Machine Learning Project

Narendra Jhabakh
Civil and Environmental Engineering
njhabakh@andrew.cmu.edu

Shubhaditya Majumdar
Mechanical Engineering
shubhadm@andrew.cmu.edu

1 Introduction

Machine learning is a powerful tool for recognizing and classifying data using completely unsupervised and/or supervised techniques. Herein, the problem of image classification is tackled using a mixture of unsupervised and supervised techniques to develop and train learning models and subsequently test the model by classifying unlabeled images.

1.1 Background:

This Project focuses on classifying images from the CIFAR-10 data-set using concepts learnt from the class 10:601. The CIFAR-10 dataset consists of 60,000 labeled images collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Of the 60,000 images 50,000 of them are set aside for training. Each of the images are composed of 32*32 pixels in an rgb format. Hence each row of the data-set corresponds to a single image with 32*32*3 = 3072 values. The pixel values are reshaped using im_show(reshape(data(i,:),32,32,3)). For the purpose of this project, we were presented with a subset of images for training, consisting of 5000 images having an equal distribution of images across all the 10 classes.

1.2 Related Work:

For the dataset given to us, the worldwide records of classification accuracy involve deep learning techniques and convolutional neural networks with sometimes sophisticated feature detection algorithms. For example, Coates *et al.* [1]. evaluated the performance of various unsupervised learning schemes such as K-means clustering and Gaussian mixture models to extract features from images and then classified test images using Support Vector Machines. They found the best accuracy performance using soft K-means (79.6 %). The record for the highest accuracy (published) on the CIFAR-10 dataset is for model developed by Lee *et al.* [2]. using deeply-supervised nets with an accuracy 91.78%.

In this approach, we generate image features using the functions available in the in-built libraries from VL-FEAT. The extraction methods are HOG (vl_hog) and SIFT (vl_phow/dsift). Once we have extracted the features, the features are sent to a classifier for training. We implement Gaussian Naive Bayes, Logistic Regression, Bag of Words, k-Nearest Neighbours and intend to implement Support Vector Machines. Following sections explain in detail the feature extraction and classification methods we have used.

2 Method:

2.1 Feature Extraction:

2.1.1 Histogram of Gradients

We extract Histogram of Gradient (HOG) features using the vl_hog function in the VL-FEAT library. This method divides the image into blocks (specified by a user-defined cell size in pixels), calculates a histogram of oriented gradients in each cell. A final renormalization is performed for the histograms in each cell providing feature values between 0 and 1. For example, for 32*32 image, using a cell size of 4 pixels (i.e. the cell is a 4*4 image), the number of features generated is 1984. An example image of the HOG feature extraction is shown in Figure 1(b) along with the original image (Figure 1(a)).

2.1.2 Scale Invariant Feature Transform :

The Scale Invariant Feature Transform (SIFT), which gives the frames and the descriptors are extracted from each image of the training data-set. We use both vl_sift and vl_phow(dense sift), to generate the frames and their descriptors. Figure 2 shows the image with the sift (1c) and the dsift (1d) features. With dense SIFT, a SIFT descriptor is generated at every location of the image, whereas with normal SIFT, descriptors at the locations determined by Lowe's algorithm [3] are obtained.

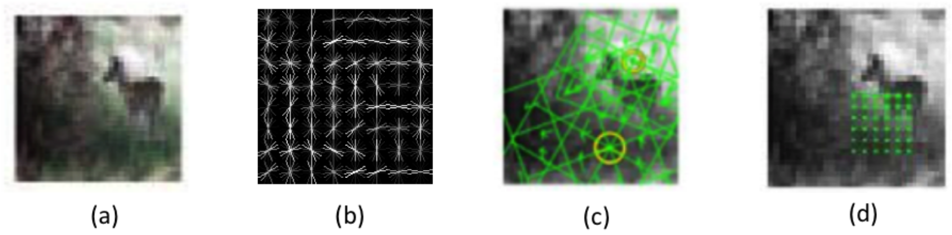


Figure 1: (a) Original image as read from CIFAR-10 set. (b) Features using HOG. (c) Features using SIFT. (d) Features using DSIFT.

2.1.3 Principal Component Analysis(PCA) and Whitening:

Here we tried to reduce the dimensionality of the images by using PCA and also used a pre-processing step called Whitening to improve accuracy. As explained in [4], in order for PCA to work well, two conditions must be satisfied, (i) The features space must have approximately zero mean, and (ii) The different features should have comparable variances to each other. As seen in normal images the pixels close to each other have a high correlation. Whitening makes these features less redundant by making adjacent features less correlated and having the same variance. Instead of using normal whitening we use something called ZPCA (zero-phase component analysis) Whitening (*Bell and Sejnowski 1996*[5]). The algorithm to compute W_{ZPCA} for an image is as follows:

X composed of n features and m training examples (X has size $[n \times m]$). The covariance Σ of each of the dimension with respect to the other is(after centering each dimension with respect to the mean):

$$\Sigma = Cov(X) = \mathbf{E}[XX^T] = \frac{XX^T}{m} \tag{1}$$

Using Eigen-Value Decomposition:

$$\Sigma = UDU^{-1} \tag{2}$$

Where every column of U ($[n \times n]$) is an eigen vector of Σ and D is a diagonal matrix of eigen values. Normal PCA whitening transformation is given by

$$W_{PCA} = D^{-1/2}U^T \tag{3}$$

As seen whitened data will remain the same after rotation, i.e. $W = RW_{PCA}$, with an orthogonal matrix R is also a whitening transformation. Under ZCA whitening, we take U as this orthogonal matrix, i.e.

$$W_{ZCA} = UD^{1/2}U^T = \Sigma^{-1/2} \tag{4}$$

The following images shows the implementation of the above method before and after whitening.

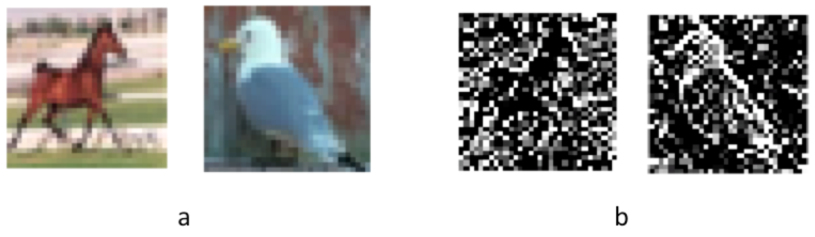


Figure 2: (a) Original images as read from CIFAR-10 set. (b) Images after ZCA-Whitening.

2.2 Classification:

2.2.1 Gaussian Naive Bayes (GNB) Model

This was the baseline classification algorithm used in this study. Let $\mathbf{X}^{\mathrm{N}}$ denote the N-th training set of features and is a k dimensional vector. The GNB model computes a set of priors, means and variances from the training examples based on which it calculates conditional probabilities to classify the test data. The mean is calculated using

$$\mu_{ik} = \frac{\sum_j X_i^j \delta(Y^j = y_k)}{\sum_j \delta(Y^k = y_k)} \tag{5}$$

where μ_{ik} is the mean for the i-th feature and k-th class, j indexes all training examples and **Y** is the vector of class labels. The variance is given by

$$\sigma_{ik}^2 = \frac{\sum_j (X_i^j - \mu_{ik})^2 \delta(Y^j = y_k)}{\sum_j \delta(Y^k = y_k)} \tag{6}$$

For classification, we then calculate the conditional probability $P(X_i|Y = y_k)$ for each feature in an image using

$$P(X_i|Y = y_k) = \frac{1}{\sqrt{2\pi\sigma_{ik}^2}} e^{\frac{-|X_i - \mu_{ik}|^2}{2\sigma_{ik}^2}} \tag{7}$$

Through the Naive Bayes assumption, we classify the training image to that label y_k for which $\prod_i P(X_i|Y = y_k)$ is maximum.

2.2.2 Logistic Regression (LR)

Here we try to learn the function directly $f : X \to Y$, where X in this case is the feature set and $Y \in (y_1, .., y_K)$ is the class-set. The conditional probabilities are as follows[6]:
For $Y \in (y_1, ...y_{K-1})$

$$P(Y = y_k|X) = \frac{exp(w_{k0} + \sum_{i=1}^{n} w_{ki} X_i)}{1 + \sum_{j=1}^{K-1} exp(w_{j0} + \sum_{i=1}^{n} w_{ji} X_i)} \tag{8}$$

When $Y = y_K$:

$$P(Y = y_k|X) = 1 - \sum_{j=1}^{K-1} P(Y = y_j|X) \tag{9}$$

Here K is the total number of classes which is 10 and w_{ji} is the weight associated with the j^{th} class and i^{th} feature X_i.

2.2.3 Bag of Words (BOW)

In this classification scheme, the features from the training set, i.e the descriptors generated from SIFT are first clustered into K clusters (words) using a k-means scheme. The features are then assigned to a particular cluster (for our study, we did not assume 'soft' membership), i.e., it is assigned to that cluster which is situated closest to it (based on Euclidean distance - L2 norm of the difference between the two vectors). Once all the features are assigned to their respective K-centres, a histogram is generated based on the cluster to which each feature (descriptor) of the image belongs to. This is used to train a Naive Bayes model using

$$P(X_i|Y = y_k) = \frac{\#D\{X = x_i, Y = y_k\}}{\#D\{Y = y_k\}} \tag{10}$$

where $D\{\}$ denotes the count. The test images can now be classified using this model after feature extraction, clustering and histogram calculation (just like for the training images).

2.2.4 k-Nearest Neighbours:

The HOG feature set for all training images is itself the training model. The HOG feature set for test images is compared to the training set in terms of its Euclidean distance. The top k nearest neighbour training images are taken and a histogram is generated for its corresponding labels. If the maximum value of the histogram bins is unique, then that label is assigned to the test image. If the maximum value of any bin is 1, the label for the training image with the minimum distance from the test image is assigned. To resolve a tie, a mean value of the distances for all training images for that bin is calculated and the label corresponding to the minimum mean distance is assigned to the test image. An alternate way of resolving a tie is to assign a weighting factor factor to each bin *i* as

$$w_i = \sum_j e^{\frac{-|X^{test} - X^j|^2}{2}} \tag{11}$$

where *j* spans the set of all images for the bin corresponding to label *i*.

2.2.5 Support Vector Machines:

Support vector machines (SVMs) are powerful classification methods in which linear and non-linear decision boundaries can be implemented. Instead of utilizing the entire data set for classification, support vectors i.e. vectors to data points closest to the decision boundary are chosen and only these are used. A hyperplane is constructed at the boundary and the distance of the hyperplane between the support vectors (known as the margin) is minimized during training. This minimization can be optimized by converting the data points to a higher dimension where a linear division exists between the support vectors of the different classes.

The minimization step to find the weights w defining the hyperplane dividing the training vectors $x_{(i)}$ (ith training image) can be analytically written as

$$\begin{aligned} &\text{min}_{(w,b,\zeta)} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \zeta_i \\ &\text{subject to } y_i(w^T z_i + b) \geq 1 - \zeta_i \\ &\qquad\qquad\qquad \zeta \geq 0, i = 1, 2, ...l \end{aligned} \tag{12}$$

where l is the number of training images, C the penalty term, ζ_i is the error term for the ith training image, and $z_i = \phi(x_i)$ where ϕ is a function transforming the features x_i into a higher dimensional space. This formulation is known as the "primal form". Now, the kernel function is defined as $K = \phi(x)^T \phi(x)$. This abstraction is useful when x_i is transformed into an infinite-dimensional feature space as in the case of an Radial basis function (RBF) kernel.

However, it is often beneficial to transform the primal formulation what is known as the "dual form" using Lagrange multipliers, which can be evaluated using quadratic programming. The dual form can be written as

$$\begin{gathered}\text{min}_{(\alpha)} \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \\ \text{subject to } 0 \leq \alpha_i \leq C, i = 1, 2, ...l \\ y^T\alpha = 0\end{gathered} \tag{13}$$

where e is a vector of 1's, Q is a $l \times l$ semi-definite matrix wherein $Q_{i,j} = y_i y_j K(x_i, x_j)$. Then, we get $w = \sum_{i=1}^{l} \alpha_i y_i \phi(x_i)$ and we classify the test images based on $sign(\sum_{i=1}^{l} \alpha_i y_i K(x_i, x))$ (+ve sign indicates a +ve classification).

1. *Gaussian (RBF) kernel*
   The kernel function is represented as
   $$K(x_i, x_j) = -\frac{exp(||x_i - x_j||^2)}{2\sigma^2} \tag{14}$$
   This maps x_i into an infinite dimensional feature space.
2. *One vs All classification*
   We use the one vs all scheme to extend our binary SVM implementation to a multi-class model. We run our SVM minimization scheme 10 times, where every time the label under classification is given the identity +1 and all other labels are given the identity -1. Now, when a particular test image is classified positively by more than one SVM run, we classify it to that label which has the maximum value of $sign(\sum_{i=1}^{l} \alpha_i y_i K(x_i, x))$.

3 Experiments and Results

3.1 Performed Procedures

We present our results for the k-NN classifier, GNB classifier and SVM classifier for the feature set created by the HOG scheme. We also attempted classification using LR (on HOG) BOW (on HOG and SIFT) and SVM (on raw pixels but clustered using k-means) but the classification accuracy were quite low.

3.2 Ensemble and Voting:

We took an ensemble set of our three main classifiers: NB, k-NN and SVM, to arrive at a final classification of the test images. For every test image, we calculate the probability of classifying it into label y_i by each classifier. We then add up the probabilities (to get weights) and classify the image into y_i that has the greatest weight.

The probabilities from each classifier is not trivial to obtain. NB directly gives the probability $P(Y = y_i|X, \theta)$, where θ is the learned classifier model. For k-NN 2, we calculate the mean distances of each class label cluster from the test image, weigh it by the number of occurrences of that class ($w_i = 1/\#D\{y_i\}$) in the k nearest neighbour set and normalize them to get a probability according to $P(Y = y_i|X) = \sum_{j \neq i} w_j r_j / Z$, where Z is a normalizing factor and r_j is the mean k-NN distance for the jth class label. For SVM, every class label having a -ve sign of $(\sum_{i=1}^{l} \alpha_i y_i K(x_i, x))$ is made to have $P(Y = y_i|X) = 0$. For all the other classes, normalize the values of $(\sum_{i=1}^{l} \alpha_i y_i K(x_i, x))$.

A less robust form of the above voting algorithm is also possible where for every test label we compare the predictions from the three classifiers. If more than one classifier predicts the same class label, we choose that. If not, we compare the probabilities associated with the predictions of each classifier and choose the highest one.

It turns out that only the latter voting method is working on Autolab, eventhough both work on our local machine. We present the results in Table 1.

3.3 Boosting:

In this method we again use a combination of the above three classifiers. We consider kNN and GNB to be weak classifiers when compared with the SVM. Based on the work by Robi Polkar [8], the following algorithm is performed: Classifier 1 is used to train a set of sample data. This sample data is again used for testing. All the misclassified samples and a few of the correctly classified samples (we took two cases of 50% and 75% of the correct samples) are used as training set for the weak Classifier 2. The same samples are again used for testing on the 2nd classifier. Now the training samples for the third strong Classifier are all the misclassified samples from Classifiers 1 and 2. Once this has been done, the testing set is used on all three Classifiers. The results from the three are compared in the following manner: If all the predicted labels are different, the final classification is that of the strong classifier. In case of a repetition the final classification is that of the repeated label irrespective of the classifier.

3.4 Clustering of pixels

We followed the procedure outlined by Coates *et al.*[1] for generating features from the pixels of a training image using k-means clustering. We first created square sub-patches of the images and performed pre-processing steps of mean value subtraction and ZCA whitening. These patches may or may not have overlapping sections with one another but we chose them to be overlapped so that we were consistent with the method outlined by Coates *et al.*[1] Using the pixel values of every sub-patch as our training vector, we clustered these into k clusters using k-means clustering. Now, every image has been converted into a space where it is described by the classification of each sub-patch into a cluster. Coates *et al.*[1] showed that dividing these features into four quadrants further improves accuracy of final classification by isolating localized image characteristics and also reduces training time since we have a fewer number of features to train the SVM on. We follow this procedure and the results of the classification are shown in Table 1.

3.5 Testing Accuracy:

Table 1 summarizes the accuracies obtained for each of the methods for feature extraction and classification on our local system. The classifier is trained on the first 4 data sets and tested on the last one (cifar_small). Here the only difference between kNN 1 and kNN 2 is that the latter uses the weighting factor as shown in Eq (7) in the case of a tie.

Table 1: Performance Analysis

Classifier/Feature Extraction	**HOG**	**SIFT**
GNB	48.8%	-
LR	8.1%	11.4%
BOW	-	22.1%
kNN 1	57.0%	34.6%
kNN 2	57.4%	33.4%
kNN 2 *with whitening*	41.2%	-
SVM	59.2%	-
SVM *with whitening*	35.2%	-
Others:		
Voting with probabilities	62.9%	-
Voting with numbers	57.2%	
Boosting	56.2%	-
SVM with k-means	34.6%	

For the k-NN schemes, we maximized our classification accuracy with respect to k (number of nearest neighbours) and cell size (HOG). In Figure 2, we plot the classification accuracy as a function of k and find the maximum accuracy (0.57 or 57%) for the case where **k=27**.

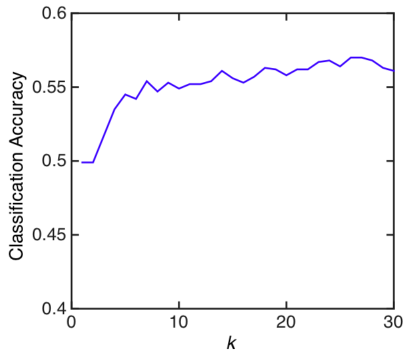
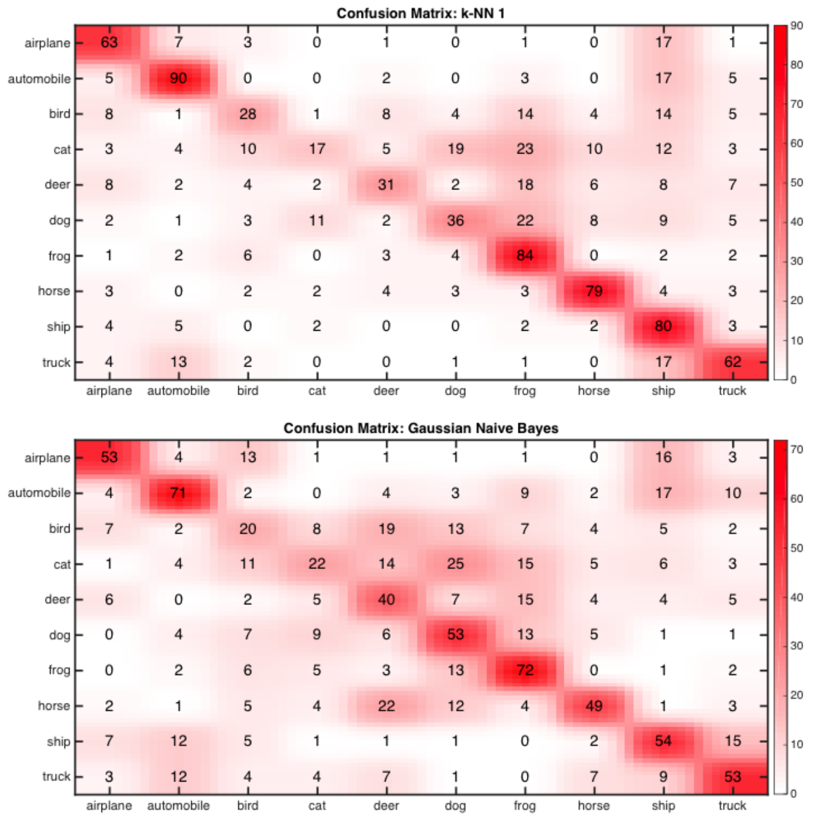


Figure 3: Classification accuracy as a function of k for HOG features with a cell size of 4 pixels.

3.5.1 Confusion Matrix: Heat Map

We plot the confusion matrix as a heat map for the best case k-NN 1 model, the baseline GNB model in Figure 4 (Mid-Term) and for the SVM model in Figure 5. The interesting feature to note is that all perform relatively well on similar labels and badly on others. For example, for the k-NN 1 classifier, a lot of 'cat' images are classified as 'dog' and vice-versa. This is probably because their shape/form are very similar and it is quite difficult to distinguish them in a 32*32 image. SVM performs very badly for 'deer' but much better on others.

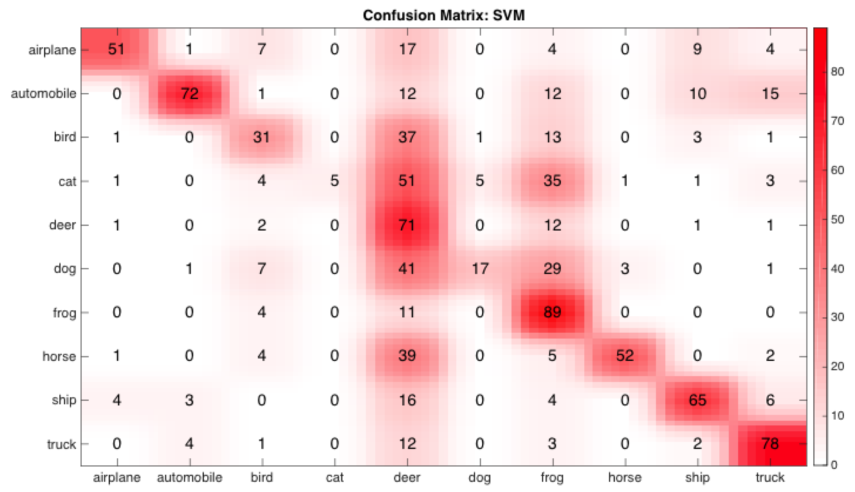

Confusion Matrix: k-NN 1

	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	63	7	3	0	1	0	1	0	17	1
automobile	5	90	0	0	2	0	3	0	17	5
bird	8	1	28	1	8	4	14	4	14	5
cat	3	4	10	17	5	19	23	10	12	3
deer	8	2	4	2	31	2	18	6	8	7
dog	2	1	3	11	2	36	22	8	9	5
frog	1	2	6	0	3	4	84	0	2	2
horse	3	0	2	2	4	3	3	79	4	3
ship	4	5	0	2	0	0	2	2	80	3
truck	4	13	2	0	0	1	1	0	17	62

Confusion Matrix: Gaussian Naive Bayes

	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	53	4	13	1	1	1	1	0	16	3
automobile	4	71	2	0	4	3	9	2	17	10
bird	7	2	20	8	19	13	7	4	5	2
cat	1	4	11	22	14	25	15	5	6	3
deer	6	0	2	5	40	7	15	4	4	5
dog	0	4	7	9	6	53	13	5	1	1
frog	0	2	6	5	3	13	72	0	1	2
horse	2	1	5	4	22	12	4	49	1	3
ship	7	12	5	1	1	1	0	2	54	15
truck	3	12	4	4	7	1	0	7	9	53

Figure 4: Confusion matrix for k-NN 1 and GNB classification schemes. The numbers correspond to the number of images in class *i* (row) classified as class *j* (column). (Mid-Term)



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	51	1	7	0	17	0	4	0	9	4
automobile	0	72	1	0	12	0	12	0	10	15
bird	1	0	31	0	37	1	13	0	3	1
cat	1	0	4	5	51	5	35	1	1	3
deer	1	0	2	0	71	0	12	0	1	1
dog	0	1	7	0	41	17	29	3	0	1
frog	0	0	4	0	11	0	89	0	0	0
horse	1	0	4	0	39	0	5	52	0	2
ship	4	3	0	0	16	0	4	0	65	6
truck	0	4	1	0	12	0	3	0	2	78

Figure 5: Confusion matrix for SVM classification scheme. The numbers correspond to the number of images in class *i* (row) classified as class *j* (column).

3.6 Submission to Autolab

The three classifiers submitted to Autolab are the ones which provided the three highest classification accuracy on our local systems. These include SVM, voted SVM + NB + kNN (with numbers), and voted SVM + kNN (with probabilities). These correspond to classification accuracies of 58.0%, 57.8% and 58.3% respectively on Autolab. Their associated train.m files have also been uploaded.

4 Conclusion

We have implemented base-line approaches (NB) for classification and compared it to a more robust k-NN based classification and SVM classification, which yield higher accuracies. We find SVM to be the best classification method for this dataset yielding the highest accuracy of 58.0% on Autolab (59.2% on our local system). To improve our accuracy, we have also experimented with ZCA whitening as a pre-processing step, k-means clustering for feature generation, and ensemble voting and boosting on a combination of our original classifiers. Among these, we find that pre-processing reduces the final accuracy whereas as the ensemble techniques just marginally beat the SVM accuracy on our local system.

References

[1] Coates, Adam, Andrew Y. Ng, and Honglak Lee. ”An analysis of single-layer networks in unsupervised feature learning.” International conference on artificial intelligence and statistics. (2011).

[2] Lee, Chen-Yu, et al. ”Deeply-supervised nets.” arXiv preprint arXiv:1409.5185 (2014).

[3] David G. Lowe, ”Distinctive Image Features from Scale-Invariant Keypoints,” International Journal of Computer Vision, 60, 2 (2004), pp. 91-110.

[4] ”Implementing PCA/Whitening.” - Ufldl. Stanford.edu, n.d. Web. 16 Dec. 2015.

[5] Bell, Anthony J., and Terrence J. Sejnowski. Edges Are the Independent Components of Natural Sciences. (1996)

[6] Mitchell, Tom M. ”Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression.” Machine Learning (2015), Chapter 3

[7] Vedaldi, Andrea, and Brian Fulkerson. ”VLFeat Vedaldi Tutorial.” (2015) Academia.edu.

[8] Polikar, Robi. ”Ensemble Learning.” Scholarpedia. MediaWiki, 2009. Web. 17 Dec. 2015.