REPORT ON

VLAD implementation (vs BvoW) on CIFAR-10

MOHD ZAHID FAIZ : MT2019514
SUBARNA KANTI SAMANTA : MT2019523
PRANJAL KUMAR : MT2019079

UNDER GUIDANCE OF:

PROF. DINESH BABU JAYAGOPI

Goal:

- Classify image whether its belong to right category of class using two approaches:
  - Bag of Virtual Words(BoVW)
  - Vector of Locally Aggregated Descriptors(VLAD)
- Compairing both the results

Concept:

BoW involve simply counting the number of descriptors associated with each cluster in a codebook(vocabulary) and creating a histogram for each set of descriptors from an image, thus representing the information in a an image in a compact vector

VLAD is an extension of this concept. We accumulate the residual of each descriptor with respect to its assigned cluster. In simpler terms, we match a descriptor to its closest cluster, then for each cluster, we store the sum of the differences of the descriptors assigned to the cluster and the centroid of the cluster.

- Mathematical Formulation for VLAD:
  - $C = \{c_1, c_2, c_3, ...c_k)$ where k is the no. of clusters in K-means
  - we accumulate the difference $x - c_i$ where for each x, $c_i = NN(x)$
  - $v_{ij} = \sum_{x/x = NN(ci)} (x_j - c_{ij})$
- Comparision between VLAD and BoVW:

  The primary advantage of VLAD over BoW is that we add more discriminative property in our feature vector by adding the difference of each descriptor from the mean in its voronoi cell. This first order statistic adds more information in our feature vector and hence gives us better discrimination for our classifier. This also points us to other improvements we can adding higher order statistics to our feature encoding as well as looking at soft assignment,i,e. assigning each descriptor multiple centroids weighed by their distance from the descriptor.

Dataset:

- Cifar – 10:
  - Training Images (50000)
  - Test Images (10000)
  - Classes : 10 (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck)

Steps Involved for BoVW:

- Importing necessary libraries:
  - opencv (image operations)
  - numpy (array operations)
  - pandas (dataframes)
  - KneighborsClassifier from sklearn.neighbors(model training)
  - kmeans from sklearn.cluster (clustering)
  - StandardScalar, Preprocessing from Sklearn (normalization purpose)

- Feature Descriptors:
  we have used **SIFT** to extract features from our whole dataset of images and append them all **(vertically stack)**.

  We have done in this two respective training and test sets.

- Getting cluster centres(Bag of Visual Words):
  Now that we have extracted feature vectors from each image in our dataset, we need to construct our vocabulary of possible visual words.

  Vocabulary construction is normally accomplished via the **k-means clustering algorithm** where we cluster the feature vectors obtained.

  The resulting cluster centers (i.e., centroids) are treated as our *dictionary* of visual words.

  This is done only in **Training set.**

- Vector Quantization:
  Assign codes from a code book to observations.

  Assigns a code from a code book to each observation. Each observation vector in the M by N obs array is compared with the centroids in the code book and assigned the code of the closest centroid.

  It returns a list comprising two matrix:

  - cluster labels
  - distance of the individual descriptors from its nearest centroid of its respective clustures as mentioned in label.

  We have done in this two respective training and test sets.

- Histogram:

  With the help of Vq function we have made a test feature matrix which will be further processed and used for model training.

  - We used the first part of vq function i.e. cluster_labels to make our histograms for our training and testing set.

- TF-IDF:

  A problem with scoring feature frequency is that highly frequent features start to dominate in the document (e.g. larger score), but may not contain as much “informational content” to the model as rarer but perhaps domain specific features.

  One approach is to rescale the frequency of features by how often they appear in all documents, so that the scores for frequent features are penalized.

This approach to scoring is called **Term Frequency – Inverse Document Frequency**, or TF-IDF for short, where:

Term Frequency: is a scoring of the frequency.
Inverse Document Frequency: is a scoring of how rare the features are.

The scores are a weighting where not all features are equally as important or interesting.

The scores have the effect of highlighting features that are distinct.

- Normalization:

  Finally we normalize our training matrix to get better results using StandardScaler for our both training and test matrix.

- Model Training and Results:

  We have used **KNN**(k-nearest neighbors)

  We have already train and test set separated.

  After hypermater tuning we get our best result for BoVW :

  Accuracy:

  KNN 19.00%

  SVM 25.79%

  n_neighbours = 9

  number of clusters = 40

Steps Involved for VLAD :

Steps which are involve till vector quantization are same, then an extenxion is done instead of histograms we are using summation of residuals.

We accumulate the residual of each descriptor with respect to its assigned cluster. In simpler terms, we match a descriptor to its closest cluster, then for each cluster, we store the sum of the differences of the descriptors assigned to the cluster and the centroid of the cluster.

Subsequently l2 normalization is also done.

- Results:

  With same parameters as used in BoVW , accuracy came out
  KNN 25.39%
  SVM 34.96%