Content Based Image Retrieval Based On Bag of Features Representations of Descriptors From SIFT and Dense SIFT

Ertugrul Kara - 2099125

***Abstract*— In this assigment, I aimed to extract extract local descriptors from images using SIFT and Dense SIFT methods. Then clustered this descriptors to turn them into Bag of Features(BoF) representation. Then for validation images, results are compared against the ground truth values and mAP scores are evaluated.**

I. INTRODUCTION

Image retrieval problem is one of the critical and most studied problems of computer vision. Using bag of features representations of images method is adapted from natural language processing area. Bag of Features(BoF), or known as Bag of Visual Words, is simply a method that ignores spatial locations of objects and is similar to histogram. But for images, we cannot directly apply pixel values to represent images to create a histogram. Instead we extract local descriptors from image using SIFT algorithm and Dense version of this algorithm. SIFT creates scale, rotational and orientation invariant image descriptors. Algorithm first finds interest points. Interest points are where the signal in 2D space has variation that exceeds some threshold criterion and is superior to simple edge detection. SIFT uses Difference of Gaussians which actually approximates Laplacian of Gaussians to find interest points. In SIFT algorithm, we trust in the algorithm to find keypoints in the image. But in Dense SIFT, we tell the algorithm to look for SIFT descriptors in every step size we want. Therefore it is denser hence the name. For the evaluation of the results, because of my computer's specifications -Chromebook with almost no CPU- I had to use cloud computing and I have chosed Google Cloud Platform(GCP) for this purpose, because I have experience with it earlier. For the SIFT and Dense SIFT algorithms, I have chosed cyvlfeat which is a wrapper around VLFeat library. OpenCV also has SIFT implementation but in newer versions of OpenCV, Dense SIFT is not included. To solve the performance and timing issues, I have sampled SIFT descriptors while creating BoF representations with KMeans algorithm and could not used very dense descriptors for Dense SIFT. Citations of the libraries that I have used to calculate the results can be found in the references section.

II. SIFT RESULTS

When extracting SIFT, I have sampled the local descriptors and chose 20% of the descriptors to fit into the memory and speed-up the KMeans algorithm running time to create Bag of Features representation. Than, from this BoF representations, I have found cluster centers with KMeans and tried out different cluster sizes such as; 32, 64, 128 and 256. When querying images, I did not used sampled versions of the descriptors because that led to bad results such as 0.04 mAP value. mAP scores with different cluster sizes can be seen in the graph below. With this settings, while creating cluster centers, I had ¿340k data points.

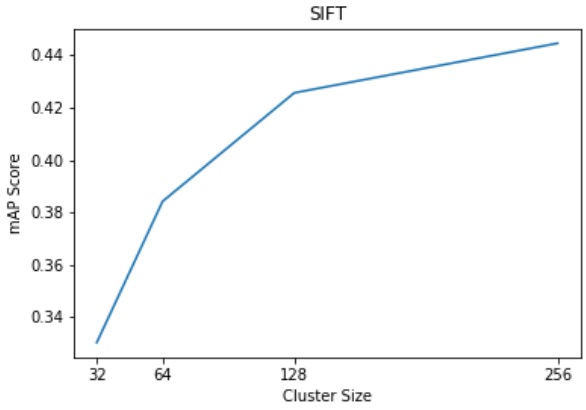

Fig. 1. mAP score with respect to the different cluster sizes that is tried using SIFT extractor.

As we can see that, in this cluster size range, increasing the cluster size also increases the precision and leads to higher accuracy. Not sampling the descriptors while creating the cluster centers could lead to higher mAP score.

III. Dense SIFT Results

When extracting Dense SIFT, I have 2 parameters that I can fine-tune. They are step and size. Step is a parameters that allows me to choose in which pixel intervals, in [size, size, size, size] square the algorithm will look for a SIFT local descriptors. I have tried different settings for these parameters and I will discuss and compare the results against the SIFT. While tuning the parameters again, I had to consider performance and timing in mind. Because I have used Jupyter Notebook and while computing the results internet connection is lost, re-connecting to the existing notebook is not implemented so it cannot take long. For the experiments below, I have tried to comply with the same amounts of data points as in SIFT experiment. For the first three, I had data between 200k and 500k and did not sampled the data points. For the last experiment with step size of 10, there were more than 4 million data points.

A. *Step:50, Size:3-10-30*

I have chose this parameters to see how Dense SIFT performs with the same number of parameters. And it performed worse.

For the first experiment, I have chose step of 50 pixels and size of 3. That means lots of information is lost on the way. Because I pass 50 pixels in each run but look at 3x3 squares. Results also complies my assumptions.

For the second experiment, I have not changed the step size but increased the square size. That, as expected, leads to increased mAP score because now, less information is lost.

For the third experiment with step size 50, I have increased the size of the square to 30x30. That means now, a lot less information is lost but still some is lost. That leads to higher precision again. This computation generates less descriptors but since the square edge size is tripled, creating of these descriptors took long.

As we can see from the figure, when step size is 50 and square size is 3, lot of information is lost and therefore precision goes way low when compared to default SIFT and other parameters. While keeping the step size the same, when square size is increased mAP score goes higher which complies with my reasoning, that is, because less information is lost now mAP is higher.

Other thing that we can observe is that even though large size value, it still performs worse than how the default SIFT performed. I thought that would be because of large information loss between step and size which is 20 pixel for every square. But I could not try higher size values since it takes ages. But I have tried to decrease the step size which leads to my second experiment set.

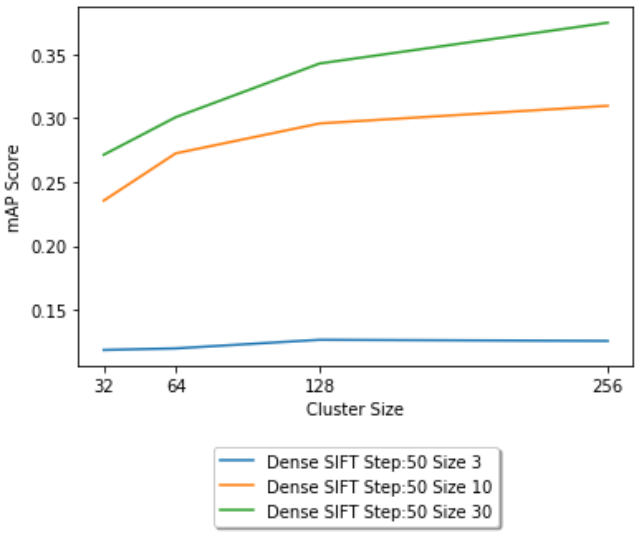

Fig. 2. mAP score with respect to the different cluster sizes that is tried using Dense SIFT extractor with step of 50 and sizes of 3, 10 and 30=.

B. *Step:10, Size:1-2-3-8*

For the last set of experiments, to understand the effects of changing step size better, I have set step size to 8 and changed the size from 1 to 8. This way, I hope to compare the effect in relation to upper experiments. I had to sample, randomly, every 20th descriptor to comply with the data point count with other experiments while sacrificing the precision.

Since step size is lower, when sliding the windows, less information is lost and it is dense as the name suggests. We can clearly see that, when step size is 50, incrementing the square size increased the performance, when step size is 8, incrementing the square size also increases.

As we can see from the figure, when step size is 10 and square size is 4, information loss is greater and thus, lower precision score among these experiments. When I increase the size parameter, mAP score is increased. I think that, this is proving my earlier remarks about information loss. Larger precision score that I have got is when step size 10 and size 12. That is there are no actual information loss. But we can say that, although no information is lost, since we are looking at a huge (12x12) window, some information might not be represented enough.

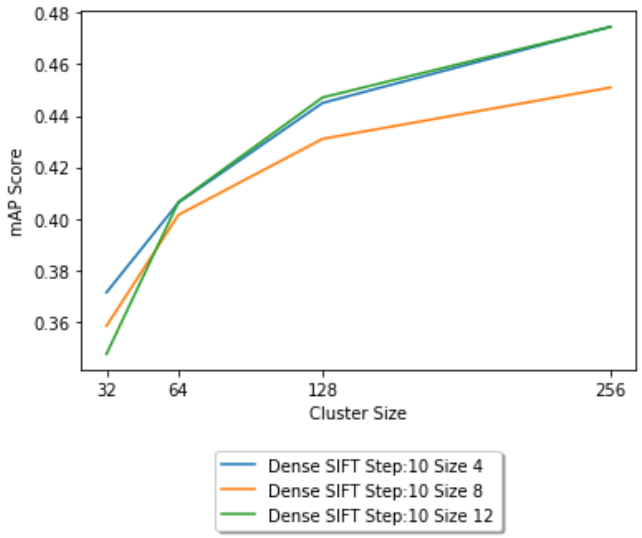

Fig. 3. mAP score with respect to the different cluster sizes that is tried using Dense SIFT extractor with step of 10 and sizes of 4, 8 and 12.

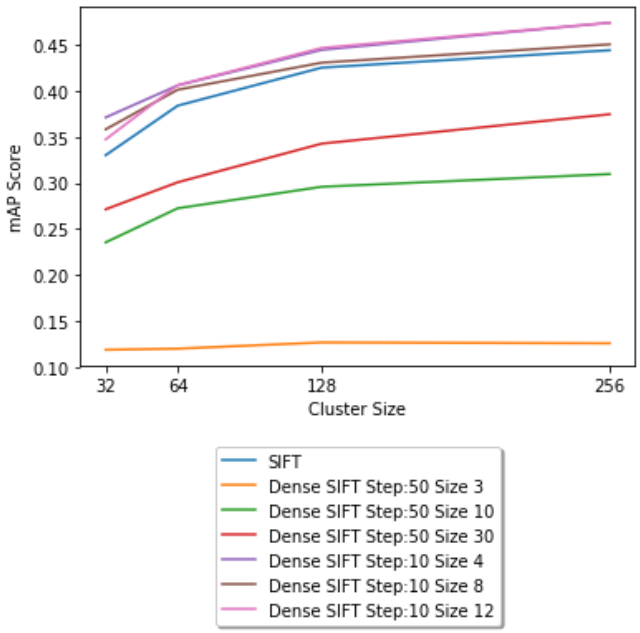

Fig. 4. mAP score with respect to the different cluster sizes and different setups.

IV. CONCLUSIONS

First, I wanted to discuss the effect of cluster size. As it can be seen in the graphs, increasing the cluster size increases the accuracy. But as Elbow Method suggests, cluster size of 128 might be the best choice. Cluster size of 256 might lead to overfitting which we do not want.

In conclusion, my SIFT performance was as expected. I have sampled local descriptors and took 1 over 5 of the descriptors.

While experimenting the Dense SIFT, I thought that it might be a good idea to see how Dense SIFT performs with the same size of descriptors and thus, chose a big step number such as 50. When I increase the size parameter, it slowed down hugely while extracting the descriptors and that is due to the performing SIFT calculations on a larger window at each iteration. However, with the same number of parameters, Dense SIFT could not cope with the SIFT results. Maybe it would have if I have increased the step size to an even larger number such as 60 or even higher.

In my second experiment set, I have chose a step size of 10. That lead to huge number of descriptors such as 4.9 million descriptors. In this experiment getting the clustering results was too long as expected. But it outperformed SIFT even when the square size is 4. Even though there is loss of information because the algorithm steps 10 pixels and looks at 4x4 square at each iteration, it performed better than the default SIFT scores. That is probably due to the larger number of the descriptors.

When I decreased the step size, the precision score increased. When I increased the size value with respect to the step size, the precision score increased. That leads to the result that, with a small step size and relatively large size, we can achieve greater scores. The default Dense SIFT as stated in the paper, step size of 1 and size of 3 probably would perform much better than everything that I have tried. But that generates huge amounts of data and takes too long to extract descriptors for a single image.

In summary, my thoughts are that SIFT performs well, extracts good local descriptors, interest points but with good set of parameters, Dense SIFT looses less information about the image and performs even better.

REFERENCES

[1] David G. Lowe. 2004. Distinctive Image Features from Scale-Invariant Keypoints. Int. J. Comput. Vision 60, 2 (November 2004), 91-110

[2] Fei-Fei, Li, and Pietro Perona. "A bayesian hierarchical model for learning natural scene categories." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 2. IEEE, 2005.

[3] Travis E, Oliphant. A guide to NumPy, USA: Trelgol Publishing, (2006).

[4] Fernando Prez and Brian E. Granger. IPython: A System for Interactive Scientific Computing, Computing in Science & Engineering, 9, 21-29 (2007),

[5] Fabian Pedregosa, Gal Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, douard Duchesnay. Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, 12, 2825-2830 (2011)

[6] Jones E, Oliphant E, Peterson P, et al. SciPy: Open Source Scientific Tools for Python, 2001-

[7] A. Vedaldi and B. Fulkerson. VLFeat: An Open and Portable Library of Computer Vision Algorithms, 2008