the second holds the sift vectors dictionary which has the descriptors separated class by class.

Picture 2.1: SIFT feature matching for images containing elephants.



***Comments*:** The image on the left shows the keypoints SIFT has selected. As we can see there are many points which not represent an elephant, but the land surrounding him. On the right we can see the matched features for a different picture with two elephants. The keypoints that was on pixels with land aren't matched with the second image.

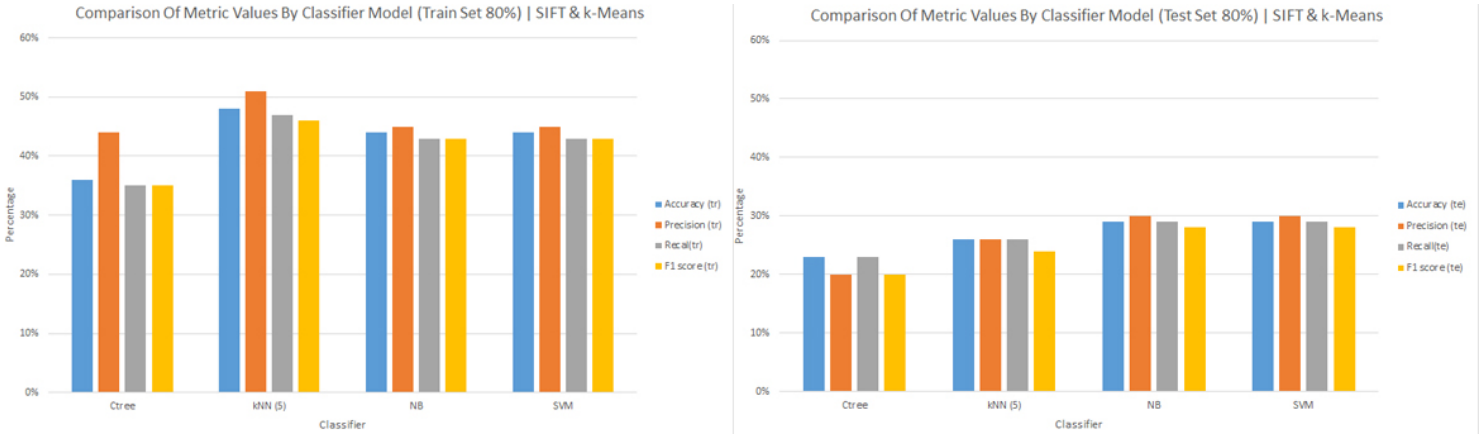
Picture 2.2: SIFT feature matching for images containing bears.



***Comments*:** As with *Picture 2.1*, here it's clear that SIFT didn't only selected detectors for the bear, but also for the grass too. Comparing those local points with the second image, we see that most of them are only matched on the bear.

Following SIFT, for the first experiment we'll build our vocabulary clustering the image keypoints using k-Means. What this algorithm does is that it finds representatives of the gathered features, a method called quantization. The centers of each k clusters are the "visual words".

Figure 2.1: Bar chart comparison of metrics by model (Train 80%) SIFT & k-Means



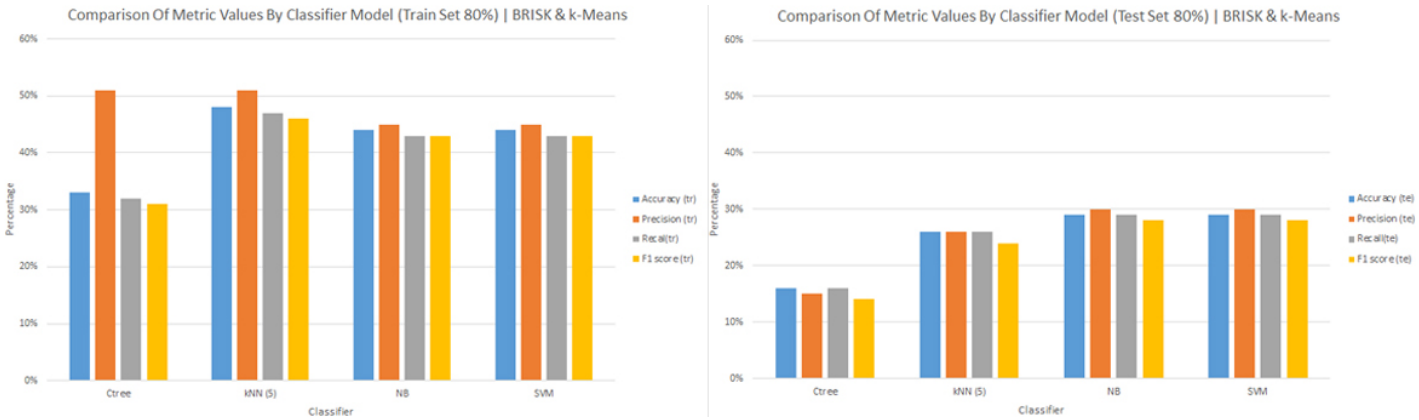
***Comments*:** The above bar chart gives us a clear look on the classifier's performance scores. Each and one of them appear to have higher values on the training images than the testing ones. This leads us to the conclusion that there's definitely some overfitting on the training sample. The two classifiers that have slightly better results are Naïve Bayes and Support Vector Machine. Unfortunately, F1 test scores are not over 30% something that definitely concerns us.

SIFT algorithm is amongst the most famous if not the most famous algorithm for feature extraction. The problem with it is that it's computationally demanding and it's also a patented algorithm, meaning that you can't really use it for commercial use. Of course, there are several others and one of them is BRISK. This free alternative to SIFT is implemented in famous CV libraries such as OpenCV. Scale and rotation invariance are two of its main features. BRISK sampling pattern is composed out of concentric rings, by taking a small patch around each sampling point and apply Gaussian smoothing to it. Just like the previous feature extraction algorithm, BRISK downsamples its images. We distinguish between short and long pairs the distances that help us determine intensity and orientation while using BRISKs' sampling pattern. The descriptor is built based on those two.

In general, interest points are detected by computing a score called FAST for each pixel in the image. If the pixel is above the pre-defined score threshold, then it is detected as an interest point. Some brightness comparison tests are done and interest points are then matched between image pairs by computing the Hamming distance between the feature descriptors. With these characteristics we have a less computational demanding algorithm.

Continuing the first experiment we'll cluster the keypoints BRISK provided us using again k-Means.

Figure 2.2: Bar chart comparison of metrics by model (Train 80%) BRISK & k-Means



***Comments*:** In *Figure 2.2*, we can observe that nearly nothing changed with BRISK in terms of metric scores and the overfitting of our model. The classifiers we used perform as low as SIFT, with Naïve Bayes and Support Vector Machine almost touching 30%.

In our effort to compare different models of Bag of Visual Words image classification we tried to build our vocabulary and with a clustering technique other than k-Means. Unfortunately, after trying all of the available clustering methods from *sklearn.cluster* library for python, none of them worked. Most of the clustering methods were being executed long enough (2hrs) that it wasn't normal for such a small dataset, while others gave us RAM crash errors (Affinity Propagation, Spectral Clustering). Even though we're not sure why this happened, we could guess that density-based clustering algorithms fail in case of varying density clusters. At this point it should be mentioned that all of our experiments were running on Google Colab which provides 12GB of RAM and a processor faster than most usual laptops. As expected, trying to run the code locally, didn't result in the desired outcome.

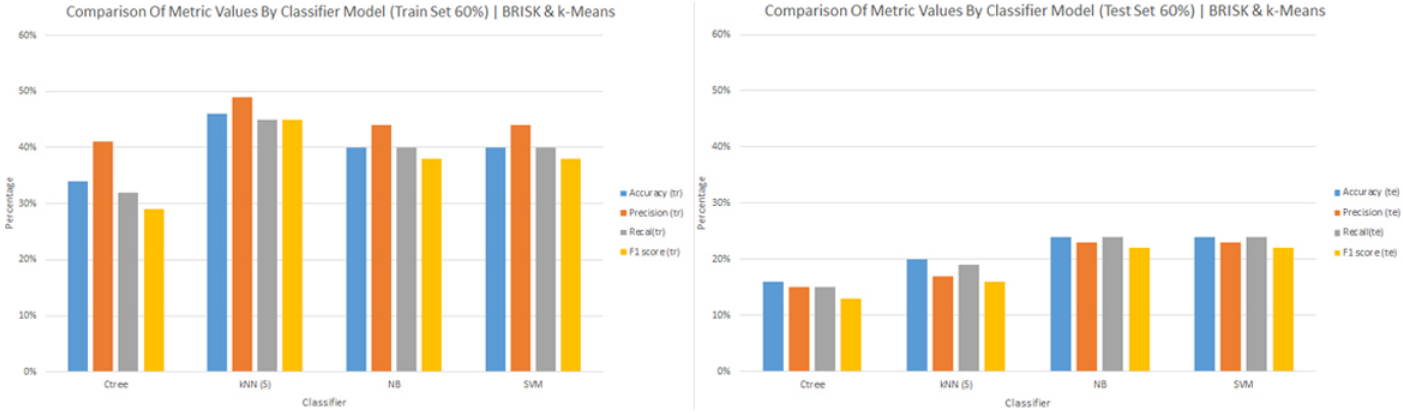
For our second experiment our train to test ratio was 60% - 40%. Again, we tried SIFT and k-Means for our first comparison model and the outcome results were not better than the first train-test splitting.

Figure 2.3: Bar chart comparison of metrics by model (Train 60%) SIFT & k-Means

***Comments*:** *Figure 2.3* offers us the chance to see that Naïve Bayes and Support Vector Machine classifiers have scores not far away than those with the 80% - 20% ratio on train – test data. Of course, metrics results for the training set are higher, meaning that overfitting with such a little data is inevitable.

Finally, we tried the combination of BRISK and k-Means with all the above classifiers and the results were worse.

Figure 2.4: Bar chart comparison of metrics by model (Train 60%) BRISK & k-Means



***Comments*:** In the above Bar Charts we can see that BRISK didn't behave better with the change in the ratio of train and test data. The results are the worst of all the experiments we run and compared. Naïve Bayes and SVM F1-Scores are even below 25%.

3 Conclusions

In conclusion, our best performing model was implemented with SIFT & BRISK as a feature extraction algorithm, k-Means as a clustering technique and both Naive Bayes and Support Vector Machines as final classifiers.

Nonetheless, the results were far from satisfactorily since the best F1 Score we got was 28%. We can attribute the poor performance of individual techniques to a number of factors. First of all, an image dataset with a total of 830 images and 10 classes is indeed small and not really enough for our classifiers to get used to the characteristics of each class. If we also take into account the splitting of the dataset into training and test sets, the total number of train data is lowered. This could be a major problem.

Also, those results led us to dive into the set of images class by class in order to find if there were any errors. Investigating the folders, we found out that images were incorrectly allocated into many folders. For example, in the folder of dogs there is a bear and several squirrels. The folder containing lions does not only have jungle lions but also sea lions. The examples go on and on leading us to a final conclusion.

The algorithms we used for obtaining image local features, clustering and classify them may performed better than we thought on a different dataset. Ours was small, with many classes and fairly poor distribution of images.

Lastly, we can’t say that our experiments were complete since we were unable to use other clustering technique other than k-Means due to low computation resources.