

Unsupervised Feature Learning For Image Recognition

Shuang Ao

Supervisor: Professor Charles Ling

November 7, 2014

Contents

1	Introduction to Image Filter	2
1.1	Kernel Based Method for Image Recognition	4
2	Deep Learning Feature Representation	8
3	K-means Feature Representation	12
3.1	learning the patches with K-Means	12
3.2	Patch Combination	13
3.3	Patch Ranking	17
3.3.1	Unsupervised Patch Ranking	17
3.3.2	Supervised Patch Ranking	19
3.4	Encoding with the centroids	22
4	Conclusion	25

Abstract

In image recognition, the success of machine learning algorithm is mainly depends on the image representation. Different representations can lead to different views of the image data. Although, the specific domain knowledge can help us for representation, sophisticate method for image representation is still crucial. In this TSP, firstly the background knowledge is introduced, including some basic feature extraction methods. Deep learning, which is introduced in Section 2, is the most popular method for feature extraction technique which has gained great reputations in both academic and industrial area. Finally, I present a novel method that is inspired by the idea of deep learning. In this method, I try to use patch combination as well as some ranking methods for image representation. The defects of this method can be my future work.

1 Introduction to Image Filter

Image recognition is an important part in Computer Vision which includes many advanced machine learning techniques. A pattern recognition system (PRS) is an automatic system that aims at classifying the input pattern into a specific class[27]. An image recognition system always proceeds in two steps: (1) generate several important features (descriptors or a book of code) (2) use a classifier to distinguish the patterns with all these features. Compared to our human learning procedures, some prior knowledge should acquired before the recognition which turns the learning procedure into be a supervised learning.

There are four major methodologies in image recognition:

- Statistical approach. Here features are converted into a vector to present the pattern which can be a very easy to handle. This method is wildly and intensively used in practice.
- Syntactic approach. This method analysis the relationship between features and patterns are described in a hierarchical structure. For example (see Figure 1), the image on the right hand can be presented as the string *dbabcbabdbabcbab*.The system parses the set of extracted features using a kind of predefined grammar. If the whole features extracted from a pattern can be parsed to the grammar then the system has recognised the pattern. Unfortunately, grammar-based syntactic pattern recognition is generally very difficult to handle.

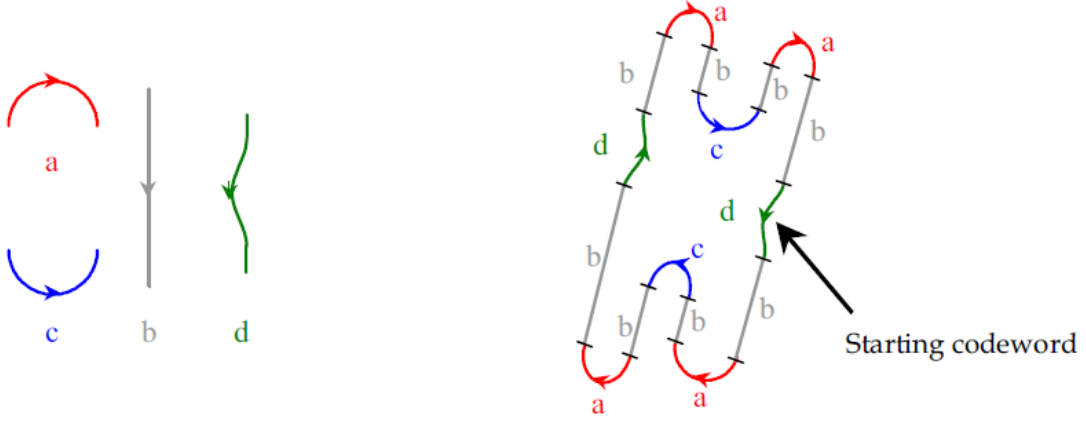


Figure 1: Example of syntactic description features

- Template matching. This can be the most intuitive way to localize and identify shapes in an image. Here, the method looks for the parts of the image which match the template. For each possible position in the image, it compares with each possible rotation or each possible geometric transformation of the template. According to certain metrics that can measure the similarity of these, the approach returns the most similar one. Since it has to compute all kinds of possibilities, this could be a very computational expensive approach.
- Neural networks. The an artificial neural network (ANN) had been used extensively before 2006 for image recognition. In 2006, G. Hinton introduced "deep learning", a new method that can learning image features with fast unsupervised procedure which led to a revolutionary in image recognition[26]. We will discuss this method in details in section 2.

There are some simple dependencies when representing the image patterns. One reason why explicitly dealing with image patterns is interesting is because they can be convenient to express many general priors about the world around us, i.e., priors that are not task specific but would be likely to be useful for a learning machine to solve AI tasks[4]. These general priors can sometime be used to help us design more powerful learners to discover the underlying factors that the data may reveal. Here are some examples of the general-purpose priors:

- Smoothness: Assumes that the function f to be learning for evaluation must satisfy that for any two patterns x, y , if $x \approx y$, $f(x) \approx f(y)$.
- A hierarchical organization of explanatory factors: Some patterns are more abstract than the others. These abstract patterns can be defined in terms of other patterns hierarchically.

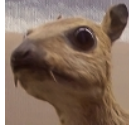
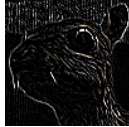
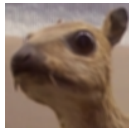
original	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
edge detect	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
blur	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

Table 1: Apply filters on image

- Natural clustering: Different values of categorical variables such as object classes are associated with separate manifolds. Humans have named categories and classes because of such statistical structure (discovered by their brain and propagated by their culture), and machine learning tasks often involve predicting such categorical variables.
- Sparsity: For any given observation x , only a small fraction of the possible factors are relevant[42]. This means just a small fraction of the patterns can represent a image well.
- Simplicity of factor dependencies: The good abstract patterns can be related to each other through simple, typically linear combination. This can be seen in many neural networks when the top layer is just linear combination of the hidden units.

1.1 Kernel Based Method for Image Recognition

No matter what classification techniques it is used, the images have to be transformed into certain kind of feature vectors with one of the methodologies mentioned above. Image filter (also called *kernel*) allows you to apply various effects on photos. In a 2D image, the 2D filter matrix f can be applied with a convolutionary operation on the image I :

$$(f * I)(x, y) = \iint f(x, y) I(x - t, y - t) dt \quad (1)$$

For a image classification task, the key issue is to find the most useful features to present the image. The results of the classification results can be greatly affected by the type of the kernels

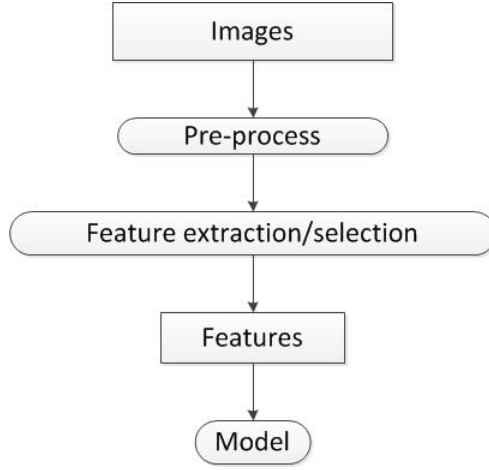


Figure 2: Image Classification Procedure

used. So there are two essential parts here: how to get the kernels and how to present the image with these kernels and the second part of the problem can be considered as the encoding the features. There are 3 different kernels defined in Table 2. After applying these kernels on the original image, the results are shown in the 3rd column. The size of the kernel can affect the image: the smaller the kernel is, it is more sensitive to some features in the image. However, the larger one is also useful while there are some noisy in the image as it is less possible to detect those tiny noisy pixels. Furthermore there are many different kinds of kernels (with different properties and sizes) that can be applied on the image to detect different features. For a specific problem, it is very difficult and inefficient to define any kernels even though we may acquire sufficient prior knowledge for this problem. And as the development of the power of the computer and the complexity of the problem, it is not possible for one to get enough prior knowledge.

For a more heuristic way, many researchers try to learn the specific kernels for a particular task with either supervised or unsupervised learning strategies [8][10][26][34][33]. In kernel methods, the problem is modeled as a pairwise relation between data points which is captured in kernels. Thus, kernel functions (or simply kernels) have a profound impact on the performance of these learning algorithms [1]. However, tuning the kernels to fit the underlying hypotheses of the dataset can be a great challenge. As the images are taken in different situations, the light condition, the angle of the view and the shadow of the objects in the image would affect the performance of the kernels. On the other hand, mathematically kernel learning can be a constrained optimization task. Good kernels can map the images into high dimensional space which makes them more distinguishable. In an image recognition task, the image always includes at least thousands of pixels and there





Original	Vertical Kernel 1	Vertical Kernel 2	Vertical Kernel 3
$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & -1 \\ 5 & 0 & -5 \\ 1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 0 & -2 & -1 \\ 4 & 8 & 0 & -8 & -4 \\ 6 & 12 & 0 & -12 & -6 \\ 4 & 8 & 0 & -8 & -4 \\ 1 & 2 & 0 & -2 & -1 \end{bmatrix}$
			

Table 2: Different Kernels for Edge Detection

are also thousands of images for training. Optimization on these large high dimensional dataset, taking gradient descent for instance, can easily stuck into local minimum no matter how simple the objective function is. Also kernel learning can be considered as a problem similar to distance metric learning as the pixels in the image can be "activated" by a kernel if they are similar. But distance metric learning is trivial in image recognition which would be affected by many simple geometrical transformation such as rotation, shift or zoom in/out. In human brain, we do this unconsciously and parallelling. Many conventional distance metrics have been used in computer vision even though none these metrics can handle geometrical transformation. Thus before measuring the distance of the original image and the kernels, some transformations have to be apply on the original image such as rotation with different degrees (like from -30 degree to 30 degree with 1 degree interval), shift around (within a 9×9 window) and scaling in order to fit the kernel. Within these operations, maybe just one set of the combinations can have a high response which we couldn't know in advance. Yann LeCun *et al* introduced *ConvNet* which applies convolutionary operation on the whole image to learning the kernels[36]. Still ConvNet is one of the most popular methods with the best performance in real world image recognition[13][48][29].

City Block Distance/Manhattan Distance

$$d(P_1, P_2) = \sum_{i=1}^n |p_1^i - p_2^i| \quad (2)$$

Euclidean Distance

$$d(P_1, P_2) = \sqrt{\sum_{i=1}^n (p_1^i - p_2^i)^2} \quad (3)$$

Cosine Distance

$$d(P_1, P_2) = 1 - \frac{P_1 P_2^T}{\|P_1\| \times \|P_2\|} \quad (4)$$

As each image normally contains thousands of pixels, cutting the image into small patches can be a more realistic way when learning the kernels.

2 Deep Learning Feature Representation

In 2006, a new feature learning and deep learning method was initiated by G.Hinton and has

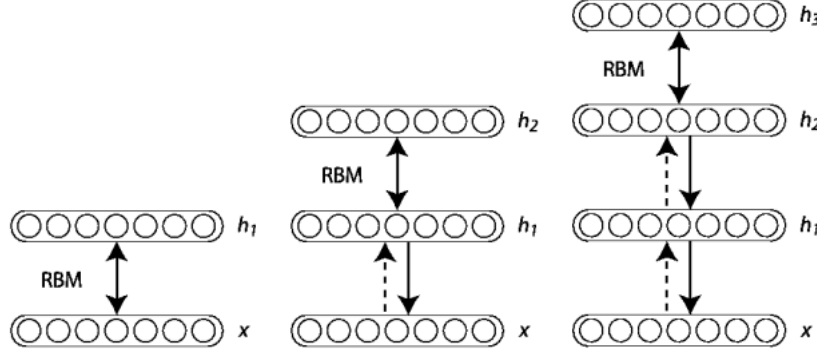


Figure 3: Deep learning procedure

brought great impact in both academic and industrial field. This deep learning method is a kind of deep neural network which contains several layers. The core idea, referred to as *greedy layerwise unsupervised pretraining* was to learn the bottom to top hierarchical features in a deep neural network. In deep learning, each layer is trained separately which makes it really fast and the output of the lower layer is the input of the higher one. When using greedy layerwise unsupervised pretraining, the two conjunctive layers can be regarded as a Restricted Boltzmann Machine (RBM) and the *Energy* of the RBM is defined as [25]:

$$E(v, h) = - \sum_{i \in \text{visible}} a_i v_i - \sum_{j \in \text{hidden}} b_j h_j - \sum_{i, j} v_i h_j w_{ij} \quad (5)$$

Here, v_i , h_j are the visible value (input) and the hidden value (output) respectively, and a_i , b_j are their bias. w_{ij} is the weight of the visible node i and hidden node j . The probability that the RBM can recall the visible vector, v , is defined as:

$$p(v) = \frac{1}{Z} \sum_h e^{-E(v, h)} \quad (6)$$

$$Z = \sum_{i, h} e^{-E(v, h)}$$

The derivative of the log probability of a training vector with respect to a weight is:

$$\frac{\delta \log p(v)}{\delta w_{ij}} = \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}} \quad (7)$$

$$\langle v_i h_j \rangle_{\text{model}} \approx \langle v_i h_j \rangle_{\text{recon}}$$

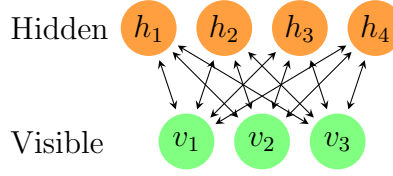


Figure 4: Restricted Boltzmann topology with 3 visible units and 4 hidden units.

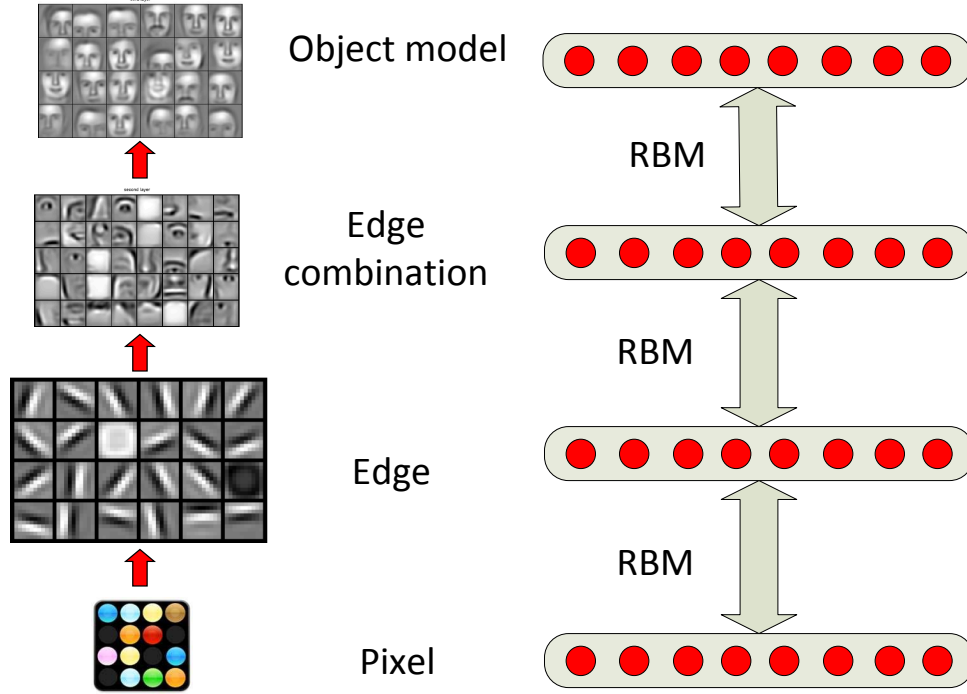


Figure 5: Hierarchical Representation of Deep Learning

Even though, the $\langle v_i h_j \rangle_{model}$ can't be calculated, Hinton proposed a fast learning procedure, called *Contrastive Divergence*, that can estimate it by ignoring the tricky term in the objective function and proved that this can lead to the convergence. Sutskever and Tieleman have shown that this procedure is not following the gradient of any function[53].

After this greedy layerwise unsupervised pretraining, the configuration of the deep network can be the initial of the supervised deep neural network predictor[52]. And the stochastic gradient descent can be applied for further optimization. The initial configuration after greedy layerwise unsupervised pretraining can make this deep network get rid of the local minimum since the supervised procedure is not based on any gradient method.

It is interesting to ask why this greedy layerwise unsupervised pretraining can help in supervised learning[19]. The curriculum greedy layerwise unsupervised pretraining tries to build some intermediate representations rather than some distinguishable representations. This is nicely related to curriculum learning idea which indicates that it is much more easier to learn some simpler concept first and build a much more complicated ones on top of the simpler ones[5]. For a deep supervised classifier, in the unsupervised training procedure, the deep neural network is the stacked RBMs which can initialize in a *good* local minimum where the *good* refers to low generalization error[3].

Stacked RBMs has been developed for many large learning problems as the feature extractor for other supervised classifiers. In these cases, stacked RBM is used as both feature extractor to learn the hierarchical representations and autoencoder to evaluate the similarity between the features and the images. However, single-layer RBM can still be a good predictor for a supervised learning problems. H.Larochelle used the RBM stand-alone to build the discriminative RBM on character recognition without replying on any other classifier[30]. Even though, for the large learning task, deep neural network with millions or even billions of nodes can get some good results[28][33], in some simple shadow network can still do well. It's a big waste to use a cannon to kill a mosquito.

When training the deep neural network, there are many details that can speed the convergency of the stacked RBMs to achieve a better local minimum. Cho et al. proposed a adaptive learning rate for the RBM training as well as a novel gradient estimator that takes into account the invariance of the model to flipping hidden unit bits and inverting signs of corresponding weight vectors[11]. For the choice of the hidden units, several experiments have shown that nonlinearity could influence both training and generalization performance[21][23][41]. Moreover, the initial weights can affect the training results. A good initialization can substantially reduce the difficulty of training the deep neural network. With a sparse initialization or using often saturated nonlinearities, it is more promising to make the hidden units specialized[39]. For the training method, there is still a debate between online learning method such as stochastic gradient descent and using large mini batch(thousands of samples)[32], even though, the stochastic gradient descent and its invariants are more popular and have won many competitions. As we can see in Equation (2), most of the computation in updating the weights can be done with matrix operations, it is more convenient to use GPU to do these jobs which is designed to accelerate the matrix operations[47]. Normally, training the huge deep network on GPU can be at least 10 times faster than training on CPU.

Despite great achievement in both academic and industrial area, there are still some criticisms, one of which is that there are so many hyperparameters and variants that exploring the configuration and the architecture is really an art[35]. To solve this problem, many automating hyperparameters searching works have been proposed recently which makes it more convenient and efficient[7][6][51]. To deal with large quantities of image recognition problem, deep learning can be the first choice. As a neural network, it can easily get rid of the local minimum, which is the biggest issue in all the large neural network training procedure, with just a little prior knowledge through the efficient unsupervised training. However, recent work on large quantities of labeled data shows that with a proper initialization, the pure deep supervised network can still be successfully trained without any layerwised pre-training[12][22][50]. And in this situation, the layerwised unsupervised pretraining can bring little or even no improvement over the supervised one when training for long enough.

In addition, deep learning will be the most promising method for image recognition in the next few years. Even though pure supervised learning can compete with deep learning while there are enough labeled data for training, still deep learning and unsupervised pre-training for image recognition would be the future. As in real world, we can always obtain enough data while just a very small fraction can be labeled. Unsupervised feature extraction techniques can be a more realistic method.

3 K-means Feature Representation

Deep neural network has shown its great power on large scale image recognition tasks, but sometimes it is more important to get good representation of the image rather than learning the complex hierarchical representations[14]. The main drawback of learning the high-level representation for many systems is their complexity and expense. Moreover, the multi hyper-parameters of these algorithms which would affect the performance significantly require many prior knowledge as well as computational expensive validation to determine their values. Since the unsupervised pre-training can learn some important representations, as one of the most basic and fast unsupervised learning strategy, K-Means can also learn the representation well through a self-taught procedure. For a K-Means algorithm, the only parameter K decides the number of the centroids that the data can generate. As a mature algorithms, there are many techniques that can accelerate its convergence[43][44][20]. K-Means has been defined as a successful method to learn features from images by computer vision researchers. The term of "bag of features" is very popular in computer vision communities[15][31].

In our system, the feature representation is learned as the following steps:

1. learning the bag of features from sub-patches with K-Means to get a low level features.
2. combining the low level features to generate a higher-level features.
3. mapping the image to the feature space and do the classification.

The concept of an ROIs (region of interests) is commonly used in many application areas especially in medical image (magnetic resonance imaging, MRI) process[17]. It is based on a simple and intuitive hypothesis: there could always be some objects or features that appear in some certain location of the image which can be used to identify the image. This hypothesis can be applied to many other real world applications such as hand digital recognitions.

3.1 learning the patches with K-Means

The classic K-means clustering algorithm finds cluster centroids that minimize the distance between data points and the nearest centroid. Also called "vector quantization", K-means can be viewed as a way of constructing a dictionary $D \in R^{n \times k}$ of k vectors so that a data vector $x(i) \in R_n, i = 1 \dots m$



Figure 6: The first 100 images in MNST dataset

can be mapped to a code vector that minimizes the error in reconstruction.

$$\min_S \sum_i^k \sum_{x_j \in S_i} \|x_j - \mu_i\|^2 \quad (8)$$

μ_i is the mean of S_i .

In our task, the images are 28×28 pixels grayscale images and we use 14×14 pixels patch presented as a 196 pixel intensities for the K-means to learn the features. We select 12 location to generate the patches (shown in Figure 3.1). For each location we use K-means to get 100 centroids with *K-means++* which can avoid the sometimes poor clusterings found by the standard k-means algorithm.[2].

3.2 Patch Combination

The initial intuition of patch combination is the hierarchical feature representation. The higher level features can be presented as the combination of the lower ones. The idea of sparse coding is also to use the lower level features to represent the higher ones with linear combination[24]. In sparse coding, the lower features are linearly overlapped on each other to generate the complex higher level ones and the dimensions of the new feature will never change. In our task, we don't overlap the lower ones. Alternatively, as we already know the specific location of each patch in the original image, we can combine the lower features in different locations of the image and generate

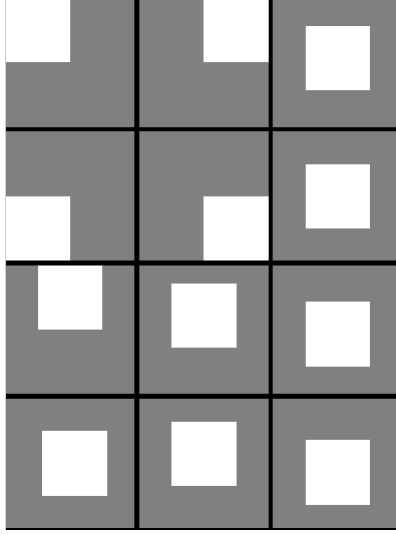


Figure 7: The 12 locations of the patch

the higher ones. When combining two low level features, the most simple method could be just consider the number of the common instances that appear in both. If they have a lot of instances in common, these two low level features can be combined as a new one. Consider the image dataset D has n instances ($D \in R^{n \times m}$) and for each candidate patch location l which are clustered into k clusters, D can be represented as a sparse binary matrix $D^l \in R^{n \times k}$

$$D_{i,j}^l = \begin{cases} D_{i,j}^l = 1, & \text{if } D_i \in \text{cluster } j \\ D_{i,j}^l = 0, & \text{if } D_i \notin \text{cluster } j \end{cases} \quad (9)$$

When we have two patch location candidates l_1 and l_2 , $C^{com} = D^{l_1} \times D^{l_2^T}$ and $C_{i,j}^{com}$ denotes the number of the common instances that for i_{th} cluster in l_1 and j_{th} cluster in l_2 . However, C^{com} just indicates how many instances that both of these two clusters hit but ignores the how many instances actually there are in each of the clusters. For instance, cluster A and cluster B has 100 and 200 instances respectively and $C_{A,B}^{com} = 100$, if cluster C and cluster D has 1000 and 800 instances respectively and $C_{C,D}^{com} = 150$, it is hard to say that the combination of C and D is better than A and B.

Alternatively, to evaluate the quality of the combination of the k_1 cluster in l_1 and the k_2 cluster in l_2 , we can get the k_1 and k_2 column of the sparse matrix D^{l_1} and D^{l_2} and get the confusion matrix as Table 3. In the method we mentioned above, the # of instance in common just take the *True Positive* into account while ignoring the impact of *False Positive* and *False Negative*. In pattern recognition, the concept of *precision* and *recall* which are defined as Equation

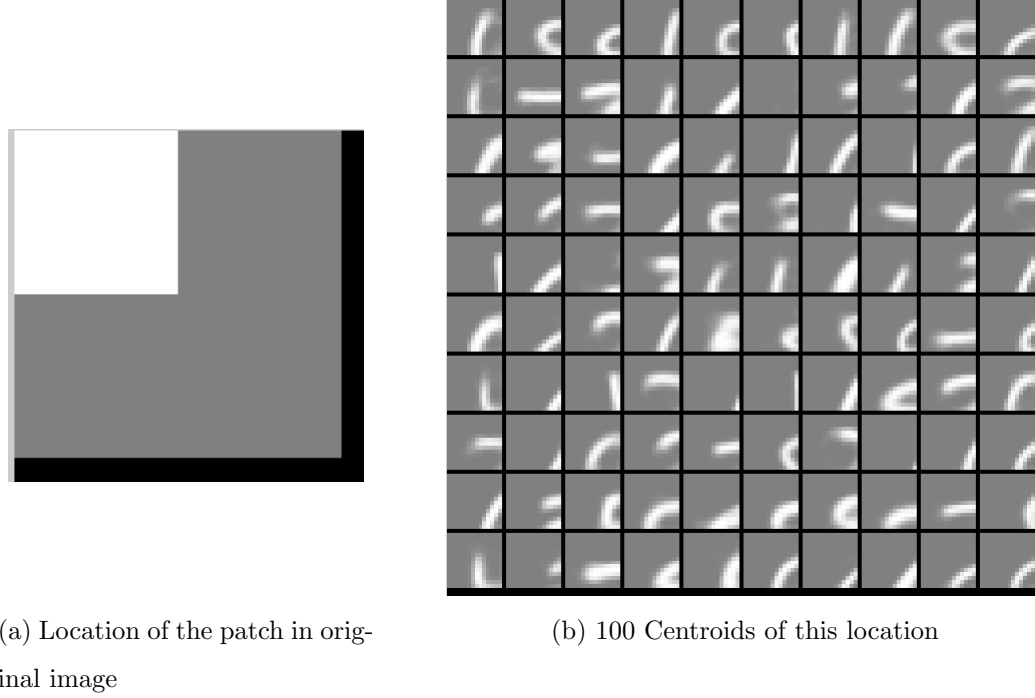


Figure 8: The Centroids for location 1.

(10), can solve this problem[46]. From Table 3, the precision and recall are switched when we switch cluster i and j . But we want some score which can satisfy $score(i, j) = score(j, i)$. Here precision and recall refer to the fraction of the "hit" in both clusters. And F -measure can take both recall and precision into account which is defined as Equation(11). If we set $\beta = 1$, F_1 is symmetric for precision and recall. So we can get $F_1(i, j) = F_1(j, i)$. For each pair of location candidates l_1 and l_2 , each of which consists of k centroids, we can get a $k \times k$ symmetric matrix $ComScore = \{ComScore_{i,j} | ComScore_{i,j} = F_1(i, j), i, j = 1, 2, \dots, k\}$ where $ComScore_{i,j}$ indicates the F-measure for the combination of i_{th} cluster in l_1 and the j_{th} cluster in l_2 . If $ComScore_{i,j}$ is great than some threshold θ , then the new location $l_1 + l_2$ will include the centroid $k_1 + k_2$ and the membership of the new centroid would be the instances in both of these two clusters.

$$\begin{aligned} precision &= \frac{TP}{TP + FP} \\ recall &= \frac{TP}{TP + FN} \end{aligned} \quad (10)$$

$$F_\beta = (1 + \beta^2) \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \quad (11)$$

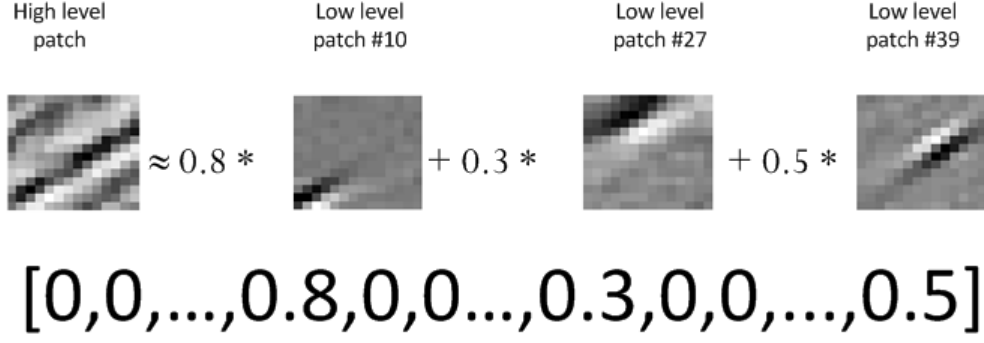


Figure 9: The idea of sparse coding



Figure 10: The combination of patches in different locations of the image

	j	$\neg j$
i	# of instances both has i_{th} and j_{th} feature (True Positive)	# of instances only has i_{th} feature (False Positive)
$\neg i$	# of instances only has j_{th} feature (False Negative)	# of instances has neither i_{th} nor j_{th} feature (True Negative)

Table 3: Confusion Matrix for cluster i and cluster j

From Figure 11 we can see that, the combination of the patch location are very flexible, which can create some ambiguous shapes of patches that can describe some important features in the image. Compared with the conventional convolutionary method, which requires large computing and is very sensitive to the size of the kernels, our method can generate kernels with any size and any shape from some small lower patches.

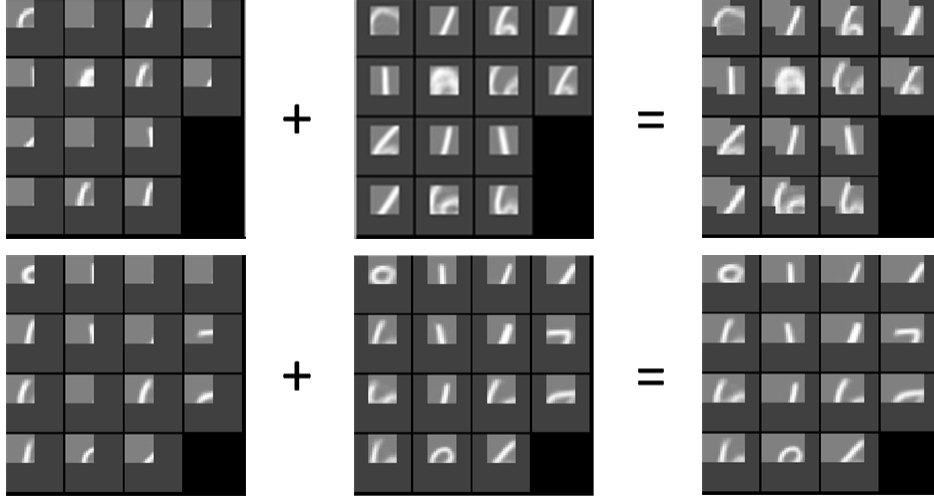


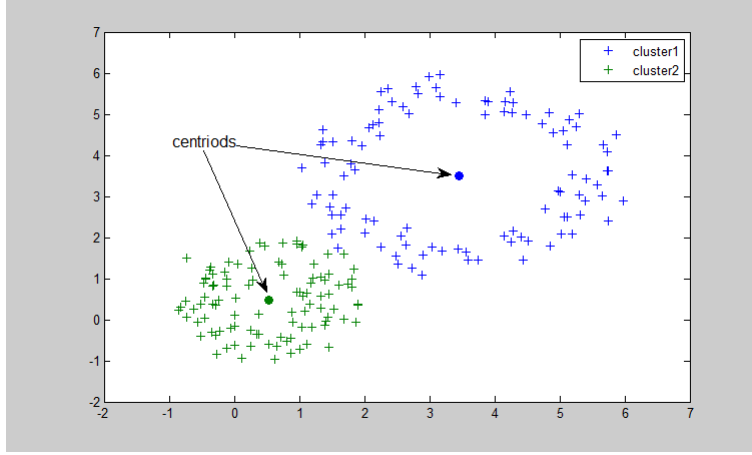
Figure 11: Successfully combined patches

3.3 Patch Ranking

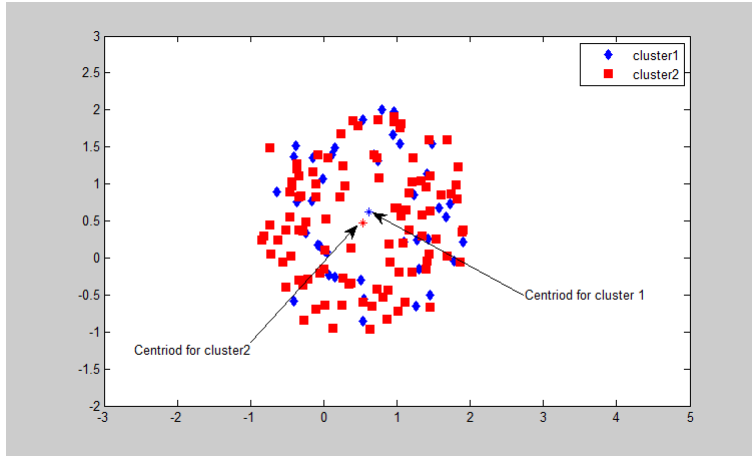
From the patch combination, for a particular threshold θ , we can get many features with ambiguous size and shape. These centroids, generated from either k-means (low level features) or combination of the lower feature(high level features), are used as the template for the image representation. But too many features can lead the algorithm to be very slow as well as some duplicate features. Therefore, some less informative centroids (templates) should be removed. These centroids should be redundant, duplicate or irrelevant. Here both the supervised and unsupervised patch ranking techniques can be used. Unsupervised patch ranking is used to eliminate the centroids whose clusters are either too sparse or too small. Supervised patch ranking is used to eliminate those centroids that are duplicate which can no longer provide extra information after the class label is introduced.

3.3.1 Unsupervised Patch Ranking

In all these centroids we get from patch combination, some of the lower ones would be less informative after they are successfully combined to generate the higher ones. For instance, if k_1 centroid in l_1 and k_2 centroid in l_2 have identical membership in the original image set and they can definitely generate a higher centroid k_3 in location l_3 which will contain identical membership as k_1 and k_2 . In this situation, k_1 and k_2 would be the duplicate patches as the higher level centroid k_3 can provide the information as much as them. However, we can not simply assume that all the lower features are less informative than the higher ones. Normally lower clusters can have more



(a) Clusters with the same centroids but different μ



(b) Clusters with the almost the same centroids but different size

Figure 12: Two situation to evaluate the cluster/centroid

instances belong to compared with higher ones. In Figure 12b, cluster 1 has 40 instances while cluster 2 has 100 instances. The distance of them are very near and they all have the almost the same average distances to the centroid. Here, the cluster 2 is more general and its centroid can better describe the feature of its member. A more complicated situation could be the comparison of the centroids with different shape. Here we introduce the following equation for evaluation of the centroid.

$$Dist(x, C) = \underbrace{\frac{1}{n} \sum_{i=1, x_i \in C}^n \frac{x_i \times C}{|x_i| |C|}}_{\text{mean cosine similarity}} + \alpha \cdot \underbrace{\log(n) * \log(C_{size})}_{\substack{\text{size of the cluster and} \\ \text{size of the centroid}}} \quad (12)$$

Here, the x is all the cluster members and C is their centroid. n denotes the size of the cluster and the C_{size} is the size of the number of pixels of the centroid C which is positive proportional

to the level of the feature. The first part of Equation (12) is the mean cosine distance of the cluster members to the centroid which describe the similarity among the cluster members. Cosine Similarity is to calculate the orientation of two vectors instead of the magnitude which is less sensitive to the size of the pixels evaluating the centroids with different number of pixels. The second part is for normalization which involves the size of the cluster and the size of the centroid. Our intuition is that if two clusters have the same similarity among their members, the higher level feature/centroid and the cluster with more instances has a higher rank.

Even though, there are many mature techniques for cluster analysis (like DaviesBouldin index[16], Dunn index[18], etc), which is the essence method to obtain the templates, we still want to use the most basic criterion, mean distance to the centroid, as our fundamental criterion. The unsupervised ranking is used here as the prior method of the supervised one which can be more sophisticated and useful than the unsupervised one. And it is better to use to unsupervised ranking before the supervised one as there could be too centroids, if we can't eliminate some of them with the fast unsupervised one, it could be very slow for the supervised ranking method to deal with a lot of centroids.

3.3.2 Supervised Patch Ranking

After eliminating the sparse centroids, the supervised patch ranking can eliminate more duplicate and irrelevant centroids to improve the results. This procedure can be considered as the feature selection which is very important in the pattern recognition system. Our ultimate goal is to improve the classification result after the patch ranking. However, without supervised ranking, there could be some embarrassing situations, like in Figure 13. For the classification of digital number 1 and 3, some important features for both of the classes should be found before the final classification. But after the unsupervised ranking, most of the features with higher rank are for the digital 1. There is just one feature (in red box) for digital 3. Even though this matters a little for the binary situation, this could be a serious problem for multi-class problem as some of the classes may not get enough features to be distinguished. So the goal of the supervised ranking is to found those centroids that can represent each of the classes and eliminate the redundant and irrelevant ones.

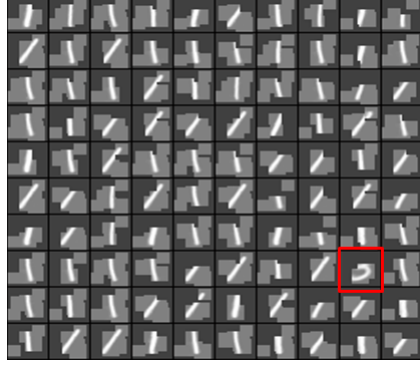


Figure 13: Top 100 unsupervised ranking centroids for classification of digital number 1 and 3

Significant Test The typical supervised feature selection can be formulated like: given the dataset D that contains N samples with M features $F = f_i, i = 1, \dots, M$ as well as the target class c , feature selection is to find the optimal subspace m of the $M - dimension$ that can best characterized c . As there could be 2^M candidates, it is really hard to search the space exhaustively. Here the *optimal* leads to the *minimal classification error* which is to find the maximal statistical dependency of the target class c on the data distribution in the subspace R^m . From this, the most simple method could be the significant test for each feature with specific target class c .

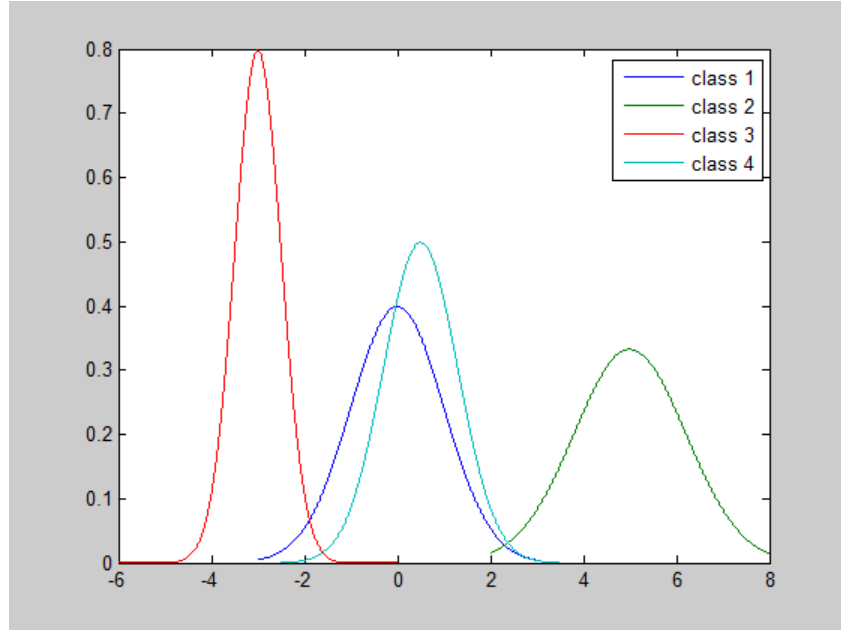


Figure 14: Significant test for feature selection for the 4 classes situation for specific feature f_i

From Figure 14 it is easy to find that the class 1 and class 4 are almost overlapped and with some significant test (like $t - test$), class 1 and class 4 can not be distinguished with the feature

f_i . Still, there should be some criterion for the supervised ranking with significant test:

1. Features that are significantly distinguishable for all classes should be rank the highest. These features are global features that can be used to "vote accept" for the every class.
2. Features that are significantly distinguishable for more classes should be rank higher. These features are local feature for specific one or more classes and they can "vote reject" for at least one class.
3. Features that are not significantly distinguishable for all classes are the redundant and irrelevant ones which should be eliminated and rank the lowest.

Mutual Information The significant test can check the dependency of target class on the data distribution for some features. It is easy to check for a single feature, as we want to find the best m features and the final performance is determined by the joint result of the features, the significant test can't solve this well. In information theory, the mutual information of two random variables is a measure of the mutual dependence of the two random variables. Given two random variables X and Y , their mutual information is defined in terms of their probabilistic density functions $p(x)$, $p(y)$, and $p(x, y)$:

$$I(X, Y) = \int_Y \int_X p(x, y) \log \frac{p(x, y)}{p(x)p(y)} dx dy \quad (13)$$

Intuitively, mutual information measures the information that X and Y share: it measures how much knowing one of these variables reduces uncertainty about the other. For example, if X and Y are independent, then knowing X does not give any information about Y and vice versa, so their mutual information is zero. At the other extreme, if X and Y are identical then all information conveyed by X is shared with Y : knowing X determines the value of Y and vice versa. As a result, in the case of identity the mutual information is the same as the uncertainty contained in Y (or X) alone, namely the entropy of Y (or X : clearly if X and Y are identical they have equal entropy) [38]. Here Y can be the target class c and X can be any dimension (feature) in the $M - dimension$ space. For each individual feature, $I(X, c)$ reflects the dependency of the class on the feature X . the top m $I(X, c)$ in descending order are the features to be selected.

It is clear that in neither significant test nor mutual information criterion, the top m selected features are not necessarily to be the "m best features" as the relationship between the features is not considered. To obtain the "m best features", it is to search the best m combination of the

features which can be applied in two opposite directions with the filter: forward and the backward selections [45].

1. The forward selection tries to select a subset of m features from M in an incremental way. At first we set the candidate set $S_{cand} = M$ and $S_{sel} = \emptyset$ as well as the criterion for the selection Cr . For each iteration, we pick up a feature x from S_{cand} and evaluate $Cr(x + S_{sel})$. The best x_{best} will be add into S_{sel} as well as remove from S_{cand} . This would stop in m iterations.
2. The backward selection manners from the other direction. It first set $S_{sel} = M$ and subtract the irrelevant features from it until stop. The backward selection is often used to select a large subset which is obviously faster than the forward selection.

It is worthy to mention that normally the result of the forward selection is not the same as the one with backward selection for the same problem and the same settings. As both of them are following a gradient descent manner, for most of the real world problems which have many local minimum, it is rare for both of them to meet at the same terminal for the same problem.

3.4 Encoding with the centroids

After obtaining all these centroids, the more important thing is to map the original image into the new feature space according to the centroids which essentially turns into the similarity metric problem. The intuition of the encoding is that for images I_1 , I_2 and the template t , we have to find a similarity function, called distance function, $D(x, y)$ must satisfy that if I_1 is more similar to t than I_2 , $D(I_1, t) > D(I_2, t)$. However, similarity is much complicated when described in the mathematical function than it looks. When dealing with similarity in our human brain for the images, we can do the shifting, scaling, rotation and other geometric transformation in parallel and get the result almost on real time. For computer vision researchers, each of these transformation is applying a kernel matrix on the original image and combine each possible results together. In practical, even though there could be infinite possible combinations, we can set some fix interval for each of the transformation, so each transformation can be divided into several intervals which is still a cost computation process.

However, it is still widely believed that the simplest method is the most practical one. Complexity can lead to uncertainty. Therefore, original Euclidean Distance is still the most popular method as the similarity metric. Even though, it is a naive method which only depends on the

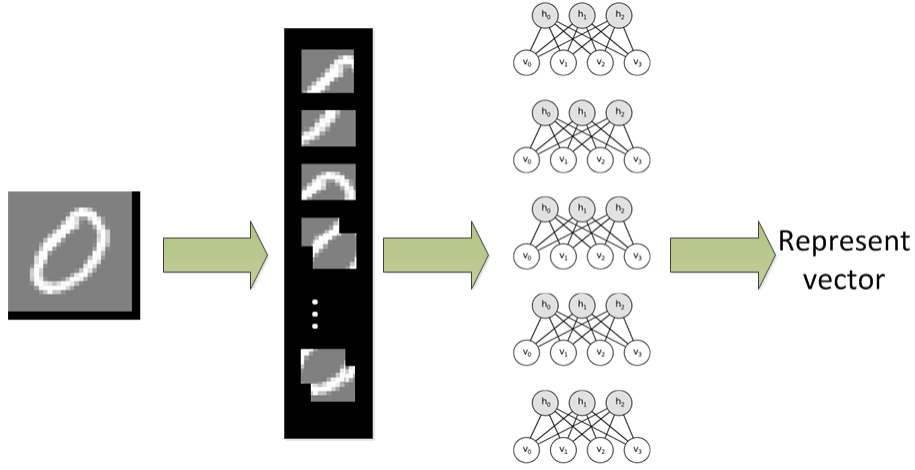


Figure 15: Use free energy of RBMs as the encoder

intensity of the pixels and is very sensitive to the deformation. There are many variance of it such as[9][40]:

$$d(I_1, I_2) = (I_1 - I_2)^T G (I_1 - I_2) \quad (14)$$

This can be considered as the weighted Euclidean Distance for images I_1 and I_2 . The weight matrix G is computed by the pixel dependency of the pixels in both images. As the weight matrix can be defined by users, it can be more flexible. For instance, if G is computed by the coefficient of the pixels in the image, the weighted Euclidean Distance can fit small deformation. After adding the weight matrix, the weighted Euclidean Distance can be less sensitive.

Neural network is considered as the efficient encoder for many researchers[49][52][25]. RBM, as the one layer neural network can be the efficient encoder. The free energy of the RBM can be used to determine whether the model is overfitting as well as for discrimination directly. The free energy is defined as:

$$F(v) = - \sum_i v_i a_i - \sum_j \log(1 + e^{x_i}) \quad (15)$$

The parameters v_i and a_i here are the same as the ones defined in Equation (5) and $x_j = b_j + \sum_i v_i w_{ij}$.

If the model is not overfitting at all, the free energy should be about the same on the training and validation set[25]. According to this, if the model is not overfitting, the free energy can be used to determine whether the test sample is similar to the data in training set. For each of the cluster, we can build a RBM and use the its free energy to encode each individual sample.

Overall, no matter the weighted Euclidean Distance or the RBM encoder with free energy,

they are trying to find the proper metric to evaluate the similarity for images. SIFT can detect the same objects in two images and get rid of the affects of rotation, scaling and lighting condition [37]. However SIFT can only detect identical objects in the two images. When the two objects are not that similar, like the handwritten digital numbers written by different people, SIFT still can do nothing. Still there are a lot of researchers that are focusing on how to encoding the image effectively. Encoding the image with the centroids is still a challenge work and can be done in future works.

4 Conclusion

This TSP has mainly cover three parts of work:

- In Section 1, some concepts and the kernel based image filtering technique are introduced. The kernel based image process method is the most important technique in image recognition which is widely used in feature extraction. Kernels are often used in convolutional neural network which can be implemented very fast with the help of GPU. Therefore, the large convolutional neural network is the first choice for large real world image learning task.
- In Section 2, the deep learning method is introduced. Deep learning is the state-of-the-art and the most popular model for image recognition. It shows great power on both learning the features and image representation. Deep learning uses the stacked RBM to learning the hierarchical feature representation and use the Contrastive Divergence strategy to training the model in an unsupervised way. This feature learning idea has given inspiration to many other unsupervised feature extract methods as well as the method proposed in Section 3.
- In the Section 3, the kmeans based feature extract method is proposed. Inspired by the unsupervised feature extraction idea in Deep learning, I try to use kmeans to find the templates as the features for image representation. The patch combination can be the novel part of this method which combines the centroids in different location to generate the "super patch". After patch combination, the unsupervised and supervised patch ranking methods are used to eliminate the sparse, redundant and irrelevant centriods. Finally the encoding methods is introduced.

In the kmeans based feature representation method, there are less hyperparemeters so that the background knowledge becomes less important. Still there are many challenges for this method:

- In patch ranking, even though, the bad centroids and irrelevant ones are ranked low, it is still unknow for us to know what is the optimal number of features get for learning the specific problem without the help of learning curve. Thus there are a lot of work to determine the optimal number of features for this method. In my future work, some score may be used to evaluate the centroids.
- There is still no good solution for the encoding problem as its complexity. There are too

many variables that need to be considered and still many work for me to find the best metric for the similarity measurement.

In the next few terms, all these problems above should be the main work for me and hopefully, with the supervision of Dr. Charles Ling, I can find some solutions for them and improve this method.

References

- [1] M. E. Abbasnejad, D. Ramachandram, and M. Rajeswari. A survey of the state of the art in learning the kernels. *Knowl. Inf. Syst.*, 31(2):193–221, 2012.
- [2] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.
- [3] Y. Bengio, A. C. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013.
- [4] Y. Bengio and Y. LeCun. Scaling learning algorithms towards ai. *Large-Scale Kernel Machines*, 34, 2007.
- [5] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *ICML*, page 6, 2009.
- [6] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *NIPS*, pages 2546–2554, 2011.
- [7] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.
- [8] P. Berkes. Handwritten digit recognition with nonlinear fisher discriminant analysis. In *Proceedings of the 15th international conference on Artificial neural networks: formal models and their applications - Volume Part II, ICANN’05*, pages 285–287, Berlin, Heidelberg, 2005. Springer-Verlag.
- [9] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 60–65. IEEE, 2005.
- [10] O. Chapelle, P. Haffner, and V. Vapnik. Support vector machines for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5):1055–1064, 1999.

- [11] K. Cho, T. Raiko, and A. T. Ihler. Enhanced gradient and adaptive learning rate for training restricted boltzmann machines. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 105–112, 2011.
- [12] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural computation*, 22(12):3207–3220, 2010.
- [13] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *IJCAI*, pages 1237–1242, 2011.
- [14] A. Coates, A. Y. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. *Journal of Machine Learning Research - Proceedings Track*, 15:215–223, 2011.
- [15] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, page 22, 2004.
- [16] D. L. Davies and D. W. Bouldin. A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (2):224–227, 1979.
- [17] R. S. Desikan, F. Ségonne, B. Fischl, B. T. Quinn, B. C. Dickerson, D. Blacker, R. L. Buckner, A. M. Dale, R. P. Maguire, B. T. Hyman, et al. An automated labeling system for subdividing the human cerebral cortex on mri scans into gyral based regions of interest. *Neuroimage*, 31(3):968–980, 2006.
- [18] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. 1973.
- [19] D. Erhan, A. C. Courville, Y. Bengio, and P. Vincent. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research - Proceedings Track*, 9:201–208, 2010.
- [20] F. Gibou and R. Fedkiw. A fast hybrid k-means level set algorithm for segmentation. In *4th Annual Hawaii International Conference on Statistics and Mathematics*, pages 281–291, 2002.

- [21] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research - Proceedings Track*, 9:249–256, 2010.
- [22] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. JMLR W&CP Volume*, volume 15, pages 315–323, 2011.
- [23] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. *Journal of Machine Learning Research - Proceedings Track*, 15:315–323, 2011.
- [24] R. Grosse, R. Raina, H. Kwong, and A. Y. Ng. Shift-invariance sparse coding for audio classification. In *UAI*, pages 149–158, 2007.
- [25] G. E. Hinton. A practical guide to training restricted boltzmann machines. In *Neural Networks: Tricks of the Trade (2nd ed.)*, pages 599–619. 2012.
- [26] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.
- [27] K. Kpalma, J. Ronsin, et al. An overview of advances of pattern recognition systems in computer vision. *Vision Systems*, 2007.
- [28] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114, 2012.
- [29] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012.
- [30] H. Larochelle and Y. Bengio. Classification using discriminative restricted boltzmann machines. In *ICML*, pages 536–543, 2008.
- [31] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. IEEE, 2006.
- [32] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng. On optimization methods for deep learning. In *ICML*, pages 265–272, 2011.

- [33] Q. V. Le, M. Ranzato, R. Monga, M. Devin, G. Corrado, K. Chen, J. Dean, and A. Y. Ng. Building high-level features using large scale unsupervised learning. In *ICML*, 2012.
- [34] Y. LeCun. Learning invariant feature hierarchies. In *ECCV Workshops (1)*, pages 496–505, 2012.
- [35] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade (2nd ed.)*, pages 9–48. 2012.
- [36] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, pages 319–, 1999.
- [37] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.
- [38] D. J. MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.
- [39] J. Martens. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 735–742, 2010.
- [40] C. R. Maurer Jr, R. Qi, and V. Raghavan. A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(2):265–270, 2003.
- [41] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.
- [42] B. A. Olshausen et al. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.
- [43] D. Pelleg and A. Moore. Accelerating exact k-means algorithms with geometric reasoning. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 277–281. ACM, 1999.
- [44] D. Pelleg, A. W. Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, pages 727–734, 2000.

- [45] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1226–1238, 2005.
- [46] D. Powers. Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.
- [47] R. Raina, A. Madhavan, and A. Y. Ng. Large-scale deep unsupervised learning using graphics processors. In *ICML*, volume 9, pages 873–880, 2009.
- [48] M. Ranzato, C. S. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In *NIPS*, pages 1137–1144, 2006.
- [49] M. Rehn and F. T. Sommer. A network that uses few active neurones to code visual input predicts the diverse shapes of cortical receptive fields. *Journal of computational neuroscience*, 22(2):135–146, 2007.
- [50] F. Seide, G. Li, and D. Yu. Conversational speech transcription using context-dependent deep neural networks. In *INTERSPEECH*, pages 437–440, 2011.
- [51] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. *arXiv preprint arXiv:1206.2944*, 2012.
- [52] N. Srivastava, R. Salakhutdinov, and G. E. Hinton. Modeling documents with deep boltzmann machines. *CoRR*, abs/1309.6865, 2013.
- [53] I. Sutskever and T. Tieleman. On the convergence properties of contrastive divergence. *Journal of Machine Learning Research - Proceedings Track*, 9:789–795, 2010.