Mobile Visual Search System Optimization

Chang Liu
Department of Computer Science
University of Massachusetts, Lowell
Lowell, MA, 01854, USA
chang_liu@student.uml.edu

Yu Cao
Department of Computer Science
University of Massachusetts, Lowell
Lowell, MA, 01854, USA
ycao@cs.uml.edu

***Abstract*— Mobile visual search system's user experience has been a long-term popular research topic in recent years. The strategy of improving the mobile visual search system includes several methodologies and there have been a lot of researchers focusing on this issue. Recently, studies have suggested that system latency, power consumption and network bandwidth are the most important factors which greatly affect the user experience. Previously a lot of researchers have been researching in the influence of one factor on the performance of the search system. Some research focus on the influence of network bitrate on the accuracy of searching, some focus on the influence of different features' influence on the power consumption or system latency area. However, few of them evaluate multiple factors' influence on multiple objects under acceptable accuracy. In this paper, we present a strategy of evaluating all of these factors at the same time to give the best dynamic scheduling algorithms and system implementations to improve user experience. Our system is the first one to explore multi-factors' (like different features, image size, bandwidth) combining influence on the system and have shown great potential in healthcare monitoring, daily activity log and mobile medical system.**

Keywords—mobile visual search system; system scheduling; dynamic system; healthcare monitoring; optimization

I. Introduction

Mobile visual system have been a very popular topic in recent years. By using smartphone, smartwatch, wearable devices and cloud server, it's easy to build a client-server system to complete a complicated task like mobile visual search, automatic image recognition and classification etc. According to the statistic observation, there have been many researchers and scientists working in this field to maximize the function and feasibility of mobile visual system.

To build a complete client-server system, there are many different strategies for a mobile visual search system. We could summarize the system into the following three categories:

1) all the computing and searching jobs are done on the client side. This kind of architecture requires the mobile devices a high computing ability, large available memory, sufficient battery and large storage to complete the whole process of computing as well as searching.

2) some jobs are done on the client side, some jobs are done on the server side. When the client finish its job, transmit the necessary data to the server to finish the processing. After the server finish the classification, transmit the data back to the client to show the results. Which part should be done in the client and which part should be done in the cloud is the most challenging job, as it depends dynamically on the specific hardware and software environment.

3) all the jobs are done on the cloud side. This strategy requires the client to send the whole original image to the server and then after the server completes the whole processing, return the result back to the client to show. Since most of the jobs are done in the cloud, the main cost of client is that it requires much time for the client to send the whole image.

Back to the architecture, even though there are so many strategies for this system, the most common structure is the second one, since it utilizes the mobile and server devices with considerations about the power consumption and system latency. But even for the second strategy, how to design and schedule is various since there are so many different running environment, when considering a strategy, we need to take those factors into consideration, especially the limitations of the mobile devices.

There have been many limitations for mobile devices, especially for smartphones and smartwatch. According to the observation , there are mainly four limitations for mobile devices: 1) limited computing ability; 2) limited storage; 3) limited network bandwidth; and 4) limited battery power/energy. To build a user-friendly mobile system, it is important to give the user accurate result in a short time using least resources like network bandwidth and power energy. Many researchers have put much efforts in exploring one factors' influence in the system[1][2][9][10][11][12][13]. In these paper, the authors proposal a terminology of bitrate to compare the performance of different features under same bandwidth. However, none of them have done research on how to take all the factors into considerations and build a fully dynamically scheduling system to best fit the current state of mobile device and environment.

In this paper, we propose a dynamic scheduling algorithms which could take all necessary conditions into consideration, including devices' hardware configurations(CPU information, memory usage), current network bandwidth, current power consumption state and other factors. The result of our experiment shows that it is more feasible to build such a system under mobile environment and it shows that our algorithm and system has a wide application in healthcare monitoring and daily activity logging, medical care improvement.

II. Related work

There have been many researches on mobile visual search system, especially in the area of exploring the influence of different bandwidth and image features when processing the retrieval jobs. For these researches, they share a common feature, which is focused on the second aspect, designing an evaluation of mobile visual system, or developing an approach for efficient mobile system.

In [1], the author presents a basic mobile system architecture and its technique details in each sub-system, as well as the evaluation of system latency, power consumptions in each specific environment under fixed configurations of other variables. In their system, there are mainly three parts, which include feature extraction, feature matching and geometric verification. In this paper, by using CHoG (Compressed Histogram of Gradients) descriptors, it could achieve better performance in that scenarios using less data transmission. After the feature extraction, the paper also present the methods which involves generating of vocabulary tree and inverted index by using k-means algorithm. At last it gives a fast geometric re-ranking algorithm to make sure the classification is under acceptable precision. Regarding the evaluation of system accuracy and power consumption, the paper presents the change of accuracy by bitrate, the change of transmission latency by query data size and the change of energy by different network bandwidth using mobile network or wireless network.

In [2], the author presents a new concept called "low bitrate", which is used to estimate the transmitted data under the same bandwidth. According to their research, using less data transmitting could reduce less computation in mobile device, thus creating less response time and less power consumption as they described. In this way, they figure out the criterion of "bitrate" and use it to evaluate the performance of a system.

In [9], the researchers propose a framework of MVS (Mobile Visual Search) based on the weighted matching of dominant descriptor. First they present an affinity propagation based algorithm for dominant descriptor selection. After that, they come up with a weighted feature matching method to consider the differences of dominant descriptors in feature matching. In this way, the result shows a network latency reduction and avoidance of transmitting useless descriptors to improve the retrieval accuracy of MVS.

Apart from these work, some researchers also proposed new features to achieve high accuracy and less response time. In [10], they propose a scalable mining method by exploiting Thread of Features (ToF). By using this new features, they could reduce the noise, discover patterns and speed up the processing. This new feature is more effective than other methods in mining small instance and could also reach the goal of reducing response time.

Another improvement of the system also includes the algorithm for quantization scheme [11]. In image classification, Bag-of-Words (Bow) has been widely use to represent an image. In order to achieve high accuracy as well as efficient retrieval, local features needs to be extracted and quantized to visual words. In this paper, the author investigate the reliability of each bit in scalar quantization, and then presents to incorporate the reliability in both index word expansion and feature similarity. This new approach could not only accelerate the search speed but also improve the accuracy, which also gives a good result by reducing the system latency.

Developing a new framework for the mobile system is also another focus for these research, by using different architecture, it is possible to reach the goal of reducing system latency. In [12], the author presents an e-health framework to conduct therapy session. They collect live therapeutic data of patients in a non-invasive way and this facilitate the therapist to model complex gestures by mapping them to a set of primitive actions and generate high-level therapies. Their framework includes two 3D motion tracking sensors, a Kinect and a Leap to collect data as well as a PC or server to do offline analysis and reporting.

Even though as the previous work focus on the compression of features, many of the proposed new methods involves the designing of new methods, some work focus on the architecture improvements. In [13], they present vocabulary decomposition by which they decompose the large data vocabulary into several ones satisfying storage constrains on mobile devices. In this way, it could reduce more than 95% of the transmission overhead without losing much data information. By this way they also achieve low system latency for the mobile search system.

There are also other work involving in evaluating one factors like bandwidth or power consumption in specific scenarios, most of them focus on the aspects I mentioned above, by developing new features or by improving the processing framework and algorithms, many of them just focus on the evaluation of accuracy and processing speed, with little consideration of power consumption, not even the combination of all those factors

As most of the previous work are focused on the design of the whole system or the influence of one factor on the power consumption or system latency. Due to the complexity of the multi-factor's influence, few of them take all the necessary factors into consideration, which lacks practical guidance for the system implementation in real world. Our work is the first one who give an overall view of the system and detailed scheduling algorithm for the system.

III. Overview of System

Our system has a traditional CS(client-server) architecture, which includes a client-end using smartphone, smartwatch or both of them, and a server-end machine, which is running the computational intensity jobs in the cloud. By using the network, client and server is connected and their running state as well as data are shared between each other to complete the whole retrieval task. In the next phrase, this section will describe the system architecture in details. Section A gives an overview of the system. Section B give technique details about the evaluation methods. Section C gives a comparable result of the experiment.

A. *System architecture*

In this section, an detailed system design and water flow graph is given to illustrate the flow of executing. By combining the client and server, it contains a large number of jobs when doing image classification. Here, we just give a specific scenarios, which is the image classification application. In theory, our model and algorithm could be applied in many similar application with different purpose.

Generally speaking, a simple mobile visual classification has some shared process to proceed when deploying them in the client and server. Here, Figure1 gives a simple illustration.[2]

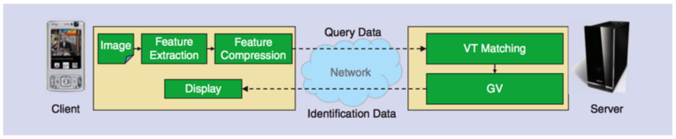

Figure1. Image retrieval system design

From the figure, we could know that for a typical system, it could be divided into the following parts:

1) *feature extraction*
2) *feature compression(transformation)*
3) *feature matching(classification)*

For any simple system like the above design, it is essential to contain these parts as they're the basic process for image processing and pattern recognition. Each sub-process of these system could have its unique technique, for example, in feature extraction sub-process, there could be a lot of feature selections to get a faster speed or a more precise accuracy; in feature compression sub-process, different schemes could be chosen to achieve different compression rate, thus creating different data transmission package; in feature matching/classification process, different methods could be applied to achieve a better classifier linear SVM or SVM with Gabor kernels. In the following parts, this subsection would give a description of these evaluations.

A.1 feature extractions

There have been a lot of global features and local features to describe an image. Different feature could give a totally different representation in the form of matrix or vector in mathematic theory. However, it is all well-know that generally local feature performs better than global feature in many scenarios, as in many cases global feature could not give a fully detailed information about the local characteristics. In our experiment, to make our data more reliable and more practical in most of our applicable case, we explore a lot global and local features to give a comparison. Basically, these could be divided into the following categories:

1) Global Features. Global features include all features that are non-local. While there exists a large number of global features ranging from color, texture to edge features, here we choose some of the most widely used general-purpose feature as follow:

a) Color histogram. We use 24-dimension color histogram in HSV color space where the histogram on each dimension is computed separately and then concated. For quantization, H is divided into 18 levels, S and V are divided into 3 levels.

b) Color Moment. We use 225-dimension grid color moment; each image is divided into 5x5 grids and the first to third moments in RGB color space are extracted from each grid respectively.

c) Gabor. We use 48 dimension log Gabor coefficients as features. 24 filters with 6 orientations and 4 scales are used to compute the response, and for each filter response, the first and second moment are extracted.

d) LBP. We use local binary pattern with uniform patterns extension, which results in a 59-dimension histogram.

e) PHOG. We use 3400 dimensional pyramid histogram of oriented gradient with 4 bin histogram and 3 level pyramid ($K = 4$, $L = 3$) .

2) Local Features. Local features have become the standard component of state-of-the-art visual recognition systems. Among the wide range of detectors as well as descriptors, we choose the following combinations in our evaluations:

a) DoG. Difference of Gaussian detector + SIFT descriptor.

b) HA. Hessian Affine detector + SIFT descriptor.

c) Dense. Extract SIFT features with dense sampling, using 20×20 patches and overlapping windows shifted by 10 pixels. We use the VLfeat library for Dense SIFT extraction.

d) SURF. SURF detector + SURF descriptor.

After choosing these different global and local features, we first implement them in our mobile client end, as most of current mobile devices have already enough to give result for a specific image. We use VLfeat and OpenCV, as well as Android NDK package to fully ingredient these C/C++ library to the Android platform. Most of these feature extractions have been implemented in these libraries, so most of our work focus on the transplantation from C/C++ platform to Android platform.

Our result shows that it's feasible and acceptable to compute and extract feature in the mobile client. For a typical SIFT feature, using Samsung Galaxy S3, it takes nearly 30s to get all the features for a

800x600 RGB images. Before doing the feature extraction method, it is necessary to do scaling and reduce the computation of the whole features.

A.2 feature compression (transformation)

Feature transformation is a group of methods that create new features (predictor variables). The methods are useful for dimension reduction when the transformed features have a descriptive power that is more easily ordered than the original features. In this case, less descriptive features can be dropped from consideration when building models.

Feature transformation methods are used where dimension reduction is achieved by computing an optimal subset of predictive features measured in the original data.

The methods presented in this section share some common methodology. Different methods have different goals, as follows:

1) Nonnegative matrix factorization is used when model terms must represent nonnegative quantities, such as physical quantities.

2) Principal component analysis (PCA) is used to summarize data in fewer dimensions, for example, to visualize it.

3) Linear Discriminant Analysis (LDA) is a linear transform that we mainly use to reduce dimensionality of our input features. LDA assumes that we have classes (labels) and that features of different classes have a Gaussian distribution with the same covariance matrix but different mean [6]. The main difference between LDA and Principle Component Analysis (PCA) is that PCA tries to keep as much structure of the features (variance). PCA decorrelates the feature space and orders the dimensions with decreasing variance. In case we want to reduce the dimensionality and choose the first N dimensions we keep the most structure of the data. LDA focus on dimensions that separates classes and orders dimensions according to class separation.

4) Factor analysis is used to build explanatory models of data correlations.

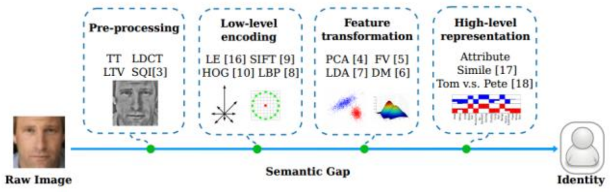

Figure2. Methods adopted by feature transformation [7]

Here normally we could choose the appropriate method for our different implementation. Our wide system could have two different method:

a) *traditional machine learning methods*

b) *deep learning methods*

For a), we could use PCA to reduce dimension and get a visualized representation of features, thus getting a less-size features. For b), it is optimal for our designed models. When using CNN (convolutional neutral network), in [7], by using wavelet transform in CNN, a higher feature transform is achieved.

About the general steps that PCA have performed, as the following shows, it is general acceptable in our previous study:

1) Take the whole dataset consisting of d-dimensional samples ignoring the class labels

2) Compute the d-dimensional mean vector (i.e., the means for every dimension of the whole dataset)

3) Compute the scatter matrix (alternatively, the covariance matrix) of the whole data set

4) Compute eigenvectors and corresponding eigenvalues

5) Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a d x k dimensional matrix W (where every column represents an eigenvector)

6) Use this d x k eigenvector matrix to transform the samples onto the new subspace. This can be summarized by the mathematical equation:

$$y = W^T \times x$$

At which x is a d x 1 -dimensional vector representing one sample, and y is the transformed k x 1-dimension sample in the new subspace.

In our system, we use PCA as it is a practical methods and it has some common implementation in VLfeat library. In our future work, since we will use R-CNN for deep learning, and then transplanted part of the algorithm in the mobile side, future evaluation will be conducted to get the best computing speed.

A.3 feature matching (classification)

In the last step of our system, a feature matching or classification is performed to categorize the query image into some category. A classification is a computing process, which calculates the distance between the query image and the dataset image, after the calculation, the result is compared to a reasonable threshold, where the targeted image is selected when the distance is smaller than the threshold.

In image processing area, there have been many methods to perform a feature matching and image classification. We could divide them into two categories: 1) supervised classification; and 2) unsupervised classification.

The difference between supervised and unsupervised classification is that supervised classification uses pre-applied knowledge to label the data, while unsupervised classification first classify the data and then apply these knowledge. The following Figure3 could give a detailed instruction about the difference.

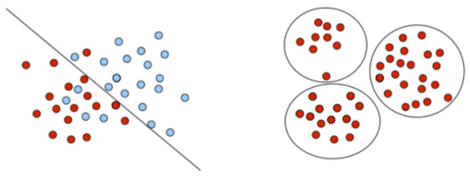

Figure3. Difference of classification

In Figure3, the left one is the supervised classification, where we could know the data type and its label before the classification, so it is easy to find the best fit to divide these two or more different types. As the linear line shows in the left, it is the division and classification of the data.

In the right one, as it shows, it is an unsupervised classification, which means that we don't know the type of the data and by using methods like k-means, we could divide them into several groups ,and in this way, we will find there similarities and its belong group. Then it is easy to find which data belongs to which group.

There are many classifier corresponding to the type of classifying methods, as a practical architecture of classifying procedure shows below [5]:

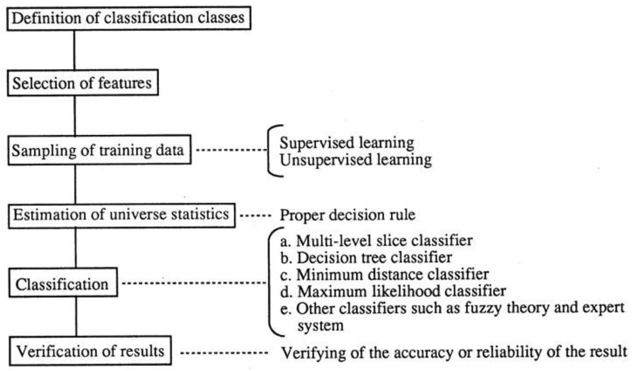
Definition of classification classes
Selection of features
Sampling of training data
Supervised learning
Unsupervised learning
Estimation of universe statistics
Proper decision rule
Classification
a. Multi-level slice classifier
b. Decision tree classifier
c. Minimum distance classifier
d. Maximum likelihood classifier
e. Other classifiers such as fuzzy theory and expert system
Verification of results
Verifying of the accuracy or reliability of the result

Figure4. Procedure of classification [5]

From the figure, we could know some common types of classifiers, as below shows:

1) Multi-level slice classifier

2) Decision tree classifier

3) Minimum distance classifier

4) Maximum likelihood classifier

5) Other classifiers such as fuzzy theory and expert system

Each classifier has its different implementation in our system. Here we use linear SVM (support vector machine) to train the data to get the classifier, which is widely used an accepted as its precision and accuracy. In this way, we know that SVM classifier is a linear classifier, which should have 1) Wide margin; 2) Cost function; 3) Slack variables; 4) Loss functions revisited; and 4) Optimization.

As there are many applications using linear SVM under different platforms like Java/C++/Python, we choose do pre-train our data using Python in the server-side, because it takes so much time to train even on highly-equipped server, even the smartphone couldn't handle these computing using their limited CPU and memory as well as the battery.

B. *Evaluation function*

In our previous description, most of related work is only focused on the evaluation of the one factor. For example, in [2], the author evaluates the accuracy under fixed bandwidth using different features and descriptors. In this way, they come up with the best solutions for the bandwidth environment to get a reasonable precision result.

However, in our real world, it is not enough since the complicated system have many other factors, which could affect the performance of the system. For example, when the smartphone runs out of battery, it is essential to optimize the algorithm to reduce the computing and make it last longer, in this way the server have to give a quick response to the client so that it is feasible to get a precision result under this situation. In other ways, when the battery is nearly full, it is necessary to do the computing or pre-processing more in the phone, so that we could reduce the work and data transmission in the network and server-side, since sometimes the mobile network is not so good compared with wireless network, it is also necessary to take all these into consideration.

In our design phase, we want to simplify the architecture of the system, so our system only includes smartphone and server, but in the future, we should extend these to include smart devices like smart watch, google glass, Fitbit or other wearable devices.

To make it clear, in this phase, we first give a short list of parameters that we think is the most important in our experiment, which is:

1) *Image size*
2) *bandwidth*
3) *feature selection*
4) *hardware configuration(CPU, memory)*

Other factors like storage and available resource in the smartphone will be taken into consideration in the future.

After deciding these factors, we should give the target of our evaluations, which is system latency and power consumption. In our survey, most of the user are most concerned about these two factors, because the first is most in relation with the response time, whether it is acceptable for a mobile system is that its ability to give the user the acceptable result in least time. For a bad system design, the biggest drawback is that they give the result in a long time. Another evaluation criterion is the power consumption, since it is the most common complain of users that the power runs so quickly and a time-consuming and power-consuming application is not acceptable for the users.

As the above shows, we want to evaluate these two targets using the parameters listed above. However, for a system, the precision is very important. Even if we could give a best schedule for system using least time and power, if the result data is too bad, it is also useless than not scheduling so much. So we should do our experiment under the conditions that the precision of the result is under reason range, here we first give a not so precise range, every precision rate above 80% is acceptable for our system. In this way, we could pay more attention in our scheduling, not worrying too much about this conditions.

The following give the evaluation object function of system latency, which indicates the response time or running time:

$$f_m = \sum_{n=1}^{N} (cost_n \times T_n) \qquad (1)$$

The equation gives a practical evaluation of the cost/object function, where $cost_n$ is the weight, which we give for each sub-process, those we want to evaluate more should get a higher weight, and T_n is the running time of each sub-process, as we illustrated above. The sub-process includes feature extraction (client-side), transmission time (network), classification time (server-side), the n is size of each portion, m is the size of configuration. For example, we have 9 different schemes of algorithms to schedule the jobs of the client-server system, each schemes could be divided into 3 sub-process, then here m is 9 and N is 3. By calculating each $f_m(x)$ in different schemes, we could get the distribution of the system latency. And each scheme's configuration is related with the parameters listed above, so the distribution is in a spatial dimension. After getting all the necessary result and data, we could then visualize this distribution and find the minimal value of system latency, with the specific parameters.

Another object/cost function is the power consumption, which we mainly evaluate the client-side's, because the server-side is not apparently different considering its large clustering and deployment.

$$g_m = \sum_{n=1}^{N} (cost_n \times P_n) \qquad (2)$$

Similarly, here $g_m(x)$ is the power consumption for different schemes. m is the size of schemes, n is the size of ports of each power-consumption, $cost_n$ means the weight of each part of the power consumption sub-parts, P_n is the evaluated power consumption amount. Since we only evaluate the power of client-side, so it is necessary to clarify the parts of power energy that we take into consideration. Here we mainly evaluated the total running power, which many includes: 1) application running/computing power; 2) network transmission and maintaining power; 3) system consumption

Here we also use an open source software called PowerTutor, which could give an estimation of the running energy consumption of each parts, even each applications. In this way, we could get the distribution of power consumption as well.

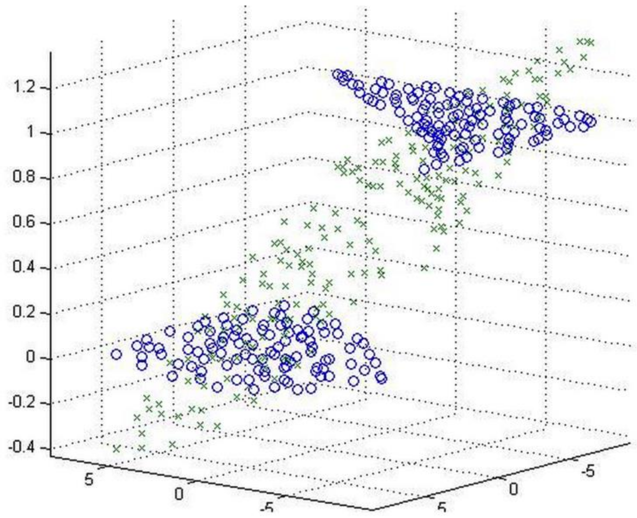

Figure5. Linear regression of data

After using linear regression or other methods to get the trend of such plot, we could try to figure out what it is the best fit for the lowest power consumption and system latency value. As we all know that for this graph, it is reasonable that system latency and power consumption graph is not so similar, so when the system latency reaches the lowest value, maybe the power consumption not. How to minimize both of them is another questions that bothers us. So here we give another evaluation function to give the total evaluation:

$$\varphi = w_1 \times f_m + w_2 \times g_m \quad (3)$$

As the equation (3) shows, the total evaluation object is subject to two factors and its corresponding weight w_1 and w_2, where:

$$w_1 + w_2 = 1 \quad (4)$$

In this way, we could give one of the factors with higher priority and the other one with little. By this way, it is easy to estimate the overall performance and balance the evaluation between system latency and power consumption.

In addition, all those plots are drawn while the precision is under the acceptable range, which is pre-defined as 80%, in the case that the precision is under 80%, we will not take those data into consideration, since they convey little helpful result in our system, and it's of less importance to evaluate these data and its relationship.

C. *Server side deployment*

After the description of the whole architecture of the system, it is necessary to describe our server side deployment and its work mechanism, since it contains a lots of variances comparing with the simple client side environment.

Here we have two strategies to deploy our system, the first is as the above describe, using traditional server side implementation by using linear SVM to train data and get its classifier, after receiving the data from client side, then doing necessary transformation and use the feature vector and this classifier to detect the image category.

Another deployment is that we use highly convolutional neutral network and its configuration in the server side to get our classifier. This classifier is trained by regional convolutional neutral network, and the feature is not the hard-coded representation of image, but from the learning of the features.

Here I will give two short descriptions about the two different architecture of these system.

C.1 traditional classification system

In this traditional classification system, we could easily know that it is handy and easy to classify an image by using linear SVM classifier. First, we could learn from the following graph[2].

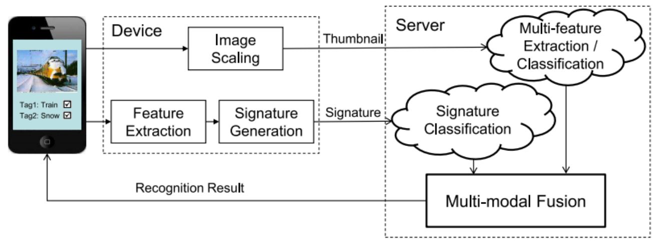

Figure6. image classification system

In this system, as it may not show, the pre-training and its classification is performed by the server side before the processing, so all the work is done offline when we want to get the classifier before the query. In this experiment, the work is done by linear SVM training using PFID dataset, so all these labeled data is used to generate the model and classifier. When using linear SVM, we choose the open source software libSVM as it is the most widely used tool to get the classifier, it also has Python/MATLAB/C++ library for different platforms.

C.2 deep learning system

Deep learning is a very hot topic in recent years in machine learning and pattern recognition area, as it gives more and more improvement for the precision and complexity of the problem. For our mobile system, even though there are few application system running on the mobile devices, we plan to use our outstanding system design to incorporate the mobile device and server implementation to give a better result with acceptable response time and power consumption.

Here I will give a detailed description about the R-CNN(region convolutional neutral network) and its implementation in our system.

R-CNN, also known as regional convolutional neutral network, is a new convolutional network which take full use of region features. In [8], the author gives a very detailed evaluation and verification of the network architecture with its performance comparison with traditional neutral network, this shows the wide potential of its application.

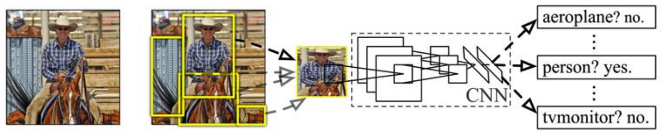

Figure7. R-CNN system overview

From the system graph[8], we could know there are few steps to do the object detection and classification:

1) *extraction region proposals(~2k/images).* In this step, by using various methods like selective search or objectiveness, category independent object proposal etc, it is easy to extract different regions in the query images.

2) *For each region, using CNN(convoluational neutral network), calcuate the CNN features to do further classification.* By the way, before we do the feature extraction process, the region should be scaled to a fixed size 227x227, in order to the same vector dimension in our further handling.

3) *After the above handling, it should generate a 4096-dimensional fc7 feature vector.* Using this vector, and applying the linear classifier like SVM or softmax, it easy to judge the likelyhood of which category this region belongs to.

From this simple illustration we could know that the theory is quite simple comparing with other algorithms, however, the convolutional neutral network is a complex system, which takes time to implement and learn. Luckily, there have been some work in caffe and its extension R-CNN library using caffe, we could easily use these library to train very large dataset and get the corresponding classifier.

In the following part of this section, I will give another description about the training of R-CNN classifier and the makeup of R-CNN network.

As we know from above, the R-CNN classifier is the last step that we should use to categorize the image. How to get the applicable and precise classifier is essential for our system[8].

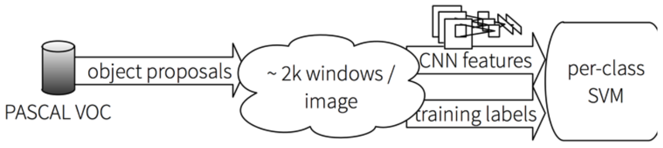

Figure7. pre-training R-CNN

From the Figure7[8], we could know that the pre-training of R-CNN could include 3 steps:

1) supervised training on a large dataset using CNN
2) fine-tuning the R-CNN feature fore detection using a smaller dataset
3) train detection SVMs.

After the above 3 steps, it is easy to get the classifier and accurate result of query.

To understand how CNN is working, we should know the below network architecture for R-CNN design. Below is the typical structure of CNN, it gives us the basic structure of neutral network. There are five layers and additional two fully-connected layers, which consists the total structure of the network. The 1st, 2nd, 5th layer is used for max pooling, and the last two layers are used for improvement.

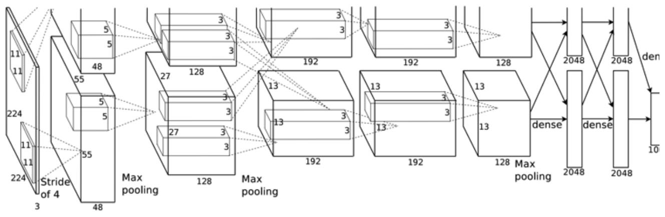

Figure8. R-CNN structure[8]

D. *Dataset*

Here for our experiment, we use PFID food dataset to do the training, we first use linear SVM to get the right classifier and test it locally then. Then we want to extend it to R-CNN network and test its performance and accuracy in that deployment, our experiment is still under conduction and need to be explored further to achieve a complete and precise conclusion.

IV. FURTURE WORK

Even though we have designed a lot of system architecture and done many experiments in the mobile device and server side, it is still not finished as the complexity of the system. We plan to do more testing using PFID dataset using R-CNN under cloud server-side. We also plan to transplant the classifier to be running on the mobile devices to minimize the transmission time. And the dynamic algorithm still needs to be tested by a lot of different scenarios and configurations as we may mention above.

ACKNOWLEDGMENT

The work is sponsored by Prof. Yu Cao's instruction and his funding. I'd like to show our sincere thanks to him.

REFERENCES

[1] Girod, B., Chandrasekhar, V., Chen, D. M., Cheung, N. M., Grzeszczuk, R., Reznik, Y., ... & Vedantham, R. (2011). Mobile visual search. *Signal Processing Magazine, IEEE*, *28*(4), 61-76.

[2] Su, Y. C., Chiu, T. H., Chen, Y. Y., Yeh, C. Y., & Hsu, W. H. (2013, October). Enabling low bitrate mobile visual recognition: a performance versus bandwidth evaluation. In *Proceedings of the 21st ACM international conference on Multimedia* (pp. 73-82). ACM.

[3] Sirovich, L., Kirby, M.: Low-dimensional procedure for the characterization of human faces. JOSA A 4(3) (1987) 519–524

[4] Simonyan, K., Parkhi, O.M., Vedaldi, A., Zisserman, A.: Fisher vector faces in the wild. In Proc. BMVC. Volume 1. (2013) 7

[5] Barkan, O., Weill, J., Wolf, L., Aronowitz, H.: Fast high dimensional vector multiplication face recognition. In: Proc. IEEE Intl Conf. Computer vision. (2013)

[6] http://en.wikipedia.org/wiki/Linear_discriminant_analysis.

[7] Levinskis, A. (2013). Convolutional Neural Network Feature Reduction using Wavelet Transform. *Electronics and Electrical Engineering*, *19*(3), 61-64.

[8] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv preprint arXiv:1311.2524*.

[9] Lan, G., Qi, H., Li, K., Lin, K., Qu, W., & Li, Z. (2014, November). A Framework of Mobile Visual Search Based on the Weighted Matching of Dominant Descriptor. In *Proceedings of the ACM International Conference on Multimedia*(pp. 1181-1184). ACM.

[10] Zhang, W., Li, H., Ngo, C. W., & Chang, S. F. (2014, November). Scalable Visual Instance Mining with Threads of Features. In *Proceedings of the ACM International Conference on Multimedia* (pp. 297-306). ACM.

[11] Guan, Jun.,Shi, &Ning (2014). Scalable Image Search with Reliable Binary Code. In *Proceedings of the ACM International Conference on Multimedia Pages 769-772*, ACM New York, NY, USA.

[12] Qamar, A., Afyouni, I., Hossain, D., Ur Rehman, F., Toonsi, A., Abdur Rahman, M., & Basalamah, S. (2014, November). A Multimedia E-Health Framework Towards An Interactive And Non-Invasive Therapy

Monitoring Environment. In *Proceedings of the ACM International Conference on Multimedia* (pp. 745-746). ACM.

[13] Qi, H., Stojmenovic, M., Li, K., Li, Z., & Qu, W. (2014). A Low Transmission Overhead Framework of Mobile Visual Search based on Vocabulary Decomposition.