Project 3: Feature Encoding for Image Classification

Xu Jiayi , Gao Yifeng, Tang Qidong, Bi Wendong

I. Local descriptor extraction

A. SIFT

SIFT (Scale-Invariant Feature Transform) [1] a method for extracting distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene. The features are invariant to image scale and rotation, and are shown to provide robust matching across a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination.

Features extracted with SIFT have the following main properties:

1) Scale-Invariant Feature Detection
   Compared to some common feature detectors, such as Harris corner detector, a main advantage of SIFT is that the features extracted with SIFT avoid scale variance. Mainly Same features can be extracted regardless of their size. And the features are also immune to rotation.

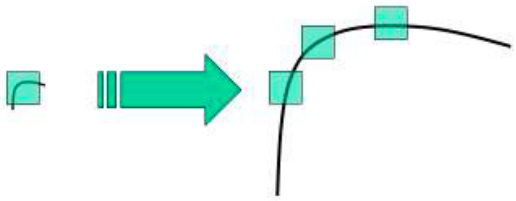


Fig. 1. Scale Variance

2) SIFT features are robust illumination, noise and minor changes in viewpoints. They are highly distinctive, allowing for object identification.
3) Recognition can be performed in close-to-real time on modern computation device.

The algorithm of SIFT is composed of several parts:

1) Scale-space extrema detection: The input image is convolved with Gaussian filters at different scales, and then the difference of successive Gaussian-blurred images are taken. Keypoints are then taken as maxima/minima of the Difference of Gaussians (DoG) that occur at multiple scales. A DoG image $D(x, y, \sigma)$ is given by

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

Where $L(x, y, k\sigma$ is the convolution of the original image $I(x, y)$ with the Gaussian Blur $G(x, y, k\sigma$ at scale $k\sigma$, i.e.,

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

Hence a DoG image between scales $k_i\sigma$ and $k_j\sigma$ is the difference of the Gaussian-blurred images at scales $k_i\sigma$ and $k_j\sigma$. For scale space extrema detection in the SIFT algorithm, the image is first convolved with Gaussian-blurs at different scales. The convolved images are grouped by octave (an octave corresponds to doubling the value of σ), and the value of k_i is selected so that we obtain a fixed number of convolved images per octave. Then the Difference-of-Gaussian images are taken from adjacent Gaussian-blurred images per octave.

Once DoG images have been obtained, keypoints are identified as local minima/maxima of the DoG images across scales. This is done by comparing each pixel in the DoG images to its eight neighbors at the same scale and nine corresponding neighboring pixels in each of the neighboring scales. If the pixel value is the maximum or minimum among all compared pixels, it is selected as a candidate keypoint.

2) Keypoint localization: At each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability.

3) Orientation assignment: One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.

4) Keypoint descriptor: The local image gradients are measured at the selected scale in the region around each keypoint. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination.

B. SURF

SURF (Speeded Up Robust Features) [2] is partly inspired by the SIFT descriptor. The standard version of SURF is several times faster than SIFT and claimed by its authors to be more robust against different image transformations than SIFT.

The SURF algorithm is based on the same principles and steps as SIFT; but details in each step are different. The algorithm has three main parts: interest point detection, local neighborhood description and matching.

1) Detection: SURF uses square-shaped filters as an approximation of Gaussian smoothing. (The SIFT approach uses cascaded filters to detect scale-invariant characteristic points, where the difference of Gaussians (DoG) is calculated on rescaled images progressively.) Filtering the image with a square is much faster if the integral image is used:

$$S(x, y) = \sum_{i=0}^{x}\sum_{j=0}^{y} I(i, j)$$

The sum of the original image within a rectangle can be evaluated quickly using the integral image, requiring evaluations at the rectangle's four corners.

SURF uses a blob detector based on the Hessian matrix to find points of interest. The determinant of the Hessian matrix is used as a measure of local change around the point and points are chosen where this determinant is maximal. In contrast to the Hessian-Laplacian detector by Mikolajczyk and Schmid, SURF also uses the determinant of the Hessian for selecting the scale, as is also done by Lindeberg. Given a point $p = (x, y)$ in an image I, the Hessian matrix $\mathbf{H}(p, \sigma)$ at point p and scale σ, is:

$$H(p,\sigma) = \begin{pmatrix} L_{xx}(p,\sigma) & L_{xy}(p,\sigma) \\ L_{yx}(p,\sigma) & L_{yy}(p,\sigma) \end{pmatrix}$$

where $L_{xx}(p, \sigma)$ etc. is the convolution of the second-order derivative of Gaussian with the image $I(x, y)$ at the point p.

The box filter of size $9*9$ is an approximation of a Gaussian with $\sigma = 1.2$ and represents the lowest level (highest spatial resolution) for blob-response maps.

2) Scale-space representation and location of points of interest: Images are repeatedly smoothed with a Gaussian filter, then they are subsampled to get the next higher level of the pyramid. Therefore, several floors or stairs with various measures of the masks are calculated:

$$\sigma_{\text{approx}} = \text{current filter size} \times \left(\frac{\text{base filter scale}}{\text{base filter size}} \right)$$

The scale space is divided into a number of octaves, where an octave refers to a series of response maps of covering a doubling of scale. In SURF, the lowest level of the scale space is obtained from the output of the $9 * 9$ filters.

3) Descriptor:

- Orientation assignment: In order to achieve rotational invariance, the orientation of the point of interest needs to be found. The Haar wavelet responses in both x- and y-directions within a circular around the point of interest are computed.
  The obtained responses are weighted by a Gaussian function centered at the point of interest, then plotted as points in a two-dimensional space. The dominant orientation is estimated by calculating the sum of all responses within a sliding orientation window of size $\pi/3$. The horizontal and vertical responses within the window are summed. The two summed responses then yield a local orientation vector. The longest such vector overall defines the orientation of the point of interest.
- Descriptor based on the sum of Haar wavelet responses: To describe the region around the point, a square region is extracted, centered on the interest point and oriented along the orientation as selected above. he Haar wavelet responses are extracted at 5x5 regularly spaced sample points.

C. Deep learning

In deep learning methods, we first extracted proposals from each sample image by selective search implemented by Matlab, and then we use these proposals as input for deep learning neural network and output their feature as 2048-dim vectors.

Here we use Resnet101 as our model, and we firstly train 100 epochs using original dataset, and get an classification accuracy of 95% on test set. And then we use proposals as input and output their feature vector after convolutional layers of neural network after view function(2048-dim vectors) while just before fully-connected layers (classifier).

1) Selective Search by bounding box: For better image segmentation, we first convert original RGB image into gray image using binaryzation. And then use selective search methods to get bounding box of different proposals for each gray image [3]. Because image after binaryzation is easier to compute bounding box region. We then apply bounding box on the same region on original RGB image.

Finally, we crop the RGB image with bounding box to get proposals. Because some of the proposals are very small and less meaningful, we use a threshold of image size to limit the number of proposals get from each image. And we can successfully get some proposals such as the eyes, hands or ears of dogs as shown in Fig. 2.

We implement seletive search by Matlab2016a.

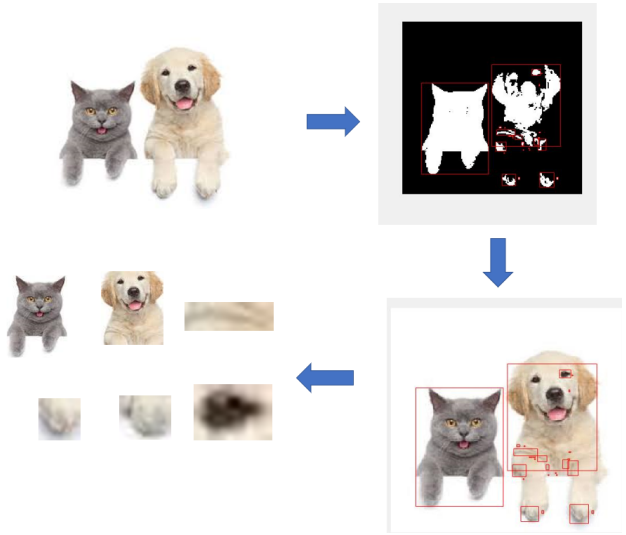


Fig. 2. Selective search

2) Deep learning neural network: Resnet101 [4]: In this part, we first use Resnet101 model pretrained by ImageNet and add parts of images for training (60%) and others for testing. After training 100 epochs, we get an classification accuracy of 95% (Fig.4) on testing set by classifier composed of one fully connected layer.

After model preparation, we saved the trained model and use this model to generate deep learning feature for proposal images.

We implement this part with environment pytorch1.0.1 .

Layer name	101-layer resnet
conv1	7x7, 64, stride 2 3x3 max pooling, stride 2
Conv2_x	1x1, 64 3x3, 64 1x1, 256 x3
Conv3_x	1x1, 128 3x3, 128 1x1, 512 x8
Conv4_x	1x1, 256 3x3, 256 1x1, 1024 x36
Conv5_x	1x1, 512 3x3, 512 1x1, 2048 x3
Classifier	Average pool , 1000-d fc , softmax

Fig. 3. Structure of Resnet101

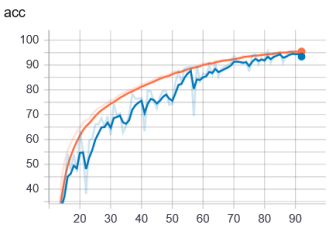


Fig. 4. Train(red) and test(blue) accuracy

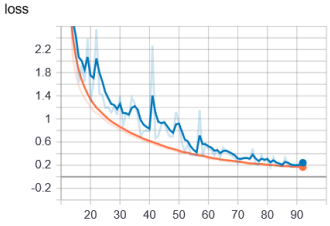


Fig. 5. Train(red) and test(blue) loss

II. Feature encoding

A standard approach to describe an image for classification and retrieval purposes is to extract a set of local descriptors, encode them into a high dimensional vector and pool them into an image-level signature.

A. Bag-of-word

The bag-of-word(BoW) [5] method is largely inspired by the bag-of-words concept which has been used in text mining. In BoW model, each image is represented as a collection of local properties. The two main steps are code book training and feature pooling.

1) Train a Codebook
   - Feature extraction: This is what we stated in Section refsec:extraction. Through SIFT or other extraction method, we can get the local descriptors of all the training images.
   - Feature clustering: Through clustering methods such as K-means, all derived local descriptors can be divided into K clusters. The center of each cluster is called a word in BoW model. All the words form a codebook.
2) Feature Pooling
   - Feature extraction : The extraction process is totally the same as what we do before. However this time, the extraction objects are not only training images but also test images.
   - Feature encoding : For each local descriptor, we compute its distance to each word in our codebook. And we classify the local descriptor to the cluster whose center is nearest. By this way, each image can be represented by the BoW frequency histograms of the visual vocabulary of the codebook.

BoW model defines BoW histograms as the feature representation for images. The similarity of images can be measured by comparing between the BoW histograms. The dimension of BoW vector is K, the size of codebook and also the number of clusters in cluster model (K-means in this project).

B. VLAD

Vector of locally aggregated descriptors (VLAD) starts by vector quantizing a locally invariant descriptor such as SIFT. It differs from the BoW image descriptor by recording the difference from the cluster center, rather than the number of SIFTs assigned to the cluster [6].

The algorithm for VLAD is shown in Alg. 16.

Algorithm 1 An example for format For & While Loop in Algorithm

Input: A set of descriptors $x_1, \ldots, x_T$
A set of centroids (codebook) $\mu_1, \ldots, \mu_K$
for all $i = 1, \ldots, K$ **do**
  $v_i := 0_d$
end for
# accumulate descriptor
for all $t = 1, \ldots, T$ **do**
  $i = \arg\min_j ||x_t - -\mu_j||$
  $v_i := v_i + (x_t - \mu_i)$
end for
$V = [v_1^T \ldots x_K^T]$
# apply power normalization
for all $j = 1, \ldots, Kd$ **do**
  $V_j := sign(V_j)|V_j|^\alpha$
end for
# apply L2 Normalization
$V := \frac{V}{||V||^2}$

C. Fisher vector

Fisher vector [7] describes local descriptors by their deviation from an universal generative Gaussian mixture model. This representation has many advantages over Bag-of-word: it is efficient to compute, it leads to excellent results even with efficient linear classifiers, and it can be compressed with a minimal loss of accuracy using product quantization.

For a picture, suppose it has T extracted local descriptors, each local descriptor is of dimension D, Then we can describe this picture with the matrix $\boldsymbol{X} = \{\boldsymbol{x}_t, t = 1, \cdots, T\}$.

To get the fisher vector of each image, first, we should generate the GMM model, a mixture of K Gaussian distribution

with dimension of D from all images' local descriptor matrix. Suppose the learnt GMM model's parameter set is:

$$\lambda = \{\omega_k, \mu_k, \Sigma_k, k = 1, \cdots, K\}$$

We have a hypothesis here: the T local descriptors are independent and identically distributed (i.i.d), therefore:

$$p(\boldsymbol{X}|\lambda) = \prod_{t=1}^{T} p(\boldsymbol{x}_t|\lambda),$$

where $p(\boldsymbol{X}|\lambda)$ represents the probability of the image $\boldsymbol{X}$ generated from the GMM model specified by parameter set λ.

Take the logarithm of it, then we can get:

$$\mathcal{L}(\boldsymbol{X}|\lambda) = \sum_{t=1}^{T} logp(\boldsymbol{x}_t|\lambda)$$

Since we use a mixture of K Gaussian distributions to describe the distribution of local descriptors, the GMM model can be represented as follows:

$$p(\boldsymbol{x}_t|\lambda) = \sum_{i=1}^{K} \omega_i p_i(\boldsymbol{x}_t|\lambda_i),$$

where ω_i is the i-th Gaussian distribution's weight, sum(ω_i) = 1. p_i is the i-th Gaussian distribution:

$$p_i(\boldsymbol{x}|\lambda) = \frac{exp\{-1/2(\boldsymbol{x} - \boldsymbol{\mu}_i)' \sum_i^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_i)\}}{(2\pi)^{D/2} |\sum_i|^{1/2}}$$

We can calculate the probability that the local descriptor $\boldsymbol{x}_t$ is generated by the i-th Gaussian distribution:

$$\gamma_t(i) = p(i|\boldsymbol{x}_t, \lambda) = \frac{\omega_i p_i(\boldsymbol{x}_t|\lambda)}{\sum_{j=1}^{N} \omega_j p_j(\boldsymbol{x}_t|\lambda)}$$

Then, we can get the following partial derivatives:

$$\frac{\partial \mathcal{L}(\boldsymbol{X}|\lambda)}{\partial \omega_i} = \sum_{t=1}^{T} \left[\frac{\gamma_t(i)}{\omega_i} - \frac{\gamma_t(1)}{\omega_i} \right] \quad \text{for } i \geq 2$$

$$\frac{\partial \mathcal{L}(\boldsymbol{X}|\lambda)}{\partial \mu_i^d} = \sum_{t=1}^{T} \gamma_t(i) \left[\frac{x_t^d - \mu_i^d}{(\sigma_i^d)^2} \right]$$

$$\frac{\partial \mathcal{L}(\boldsymbol{X}|\lambda)}{\partial \sigma_i^d} = \sum_{t=1}^{T} \gamma_t(i) \left[\frac{(x_t^d - \mu_i^d)^2}{(\sigma_i^d)^3} - \frac{1}{\sigma_i^d} \right],$$

where i represents the i-th Gaussian distribution, d represent the d-th dimension of $\boldsymbol{x}_t$.

After calculating the partial derivatives, we should normalize them, and then we get the fisher vector composed of those partial derivatives. If we count in the weight of each Gaussian distribution in the GMM model, then the dimension of fisher vector is $(K + 2 * K * D)$, while if we ignore the weight, the dimension of the fisher vector is $(2 * K * D)$.

III. Experiments

In this section, we will introduce the details about the experiment procedure, and the results of different methods will be displayed and compared.

A. About the Dataset

The dataset we use is Animals with Attributes (AwA2) dataset, which consists of 37322 images of 50 animals classes. 60% of the images are divided for training and 40% for testing. The training/test split is completely the same as that in Prj. 1 and Prj. 2. The features we use in Prj. 1 and Prj. 2 are pre-extracted deep learning features, while local descriptors are used for feature encoding in this project. The local descriptors are extracted by SURF, SIFT and deep-learning method.

B. Selection of Extraction Methods

*1) **SIFT vs. SURF**:* As we introduce in Sec. I, SURF and SIFT are two feature detection algorithms to detect and describe local features in images. We first do some experiments based on BoW to compare the two local feature extraction methods. Considering the size of the image are different, we resize each image to 224×224 pixels.

TABLE I
ACCURACY ON SURF AND SIFT (CODEBOOK SIZE:1000, PCA DIMENSION: 128)

Method \ Standardization	L2	Z-Score	L2+Z-Score
SURF	0.16846	0.15547	0.16290
SIFT	0.16753	0.16345	0.16456

As we can see from Table I, the performances between SURF and SIFT do not make much difference. Although SURF with L2 Normalization gains the highest accuracy, average performances of SIFT are better than SURF. Generally, SURF is better than SIFT in rotation invariant, blur and warp transform. SIFT is better than SURF in different scale images [8]. The images of AwA2 are of different size however nearly without rotation or intentional blur. We think this is why SIFT performs a little bit better. Also considering the requirement is to extract local descriptor using SIFT, thus we do not pay any attention to SURF anymore.

*2) **SIFT vs. Deep-Learning**:* In addition, we make an effort to extract local descriptors by deep learning method. We believe that the neural network ResNet101 is of no problem. The classification accuracy on testing set (95%) also demonstrates this. However, the Selective Search process falls short of the desired effect. On average, the number of proposals searched from each images is around 10.

10 local descriptors per image is not enough for feature encoding. It is not a good idea to set the size of the codebook too small. It is unlikely to get a good result if the codebook size is far below the number of animals(also the number of labels). However, if we set the size of the codebook large, then

most of the bits in a feature vector derived by BoW or VLAD will be 0. It is unsatisfactory too. As a result, whether we set the size of codebook large or small, the classifier accuracy is around 0.120 in BoW, and around 0.140 in VLAD.

In addition, we use the 2048-dim feature output by ResNet101 as the local descriptor of the image. That is to say, one image with only one proposal, derives one 2048-dim local descriptor. Then, we perform KMeans on the 2048-dim feature vector. Here, KMeans is equivalent to BoW because each image has only one local descriptor. The results shows that the classification accuracy reaches 0.82055. The accuracy is defined as that the probability of the original class is still a class in KMeans. This result shows the power of deep learning.

On conclusion, these local descriptors extracted by us is not good. Though, We do not give up the deep learning extraction method, because we believe well-trained deep learning model will perform much better than traditional method (Also proved from the side by the last paragraph). Unfortunately, the time is not enough for us to derive a decent result right now. However, we will still stick to deep learning method and make an effort to make distinguished progress.

C. Selection of Local Descriptors' Form

SIFT returns a Matrix $\boldsymbol{S}_{m\times k}$ for each image, in which m is the number of local descriptors and k is the dimension of a local descriptor vector (128-dim) .There is a large difference between the value of m for different image.m for some images reaches 10000 while for some is less than 100. We think there are two main reasons:

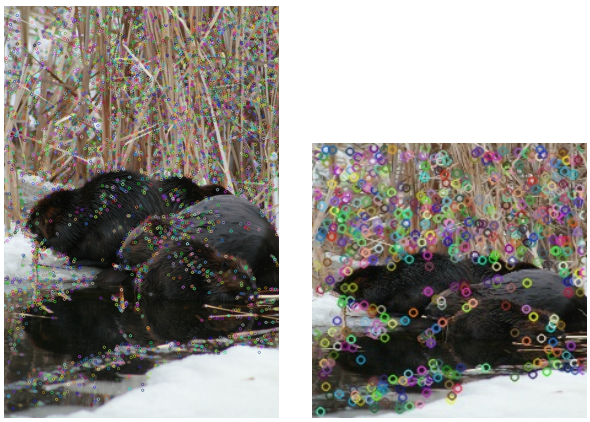
- The size of images in AwA2 varies greatly.
- Different kinds of animals vary greatly. Some animals possess obvious characteristics of each part of the body while the others do not.

To explore the effect of the unbalance of the number of local descriptors, we did a series of experiment. We extract all the local descriptors of the images using cv2 Library Function. This $80GB$ file represents **All-feature form**. We also take the first 100 features (100^{st}**-feature form**), hoping to capture the key features as well as make a balance of different sample on the number of local descriptors. Moreover, we resize all the pictures into 224×224 pixels (224×224**-feature form**), hoping to resolve the inequality of image size. This operation reduces the file size by 14 times $(5.7GB)$ and makes the number of different sample's local descriptors more balanced.

TABLE II
ACCURACY ON DIFFERENT FORM (CODEBOOK SIZE:1000, PCA DIMENSION: 128)

Feature \ Standardization	L2	MaxMin-Score	L2+MaxMin-Score
All-feature	0.34700	0.31776	0.34168
100^{st}-feature	0.18277	0.18646	0.18725
224×224-feature	0.16846	0.15547	0.16290

Table II shows that the performance of 224×224-feature is the worst. We think this is because the degree of distortion for different image is not the same. The original size differs greatly, such as 1024×683, 683×1024, 1024×768 and 1000×607. When they are resized to 224×224 pixels, the force of the scaling on different directions for different images are not the same. For example, images with size of 1024×683 are stretched on horizontal-axis compared to vertical-axis shown in Fig 6, while images with size of 683×1024 is stretched on vertical-axis in Fig. 7.



(a) original image (b) resized image

Fig. 6. 1024×683 image being stretched on horizontal-axis

(a) original image (b) resized image

Fig. 7. 683×1024 image being stretched on vertical-axis

The performance of the 100^{st}-feature is far behind the origin feature. This demonstrates the latter features are not noise. On the contrary, they contain a lot of valid information. Although the number of local descriptors in All-feature is severely unbalanced. By applying Standardization to each feature vector, the numerical value in feature vector could represents the frequency of each type of the local descriptor. Plus the complete information of the images, All-feature obtains a relatively competitive performance.

To cluster all the local descriptors in our computer is nearly impossible. So in All-feature experiment, we just randomly select 5% local descriptors in training set to be fed to the Kmeans cluster model and use all the local descriptors for encoding. Even so, it is also a burden for us. Therefore, we

run experiment on 100^{st}-feature most time, and All-feature is only used when we need final highest accuracy.

D. Feature encoding method

1) BoW:

- **Effect on the Normalization**
  Considering the number of local descriptors for each image varies greatly. For example, two images are alike with different size. Then the type of local descriptors extracted from them are likely to be similar but the number of each type of local descriptors may be different. Therefore, just use the BoW histogram as the feature representation for images may be unreasonable. The ratio of each type of local descriptors may become a better choice. Therefore, we try to perform normalization for each feature vector. Here we use L2-Normalization.

$$L2 : dst(i,j) = \frac{src(i,j)}{\sqrt{\sum src(x,y)^2}}$$

  Before the features are fed to PCA, it is usually need to perform standardization. This is different from above. **The L2-Normalization is applied to each sample (a row in the dataset). Here the standardization is applied to each feature dimension of all samples (a line in the dataset).** Here we try StandardScaler and MinMaxScaler.
  The StandardScaler uses Z-score method. The score of a sample x is calculated as:

$$z = (x - u)/s$$

  where u is the mean of the training samples and s is the standard deviation of the training samples .
  For MinMaxScaler:

$$X_std = (X - min)/(max - min)$$

$$X_scaled = X_std * (max - min) + min$$

  where max and min are the maximum and minimum of the line(axis=0).

TABLE III
ACCURACY ON DIFFERENT STANDARDIZATION (100^{st}-FEATURE, CODEBOOK SIZE:1000, PCA DIMENSION: 128)

Method	L2	Z-score	MinMax	L2 + Z-score	L2 + MinMax
Accuracy	0.18277	0.13820	0.14844	0.14817	0.19267

  From Table III, we can see that, L2 performs best when 3 methods are used alone. This is in agreement with what we state in the previous paragraph. The standardization within each vector sample is more important than standardization for each feature dimension of all samples. In addition, MinMax standardization performs better that Z-score standardization. We think this is because the data do not obey Gaussian distribution. MinMax standardization is more closer to the ratio, which is more suitable in this project.
  Based on this, we perform L2 and MinMax standardization to the All-feature dataset. Results are shown in Table IV. The results are similar to those on 100^{st}-feature, L2- normalization for each sample is more essential.

TABLE IV
ACCURACY ON DIFFERENT STANDARDIZATION (ALL-FEATURE , CODEBOOK SIZE:1000, PCA DIMENSION: 128)

Method	L2	MinMax	L2 + MinMax
Accuracy	0.34700	0.31776	0.34105

- **Effect of the value C in SVM**
  In this project, we use Linear SVM as the classifier. For the consistency with the previous Projects, we set the linear kernel as fixed. Among the other parameters of SVM, we find C has a huge impact on the classification accuracy.

TABLE V
ACCURACY ON DIFFERENT VALUE OF C (ALL-FEATURE , CODEBOOK SIZE:1000, PCA DIMENSION: 128)

Value of C	1.0	5.0	10.0
Accuracy	0.34700	0.35523	0.34105

2) VLAD:

- **Effect of the Size of Codebook and PCA dimension**
  For VLAD method, the dimension of feature vector is $(K * D)$, K is the number of clusters in KMeans, also the size of codecook. Considering the $(K * D)$ is a long vector, We explore the effect of the size of codebook and PCA dimension. The feature we use is the 100^{st}-feature. As we can see, with the size of codebook varying from 32 to 128, and the PCA dimension equaling to 128 or 256, the accuracy does not change greatly. All 6 accuracy values are between 0.22 to 0.23, which demonstrates that these two factors do not have a significant impact.
  On the whole, PCA_dim $= 128$ is better than PCA_dim$=$ 256, whatever the size of the codebook. This phenomenon is a proof that the VLAD vector is of redundancy. Reducing it to a relatively low-dimension space not only removes the noise but also speeds up the computing process for SVM. The impact of the codebook size is even weaker. The range of accuracy change caused by codebook size is around 0.001. Therefore, we are confident that 32 is a large enough codebook for 100^{st}-feature.

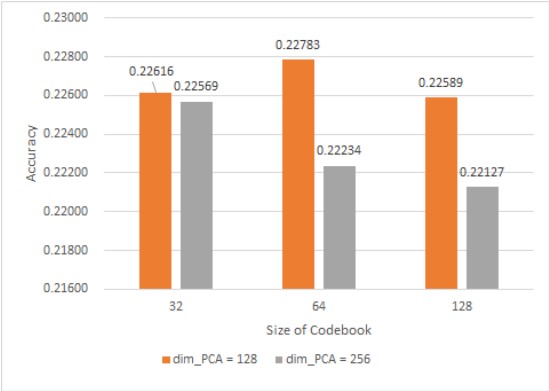


Fig. 8. Effect on the Size of Codebook and PCA dimension

Final Result of VLAD

Based on the previous experiment, we know L2 + MinMaxScaler is a good standardization. $C = 5$ in SVM is a good value than default. We also set these for VLAD on All-feature. These settings do give us a comparatively good result. In this part, one of the point we find that is different from the previous experiment is the size of codebook. When we do experiment on 100^{st}-feature, we find that codebook size does not make much difference. However, the size of codebook in All-feature has larger significance. When it is set as 128, VALD obtains the highest classification accuracy 0.37712. We think this is because the number of local descriptors are larger than 100^{st}-feature. The modality of local descriptors must be more abundant too. Therefore a large codebook makes a positive effect.

TABLE VI
ACCURACY ON DIFFERENT CODEBOOK SIZE (ALL-FEATURE , PCA DIMENSION: 128, C = 10)

Codebook Size	32	64	128
Accuracy	0.35425	0.34093	0.37712

3) Fisher Vector:

- **Effect of normalization**

  Since the number of local descriptors of each image varies greatly, we should do normalization on each fisher vector. Similarly, each feature of the codebook differs from each other, thus we should do normalization on each feature as well. The total dataset after feature encoding with fisher vector is a matrix with $2*K*D$ columns and N rows, where K represents the number of features in the codebook, D represents the dimension of each feature in the codebook, and N represents the number of all samples in the dataset.

  We do 'max' normalization and 'MinMax' normalization on each row and each column.

  Suppose $\boldsymbol{X}$ is a vector of length n, and say that the normalized vector is $\boldsymbol{y} = \boldsymbol{x}/z$. The 'max' norm will use $\max \boldsymbol{x}_i$ to denote z, while the 'MinMax' norm will use $y = (\boldsymbol{x} - \min \boldsymbol{x}_i)/(\max \boldsymbol{x}_i - \min \boldsymbol{x}_i)$ to represent the normalized vector. The classification results can be seen as Tab. VII

TABLE VII
ACCURACY ON DIFFERENT NORMALIZATION (CODEBOOK SIZE:3, CLASSIFICATION METHOD: LINEAR SVM)

Norm	without norm	max on column	max on row	MinMax on column	MinMax on row
Accuracy	0.14918	0.12466	0.12278	0.13169	0.13866

As we can see, the dataset without normalization achieved the highest accuracy. 'MinMax' norm achieves better result than 'max' norm. Maybe that is because when the feature data has tiny difference from each other, 'max' norm cannot distinguish them, thus that feature is useless in the classification step. Overall, doing normalization on row performs better than doing normalization on column. This is also in agreement with what we state in the previous paragraph: the standardization within each vector sample is more important than standardization for each feature dimension of all samples.

- **Effect of PCA dimension**

  For fisher vector method to do feature encoding, the dimension of each feature vector is $(2*K*D)$. Since $(2*K*D)$ is a long vector, we use PCA to do dimension reduction. The classification results can be seen as Tab. VIII.

TABLE VIII
ACCURACY ON DIFFERENT PCA DIMENSIONS (CODEBOOK SIZE:3, CLASSIFICATION METHOD: LINEAR SVM)

PCA dimension	without PCA	PCA-64	PCA-128	PCA-256	PCA-512
Accuracy	0.14918	0.13953	0.14502	0.14924	0.14924

As we can see, the dataset with dimension reduced to 256 and 512 using PCA achieves highest accuracy. This result reveals that fisher vector is of redundancy as well. Reducing the dataset to a relatively low-dimension space will not only remove the noise, but also reduce the computation of SVM classification.

- **Effect of SVM kernel**

  When we use SVM to do classification and the dataset is not linear separable, usually we map the dataset to a higher dimension, and make the dataset separable there. Kernel function can help us map the dataset to higher dimension. There are many kernel functions. Linear kernel is usually used when the dataset is linear separable and the feature space is huge. Polynomial kernel is usually used in image processing tasks. Gaussian radial basis kernel (RBF kernel) is a commonly-use kernel in linear inseparable situations, and the feature space is not so huge. Sigmoid kernel is kind of similar to RBF kernel.

The classification results of different kernel functions for SVM can be seen as Tab. IX.

TABLE IX
ACCURACY ON DIFFERENT KERNEL FUNCTIONS (CODEBOOK SIZE:3, CLASSIFICATION METHOD: SVM)

Kernel	Linear	Polynomial	RBF	Sigmoid
Accuracy	0.14918	0.04407	0.04407	0.06370

As we can see, SVM with linear kernel performs best. Maybe that is because each dimension in fisher vector is a representation of a Gaussian distribution, they are naturally separable, thus there is no need to map the dataset to a higher dimension to get high accuracy.

- **Effect of different codebook size** In VLAD, $K = 32$ is a good choice for 100_{st} feature and $K = 128$ for All-feature. A little bit large K makes higher accuracy. However, fisher vector is very time consuming when using the features we extracted, so we only try $K = 3$ and $K = 5$. The results are shown as Tab. X.

TABLE X
ACCURACY ON DIFFERENT K FOR SIFT FEATURE (PCA DIMENSION: 512, CLASSIFICATION METHOD: LINEAR SVM)

Codebook size	3	5
Accuracy	0.14918	0.12399

IV. Conclusion

In this project, we implement three different feature extraction methods: SURF, SIFT, and deep learning. For deep learning, we implement two distinct methods. One is to extract proposals first and then use DNN to output the proposals' features, the other is to use an end-to-end DNN and output one feature vector for one image.

Through feature selection, we find that the feature extracted by SIFT method has best overall performance in the following encoding and classification step. Therefore, we verify the performance of different feature encoding methods using SIFT feature. The three feature encoding methods we verify are BoW, VLAD and fisher vector. The highest classification accuracy using SVM on their encoded features can be seen as Tab. XI.

TABLE XI
BEST PERFORMANCE OF DIFFERENT FEATURE ENCODING METHODS

Method	Accuracy
BoW	0.35523
VLAD	0.37712
Fisher vector	0.14924

Although fisher vector does not perform better than the other two feature encoding methods according to the table above, it has better performance with the same codebook size.

REFERENCES

[1] P. C. Ng and S. Henikoff, "Sift: Predicting amino acid changes that affect protein function," *Nucleic acids research*, vol. 31, no. 13, pp. 3812–3814, 2003.

[2] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[3] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[4] Z. Lu, X. Jiang, and A. Kot, "Deep coupled resnet for low-resolution face recognition," *IEEE Signal Processing Letters*, vol. PP, no. 99, pp. 1–1, 2018.

[5] F. Jurie and B. Triggs, "Creating efficient codebooks for visual recognition," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 1. IEEE, 2005, pp. 604–610.

[6] R. Arandjelovic and A. Zisserman, "All about vlad," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2013, pp. 1578–1585.

[7] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *International journal of computer vision*, vol. 105, no. 3, pp. 222–245, 2013.

[8] D. Mistry and A. Banerjee, "Comparison of feature detection and matching approaches: Sift and surf," *GRD Journals- Global Research and Development Journal for Engineering*, vol. 2, pp. 7–13, 03 2017.