CV Assignment05

Lasse Haffke, Moritz Lahann, Emilio Brambilla

February 2021

5.1

In this task, we take a look at Flickr's *similarity search* feature.

1. The search has to discern what is meant by similarity first. There is semantic similarity, where images with the same content/tags are shown, e.g. multiple images of the same bird, or images featuring a hot air ballon. The images could also be similar in terms of color palette used, where every similar image features the same colors with potentially different content. For example a hot air ballon might be red in front of a blue sky, so a similar pallete can be found in a red ship crossing the ocean, making these to images similar despite different content. There are more measures for similarity than these two, so which did the people at Flickr choose ?
   That's not easy to say. They already had a neural net in place to categories images and label them. But instead of using the tags, this network does put out, they grabbed the last vector prior to the final output. One can not say what each part of this vector means, as the neural net is a blackbox. It also has a high amount of dimensions as it features output from every node in that last layer instead of the convolved tag probabilities. As the neural net was trained for scene recognition, the difference (euclidean distance) between similar scenes should be smaller, than between dissimilar scenes, but there is no direct notion of similarity used.

2. Now to find similar images, one would intuitively compute the distance for the feature vector of the current images and every vector in the database. There are multiple reasons why this is infeasible. The two main ones are the amount of storage required for the feature vectors, and the pure computation time. Each feature vector is high-dimensional and stores floating point numbers, imagine now multiple billions of feature vectors stored. The disk space cost is unreasonable high, but for efficient comparison, they all need to be in memory at the same time, making the entire process impossible. Further to find the minimum distance, every feature vector has to be looked at, as there are too many dimensions to sort them. And there is no case in which one can exit early. So every similarity search would need to recompute the distance to every feature

vector and search for the minimum. When done over billions of feature vectors, this is infeasible.

3. So, to speed up the comparison, the people at Flickr implemented something called *Product Quantization*. The idea is that instead of comparing to every vector known, one takes only a subset of them, and calculates the minimum distance in this subset. Clustering the vectors into these subsets can be done beforehand, so that every new vector only needs to compute its distance to the center of the subset and choose the closest subset for closer comparison. Product quantization now mans that the vector is spit up and each part is compared to the subsets. That way only a subset of comparisons has to be done to get the same granularity as when comparing the full vector. So the neighbour search is sped up by only doing a subset of comparisons. In the given example, there are only 2000 comparisons done, instead of 1.000.000.