Photo Auto-Tagging with Dictionary Learning

Bei Li BEI.LI@OU.EDU

Yohann Burette YOHANN.BURETTE@OU.EDU

Keywords: classification, sparse coding, KNN, Neural Network, Naive Bayes

1. Introduction

With the drop in prices, more and more users have now access to digital cameras. Capturing and processing digital pictures is more accessible compared to their film counterparts. This results in an ever increasing number of pictures people save on their computer's hard drive. However, as it is easy to store new pictures, it is more difficult to find a particular one.

For this, the photography industry introduced the notion of "tags" in order to help users sort their pictures. Basically, when storing a picture, the user "tags" it with words that match its content. For instance, a picture of a flower will receive the tag "flower". Similarly, a picture of a sailboat passing in front of a lighthouse will be tagged with "sailboat" and "lighthouse". The idea behind this is that, later, it will be easier to filter these tagged pictures out of an entire collection.

On the paper, photo tagging is a very simple yet powerful way of organizing pictures. Unfortunately, as the number of pictures increases, the tagging process quickly becomes tedious. And most of the users tend to skip this step eventually – worsening the problem each time.

Our idea is thus to create an automatic tagging system. By combining Computer Vision technologies and Machine Learning algorithms, the system is able to observe a picture's features and assign tags to it. The basic structure of our system is presented in Figure 1.

Although recognizing pictures is an easy task for a human being, it is much harder for a computer. In its simplest representation, an image is a matrix of numbers. This matrix can be written as a stack of vectors in order to process it with a classification algorithm. The difficulty comes from the fact that a slight modification in the original picture might change the matrix significantly. Similarly, two pictures of a same object but at different scales will have very different matrix representations. Thus, finding a better representation of the image is the key.

In computer vision, there are several popular ways of representing images such as patch-based approaches and the "bag-of-words" representation based on SIFT. In this paper, we explored a new method called "Dictionary Learning".

This study led us to design two approaches to picture auto-tagging. First, we tried to classify pictures using the Reconstruction Residual Error of Dictionary Learning. As explained in the following, it didn't work as well as we expected. Our second approach makes better use of Sparse Representation and is explained later.

It is worth noting that both approaches were not found in the literature. Most of the time spent on this project was actually spent experimenting with these ideas.

2. Related Work

In the same manner as for text, researchers have been trying to extract useful information from pictures. Multiple techniques have been explored over the years. Bag-of-words, Multiple-Instance Learning and Dictionary Learning are three promising ones.

2.1. Bag-of-Words

In (Sivic & Zisserman, 2003), an approach to object and scene retrieval similar to text retrieval is proposed. Firstly, a number of features and viewpoint-invariant features are extracted from a set of images. By k-mean clustering, k most representative features are extracted to form a vocabulary. With this vocabulary, every image can be represented in a "bag-of-words" form. Then the image retrieval task can be done in the same way as text retrieval. The feature extraction method mentioned above is mainly SIFT (Lowe, 2004), which firstly uses difference of gaussian (dog) to detect points of interest, and then use a vector (usually with length 128) to describe it. These feature vectors demonstrate

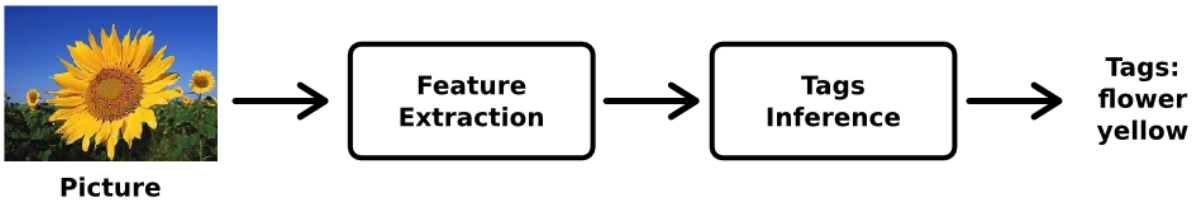

Figure 1. Structure of the system.

a certain level of scale and viewpoint-invariant ability. Then the bag-of-visterms way is adopted in broader tasks of computer vision. With the algorithm proposed in (Bosch et al., 2006), one can discover objects in images containing multiple object categories. In the information retrieval field, pLSA is employed to handle a similar problem, that is to classify a text document into different topics (Brants et al., 2002).

Fergus et al. (Fergus et al., 2005), present an approach that can learn an object category from its name by utilizing the raw output of image search engines available on the Internet. TSI-pLSA extends pLSA (as applied to visual words) to include spatial information in a translation and scale-invariant manner. This is presented in (Fergus et al., 2010).

2.2. Multi-Instance Learning

Multiple-Instance learning is another variation of supervised learning. The idea behind this approach is to have the agent learn a concept by giving it "bags of instances". If a bag contains at least one instance that represents the concept, then it is said positive. On the contrary, if none of the instances represent the concept, then the bag is said negative. In (Maron & Lozano-Prez, 1998), the authors describe a tool called Diverse Density that improves the way an agent can learn from multiple-instance examples. They give three examples of application: drug discovery, person recognition, and stock analysis. For our project, the interesting part of this paper is the bag generation. Each image forms a bag and the instances are subdivisions of that image. In the person recognition example, if the image contained the person, then the bag is labeled positive. Otherwise, it is said negative.

In (Maron & Ratan, 1998), this concept is extended to natural scene recognition. The user easily defines each bag as positive if the image represents a concept (e.g. a waterfall, a mountain). And, in the same manner, sub-regions of the image are taken as instances of the bag. This example also shows that the Diverse Density algorithm works well even when the feature extraction/representation is simple. They show that it works with a simple 15-dimensional feature space where each instance is a point. The paper also shows that there is no real difference in performance between a simple feature extraction step and a more complex one. The only difference is the amount of computation power needed. As an example of how a concept is 'learned', their agent determined that an image containing a waterfall will have a blob whose blue value is 0.5, whose neighbor below has the same blue value, whose neighbor above has the same red value, and so on. It is also interesting to note that the results are presented as precision-recall and recall graphs. The precision is defined as the "ratio of the number of correct images to the number of images seen so far." And the recall is the "ratio of the number of correct images to the total number of correct images in the test set."

A more recent approach to multiple-instance learning is given in (Qiao & Beling, 2009) where the agent learns about a concept on its own. This comes from the observation that it is difficult to find good example sets for algorithm training. The idea is to have fewer labeled examples and to use the knowledge from unlabeled bags to boost the training of the classifier. In their example, they use features such as color, texture and shape of "patches" within the image. Even though this approach requires less computation, it is interesting to see that the results with this approach are only slightly better than that obtained with other existing approaches.

2.3. Dictionary Learning and Sparse Representation

Intuitively, Dictionary Learning consists in extracting common features from a set of images and forming a dictionary. Then, each image can be represented as a linearly combination of the dictionary's elements. For example, in the text processing domain, one can scan through a set of articles and put every word that occurs more than three times into a dictionary. Any new article can then be represented in terms of the terms

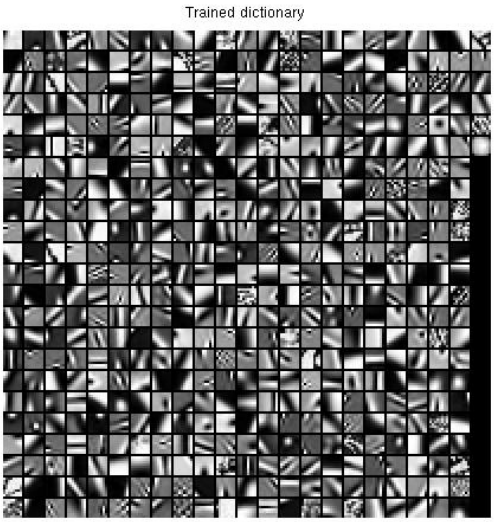

Figure 2. Example of dictionary learned from a set of random images

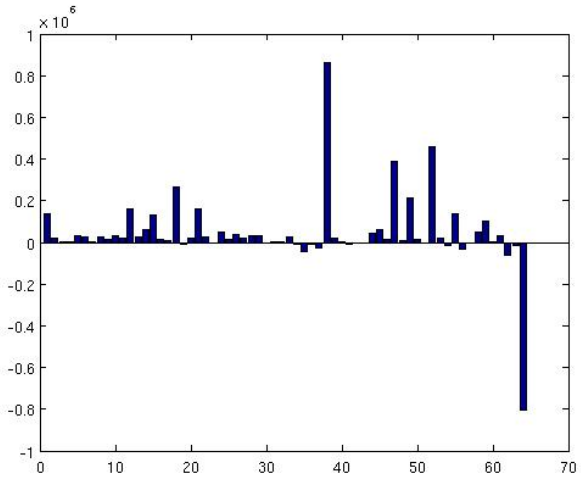

Figure 3. Sparse representation of an image.

of the dictionary instead of its original words. This also works for images; but instead of "words" we use common visual features. Figure 2 shows a dictionary learned from a set of random images downloaded from Flickr.

Using this dictionary, an image can be represented as a vector that combines elements from the dictionary. Since some of the elements are not used, we obtain a sparse representation of the image (c.f. Figure 3).

Our work explores auto-tagging of pictures using dictionary learning and sparse representation.

The magic behind Dictionary Learning and Sparse Representation is explained below. The casual reader can easily skip the end of this section as it does not affect the understanding of the following sections.

Sparse coding (SC) tries to reconstruct a signal using a dictionary and a sparse and efficient representation – i.e. a combination of only a few elements of that dictionary. The goal of dictionary learning is to train a specialized dictionary from some available data. Many research projects has suggested that dictionaries learned from data generally perform better than other off-the-shelf dictionaries such as wavelets and contourlets (Bryt & Elad, 2008; Bruckstein et al., 2009; Elad & Aharon, 2006; Krishnapuram et al., 2005; Mairal et al., 2009; 2008; Raina et al., 2007).

To state it formally, given a fixed dictionary D and n vector signals $\mathbf{x} = (x_1, x_2, \cdots, x_n)$, SC finds a sparse representation of $\mathbf{x}$ as $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n)$ by optimizing:

$$\alpha = \arg\min_{\alpha} \|\mathbf{x} - D\alpha\|_F + \lambda \|\alpha\|_p , \qquad (1)$$

where $\|\cdot\|_F$ denotes the Frobenius norm and $\|\cdot\|_p$ denotes the l_p norm after α is vectorized. Typically p is chosen as 1 or 0, where the former case ($p = 1$) is used as an approximation for the latter ($p = 0$). And λ is a weighting parameter that adjusts the importance of the two terms.

Now, to design a dictionary D so that $x \simeq D\alpha$ while $\|\alpha\|_p$ is sufficient small, the equation above can be modified as:

$$(\alpha, D) = \arg\min_{\alpha, D} \|\mathbf{x} - D\alpha\|_F + \lambda \|\alpha\|_p \qquad (2)$$

By combining the optimization of both α and D, the dictionary that is most suitable to the signal $\mathbf{x}$ can be learned. The optimization problem above can be rather involved because (2) is not simultaneously convex in terms of both α and D. However, a few optimization techniques have been proposed (Aharon et al., 2006; Engan et al., 2002).

The choice of the dictionary is extremely important and the design of such a dictionary is one of the interesting research directions that came out from this study.

3. Classify with Reconstruction Residual Error based on Dictionary Learning

Dictionary learning relies on the feature extraction technique (i.e. dictionary creation) and the reconstruction phase. The features are extracted from a set of pictures and grouped into a dictionary. From this dictionary, it is possible to reconstruct any of the original images with good precision. The difference between the original image and its reconstruction is called the residual error.

Our original idea was to use the residual error of a reconstruction in order to classify the picture. Given a dictionary per category, we would associate the picture with the category for which the reconstruction residual error is the smallest. The idea was simple enough that we could use a perceptron to determine whether or not a picture fitted a given category.

As an example, let us consider we want to classify an article as "English", "Chinese" or "Latin". One way to do this is to bring three dictionaries – one for English, one for Chinese and one for Latin – and to try to use words from each dictionary to reconstruct the article. Now, if the article is written in English, the English dictionary would match the best. It has all the words needed to reconstruct the article. The Latin dictionary would still match properly since many English words are Latin-based. On the other hand, the article would be poorly reconstructed when using the Chinese dictionary because the two languages are too different.

Using this idea, we proposed to determine which class a given image should belong to by measuring the reconstruction residual error using each dictionary. The smaller the residual error, the higher the probability of the image to belong to that category.

3.1. System Components

For this first approach, the system consists of three components:

- Training samples collector component. This component fetches training photos from websites like Flickr. After obtaining mitigated results with Flickr's tags and group pools, we decided to use an existing – and already prepared – dataset from Caltech (L. Fei-Fei & Perona). Each picture is stored in a directory reflecting its category.

- Dictionary learning component. With the pictures sorted, this component learns one dictionary per category. As we explained before, a dictionary contains "visual words" that are common to most of the images in the category. The fitness of the dictionary for a particular image is measured by the reconstruction residual error. Of course high residual error indicates low fitness. This component is as shown in Figure 4.

- Annotation component. For a given picture, a fitness value is computed against each dictionary. Our intuition was that the picture will fall into the categories with the lowest reconstruction error. Figure 5 shows an instance of good match while Figure 6 shows a mismatch. Machine Learning enters into play when determining the threshold between good match and mismatch. In this approach, we used a simple perceptron. Its input was the residual error and its output was a boolean indicating whether or not it was a match.

Given N dictionaries learned from N groups of training images $\{D_1, D_2, ...D_N\}$, reconstructing a photo I results in N reconstruction errors $\{r_1, r_2, ...r_N\}$. A reconstruction error indicates how relevant a dictionary is to the picture. For example with a dictionary learned from a set of pictures of flowers, reconstructing a new photo of a flower will have a lower residual error than reconstructing a photo of motorbike. Thus, with the residual error, we can tell if a test photo has anything to do with a dictionary.

We can building a set of training samples $\{\langle r_1, y_1\rangle, \langle r_2, y_2\rangle, ..., \langle r_N, y_N\rangle\}$ where r_i is the residual error of the reconstruction with the dictionary i and y_i is a boolean indicating whether or not the category is relevant to the picture. For example, there are three dictionaries D_{bee}, D_{lake} and $D_{montain}$. When reconstructing a photo of Moraine Lake which contains both a lake and mountains, the residual errors would be 0.8, 0.2 and 0.14 respectively. With these, we can create a training sample $\{\langle 0.8, 0\rangle, \langle 0.2, 1\rangle, \langle 0.14, 1\rangle\}$, where the second element of each pair is either 0 (irrelevant) or 1 (relevant). This process is repeated for each picture of our training data. Once done, we can train our classifier.

To classify a new photo – i.e. not part of the training data – we first reconstruct it with each dictionary separately. The residual errors are then fed into the classifier. For each relevant dictionary, a new annotation is added to the picture. More than one annotation can be added to one photo since there can be multiple residual errors that are classified as relevant.

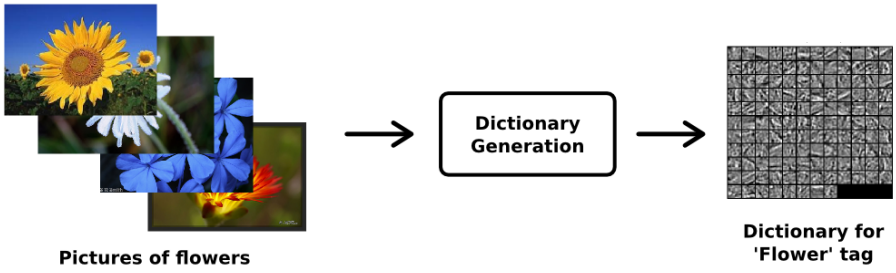

Figure 4. Dictionary Generation

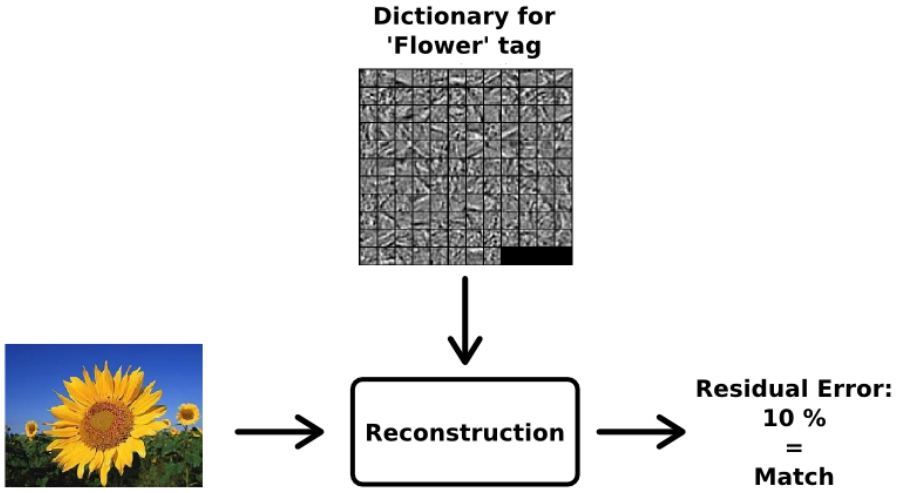

Figure 5. Good Match

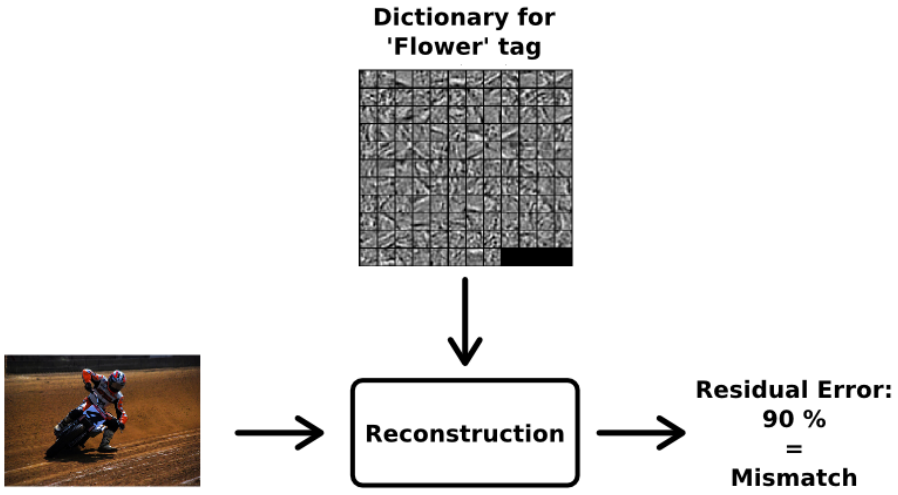

Figure 6. Mismatch

3.2. Experiment Design

In our experiment, we learned two dictionaries $D_{motorbikes}$ and $D_{flowers}$ from two sets of images. The dictionary learning was done by K-SVD which is based on k-mean clustering and Singular Value Decomposition. With these two dictionaries, we tried to reconstruct a set of test images. We expected the residual error of a picture of a flower reconstructed with $D_{flowers}$ to be lower than the residual error obtained with $D_{motorbikes}$.

3.3. Results Analysis

The results showed that, the reconstruction of flower images were always lower than reconstruction of motorbike images, no matter which dictionary was used. This result was disappointing since it meant the reconstruction error couldn't be used to determine precisely which dictionary fitted the best. Basically, the results showed that a flower is more similar to a motorbike than a real motorbike! Unfortunately, this was not the way to go.

This showed us that the analogy between text and image has its limit. In the English, Latin and Chinese example, the distinction between each dictionary is clear. But in the world of images, a dictionary extracted from one set of images can be very similar to a dictionary extracted from another set of images. One dictionary can even contain every single element of the other dictionary. Say we learn a dictionary from a white page, basically the dictionary will contain only blank elements – since this is all that is needed to reconstruct a photo of a white page. Now if we learn another dictionary for images with flowers in the center and a white page as a background, then this dictionary will contain elements for reconstructing both flowers and white page. Thus, the distinction between these two dictionaries is vague.

Based on these results, this approach could not be used for the classification.

4. Classify with Sparse Representation

Our second approach is also based on the feature extraction technique of dictionary learning. As stated above, each image can be represented as a vector of features (c.f. Figure 3). Actually, it is more a sparse representation since some features present in the dictionary might not be present in the picture.

This observation gave us the idea of merging all the dictionaries into a single one and to use it for all the categories. Figure 7 shows an instance of such a mixed dictionary.

The intuition is that two pictures from two different categories would have totally different sparse representations. And, on the contrary, two pictures belonging to a same category should have similar representations.

First, the sparse representation of the image is obtained using the common (merged) dictionary. Then its category is determined by looking how close it is to other samples in our database. Thus, our problem becomes a clustering one.

Machine learning offers several algorithms to tackle such problems. Among them, we decided to try Neural Networks, k-Nearest Neighbors and Naive Bayes, and see which one works the best.

In this study, we wrote our own implementations of Neural Networks, k-Nearest Neighbors and Naive Bayes.

4.1. System Components

The system's architecture is presented in Figure 8.

- Training samples collector component. This component is the same as for the first approach.
- Dictionary Learning component. Different from the previous approach, there is only one dictionary for all the photos. The photos that are picked for dictionary learning are randomly chosen from different classes. With this learning strategy, the dictionary contains all features for different classes.
- Sparse Coding component. Now with a dictionary, each photo can be represented using elements from the dictionary (imagining you are using words from an English dictionary to compose an article). The procedure of finding the representation is called sparse coding and the result is a sparse vector.
- Training component. The classifier can be Naive Bayes, Neural Network or k-Nearest Neighbor. Its input is the sparse representation of the image. Its output is a vector of booleans. There is a boolean per category.
- Annotation component. From the output of the classifier, this component can determine whether or not the image belongs to each category. If the boolean is 1, then the image belongs to that category.

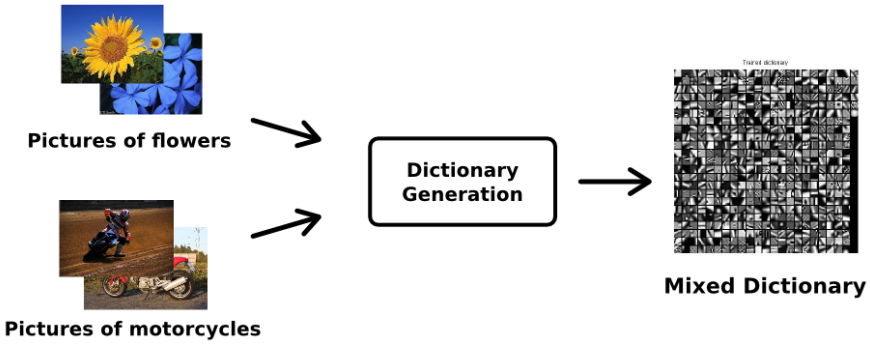

Figure 7. Merged dictionary for several categories.

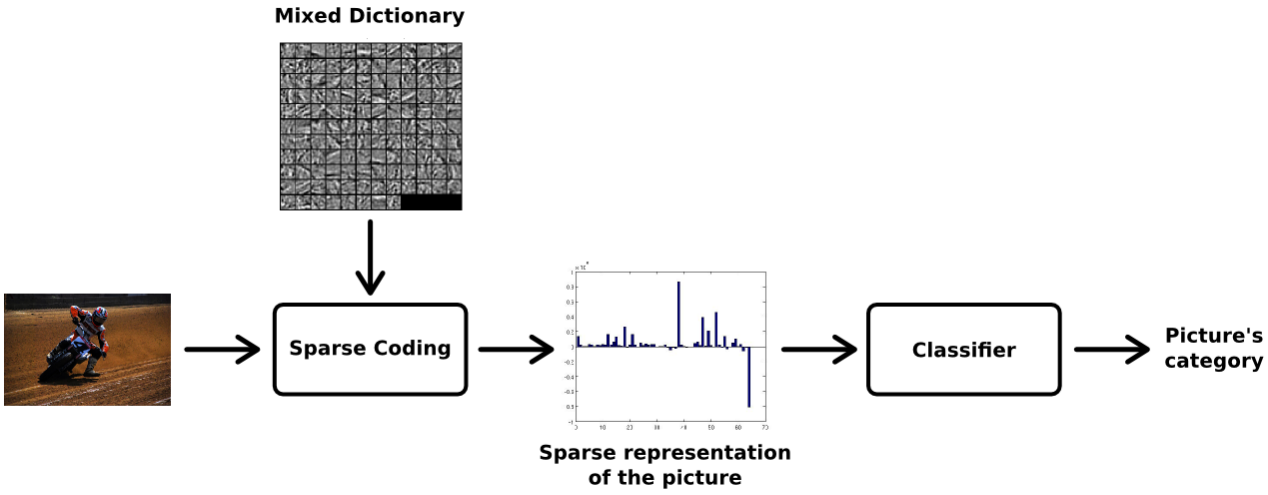

Figure 8. System's Flow Diagram

4.2. Experiment Design

In this second approach, we used six different sets of pictures to create a single dictionary. This dictionary was then used to get the sparse representation of pictures that had already been classified. Each picture was represented as a 256-feature vector. As mentioned above, each feature of the dictionary receives a value indicating how relevant it is to the picture. The resulting dataset was then split into a training set and a testing set. 75% of the pictures were used for training while the remaining 25% were used for testing.

In order to test our kNN agent, we looked at its prediction for each picture of the testing set knowing all the pictures of the training set. The tuning parameter in this algorithm is the number of neighbors that are taken into consideration when making the prediction. So, we ran the different tests while increasing k from 1 to 20.

For the neural network, it was a bit more complicated. As stated earlier, it is difficult to know how many hidden nodes it should have. So, we ran different tests using different numbers of hidden nodes. We successively increased the number of nodes from 1 to 256. We observed both the training accuracy and the testing accuracy to detect whether or not the agent was overfitting the data.

Two different datasets were used for the experiments. A first dataset consisted of only two classes with 400 samples each. This dataset allowed to determine if our approach seemed to work. The second dataset represented six different classes. Only 42 samples were used for each class. This allowed us to see how well our technique handled more complex classification problems.

4.3. Results Analysis

All three algorithms tested below were implemented for this study. No external library was used.

4.3.1. KNN

For kNN, we ran the tests on the two datasets. As shown in Figure 9, the accuracy is not really affected by k when trying to pick between two classes. With 300 samples per class (75% of the 400 samples), the agent is able to make the right prediction with about 80% accuracy.

The same tests were then run on the six-class dataset. As Figure 10 shows, the results weren't as high but still really promising. The best accuracy was around 50% which is above random – in this case 16%. It is worth noting that the kNN algorithm works best when it has multiple data points. So, with more samples, we should expect even better results.

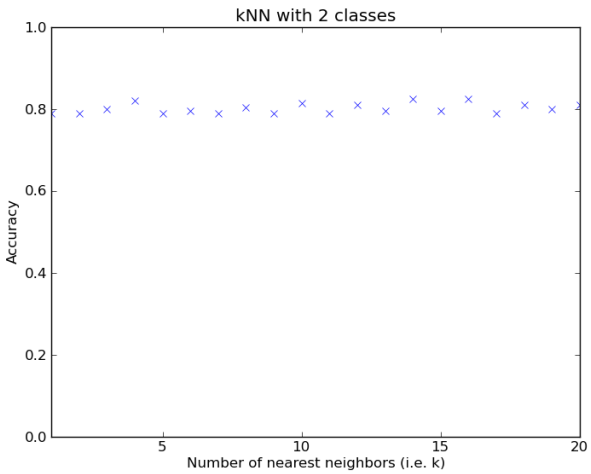

Figure 9. KNN on two classes of pictures.

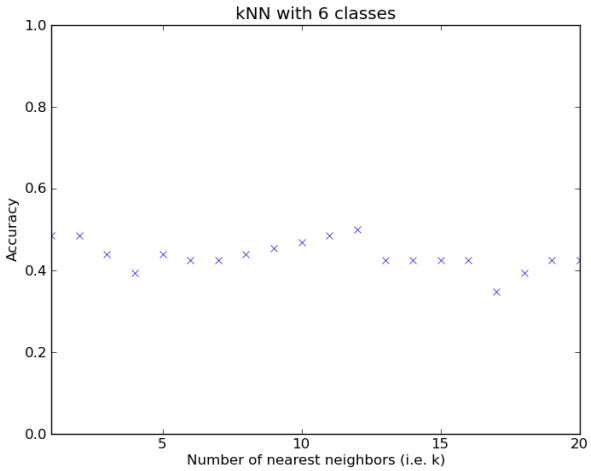

Figure 10. KNN on six classes of pictures.

4.3.2. Naive Bayes

In order for our Naive Bayes classifier to work on the continuous value we needed to decide how the CPT table will be represented. When working with continuous values, the CPT is usually represented as a conditional distribution table (CDT) which maps a certain condition to a distribution. Another way is to discretize the continuous value. This is the method adopted in this experiment.

The number of training sample is 75% of 400 samples and the rest is for testing.

The granularity of the discretization affects the prediction accuracy. So, in our test, we varied the granularity to see the impact it had on the result. As shown in Figure 11 for the two-class dataset, the accuracy goes down as the granularity increases. Naive Bayes uses training samples to estimate the true probability of values of an attribute given a class label. As the number of training samples increases, the probability approaches the true value. When the number of training samples is too low, the probability is far from the true value. With the discretization, the probability of each discrete value showing for each label is even farther from the true value. This is why it decreases when the granularity increases.

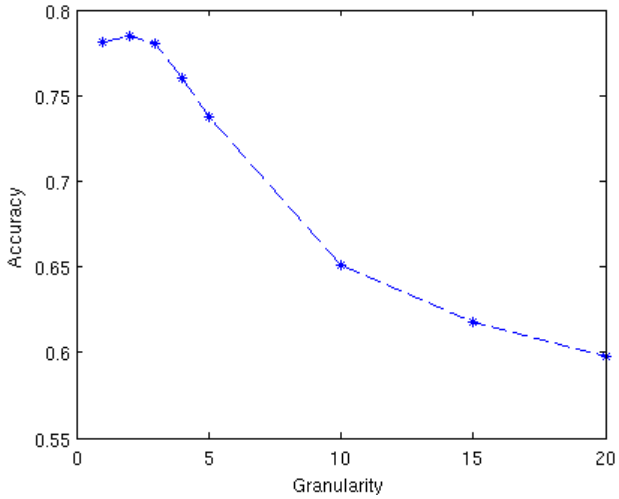

Figure 11. Naive Bayes on two classes of pictures.

The best result Naive Bayes achieved for the 2-class dataset was 78.50%.

We performed the same test on the 6-class dataset and the best classification accuracy was 64%.

4.3.3. Neural Networks

The main difficulty with neural networks is to determine the number of hidden nodes that suits the problem. Too few hidden nodes, and the neural network can't learn completely. Too many, and it is the opposite: the neural network knows too much (i.e. overfitting issue).

Tests were run in order to see how learning was affected by the number of hidden nodes. The results are shown in Figure 12 for two classes and in Figure 13 for six classes. In both cases, the best training accuracy was obtained with 256 hidden nodes – which is the input size. We set the learning rate α to 0.01 as it allows for a fair amount of exploration.

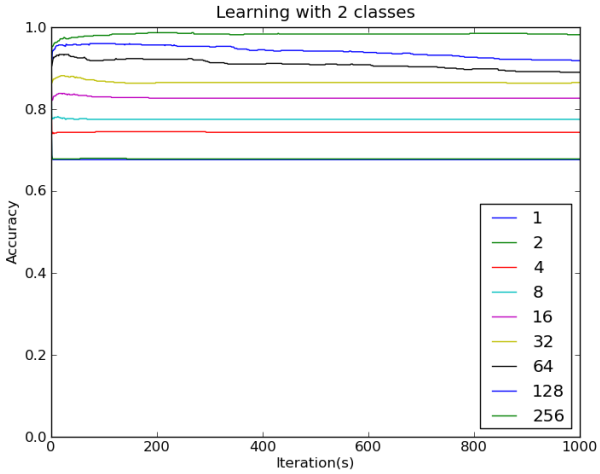

Figure 12. Neural Networks on two classes of pictures.

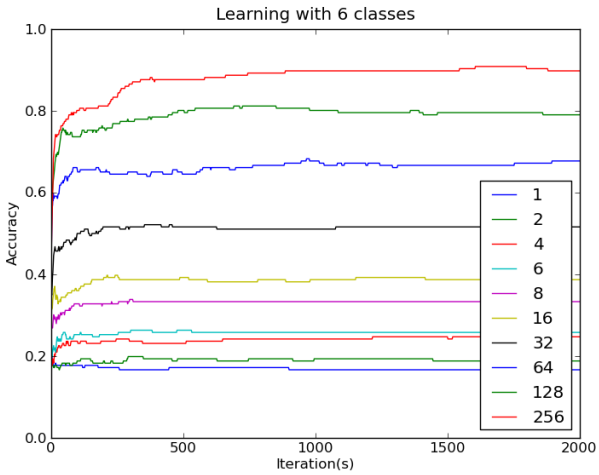

Figure 13. Neural Networks on six classes of pictures.

In parallel, tests were conducted to see how well neural networks predicted classes for pictures that they had never "seen". Basically, the neural networks were trained with 75% of the dataset and tested with the remaining samples.

In Figure 14, we can see that the training accuracy for two classes increases rapidly up to close to 1. The prediction accuracy (i.e. testing) increases more slowly but still goes above 60% in only 1,000 iterations.

When training on six classes, the best training accuracy doesn't go much above 90%. But, as shown in Figure 15, the prediction is still well above random.

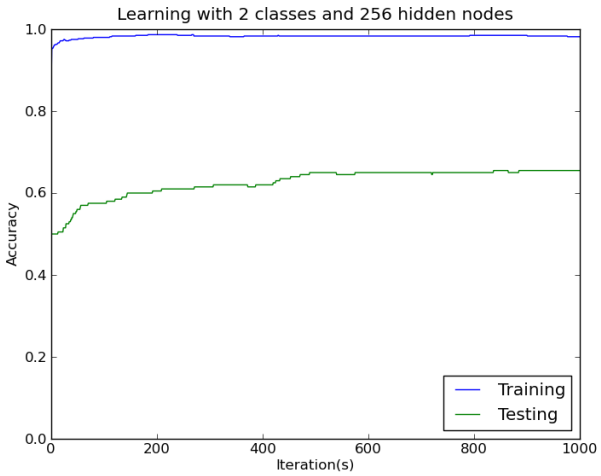

Figure 14. Training vs Testing for Neural Networks on two classes of pictures.

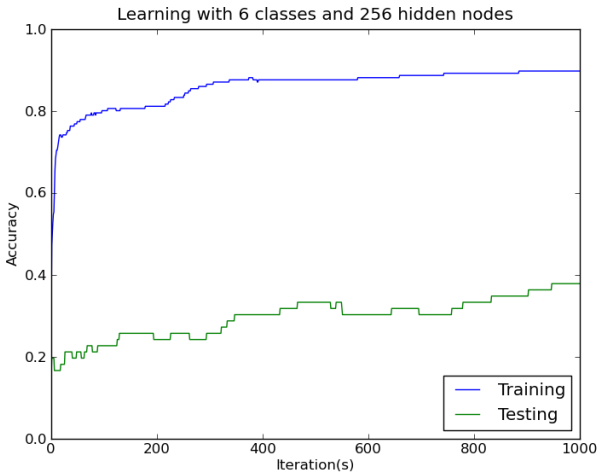

Figure 15. Training vs Testing for Neural Networks on six classes of pictures.

4.4. Discussion

A summary of the results is presented in Figure 16. The three classifiers performed better than random. In our tests, Naive Bayes seem to be the most consistent. But it is important to keep in mind that our dataset was relatively small. With a larger dataset, kNN and Neural Networks are expected to take the lead.

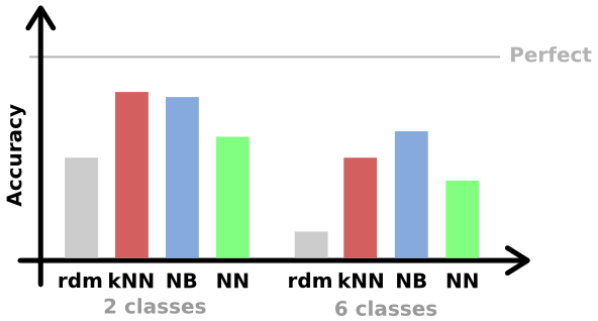

Figure 16. Comparison of the prediction accuracies obtained with k-Nearest Neighbors, Naive Bayes and Neural Network for two and six classes of pictures.

5. Future Work

The number of classes this system can automatically attached label to depends on the size and scale of training samples. Currently the system is trained only to work on 6 classes. To make this system practical, more classes need to be trained. Training data is essential in machine learning and is usually hard to collect but fortunately this is not our case. From Flickr, we can collect essentially infinite number of training data for different classes and the size of available training is growing quickly each day. For the next step, we might need to make the existing data collecting component to work automatically collecting data from Flickr in the background and trigger the training component to learn new classes periodically.

Currently our system can only attach one label to one photo but it would nice if the algorithm can detect multiple region of interest and attach label accordingly. One way to implement attaching multiple labels is to use image segmentation algorithm which can automatically separate different regions and based on these regions, the existing algorithm can run and attach labels accordingly. Another way to do it is letting users to manually choose regions of interest and run the algorithm on each region to attach label.

Finally, as suggested by Dr McGovern, we could have the three classifiers (Naive Bayes, kNN and Neural Networks) vote for the picture's category. The category with the most votes wins.

6. Conclusion

In the work we have implemented an auto-labelling system for photos. The idea behind the labelling is photo classification. We have tried two different algorithms for classification. One algorithm was on based the reconstruction residual error with dictionary learning. But because the "visual keywords" have less differential power than "written keywords", the first algorithm did not work well. The second algorithm we tried was based on the sparse representation by dictio-

nary learning and sparse coding. Based on this sparse representation, we have tried a few general classification algorithm such as KNN, Naive Bayes and Neural Network. The accuracy of classification is high.

References

Aharon, M., Elad, M., and Bruckstein, A. K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311, 2006.

Bosch, A., Zisserman, A., and Munoz, X. Scene classification via pLSA. *Computer VisionECCV 2006*, pp. 517530, 2006.

Brants, T., Chen, F., and Tsochantaridis, I. Topic-based document segmentation with probabilistic latent semantic analysis. In *Proceedings of the eleventh international conference on Information and knowledge management*, pp. 218, 2002.

Bruckstein, A.M., Donoho, D.L., and Elad, M. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM review*, 51 (1):34–81, 2009.

Bryt, O. and Elad, M. Compression of facial images using the K-SVD algorithm. *Journal of Visual Communication and Image Representation*, 19(4):270–282, 2008.

Elad, M. and Aharon, M. Image denoising via sparse and redundant representations over learned dictionaries. *Image Processing, IEEE Transactions on*, 15 (12):3736–3745, 2006.

Engan, K., Aase, S.O., and Husoy, JH. Frame based signal compression using method of optimal directions (MOD). In *Circuits and Systems, 1999. ISCAS'99. Proceedings of the 1999 IEEE International Symposium on*, volume 4, pp. 1–4. IEEE, 2002.

Fergus, R., Fei-Fei, L., Perona, P., and Zisserman, A. Learning object categories from google? s image search. 2005.

Fergus, R., Fei-Fei, L., Perona, P., and Zisserman, A. Learning object categories from internet image searches. *Proceedings of the IEEE*, 98(8):14531466, 2010.

Krishnapuram, B., Carin, L., et al. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 957–968, 2005.

L. Fei-Fei, R. Fergus and Perona, P. Caltech 101: pictures of objects belonging to 101 categories.

Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91110, 2004.

Mairal, J., Sapiro, G., and Elad, M. Learning multiscale sparse representations for image and video restoration. *SIAM Multiscale Modeling and Simulation*, 7(1):214241, 2008.

Mairal, J., Bach, F., Ponce, J., Sapiro, G., and Zisserman, A. Supervised dictionary learning. *Adv. NIPS*, 21, 2009.

Maron, O. and Lozano-Prez, T. A framework for multiple-instance learning. *Advances in neural information processing systems*, pp. 570576, 1998.

Maron, O. and Ratan, A. L. Multiple-instance learning for natural scene classification. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 341349, 1998.

Qiao, Q. and Beling, P. A. Localized content based image retrieval with Self-Taught multiple instance learning. In *2009 IEEE International Conference on Data Mining Workshops*, pp. 170175, 2009.

Raina, R., Battle, A., Lee, H., Packer, B., and Ng, A. Y. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pp. 766, 2007.

Sivic, J. and Zisserman, A. Video google: A text retrieval approach to object matching in videos. 2003.