Data Science Course Project 3 – Feature Encoding

Liwei Kang, Xing Zhao, Lanxuan Zhou, Zhengyi Li

***Abstract*— In this report, we will introduce the work we have done for this project. Roughly, it can be divided into two parts, feature encoding with SIFT descriptors and feature encoding with deep learning features of image proposals. We implemented three feature encoding methods, including Bag-Of-Word, VLAD, and Fisher Vector Encoding. Code for this project will be released at https://github.com/Olafyii/Feature_Encoding_Project in a week.**

I. INTRODUCTION

The objective of this course project is to perform feature encoding on images to get feature vectors for each image. And then use a linear SVM classifier to perform image classification task using the encoded feature vectors to evaluate how discriminative and representative the encoded features are. We perform feature encoding on both SIFT descriptors and deep learning features of extracted proposals. Then we evaluate the encoded feature vectors by using a linear SVM classifier and calculating its classification accuracy. Unfortunately, the accuracy of linear SVM classifier on encoded features based on SIFT descriptors and deep learning features of proposals are all uncompetitive (Never higher than 50%, See details in Table I, II and III) compare to classification accuracy (92.3%) on the original deep learning features from AwA2 dataset. We will discuss in details about the possible reasons to the failure of the accuracy.

II. SIFT DESCRIPTORS

A. Introduction of SIFT descriptor

The scale-invariant feature transform (SIFT) is a feature detection algorithm in computer vision to detect and describe local features in images. The SIFT descriptor is known to be invariant to uniform scaling, orientation, illumination changes, and partially invariant to affine distortion, it's a rather robust image descriptor.

B. Calculation of SIFT descriptor

Using AwA2 dataset, we have 37322 images in total, denote as $\{I_i | i = 1, 2, 3, ..., 37322\}$. We leverage the opencv library to calculate SIFT descriptors of each image. The bag of descriptors of each image i is denoted as d_SIFT_i with shape $(k_i, feature\_dim)$, where k_i is the number of key points calculated from *cv2.SIFT()* function and *feature_dim* is 128 for SIFT descriptor. If we make no limitation to k_i, some images (especially zebras) will have unreasonably large k_i being more than 10000, which leads to a total storage cost of all SIFT descriptors reaching 70 Gigabytes. With such large files, we have to cut the dataset to 10% of its size, to 3709 images, and then use 60% (2225) images for training and 40% (1484) for testing. Yet we believe that such large files (larger than image file itself) must contain a lot of redundant information, and training and testing on 3709 images might be insufficient for classifier to converge. Thus, to improve performance and reduce computational cost and information redundancy, we make such constrain to SIFT descriptors of all images:

$$k_i \leq 200, \forall i \tag{1}$$

By making this constrain, the storage of SIFT proposals is reduced to 3.7 Gigabytes. Then we can train and test the SVM classifier with the full dataset and the constrained SIFT descriptor. Figure 1 shows a demonstration of SIFT descriptors of a zebra picture and a chimpanzee picture with and without the constrain. We can see that for the zebra, the one without constrain includes many key points on the grassland, which is irrelevant to our classification object, the zebra. Thus by making the constrain on the number of key points, the SIFT descriptors become more focus on the animal than the background. For the chimpanzee picture, though making the constrain makes less improvement to the attention of the animal, most

(a) SIFT descriptor with no constrain (b) SIFT descriptor with constrain (c) SIFT descriptor with no constrain (d) SIFT descriptor with constrain

Fig. 1. Effect of the constrain on SIFT descriptor

of the irrelevant keypoints are removed, which will be beneficial when learning the codebook. Because we would like the total number of irrelevant key points of all images be as small as possible.

III. Deep learning features of image proposals

Instead of using SIFT keypoints as descriptors for an image, we can also use deep learning features as descriptors for an image. For each image, the bag of deep learning descriptors is denoded as d_DL_i with shape $(k_i, feature\_dim)$, where k_i is the number of proposals of the image and *feature_dim* is the dimension of each deep learning features. Extracting bag of deep learning features can be divided as a two-stage task, the first stage is to extract image proposals (bounding boxes) of an image, and the second stage is to calculate deep learning features for each image proposal.

A. Extract image proposals

There are many off-the-shelf methods to extract proposals for an image, such as Edge Boxes algorithm, Region Proposal Network (RPN) and selective search algorithm. In our work, we choose selective search algorithm because it is relatively efficient and does not require extra training with well-annotated data. Moreover, region proposal extraction will be conducted only once during the whole project, without introducing much computational cost. For selective search, we constrain that the area of each image proposal no less than 500, and the length of the longer edge divided by the length of the shorter edge is less than 1.5. The above constrains will make sure that all proposals are not too small or too distorted, the reason of making such constrains will be discussed later in this section. For the number of image proposals, empirically, we believe 20 proposals would be enough to describe a picture (if most of the proposals are foreground i.e. the animal, not background). Thus for deep learning features of image proposals, we also make the following constrain:

$$k_i \leq 20, \forall i \tag{2}$$

Since selective search algorithm doesn't provide a confidence for each proposal being the foreground, we don't know which proposal is more likely to be foreground and which one is more likely to be background. To meet the constrain of number of proposals of an image, we simply sample 20 proposals from the extracted proposals of selective search algorithm that satisfy the first two constrains (not too small and not too distorted) if the total number of proposals that satisfy the first two constrains is more than 20.

B. Calculate deep learning features of image proposals

There are many networks to extract image features, to be consistent with previous projects, we use ResNet101 network to extract features of image proposals. Note that the provided deep learning features of AwA2 dataset has a dimension of 2048, which is the output of the second to last layer of the ResNet101 network. However, 2048 is too large for our experiment platform (even with a server with 120G memory, still give the 'Memory error' error), because in the feature encoding stage, the dimension of the fisher vector of an image would be $2Kd$, where K is the codebook size, let's say 100, and d is the dimension of of the feature vector of the proposal, let's say 2048,

then $2Kd = 2 * 100 * 2048 = 409600$. It's even too large to feed the feature vector matrix of training set (with shape (22393, 409600), where 22393 is 0.6*37322) to a dimension reducer to do dimensionality reduction task. So to continue our experiment, we include the last fully connected layer of ResNet network to extract deep learning features. Then the dimensionality of the output of the ResNet is 1000, which is still very large, but bearable for our server. Thus each bag of deep learning descriptors d_DL_i has shape $(k_i, 1000)$, where $k_i \leq 20, \forall i$, and for each proposal p in k_i, the following constrains are also satisfied.

$$area(p_i) \geq 500 \tag{3}$$

$$\frac{2}{3} \leq \frac{height(p_i)}{width(k_i} \leq \frac{3}{2} \tag{4}$$

Another problem to consider is that for a pre-trained ResNet network, the input image matrix shoud have fixed size, yet the size of our image proposals are not fixed, thus we need to resize all proposals to a fixed size ((224,224) in our experiments) by first stretching the image proposal along the shorter edge to resize it to a square image and then resize it to (224, 224). This is the reason why we constrain to all image proposals that the length of the longer edge of the proposal divided by the length of the shorter edge of the proposal is less than 1.5 (**Equ.** 4), because if the original proposal is far from being a square image, after the resize operation it will be changed a lot visually, which is considered a defect for extracting deep learning features.

IV. Feature Encoding

Our project can be divided into two stages, the first stage is to get bag of descriptors for each image, and the second stage is to perform feature encoding on bags of descriptors. We implemented three encoding methods: Bag-Of-Word, VLAD and Fisher vector. First we will introduce some mathematical formulas of these encoding methods. Note that since we can't find a sophisticated toolkit that satisfies the requirement of our feature encoding task, we wrote all feature encoding codes by ourselves based on these following formulas.

A. Mathematical formulas

First we aggregate all descriptors together, and fit a Gaussian Mixture Model (GMM) with K (i.e. codebook_size = K) gaussian distributions, denoted as

$$\lambda = \{w_i, \mu_i, \Sigma_i, i = 1 \dots K\} \tag{5}$$

where w_i, μ_i, and Σ_i denote respectively the weight, mean vector and covariance matrix of gaussian distribution i. Let p_i be the distribution of gaussian i so that we have:

$$p(x|\lambda) = \sum_{i=1}^{K} w_i p_i(x|\lambda) = \sum_{i=1}^{K} w_i \mathcal{N}(x|\mu_i, \Sigma_i) \tag{6}$$

Let $\{x_t | x_t \in \mathbb{R}^{feature\_dim}, t = 1 \dots T\}$ be the bag of descriptors of an image, where $T = k_i$, the number of descriptors of an image and *feature_dim* is the dimension of a descriptor (128 for SIFT descriptor and 1000 for deep learning features). We denote $\gamma_i(x_t)$ as the probability that the descriptor x_t is assigned to gaussian distribution i. Using Bayes' formula, we have

$$\gamma_i(x_t) = \frac{w_i p_i(x_t|\lambda)}{\sum_{j=1}^{K} w_j p_j(x_t|\lambda)} \tag{7}$$

In the Bag-Of-Word representation, the descriptors with dimension *feature_dim* is transformed to encoded feature γ_t with dimension K:

$$\gamma_t = [\gamma_1(x_t), \gamma_2(x_t), \dots, \gamma_K(x_t)] \tag{8}$$

and the image-level encoded feature is simply the accumulation of these probabilities of all descriptors:

$$\begin{aligned} \gamma(I) &= \sum_{t=1}^{T} \gamma_t \\ &= \left[\sum_{t=1}^{T} \gamma_1(x_t), \sum_{t=1}^{T} \gamma_2(x_t), \dots, \sum_{t=1}^{T} \gamma_N(x_t) \right] \end{aligned} \tag{9}$$

This representation is called soft BOW, there is another version of BOW where the only difference is that

$$\gamma_i(x_t) = \begin{cases} 1 & \text{if } i = \arg\max_j \{\gamma_j(x_t)\} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

In our experiment, we use soft BOW.

For VLAD and Fisher vector encoding, we use the following two formulas:

$$\Gamma_{\mu_i^d}(I) = \frac{1}{T\sqrt{w_i}} \sum_{t=1}^{T} \gamma_i(x_t) \left(\frac{x_t^d - \mu_i^d}{\sigma_i^d} \right) \quad (11)$$

$$\Gamma_{\sigma_i^d}(I) = \frac{1}{T\sqrt{2w_i}} \sum_{t=1}^{T} \gamma_i(x_t) \left[\frac{\left(x_t^d - \mu_i^d\right)^2}{\left(\sigma_i^d\right)^2} - 1 \right] \quad (12)$$

The VLAD representation and Fisher vector are extensions of the BOW representation. Instead of characterizing an image by the numer of occurrences of each visual word, it is characterized by a gradient vector derived from a generative probabilistic model. The gradient of the log-likelihood describes the contribution of the parameters to the generation process.

The VLAD representation of an image I is same to $\Gamma_{\mu_i^d}(I)$ in **Equ.**11, and Fisher vector representation is concatenation of $\Gamma_{\mu_i^d}(I)$ and $\Gamma_{\sigma_i^d}(I)$ in **Equ.**11 and **Equ.**12.

B. Implementation

First, we aggregate all descriptors of training set images to one matrix. Our training set is a 60% sample (no replacement) of the full dataset, including 22393 images. For SIFT descriptors, where each image has 200 keypoints, the total number of descriptors of training set images is more than 4 millions which will take a lot of time to fit a gaussiam mixture model. So for SIFT descriptors, we sample (no replacement) 1 million descriptors from the aggregated descriptors and fit a gaussian mixture model (i.e. learn a codebook). Then we calculate BOW, VLAD and Fisher vector representations based on SIFT descriptors according to above formulas. Note that we don't do the sampling operation when encoding with the codebook.

For deep learning descriptors, since we constrain that each image has no more than 20 proposals, the total number of descriptors of training set images is in an acceptable range (about 400000 descriptors), so we don't do the sample operation and learn a codebook directly from the full aggregated matrix. Then we calculate BOW, VLAD and Fisher vector representations based on deep learning descriptors according to above formulas.

V. Classification based on encoded features

To validate the discriminability and representativeness of our encoded features of each image, we use a simple linear SVM classifier from sklearn library, with all parameters set as default. To validate such simple classifier is capable to do classification task, we perform classification task on the original deep learning features given by AwA2 dataset. The simple classifier can reach a 92.3% accuracy on testing set, which proves its ability to classify images (under the condition that image features are good).

For BOW representation, the dimension of encoded feature is equal to the number of codebook size, which is a reasonable size, so we don't perform and further preprocessing operation and feed them straight to the classifier for training and testing. For VLAD and Fisher vector representation, the dimension of encoded feature is very large, reaching more than 10000 (with large codebook size can even reach 200000), which is too large for our classifier to train and test, so we use Principle Component Analysis(PCA) to reduce the dimensionality of the encoded features.

VI. Experiment

All our experiments are performed on a ubuntu 16.04 virtual machine on google cloud platform with 16 virtual CPUs, 120G memory and a Tesla P100 graphic card.

Our experiments can be divided into three categories:

- Classification with encoded features based on deep learning features of proposals with full dataset (37322 images in AwA2 dataset)
- Classification with encoded features based on SIFT descriptors with no constrains with 10% dataset (3709 images in AwA2, sample 10% images in each of the 50 categories)
- Classification with encoded features based on SIFT descriptors with constrains with full dataset (37322 images in AwA2 dataset)

Results of our experiments are shown in Table I, II and III.

VII. Analysis

From Table I, II and III, we can see that the classification results are very uncompetitive. For

TABLE I

CLASSIFICATION WITH ENCODED FEATURES BASED ON DEEP LEARNING FEATURES OF PROPOSALS WITH FULL DATASET

codebook size	BOW	VLAD		Fisher Vector	
10	10.40%	PCA_10	4.38%	PCA_10	4.38%
		PCA_20		PCA_20	
		PCA_50		PCA_50	
		PCA_100		PCA_100	
		PCA_200		PCA_200	
50	32.03%	PCA_10	4.38%	PCA_10	4.38%
		PCA_20		PCA_20	
		PCA_50		PCA_50	
		PCA_100		PCA_100	
		PCA_200		PCA_200	
100	40.91%	PCA_10	4.38%	PCA_10	4.38%
		PCA_20		PCA_20	
		PCA_50		PCA_50	
		PCA_100		PCA_100	
		PCA_200		PCA_200	
200	45.08%	PCA_10	4.38%	PCA_10	4.38%
		PCA_20		PCA_20	
		PCA_50		PCA_50	
		PCA_100		PCA_100	
		PCA_200		PCA_200	

TABLE II

CLASSIFICATION WITH ENCODED FEATURES BASED ON SIFT DESCRIPTORS WITH NO CONSTRAINS WITH 10% DATASET

codebook size	BOW	VLAD		Fisher Vector	
10	4.38%	PCA_10	4.85%	PCA_10	23.27%
100	4.38%	PCA_100	4.38%	PCA_100	12.30%

TABLE III

CLASSIFICATION WITH ENCODED FEATURES BASED ON SIFT DESCRIPTORS WITH CONSTRAINS WITH FULL DATASET

codebook size	BOW	VLAD		Fisher Vector	
10	10.24%	PCA_10	6.00%	PCA_10	4.40%
100	14.40%	PCA_100	28.43%	PCA_100	4.40%

encoded features based on deep learning features of proposals, the BOW representation shows a reasonable classification accuracy, and the accuracy increases with the increase of codebook size. Yet the performance of VLAD representation and Fisher Vector representation is nothing but a failure. On the other side, for encoded features based on SIFT descriptors, the experiment data shows a trend that the performance of Fisher Vector representation is better than VLAD representation and VLAD representation is better than BOW representation. Yet there is no clear relationship between codebook size and performance.

Before further analysis, we first take a careful look at the experiment result and prove that our code is feasible for the task and has no logic error. We define 'feasible' as classification score being more than 20%. All our feature encoding processes share the same code framework, only with different parameters such as codebook size, feature_dim (for SIFT is 128, for deep learning feature is 1000, as mentioned above) and the target dimension for PCA dimension reduction with VLAD representation and Fisher Vector representation. From the experiment data, we can see that under some situations, BOW representation, VLAD representation and Fisher Vector representation all reaches an accuracy of 20%, which indicates that our code framework is feasible.

A. Possible Explanation I–Classifier Overfitted

Notice that many classification accuracies is around 4%, we print the predicted label of the testing images, not surprisingly, they are all the same label. Yet, though accuracy on testing set is a complete failure, we notice that the classifier on training set is incredibly good, with an accuracy of 100%. Thus we conjecture that maybe the poor accuracy is due to overfitting of the classifier. So we constrain the model complexity of the svm classifier and perform classification task again. Yet the accuracy on testing set remain unchanged. Indicating that the classifier is innocent to the failure of the accuracy.

B. Possible Explanation II–Encoded features uncapable to describe the image

Since the classifier is proved innocent to the failure of classification accuracy, we then conjecture that the encoded feature vectors are not discriminative and representative at all, which leads to poor classification result. Thus we simulate an extreme condition of lack of discriminability and representativeness: generate a random matrix with size (20000, 128) and all elements randomly selected from 0 to 100, and a label vector with size (20000,) and all elements randomly selected from 50 (i.e. 50 categories). We perform linear SVM classification on this totally random dataset, and generate a random testing set in same manner to validate the classifier. And the accuracy of classifier on testing set is 1.6%, which is not a surprise. Yet, the interesting thing is that the classification result of random testing set is all same label, which is identical to our former experiment result with accuracy around 4%. Thus it is possible that the encoded features (after PCA operation) are lack of the ability to describe an image. To see if the encoded feature is that bad as totally random, we use t-sne to reduce all encoded features (after PCA operation) of 37322 images to a 2-dimension vector and visualize them. Result of scattering these feature vectors on a same plane is shown in Figure 2. As you can see, all categories are entangled with no clear cluster shown in the image.

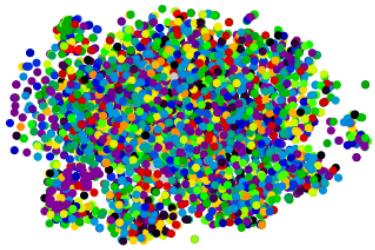

Fig. 2. Visualization of encoded feature vectors

To see clearer of the distribution of each category, we draw distribution of each label in a image, yielding 50 images, as shown in Figure 3. For comparision, we also visualize the original deep learning features from AwA2 dataset, as shown in Figure 4. We can see that for encoded features, the points of a same category is diffused in the plane, with no clear appearance of a cluster, yet for the deep learning features from AwA2 dataset, the points of a same category is well clustered in the plane. From this observation, we can further confirm that the classifier is innocent to the accu-

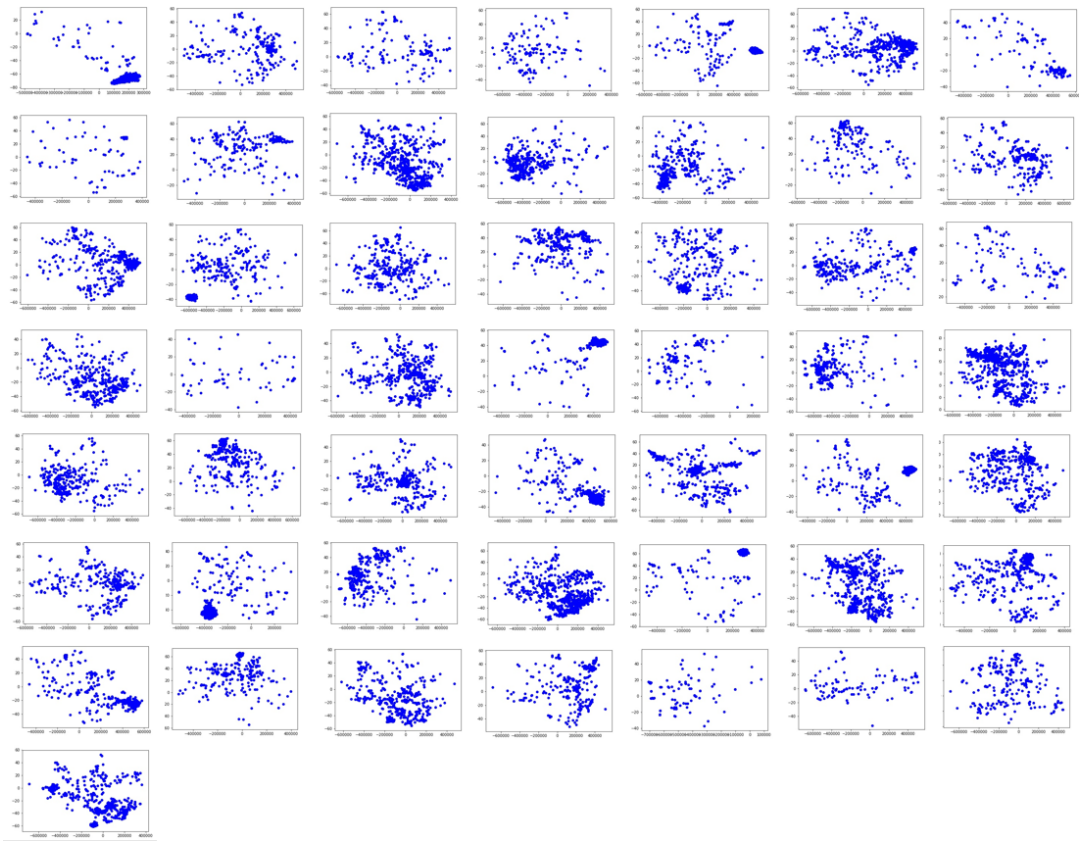

Fig. 3. Visualization of encoded feature vectors (after PCA operation), each image represents a category

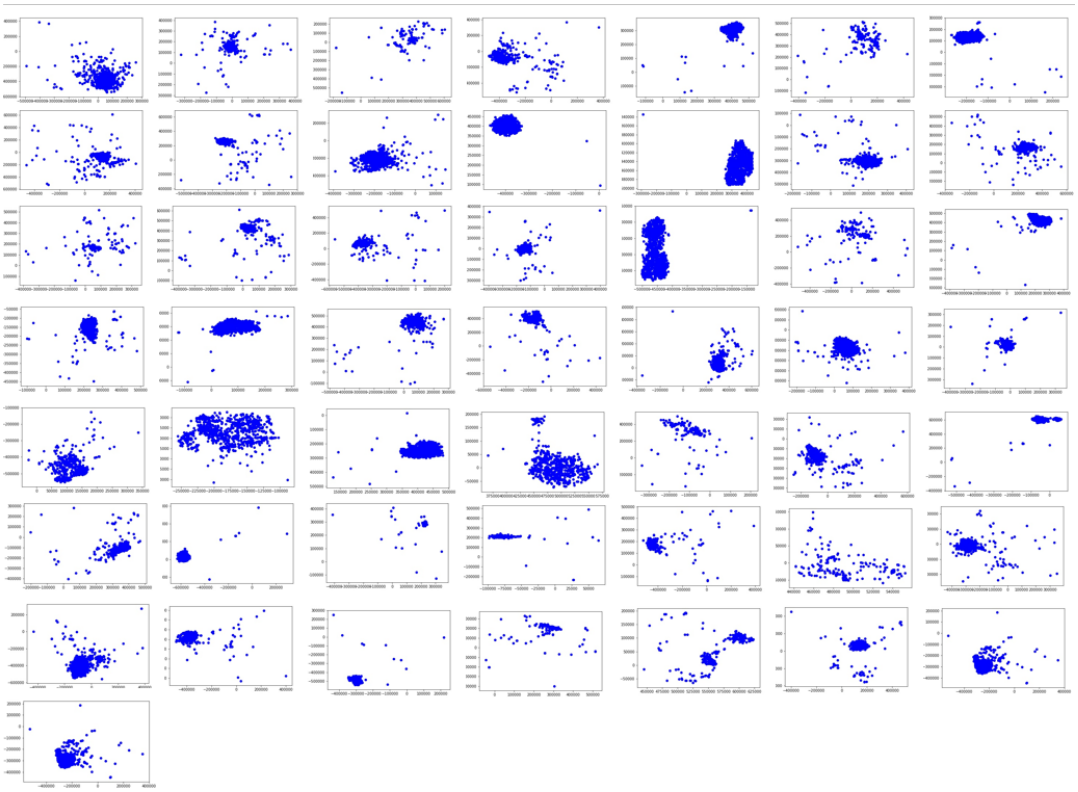

Fig. 4. Visualization of deep learning features from AwA2 dataset, each image represents a category

(a) A chimpanzee image with all proposals satisfying Equ.3 and 4

(b) A chimpanzee image with sampled 20 proposals

Fig. 5. A defect in image proposals

racy failure.

C. Possible Explanation III–The PCA operation

Theoretically, VLAD representation and Fisher Vector representation should be better than BOW representation, especially, Fisher Vector representation should be strictly better than VLAD since VLAD representation is a fraction of Fisher Vector representation (i.e. Fisher Vector representation contains all information that is contained in VLAD representation). However, look at Table III, we find that the accuracy of Fisher Vector representation is even worse than VLAD representation. The only possible explanation is that some thing is wrong with the PCA dimension reduction operation. So we did experiment with all same settings as in Table III but increase the target dimension of PCA dimensionality reduction. The results are shown in Table IV.

We can see improvements on accuracies of both VLAD and Fisher Vector representation, which very possibly indicates that the ultimate culprit for the failure of accuracy is the PCA operation! Now we can explain why in Table I the performance of VLAD representation and Fisher Vector representation are very poor. It is highly possible that a vector with dimension 200 is not capable to contain useful information in VLAD representation and Fisher Vector representation (whose dimension is more than 10000, and with large codebook size can even reach 200000). So the very possible solution to improve experiment result in Table I is to increase the dimensionality of PCA operation.

TABLE IV
RESULT OF INCREASING THE DIMENSION OF PCA OPERATION

VLAD		Fisher Vector	
PCA_10	6.00%	PCA_10	4.40%
PCA_200	14.80%	PCA_200	4.40%
		PCA_1000	18.39%

Yet due to the lack of computational resource (our google cloud platform account has run out of billings!), we can't demonstrate the improved result of Table I here.

VIII. A DEFECT IN IMAGE PROPOSALS

In this section we discuss a defect in our process of extracting image proposals. Remind that we use selective search to extract proposals and empirically constrain that the number of proposals of each image is no more than 20. And since selective search doesn't provide a confidence of each proposal of how possible it is foreground, we sample 20 proposals from the selective search output. And here comes the problem, the sample process is completely random and unreliable. See a demonstration in Figure 5. The left image shows all proposals that satisfy the constrains described in **Equ.**3 and **Equ.**4, and the right image shows the sampled proposals. We can see that many of the sampled proposals are background images that has no relationship with the chimpanzee, so by using this kind of proposals to learn a codebook and perform feature encoding and classification, the classifier might not learn to use the correct information to judge if the animal in the image is a chimpanzee. For example in Figure 5, if the classifier predict it as chimpanzee, it might not be because of the black hairy human-like creature in the image, but because of the grid fence in the background. This is will lead to unstable performance of the classifier because no one can guarantee that fence and chimpanzee will co-occur in every image. If the classifier is given a image with grid fence background and a tiger in the foreground, the classifier might still predict it as a chimpanzee.

A. Possible Solutions

One simple solution to this problem would be that we don't constrain the number of proposals of each image. Yet this solution is not clever enough to remove unrelated background proposals.

Another solution would be use other methods instead of selective search to extract image proposals, such as Region Proposal Network (RPN).

IX. CONCLUSION

From experiment data and the above analysis, we conclude that the classification accuracy of encoded features will increase along with the increase of codebook size. As for Table II where performance is decreased after increasing the codebook size, we think it is because when codebook size are increased from 10 to 100, the dimension of encoded feature (Fisher Vector representation) is also increased from 2560 to 25600, yet in Table II, the encoded feature is reduced to a vector of dimension equal to 100, which is uncapable to contain enough information of a vector with dimension equal to 25600, which leads to the accuracy decrease. And we also concluded that the performance relationship between BOW representation, VLAD representation and Fisher Vector representation is that Fisher Vector representation is better than VLAD representation, and VLAD representation is better than BOW representation. The inconsistency in Table I can be explained by too small target dimension of PCA reduction compare to the dimension of encoded features, which have dimensions varying from 10000 to 200000 depends on encoding method and codebook size.

X. MISCELLANEOUS

Experience learnt in this project:

- Problem can be anywhere, the most important part (e.g. feature encoding in this project) might not be responsible for the performance failure, yet the less important part (e.g. PCA in this project) might significantly deteriorate the performance.
- Computational resource is crucial for machine learning, yet clever minds are also very important.