Project3 Report: Feature Encoding

Hou Shengyuan 518021910604
Yan Binghao 518021910753
Wu Shiqu 518021910665

Abstract. Feature coding is a technology to construct a global feature vector from the set of local descriptors of an image. It is based on feature extraction. In this project, we tried some feature extraction methods (SIFT and Selective search) and feature encoding methods (bag of words, VLAD, Fisher vector) on the AwA2 dataset. Then use coding features to train support vector machines and deep neural networks, and analyze the performance of various feature extraction and feature coding methods. Experiments show that VLAD without clustering combined with selective search achieves the best performance which is **0.65**.

Keywords: Feature Encoding, SIFT, Selective Search, Bag of Words, VLAD, Fisher Vector, Super vector Encoding

1 Introduction

Feature extraction(Fig 1) is essentially a process of dimensionality reduction. Through this process, the initial set of original data is reduced to a more manageable group for processing. One characteristic of large data sets is the large number of variables that require a lot of computing resources to process. Feature extraction is a general term for methods that select and/or combine variables as features, which effectively reduces the amount of data that must be processed, while still accurately and completely describing the original data set. The feature extraction process is very useful when you need to reduce the amount of resources required for processing without losing important or relevant information. Feature extraction can also reduce the amount of redundant data for a given analysis. Similarly, the reduction of data and the efforts of machines in constructing variable combinations (functions) promote the speed of the learning and generalization steps in the machine learning process.

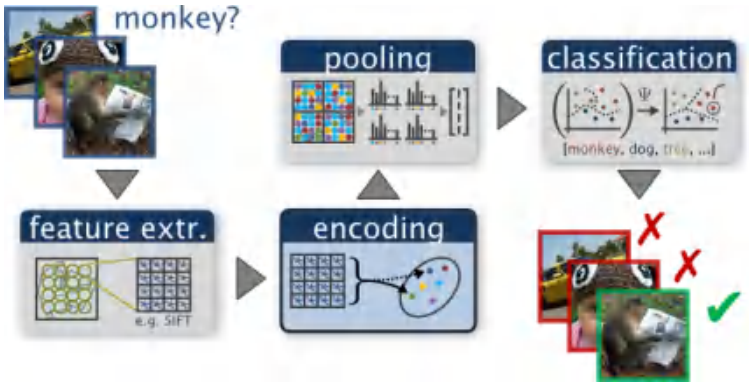


Fig. 1: Feature Extraction and Feature Encoding

For feature encoding, we need to realize that the performance of a machine learning model depends not only on the model and hyperparameters, but also on how we process and input different types of variables into the model. Since most machine learning models only accept numerical variables, coding the original variables becomes a necessary step.

In this project, we tried two feature extraction methods, namely SIFT [4] and selective search [6], and three feature coding methods, namely bag of words [2], VLAD [3] and Fisher vector [5]. For the classifier, SVM with rbf kernel and ResNet [1] for deep neural network (DNN) are used.

2 Local Descriptor Extraction

2.1 SIFT

Scale-invariant feature transform (SIFT) is a computer vision method proposed by David Lowe in 1999 and further summarized in 2004 for detecting and describing local descriptors which are position, scale and rotation invariant of the image (Fig.čite) and could be further exploited for object recognition ,video tracking and so on. Also SIFT is featured by its stability on affinity transformation, illumination, noise background and variance of the viewpoint. Generally speaking, it mainly consists of following four steps:

Step 1. Scale-space Extrema Detection

SIFT begins by finding points of interest in the image which is defined as local minima or maxima of difference value in scale space. Firstly, the scale space of the picture is constructed by applying different scale of Gaussian filter convolution so that we could capture the local detail and global feature at multiple scales simultaneously. Assume that original image is $I(x, y)$ and k-scale convolutional filter is $G(x, y, k\sigma)$, then blurred image at scale k could be written as.

$$L(x, y, k\sigma) = I(x, y) * G(x, y, k\sigma)$$

$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Next images at different blurred scales are grouped and form an octave, and the toppest one is downsampled to generate the bottom image in the next octave. Convolved images per octave is maintained a fixed number. Then the Difference-of-Gaussian images are taken from adjacent Gaussian-blurred images per octave as shown in Fig. 2.

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma)$$

The key points are then pre-extracted from DoG images across different scales by scanning its local minima/maxima point. If among eight neighborhood pixels in the same image and 18 pixels upon and below the neighborhood scale the pixel value is minimum or maximum, it is regarded as a candidate key point. However, some selected key points might have weak response or strong edge effect, therefore it's necessary to localize keypoints more precisely.

Step 2. Key point Localization

Localization step aims to specify the ratio of principal curvatures, scale, location and of every key point and filter out those that are strongly affected by edge effect or themselves noise. Discarding low-contrast key points is realized by thresholding the value of second-order Taylor expansion $D(x)$.

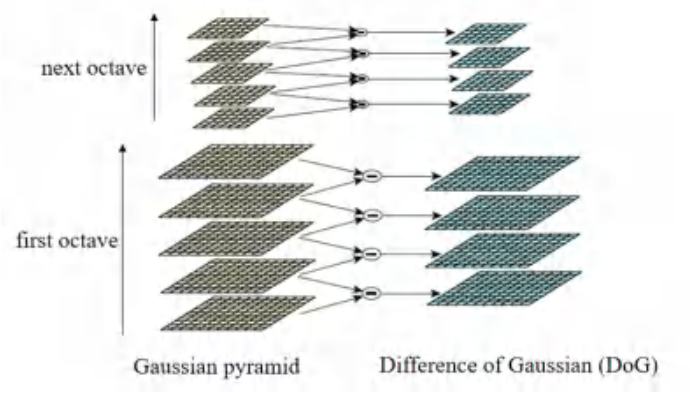


Fig. 2: Difference of Gaussians

Eliminating edge responses are realized by thresholding the ratio value $R = Tr(H)^2/Det(H)$ where H is the second order Hessian matrix.

(a) (b) (c)

Fig. 3: Image where Low-contrast and Edge Key points are Filtered Out

Step 3. Orientation Assignment

Next each key point is assigned with one or more orientations, which aims to make local key points invariant to rotation. This is achieved by describing local key points relative to this orientation. For each key point at location (x, y) and scale σ, gradient magnitude and orientation is computed from pixel differences

$$m(x, y, \sigma) = \sqrt{(L(x+1, y, \sigma) - L(x-1, y, \sigma))^2 + (L(x, y+1, \sigma) - L(x, y-1, \sigma))^2}$$

$$\theta(x, y, \sigma) = \arctan \frac{L(x, y+1, \sigma) - L(x, y-1, \sigma)}{L(x+1, y, \sigma) - L(x-1, y, \sigma)}$$

Gradient magnitude and orientation of all neighborhooding points around the key point within the same will be calculated within the same image. 360 degrees are discretized into 36 bins and an orientation histogram is constructed by neighborhood calculation results. For each pixel, what is added into the bin is weighted by the magnitude of gradient and $1.5\sigma_{octave}$ Gaussian weighted circular window. Peak in the histogram is chosen as the dominant orientation and angles that achieve 80% of peak value are assigned to key point as well.

Step 4. Keypoint Descriptor

For each key point, orientation, scale and position have been extracted so the next step is to integrate these properties into a vector that is invariant to illumination, viewpoint and so on. It should also take neighborhooding pixels into consideration. 4×4 pixel neighborhood each with 8 calculated bins are integrated into a 128 dimensional vector and the vector is processed like normalization and so on.

2.2 Selective Search

Selective search is a region proposal algorithm for target detection. It has fast calculation speed, high recall rate, and hierarchical grouping of similar regions based on color, texture, size and shape compatibility. Selective Search algorithm mainly includes two parts: Hierarchical Grouping Algorithm and Diversification Strategies.

Hierarchical Grouping Algorithm

The starting point of Hierarchical Grouping Algorithm is that regional features in images are more representative than pixels. The overall process is as follows:

1. Use the method of **Felzenszwalb and Huttenlocher** to generate the initial area of the image $R = \{r_1, r_2, \cdots, r_n\}$;
2. Calculate the similarity between all adjacent regions;
3. The two most similar regions are grouped together;
4. Calculate the similarity between the merged region and the adjacent region;
5. Repeat the 3 and 4 processes until the entire image becomes an area.

Diversification Strategies

This part is mainly about some strategies related to diversity, which makes the sampling diversified, mainly in the following three different aspects: (1) uses a variety of invariant color spaces; (2) uses different similarity measures ;(3) changes the starting area. The most important part is the 4 joint measures used to calculate the similarity. Let's briefly introduce it below:

- $s_{color}(r_i, r_j)$: This is a measure of color similarity, $C_i = \{c_i^1, \cdots, c_i^n\}$ means that each area is represented by a three-channel color histogram, and a 25 *bins* histogram for each color channel , So that each region can get a $n = 75$-dimensional vector. After using L1-norm standardization, use the following formula to calculate the similarity between regions:

$$s_{color}(r_i, r_j) = \sum_{k=1}^{n} \min(c_i^k, c_j^k)$$

he color histogram of the merged area is calculated as follows (r_i, r_j merged into r_t):

$$C_t = \frac{size(r_i) \times C_i + size(r_j) \times C_j}{size(r_i) + size(r_j)}$$
$$size(r_t) = size(r_i) + size(r_j)$$

- $s_{texture}(r_i, r_j)$: This is a measure of texture similarity, using the $SIFT - Like$ feature. The specific operation is to use the Gaussian distribution with a variance of 1 to do gradient statistics on the 8 different directions of each color channel, and then the statistical results (size and area size) Consistent) Calculate the histogram with $bins = 10$. The number of histogram intervals is $8 * 3 * 10 = 240$. $T_i = \{t_i^1, \cdots, t_i^n\}$ represents the texture histogram of each region, with 240 dimensions.

$$s_{texture}(r_i, r_j) = \sum_{k=1}^{n} \min(t_i^k, t_j^k)$$

- $s_{size}(r_i, r_j)$: This is a measure of scale similarity. In order to ensure that the scale of the region merging operation is more uniform, use the following formula and use the size similarity. The purpose is to merge the small regions as much as possible. $size(im)$ refers to the size of the region (in Pixels).

$$s_{size}(r_i, r_j) = 1 - \frac{size(r_i) + size(r_j)}{size(im)}$$

- $s_{fill}(r_i, r_j)$: This is a measure of shape coincidence. In order to measure whether the two regions are more coincident, the smaller the Bounding Box of the merged region, the higher the coincidence degree. The formula is as follows:

$$s_{fill}(r_i, r_j) = 1 - \frac{size(BB_{ij}) - size(r_i) - size(r_j)}{size(im)}$$

Finally, a combination of the above four methods is used as the final measure of similarity:

$$s(r_i, r_j) = a_1 s_{color}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + a_3 s_{size}(r_i, r_j) + a_4 s_{fill}(r_i, r_j)$$

3 Feature Encoding

3.1 Bag of Words

In computer vision, the bag-of-words model (BoW model) sometimes called bag-of-visual-words model(BoVW model) can be applied to image classification or retrieval, by treating image features as words. In document classification, a bag of words is a sparse vector of occurrence counts of words; that is, a sparse histogram over the vocabulary. In computer vision, a bag of visual words is a vector of occurrence counts of a vocabulary of local image features.

The general idea of bag of visual words (BOVW) is to represent an image as a set of features. Features consists of keypoints and descriptors. Keypoints are the "stand out" points in an image, so no matter the image is rotated, shrink, or expand, its keypoints will always be the same. And descriptor is the description of the keypoint. We use the keypoints and descriptors to construct vocabularies and represent each image as a frequency histogram of features that are in the image. From the frequency histogram, later, we can find another similar images or predict the category of the image. For feature encoding, the following procedure is used to compute bags of words:

1. **Detect features:** Extract descriptors from each image in the data set, and build a visual dictionary. The feature extractor algorithm (such as SIFT, KAZE, etc.) can be used to detect the features in the image and extract the descriptor.

2. **Learning a codebook:** We make clusters from the descriptors (we can use K-Means, DBSCAN or another clustering algorithm). The center of each cluster will be used as the visual dictionary's vocabularies.
3. **Encoding:** For each image, we make frequency histogram from the vocabularies and the frequency of the vocabularies in the image. Those histograms are our bag of visual words (BOVW).

3.2 VLAD

VLAD(Fig 4) is an extension of BoW. 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. As for bag of words, we first learn a codebook $C = \{c_1, \cdots, c_k\}$ of k visual words with k-means. Each local descriptor x is associated to its nearest visual word $c_i = NN(x)$. The idea of the VLAD descriptor is to accumulate, for each visual word c_i, the differences $x - c_i$ of the vectors x assigned to c_i. This characterizes the distribution of the vectors with respect to the center.

Assuming the local descriptor to be d-dimensional, the dimension D of our representation is $D = kd$. In the following, we represent the descriptor by $v_{i,j}$, where the indices $i = 1, \cdots, k$ and $j = 1, \cdots, d$ respectively index the visual word and the local descriptor component. Hence, a component of v is obtained as a sum over all the image descriptors:

$$v_{i,j} = \sum_{x \ such \ that \ NN(x)=c_i} x_j - c_{i,j}$$

where x_j and $c_{i,j}$ respectively denote the jth component of the descriptor x considered and of its corresponding visual word c_i. The vector v subsequently L2-normalized by $v := \frac{v}{||v||^2}$.

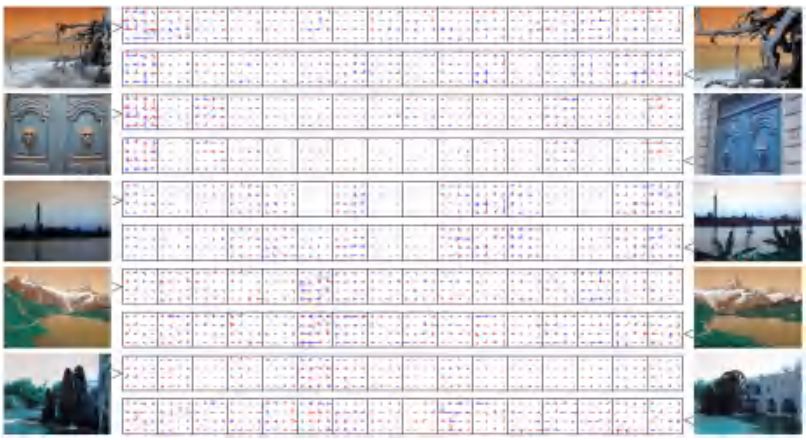


Fig. 4: Images and corresponding VLAD descriptors, for $k = 16$ centroids ($D = 16128$). The components of the descriptor are represented like SIFT, with negative components in red.

The figure above shows that the descriptors are relatively sparse (few values have significant energy) and very structured: most of the higher descriptor values are in the same cluster, and the geometry of the SIFT descriptor is visible. For images that are sufficiently similar, the proximity of the descriptors is obvious.

3.3 Fisher Vector

Fisher vector(Fig 5) essentially uses the gradient vector of the likelihood function to represent an image. The physical meaning of the gradient vector is to describe the parameter change direction that can make the model better adapt to the data, that is, the process of parameter tuning in data fitting.

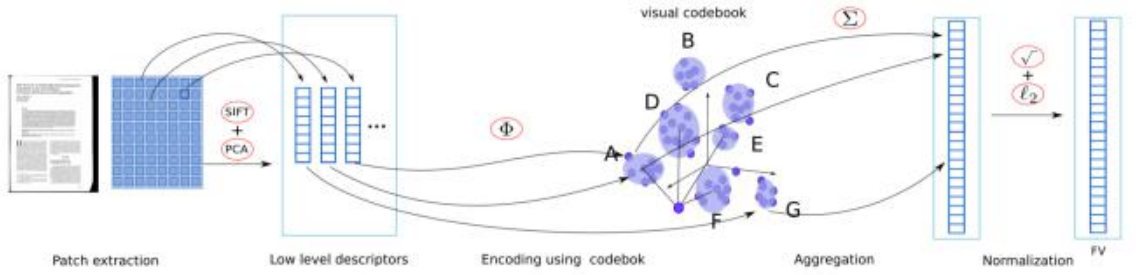


Fig. 5: **Illustration of the Fisher-vector representation.** Local patches are extracted uniformly in the document image, and then described by lowlevel descriptors (typically PCA-projected SIFT descriptors). These are encoded using a GMM-based visual codebook, and the patch-level encodings are aggregated. Finally, the representation is square-rooted and L2 normalized.

Fisher Kernel

Fisher kernel is a function that measures the similarity of two objects on the basis of sets of measurements for each object and a statistical model. Let $X = \{x_t | t = 1, \cdots, T\}$ is a sample containing T descriptors x_t. We assume that p is the probability density function, and its parameter is λ. Let $u_\lambda = p(X|\lambda)$, then the score function can be represented by the gradient vector of the likelihood function as follows:

$$G_\lambda^X = \bigtriangledown_\lambda \log u_\lambda(X) \tag{1}$$

The gradient of the log-likelihood function describes the direction in which the model parameters should be changed to better fit the current sample parameters, and can be used as a feature of the sample in any discriminative classifier. However, many discriminative classifiers that use inner products require us to input normalized feature vectors. In order to achieve the normalization of the gradient feature vector, we can use the Fisher information matrix:

$$F_\lambda = E_{x \sim u_\lambda}[G_\lambda^X G_\lambda^{XT}] \tag{2}$$

Here, we express the normalized gradient vector as $\mathcal{G}_\lambda^X = F_\lambda^{-\frac{1}{2}} G_\lambda^X$. On this basis, we can use the following Fisher kernel to measure the distance between the sample X and the sample Y, which means the similarity between the two samples:

$$K(X, Y) = G_\lambda^{XT} F_\lambda^{-1} G_\lambda^Y = \mathcal{G}_\lambda^{XT} \mathcal{G}_\lambda^Y \tag{3}$$

Fisher vector in feature encoding

Assume that $X = \{x_t | t = 1, \cdots, T\}$ are the local descriptors (such as SIFT) extracted from an image that obey the same distribution. Under the assumption of independence, we can derive the FV of the image based on formula (1) as:

$$\mathcal{G}_\lambda^X = F_\lambda^{-\frac{1}{2}} G_\lambda^X = \mathcal{G}_\lambda^X = F_\lambda^{-\frac{1}{2}} \bigtriangledown_\lambda \log u_\lambda(X) = \sum_{t=1}^{T} \bigtriangledown_\lambda \log u_\lambda(x_t) \tag{4}$$

Assuming that the above distribution obeys the Gaussian mixture model, the parameters of the model are $\lambda = \{w_k, \mu_k, \sigma_k | k = 1, \cdots, K\}$, then

$$u_\lambda(x) = \sum_{k=1}^{K} w_k u_k(x), \qquad \sum_{k=1}^{K} w_k = 1$$

$$u_k(x) = \frac{1}{(2\pi)^{\frac{D}{2}} |\sigma_k^{\frac{1}{2}}|} \exp\{-\frac{1}{2}(x - \mu_k)^T \sigma_K^T (x - \mu_k)\}$$

Here, we assume that the covariance matrix is a diagonal matrix. According to the Bayesian formula, the probability that the descriptor x_t belongs to the i Gaussian model is:

$$\gamma_t(i) = \frac{w_i u_i(x_t)}{\sum_{k=1}^{K} w_k u_k(x_t)}$$

The specific derivation will not be repeated here. The general process is to first find the formula (1) for each gradient component of GMM, then use the Fisher information matrix to obtain an approximate solution, and then combine the formula (4) to obtain the normalized Fisher Vector, as follows:

$$\mathcal{G}_{\alpha_k}^X = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} (\gamma_t(k) - w_k)$$

$$\mathcal{G}_{\boldsymbol{\mu_k}}^X = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} \gamma_t(k) (\frac{\boldsymbol{x_t} - \boldsymbol{\mu_k}}{\boldsymbol{\sigma_k}})$$

$$\mathcal{G}_{\boldsymbol{\sigma_k}}^X = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} \gamma_t(k) \frac{1}{\sqrt{2}} [\frac{(\boldsymbol{x_t} - \boldsymbol{\mu_k})^2 - 1}{\boldsymbol{\sigma_k^2}}]$$

Note that $\mathcal{G}_{\alpha_k}^X$ is a scalar while $\mathcal{G}_{\boldsymbol{\mu_k}}^X$ and $\mathcal{G}_{\boldsymbol{\sigma_k}}^X$ are D-dimensional vectors. The final FV is the concatenation of the gradients $\mathcal{G}_{\alpha_k}^X, \mathcal{G}_{\boldsymbol{\mu_k}}^X, \mathcal{G}_{\boldsymbol{\sigma_k}}^X$ for $k = 1, \cdots, K$.

4 Experiment

4.1 Experimental Setup

In the experiment, We select the **Animals with attributes 2 dataset (AwA2)** which is composed of 37322 images of 50 animals. To measure the final classification performance, We split Awa2

images 60% into training set and 40% into testing set using stratified sampling by *train_test_slpit* in *sklearn.model_selection* with random_state=1234.

One thing to mention is that during resize of the figure in selective search procedure, we figure out an image(Fig 6), which is gray-scaled and for simplicity we directly stretch it to RGB channel while maintaining it in the dataset.

Fig. 6: Gray-scale image in the dataset

4.2 SIFT

In this section, we directly use $cv2.SIFT\_create()$ API in $opencv-python$ package to extract local keypoints. Since the extractor object only accepts gray-scale image as its input, we should use $cv2.cvtColor(img, cv2.COLOR\_RGB2GRAY)$ beforehand to deal with raw RGB color images. Then it's time to call $detectandCompute()$ function to get all keypoints and their corresponding feature vectors with dimension 128.

To effectively sift out local key point features, we first perform an analytical statistics on distribution of key point amounts and draw the histogram as in Fig 7. Distribution regularity shows that key point amount of every figure appears to be distributed poissonly with most quantities between 500 and 10000.

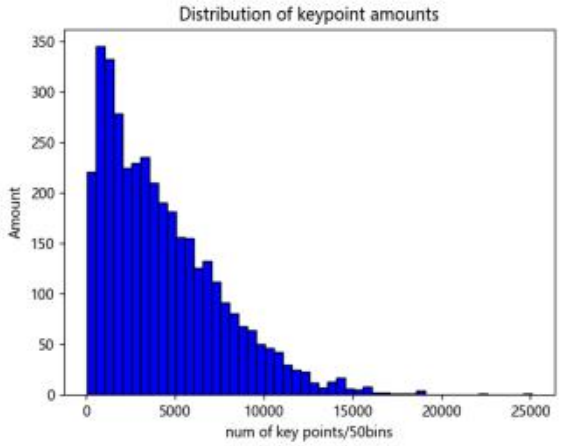
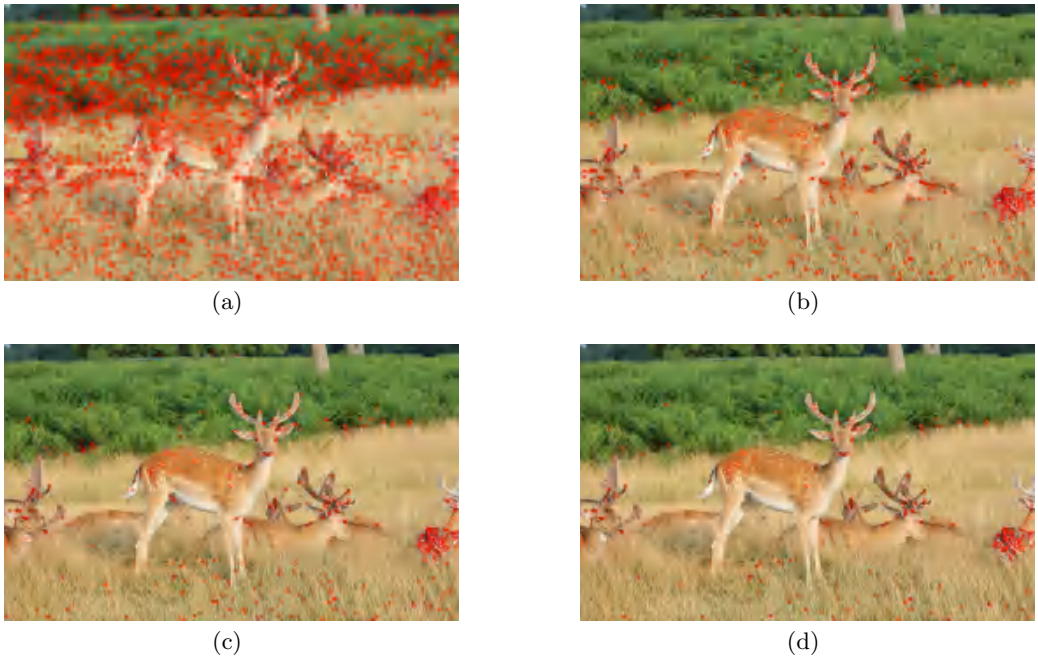


Fig. 7: Distribution of Keypoint Amounts

Further, we could take the first antelope image as an example and inspect locations and qualities of overall key points. Four detection results with (a)$N_{kp} = all$ (b)$N_{kp} = 1000$ (c)$N_{kp} = 500$ and (d)$N_{kp} = 300$ has been shown in Fig 8. After inspection we could find that with no limitation detected sift key points severely fill up all over the image which introduces significant background noise for training. When the feature number drops down to less than 1000, points are majorly composed of core animal features with slightly noisy points. Therefore, to increase efficiency and facilitate training, we fisx and restrict key point numbers N_{kp} as 100, 300 and 500 respectively.



(a) (b)

(c) (d)

Fig. 8: Detected SIFT Keypoints with (a)$N_{kp} = 6657$ (b)$N_{kp} = 1000$ (c)$N_{kp} = 500$ (d)$N_{kp} = 300$

Since tons of images generate tremendous amount of memory resources consumption and even when extracting sift keypoints we encounter 32GB memory consumption with rarely 3000 images, we sample 10 percent images which consist of totally 3400 images in a stratified way for SIFT extraction task.

In the following part, we perform BOW, VLAD, and Fisher vector respectively to integrate local descriptors into a 128-dim global feature vector. We use *sklearn.cluster.kmeans* API from *sklearn* package to execute k-means algorithm on all local descriptors. For cluster number, we fix it as 128 and set $random\_state = 0$. In fisher vector extraction, we perform GMM algorithm and also use $sklearn.mixture.GaussianMixture$ and take the same cluster number. Consequently by optionally integrating cluster assignment, derivation from centroid and covariance structure of every cluster, BOW,VLAD and Fisher Vector will return a 128-dim, 128*128-dim and 128*(128+128*128)-dim vector respectively.

After obtaining global feature vector for every sample image, we additionally take advantage of $sklearn.decomposition.PCA$ to do PCA preprocess on VLAD and Fisher vector because of huge time complexity and reduce them to 64 dimensions when cluster number is 128, 128 dimensions when cluster number is 256 and 256 dimensions when cluster number is 512. Eventually $svm.SVC()$ func-

tion in *sklearn* is used to train and test data samples. We use the *rbf* kernel and set *random_state* as 0, C as 5 and decision function shape as *ovr*. Experimental results on different N_{kp} with fixed cluster number 128 is shown in Table 1 and fig 9.

N_{kp}	BOW		VLAD		Fisher	
	Train	Test	Train	Test	Train	Test
100	1.0	0.190	0.999	0.182	0.999	**0.200**
300	0.998	0.222	1.0	**0.247**	0.999	0.216
500	0.992	0.232	0.999	**0.271**	0.999	0.222

Table 1: Performance on different N_{kp} with cluster number=128

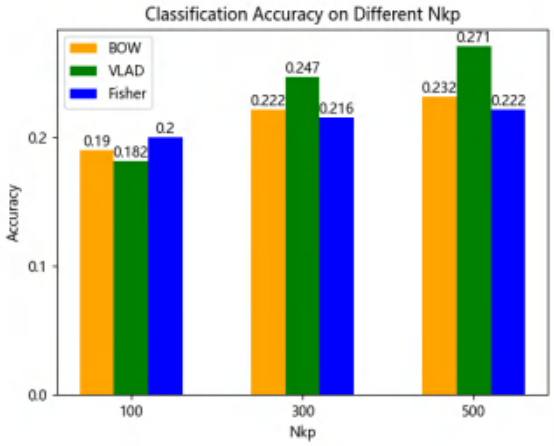


Fig. 9: Classification Accuracy of different N_{kp} on BOW, VLAD and Fisher

From the experimental result, we could find that SIFT descriptor commonly gains awful performances but when N_{kp} rises up, the accuracy attains a slight improvement. This is because in the circumstance of sparse sample data, key points detected from images might still maintain a large proportion of noise background or other disturbing objects. Therefore, useful information is hidden in the key points and the larger N_{kp}, the more decisive key points are and clustering algorithm for global feature extraction is more robust. Among three feature encoding methods when there are not sufficient key points($N_{kp} = 100$) it's unsurprising that Fisher beats other ones due to containing all three levels' information and VLAD performs worst which indicates its difference value might be disturbed by major proportion of noise points. But when N_{kp} becomes more adequate VLAD performs the best because it includes higher level information than BOW and Fisher might suffer from the non-convergence of GMM.

Also we perform experiments on different cluster numbers with the same key point numbers $N_{kp} = 100$ and $N_{kp} = 500$ respectively. Results are shown in Table 2 and Table 3. According to statistical analysis in fig 10, given small fixed N_{kp}, encoding quality tends to drop down with the increase of cluster number because with small N_{kp} feature points tend to aggregate according to its

label, but with much more clusters over category numbers will make clustering result inaccurate. As was expected, Fisher also gives the best result, followed by VLAD and BOW when $N_{kp} = 100$. When a larger N_{kp} is fixed with 500, performance starts to be overwhelmed by VLAD. This might be due to the non-convergence of GMM algorithm with respect to fisher vector as indicated in the last paragraph. Meanwhile, the classification performance is boosting overall, which of course benefits from enough information included in the image. Therefore, from above all, we could conclude that VLAD with enough key points per image will perform the best.

Cluster	BOW		VLAD		Fisher	
	Train	Test	Train	Test	Train	Test
64	0.946	0.170	1.0	0.200	1.0	**0.207**
128	1.0	0.190	0.999	0.182	0.999	**0.200**
256	1.0	0.138	1.0	0.140	1.0	**0.196**

Table 2: Performance on different cluster numbers with $N_{kp} = 100$

Cluster	BOW		VLAD		Fisher	
	Train	Test	Train	Test	Train	Test
64	0.762	0.210	1.0	**0.304**	1.0	0.247
128	0.992	0.232	0.999	**0.271**	0.999	0.222
256	1.0	0.221	1.0	0.225	1.0	**0.255**

Table 3: Performance on different cluster numbers with $N_{kp} = 500$

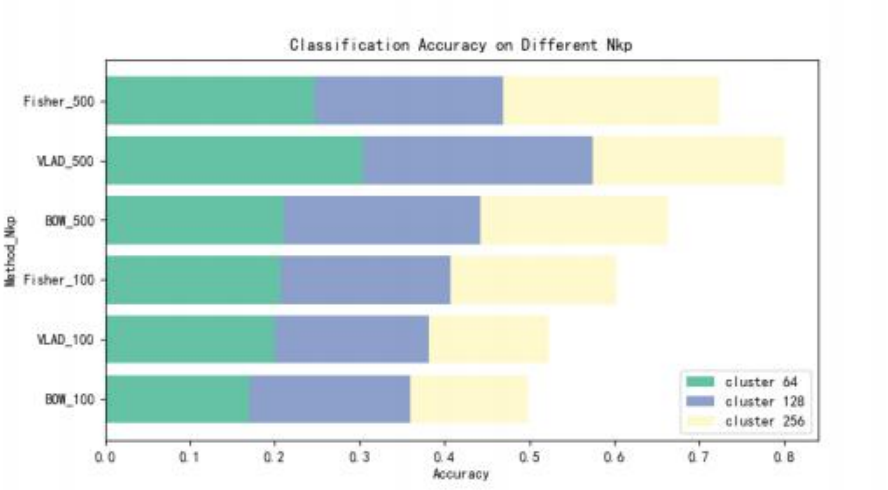


Fig. 10: Accuracy of different cluster numbers on three methods with $N_{kp} = 100$ and $N_{kp} = 500$

4.3 Selective Search

As indicated in the methodology part, selective search attempts to construct underlying useful regions by merging basic candidate areas according to similarity, as shown in Fig 13. Each local region is then fed into common Computer Vision network such as Resnet, VGG and so on which outputs an dimension-fixed feature vector as local feature. Then BOW, VLAD and Fisher Vector are then performed as in SIFT part.

It is worth mentioning that different from SIFT where the number of key points could be fixed, selective search package does not provide similar functiontality and also from the extracted region we could find that many overlapped local area is selected which incorporates redundancy and noisy location. Therefore to reduce influence of different local descriptor numbers and unnecessary information we analyze distributional feature of region size and amount in Fig 11 and Fig 12.

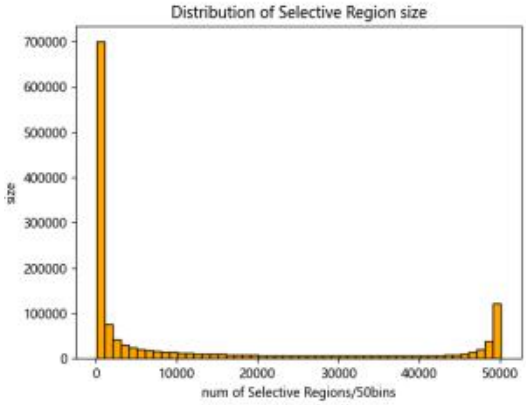


Fig. 11: Size of all local candidate regions

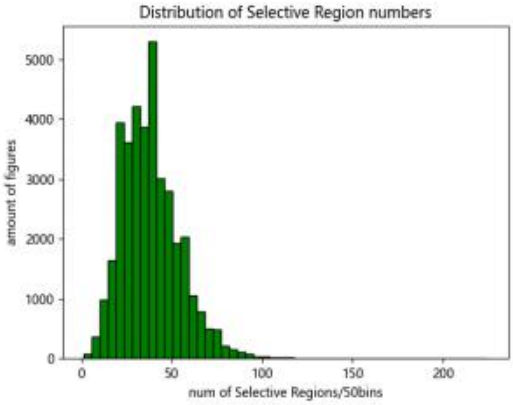
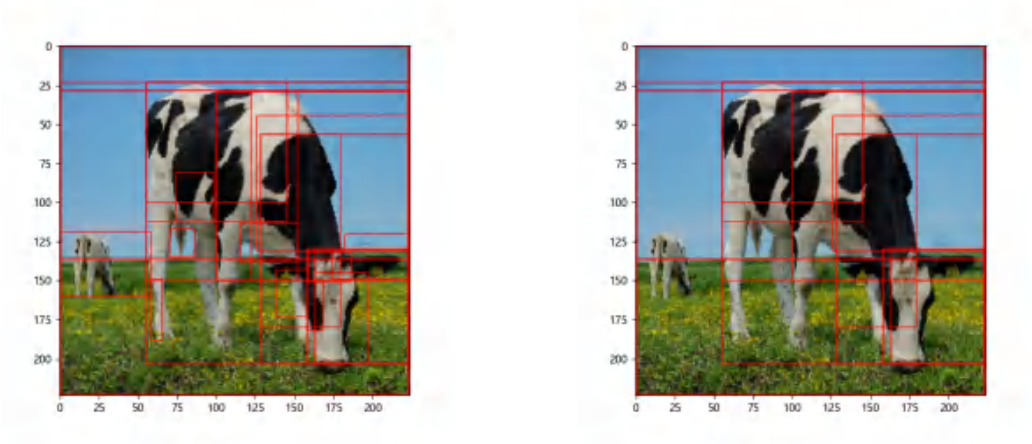


Fig. 12: Amount of regions for every figure

From the above two figures, we could find that:(1)For most regions, they appear to be aggregated on the extreme value of both sides, where the major components of regions are little ones which includes sparse information and even background noise as shown in Fig 13. This implies that we

could gather a small amount of some largest regions. (2)Most of figures possess more than 20 regions according to power law distribution in Fig 12. All in all, we decide to select top largest 20 regions for every figure and for those which owns less than 10 areas, we just ignore this bound. The example in Fig 13 shows the effect of region selection.



(a) (b)

Fig. 13: Local region extracted by Selective Search (a)original (b)selected

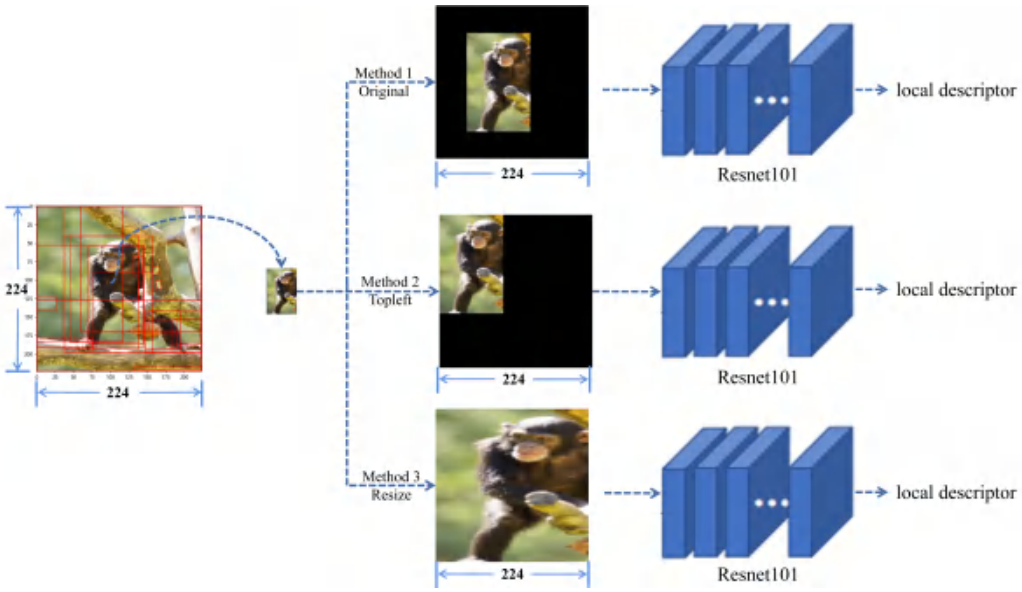


Fig. 14: Local descriptor extracted by Selective Search

In this project, to encode candidate regions into local feature vector, we use Resnet101 network of Pytorch with different initialization methods. One is directly transferred from Resnet101 where $pretrained = True$. Another one is pretrained by images in the training set where $preTrain = False$. Since Resnet only accepts input of shape 224×224, the image should be first resized by

$cv2.resize()$ function. When feeding local regions into Resnet, three resizing strategies are candidate as shown in Fig 14 :(1)Original. Remain the local region in its original position. (2)Top. Translate candidate region to the top left. (3)Resize. Resize the local part into the whole 224×224 image by interpolation.

With cluster number 128, our experimental results are shown in Table 4 and Table 6. Table 4 uses inherent parameters of Resnet101 pretrained on ImageNet while Table 6 uses parameter weights pretrained on Awa2 dataset. Fig 15 shows the performace of model using ImageNet pretrained Resnet101.

Method	BOW		VLAD		Fisher	
	Train	Test	Train	Test	Train	Test
original	0.855	0.192	0.969	0.075	0.995	**0.227**
top	0.893	0.330	0.975	0.071	0.996	**0.381**
resize	0.832	0.382	0.964	0.059	0.996	**0.420**

Table 4: Performance on ImageNet pretrained network with cluster number=128

Method	BOW		VLAD		Fisher	
	Train	Test	Train	Test	Train	Test
original	0.885	0.090	0.978	0.087	0.998	**0.144**
top	0.894	0.112	0.976	0.057	1.0	**0.122**
resize	0.915	**0.102**	0.971	0.022	1.0	0.092

Table 5: Performance on Awa2 pretrained network with cluster number=128

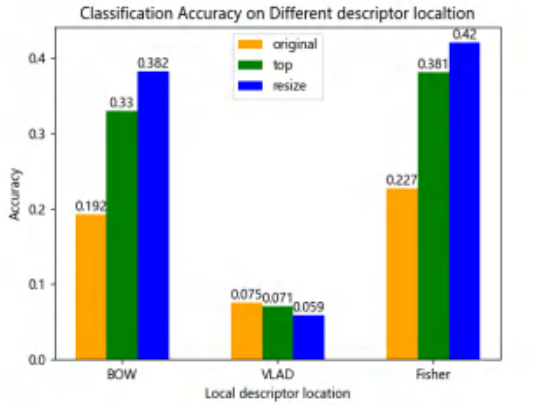


Fig. 15: Selective Search performance with pretrained Resnet101 on ImageNet

After inspection of experimental data, we could find that overall performance of Awa2 pretrained Resnet model performs much woser than Resnet101 pretrained on ImageNet dataset. This is in accordance with expectation because Awa2 dataset maintains about only 37000 images while ImageNet maintains more than 1 million images. The lack of enough images will bring about the overfitting and severe descending on local feature vector quality.

From another perspective, selective search often performs better than SIFT, where the accuracy could rise up to nearly 0.40 for BOW only and 0.42 for Fisher vector, which is significantly better than 0.304 for VLAD on the last section. Nevertheless, we inspect an exotic phenomenon that all VLAD integration method attains a terribly poor accuracy on classification, with all nearly around 0.07. In further discussion part, we will try to explore an effective way to tackle this problem.

Compared among different descriptor location settings, we could conclude that original location is the worst because it includes the positional information of the local region which might slightly disturb CNN model results. Deep CNN network extracts high level feature of the image with translation invariant property, so if positional feature is incorporated into every local descriptor vector, the clustering process and effect will be harassed. Then for the top left location, the position of local region is fixed so it's better than original one, but pure black area out of the edge of local region still incorproates noisy information. Also, different regions feature with different width and height, and the mismatch of all local region sizes give rises to more uncertainty on the performance.

Last but not least, region with resized location expectedly achieves the best performance. This is mainly for it not only tackles the region mismatch problem by resizing into identical shape 224×224, but tickle out unnecessary black pixel and fix regions' location as well.

5 Further Discussion

Do we actually need clustering for VLAD in selective search?

In this section, we try to find some tips of improving performance of VLAD on ImageNet pretrained Resnet101. Inspired by SIFT section, where the descend of cluster number will greatly improve the classification accuracy, we try to exponentially drop down the cluster number and record the experimental performance in Table 6 and Fig 16.

Cluster Number	1	2	4	8	16	32	64	128
original	**0.395**	0.383	0.341	0.307	0.265	0.222	0.164	0.075
top	**0.62**	0.605	0.587	0.523	0.430	0.388	0.269	0.071
resize	**0.654**	0.600	0.549	0.472	0.397	0.310	0.199	0.059

Table 6: Performance on ImageNet pretrained model with VLAD and different cluster numbers

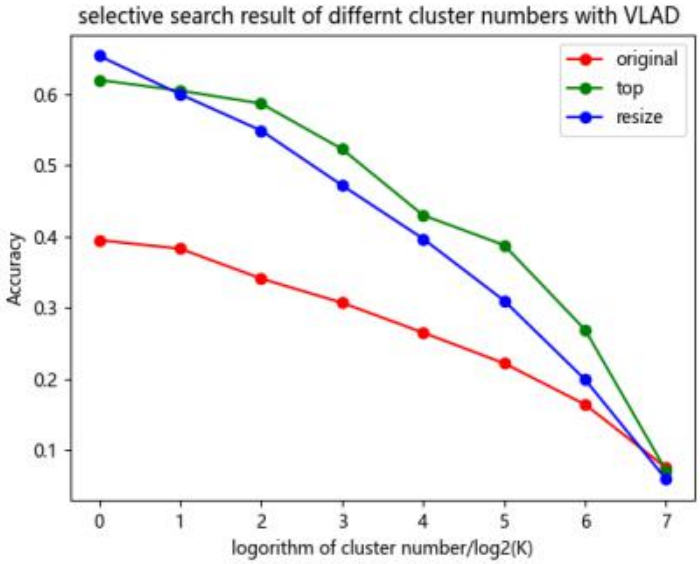


Fig. 16: Performance of VLAD with different cluster numbers on selective search

Also we try other BOW and Fisher methods and drop down the cluster number, even on the SIFT section part. Unfortunately however, we inspect very poor performance on all the other settings so we do not display them here. With rarely the experimental result from VLAD, we could surprisingly find that as the cluster number drops down, the performance of original, top and resize locations all significantly rises up, even up to 0.65, which is the best performance up to now.

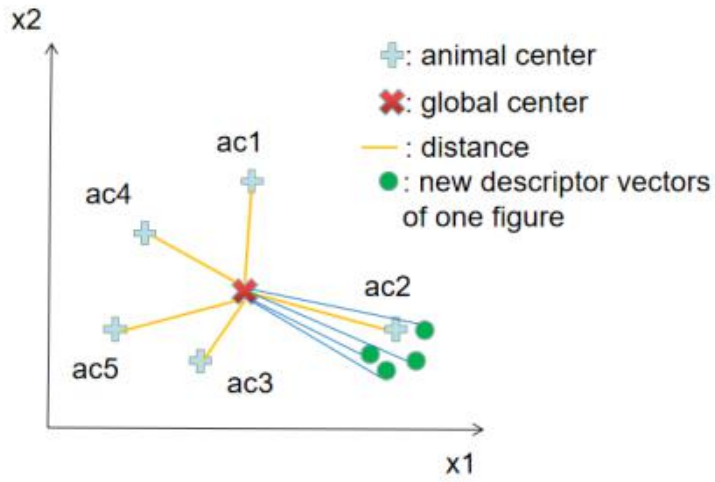


Fig. 17: VLAD with no clustering

We try to explain this in a brief way and the deeper regularity hidden behind the data remains to be further explored. Firstly, different from SIFT process, extraction of descriptor from local region by Resnet101 features with strong linear separability since it is gained by representation learning of deep neural network while SIFT just gains them by traditional feature engineering. Therefore, the feature vectors of local regions tend to cluster in high-dim space. Different clusters aggregate unevenly which means different cluster centers perform distinct distance among one another. This indicates that the summation of distance for different kind of animal performs special distributive patterns. It leads to this phenomenon: when a new local feature vectors are being integrated, their summation of difference from the single global center are very similiar to other training samples with the same label(see Fig 17). Therefore, is performance will greatly improve.

As the cluster number increases, since k-means and GMM both might suffer from the local minimum convergence, this natural clustering distinctive distribution of VLAD value will be cut into pieces and destroyed so the performance will certainly become worse.

6 Conclusion

In this project, we try different methods on feature encoding. For local feature extraction ,we perform two methods SIFT and selective search. The precedent one might include more background noise information but could fix the amount of key points for every figure. The latter one could effectively contain core area in candidate regions and could be encoded by mature deep CNN Resnet and for unification we choose the largest 20 regions of every figure. Then all local descriptor vectors are fed into three codebook learning methods: BOW,VLAD and Fisher and integrated to generate a single global vector for every figure. Finally, SVM classifer is adapted to output the final prediction result. Experimental results show that:
(1) SIFT descriptor commonly gains awful performances because of sparse sample data and background noise. When there are not sufficient key points, Fisher beats other ones containing all three levels' information and VLAD performs worst. But when N_{kp} becomes more adequate VLAD performs the best.
(2)Given small fixed N_{kp}, encoding quality drops down with the increase of cluster number. Fisher gives the best result, followed by VLAD and BOW when $N_{kp} = 100$. When $N_{kp} = 500$, VLAD performs best and the classification performance is boosting overall. Therefore, VLAD with enough key points per image performs the best.
(3)Overall performance of Awa2 pretrained Resnet model is much worse than Resnet101 pretrained on ImageNet dataset. The lack of enough images will bring about the overfitting and severe descending on local feature vector quality.
(4)Selective search often performs better than SIFT. Nevertheless, all VLAD integration method attains terribly poor accuracy, with all nearly around 0.07. In further discussion part, we explore an effective way to tackle this problem.
(5)Original location is the worst because it includes the positional information of the local region which will slightly disturbs CNN. For top left location, the position of local region is fixed so it's better than original one. The mismatch of all local region sizes give rises to more uncertainty on the performance.
(6)Region with resized location achieves the best performance.
(7)For VLAD in selective search, as the cluster number drops down, the performance of original,

top and resize locations all significantly rises up, even up to 0.65, which is the best performance up to now.

7 Contribution

Hou Shengyuan
1.Do experiment and write report on selective search part.
2.Program code, do experiment and write report on further discussion part.
3.Do vidualization for all experimental part.
Yan Binghao
1.Write all introduction,abstract and related work part.
2.Program code, do experiment on SIFT part.
3.Visualization of non-experimental part.
Wu Shiqu
1.Program code and pretrain the Resnet101 on Awa2 dataset.
2.Program code on selective search part and Write report on SIFT part.
3.Preprocessing of the dataset and typesetting of paper.

References

1. He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
2. Jurie F, Triggs B. Creating efficient codebooks for visual recognition[C]//Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1. IEEE, 2005, 1: 604-610.
3. Jégou H, Douze M, Schmid C, et al. Aggregating local descriptors into a compact image representation[C]//2010 IEEE computer society conference on computer vision and pattern recognition. IEEE, 2010: 3304-3311.
4. Lowe D G. Object recognition from local scale-invariant features[C]//Proceedings of the seventh IEEE international conference on computer vision. Ieee, 1999, 2: 1150-1157.
5. Sánchez J, Perronnin F, Mensink T, et al. Image classification with the fisher vector: Theory and practice[J]. International journal of computer vision, 2013, 105(3): 222-245.
6. Uijlings J R R, Van De Sande K E A, Gevers T, et al. Selective search for object recognition[J]. International journal of computer vision, 2013, 104(2): 154-171.