Computer Vision Assignment 9 - Image Classification

Kai-Po Lin (kl72); Department of Computer Science

2. Can you fix it? Yes, you can!

2.1 Algorithm Implementation Revisited

(1) You should aim to handle the 25-classes dataset in its entirety. If practicality should dictate otherwise, use as many classes as you are able. In any case, how you go about improving upon your existing classification algorithm is up to you. Remember that you are allowed to use code from other sources, so long as you explicitly state in your report what those sources were and what code you used. However, it is not allowed to use extra data. As a result, deep learning techniques are not recommended here since it is unfair to use pre-trained models or transfer learning. What follows are some targeted suggestions intended to improve performance with larger datasets, at least one of which you must implement:

- PCA-SIFT
- Speeded Up Robust Features (SURF)
- oFAST + rBRIEF (ORB)
- Very Fast SIFT (VF-SIFT)
- Vocabulary Tree Clustering

⇨ I chose SURF and ORB to reduce the dimensionality of the feature. **I first test on the reduced dataset to see either SURF or ORB performs better, then I will pick one to perform on the 12-classes and 25-classes dataset.** Time and accuracy are both important for my measurement. The following lines show the code I modified from the previous assignment.

SIFT:
```
[f, d] = vl_sift(gray_image);
descriptors{cnt} = im2double(d');
```

SURF:
```
points = detectSURFFeatures(gray_image);
[features, ~] = extractFeatures(gray_image, points);
```

ORB:
```
points = detectORBFeatures(gray_image);
[features, ~] = extractFeatures(gray_image, points);
features = im2single(features.Features);       % features.Features is uint8 type
                                               % kmeans only accept single or double type
```

The following table compares the three methods' accuracy, time, and their best threshold value (using kmeans() to cluster features and fetch the median value of features to the threshold policy) by running the threshold value from 0.8~1 with 0.01 increment.

Method	Average Accuracy	Time	Best Threshold
SIFT	92.50%	312.89 secs	0.95
SURF	87.92%	64.59 secs	0.97
ORB	87.08%	529.88 secs	0.88

SIFT:

Best Sum: 277.50, Best Threshold: 0.95

-------------------------- Result --------------------------

Classes	Hat	Butterfly	Airplane
Hat	90.00%	10.00%	0.00%
Butterfly	0.00%	100.00%	0.00%
Airplane	12.50%	0.00%	87.50%

Elapsed time is 312.892765 seconds.

SURF:

-------------------------- Result --------------------------

Classes	Hat	Butterfly	Airplane
Hat	70.00%	30.00%	0.00%
Butterfly	0.00%	100.00%	0.00%
Airplane	6.25%	0.00%	93.75%

Elapsed time is 64.589963 seconds.

ORB:

Best Sum: 261.25, Best Threshold: 0.88

-------------------------- Result --------------------------

Classes	Hat	Butterfly	Airplane
Hat	80.00%	0.00%	20.00%
Butterfly	0.00%	100.00%	0.00%
Airplane	0.00%	18.75%	81.25%

Elapsed time is 529.876131 seconds.

⇨ According to the result, we could see that SIFT, SURF, and ORB maintain similar accuracy but ORB and SIFT required more computational time than SURF. That is, I will pick SURF as a feature extractor on the expanded dataset.

2.2 Technical Write-up: Results and Discussion

1. Describe the salient features of your improved classification framework.

Speeded Up Robust Features (SURF) are famous **scale and rotation invariant interest point detectors and descriptors**, like the Harris corner detector or SIFT. Read the attached paper on SURF and write a summary of your understanding in a page or two. Mention the key contributions and the reasoning behind the choices made in the paper. Keep the details at a high level and don't include any equations.
Paper Cited: https://people.ee.ethz.ch/~surf/eccv06.pdf

Introduction: They presented a scale and rotation invariant interest point detector and descriptor, coined SURF (Speeded Up Robust Features). It approximates or even outperforms previously proposed schemes concerning **repeatability**, **distinctiveness**, and **robustness**, yet can be computed and compared much faster. To achieve the goal, they **developed both a detector and descriptor, which in comparison to the state-of-the-art are faster to compute, while not sacrificing performance.**

Key Contribution of the Paper:
1. Presented a fast and performant **interest point detection-description scheme** that outperforms the current state-of-the-art, both in speed (about **3 times faster**) and **accuracy**.
2. In our experiments on benchmark image sets as well as on a real object recognition application, the resulting **detector and descriptor are not only faster but also more distinctive and equally repeatable.**

Implementation:
(1) Select Interest Points => 'interest points' are selected at distinctive locations in the image, such as corners, blobs, and T-junctions. There are some methods to determine whether the point is an interesting point or not and to increase the computing speed.

(a) Instead of using Hessian Matrix to obtain the Gaussian second-order partial derivatives in the x-direction, y-direction, and XY-direction, **they use box filters to approximate similar values and highly decrease the computational time.** The result showed that the accuracy was similar but increased a huge amount at the computational speed. Furthermore, the filter responses are normalized concerning the mask size. This guarantees a constant Frobenius norm for any filter size.

(b) Traditionally, scale spaces are usually implemented as image pyramids. The images are repeatedly smoothed with a Gaussian and subsequently sub-sampled to achieve a higher level of the pyramid. **Due to the use of box filters and integral images, they don't iteratively apply the same filter to the output of a previously filtered layer, but instead can apply such filters of any size at the same speed directly on the original image**, and even in parallel (although the latter is not exploited here). **Therefore, the scale space is analyzed by upscaling the filter size rather than iteratively reducing the image size.** This guarantees to **increase the speed** of the operations.

(2) Find out the descriptors of the interest points => The proposed SURF descriptor is based on similar properties, with a complexity stripped down even further.
(a) Consists of fixing a reproducible orientation based on information from a circular region around the interesting point.
(b) Construct a square region aligned to the selected orientation and extract the SURF descriptor form it.
Note: They purposed an upright version of our descriptor **(U-SURF)** that **is not invariant to image rotation and therefore faster to compute and better suited for applications where the camera remains horizontal.**

Conclusion:
SURF (Speeded Up Robust Features) is a fast and performant interest point detection-description scheme that outperforms the current state-of-the-art, both in speed and accuracy. The descriptor is featured with repeatability, distinctiveness, and robustness, which is easily extendable for the description of affine invariant regions. The paper achieves such results by utilizing these three main points:
1. Relying on integral images for image convolutions
2. Building on the strengths of the leading existing detectors and descriptors
3. Simplifying these methods to the essential

2. Explain how your method conceptually compares with other algorithms mentioned above.

⇨ I compared the difference between SIFT, SURF, and ORB by running on the reduced dataset in the previous section (there is a table stating the difference between accuracy, time, and threshold, **please refer to question 2.1 above**).

3. Clearly and cogently document your methods and results.

The main code between this assignment and the last assignment is the same. I reused most of the code and would only specify the difference in my implementation below.

Apart from comparing different algorithms (SIFT, SURF, ORB), I also set different N clusters to the K-means algorithm on the expanded dataset (25 classes) and compare the results (please refer to the table below).

N Clusters	Average Accuracy	Time
1000	37.27%	14 min 53 secs
3000	36.79%	19 min 6 secs
6000	37.69%	21 min 25 secs
8000	36.72%	41 min 24 secs

According to the table above, I think the **average accuracy of different N clusters performs similarly**. Since increasing the cluster number will also enlarge the overall computational time, I will use 1000 clusters for K-means clustering.

In addition, I've tested different max iterations for kmeans on the 12-classes dataset and compared the accuracies (clusters = 4000, median threshold strategy).

Max Iterations	Average Accuracy	Converged?
9000	41.07%	No
12000	48.06%	No
16000	46.31%	No
24000	43.67%	No

According to the table above, even though I've already enlarged the max iterations to 24000, the kmeans model still cannot converge within those iterations. Moreover, increasing the maximum iterations on the kmeans algorithm seems to no help to the average accuracy.

4. **Include and interpret a (12 x 12) and a (25 x 25) confusion matrix (for your results on the classes with IDs 006-045 and the full dataset). If you were unable to use the full 25 classes, provide your results for the largest dataset used. In either case, comment on the issues you faced handling larger datasets.**

⇨ Result of 3-classes (SURF, clusters = 1000, city-block (L1) distance matric, 9000 max iterations, median threshold strategy)

-------------------------- Result --------------------------

Classes	Hat	Butterfly	Airplane
Hat	90.00%	10.00%	0.00%
Butterfly	0.00%	100.00%	0.00%
Airplane	0.00%	0.00%	100.00%

Best Sum: 290.00, Best Accuracy: 96.67, Best Threshold: 0.97

Elapsed time is 72.476117 seconds.

Result of 12-classes (ID: 006~045) (SURF, clusters = 4000, cityblock (L1) distance matric, 12000 max iterations median threshold strategy)

Best Sum: 576.67, Best Threshold: 1.00

Class ID	006	007	011	012	022	024	025	026	028	031	037	045
006	66.67%	0.00%	0.00%	0.00%	8.33%	8.33%	8.33%	0.00%	8.33%	0.00%	0.00%	0.00%
007	0.00%	14.29%	7.14%	0.00%	7.14%	14.29%	7.14%	35.71%	14.29%	0.00%	0.00%	0.00%
011	10.00%	0.00%	30.00%	0.00%	20.00%	10.00%	0.00%	10.00%	10.00%	10.00%	0.00%	0.00%
012	0.00%	0.00%	10.00%	90.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
022	0.00%	0.00%	0.00%	7.14%	64.29%	14.29%	7.14%	7.14%	0.00%	0.00%	0.00%	0.00%
024	0.00%	0.00%	0.00%	0.00%	0.00%	70.00%	20.00%	10.00%	0.00%	0.00%	0.00%	0.00%

025	10.00%	0.00%	0.00%	0.00%	10.00%	20.00%	50.00%	0.00%	10.00%	0.00%	0.00%	0.00%
026	0.00%	0.00%	0.00%	0.00%	10.00%	30.00%	0.00%	60.00%	0.00%	0.00%	0.00%	0.00%
028	10.00%	0.00%	0.00%	10.00%	20.00%	0.00%	10.00%	10.00%	40.00%	0.00%	0.00%	0.00%
031	26.67%	0.00%	6.67%	0.00%	20.00%	0.00%	20.00%	0.00%	6.67%	20.00%	0.00%	0.00%
037	0.00%	0.00%	0.00%	0.00%	0.00%	21.43%	42.86%	0.00%	7.14%	0.00%	28.57%	0.00%
045	7.14%	0.00%	0.00%	14.29%	14.29%	14.29%	7.14%	0.00%	0.00%	0.00%	0.00%	42.86%

Sum: 576.67
Average: 48.06%
Elapsed time is 530.434 seconds

Result of 25-classes (SURF, clusters = 1000, median threshold strategy)

Class ID	006	007	011	012	022	024	025	026	028	031	037	045	051	054	063	072	093	102	129	145	159	171	178	180	251
006	8.33%	0.00%	0.00%	0.00%	0.00%	0.00%	8.33%	8.33%	8.33%	0.00%	0.00%	33.33%	0.00%	0.00%	0.00%	0.00%	8.33%	8.33%	0.00%	0.00%	0.00%	16.67%	0.00%	0.00%	0.00%
007	0.00%	21.43%	7.14%	0.00%	0.00%	0.00%	7.14%	14.29%	14.29%	0.00%	0.00%	0.00%	7.14%	0.00%	0.00%	7.14%	21.43%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
011	0.00%	0.00%	20.00%	0.00%	10.00%	0.00%	0.00%	10.00%	10.00%	10.00%	0.00%	0.00%	0.00%	0.00%	0.00%	20.00%	0.00%	10.00%	0.00%	0.00%	0.00%	10.00%	0.00%	0.00%	0.00%
012	0.00%	0.00%	10.00%	80.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	10.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
022	0.00%	0.00%	0.00%	7.14%	35.71%	0.00%	0.00%	14.29%	7.14%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	14.29%	0.00%	7.14%	0.00%	7.14%	0.00%	0.00%	0.00%	7.14%
024	0.00%	0.00%	0.00%	0.00%	0.00%	50.00%	10.00%	20.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	20.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
025	10.00%	0.00%	0.00%	0.00%	10.00%	0.00%	40.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	10.00%	10.00%	0.00%	10.00%	0.00%	10.00%	0.00%	0.00%	0.00%	0.00%
026	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	50.00%	0.00%	0.00%	0.00%	0.00%	0.00%	10.00%	0.00%	10.00%	0.00%	0.00%	0.00%	0.00%	20.00%	10.00%	0.00%	0.00%	0.00%
028	10.00%	0.00%	0.00%	20.00%	0.00%	0.00%	0.00%	10.00%	30.00%	0.00%	0.00%	0.00%	0.00%	10.00%	0.00%	0.00%	10.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	10.00%
031	0.00%	0.00%	0.00%	6.67%	6.67%	0.00%	13.33%	0.00%	6.67%	6.67%	0.00%	6.67%	6.67%	0.00%	13.33%	6.67%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	20.00%	6.67%	0.00%
037	0.00%	0.00%	0.00%	0.00%	0.00%	21.43%	28.57%	0.00%	0.00%	0.00%	21.43%	0.00%	0.00%	21.43%	0.00%	0.00%	0.00%	7.14%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
045	0.00%	0.00%	0.00%	0.00%	0.00%	7.14%	0.00%	0.00%	7.14%	0.00%	0.00%	42.86%	14.29%	0.00%	0.00%	7.14%	0.00%	0.00%	0.00%	7.14%	0.00%	7.14%	7.14%	0.00%	0.00%
051	10.00%	0.00%	0.00%	30.00%	0.00%	0.00%	0.00%	10.00%	10.00%	0.00%	0.00%	0.00%	10.00%	0.00%	0.00%	0.00%	10.00%	10.00%	0.00%	0.00%	0.00%	10.00%	0.00%	0.00%	0.00%
054	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	10.00%	0.00%	0.00%	0.00%	60.00%	0.00%	10.00%	0.00%	0.00%	0.00%	0.00%	10.00%	0.00%	0.00%	0.00%	10.00%
063	10.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	10.00%	20.00%	0.00%	0.00%	10.00%	0.00%	0.00%	40.00%	0.00%	0.00%	10.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
072	0.00%	0.00%	0.00%	0.00%	10.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	50.00%	10.00%	0.00%	10.00%	0.00%	0.00%	0.00%	20.00%	0.00%	0.00%
093	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	10.00%	10.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	50.00%	0.00%	10.00%	0.00%	0.00%	0.00%	0.00%	0.00%	20.00%
102	20.00%	10.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	10.00%	0.00%	0.00%	0.00%	0.00%	30.00%	0.00%	0.00%	0.00%	20.00%	0.00%	0.00%	10.00%
129	0.00%	7.69%	0.00%	0.00%	0.00%	0.00%	30.77%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	7.69%	0.00%	0.00%	0.00%	0.00%	38.46%	0.00%	15.38%	0.00%	0.00%	0.00%	0.00%
145	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	40.00%	0.00%	0.00%	0.00%	60.00%	0.00%	0.00%	0.00%	0.00%	0.00%
159	10.00%	0.00%	10.00%	0.00%	10.00%	30.00%	10.00%	0.00%	10.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	10.00%	0.00%	0.00%	0.00%	10.00%
171	12.50%	0.00%	6.25%	0.00%	0.00%	0.00%	6.25%	6.25%	6.25%	6.25%	0.00%	0.00%	6.25%	18.75%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	18.75%	6.25%	0.00%	6.25%
178	0.00%	0.00%	0.00%	10.00%	10.00%	0.00%	10.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	20.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	50.00%	0.00%	0.00%
180	7.14%	0.00%	7.14%	7.14%	7.14%	0.00%	0.00%	7.14%	0.00%	0.00%	0.00%	0.00%	14.29%	0.00%	7.14%	7.14%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	14.29%	14.29%	7.14%
251	0.00%	0.00%	0.00%	6.25%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	93.75%

Sum of Accuracy: 931.68%
Average Accuracy: 37.27%

2.3 Competition

Best Results of All Different Classes

Classes	Sum of Accuracy	Average Accuracy	Threshold	Time

3	290	96.67%	0.97	1 min 12 secs
12	576.67	48.06%	1	6 min 50 secs
25	942.25	37.69%	1	21 min 25 secs

3 Grad Credits: Support Vector Machines for Image Classification

3.1 Reading

Support Vector Machines (SVMs) are discriminative image classifiers just like the KNNs you have implemented in this assignment. Unlike KNNs, which do not have any training time, SVMs need to first train a classifier (function) on the input data. However, once trained, SVMs typically have smaller run times compared KNNs. SVMs are also robust to noise and outliers in the data and have better generalization (less error) compared to KNNs.

For Grad credits, you should read the attached paper (Sections 2.4, and 2.5 are not required) and write a coherent summary of your understanding of the same. The paper has a decent amount of math but tries not to include equations in the write-up unless you think it is absolutely necessary. **Your summary should include a higher-level overview of what SVMs are learning from the data, and how they handle noise, outliers, and non-linearities.** You should also comment on **how SVMs are used by the authors of this paper for image classification, and how they handled multi-class and dimensionality**. Finally, based on your learnings in this assignment, what would you do differently compared to this paper.

Paper Cited: https://www.yumpu.com/en/document/read/10500330/support-vector-machines-for-image-classification-of-olivier-chapelle

Introduction

Given a set of points that belong to either of two classes, **a linear SVM finds the hyperplane leaving the largest possible fraction of points of the same class on the same side** while **maximizing the distance of either class from the hyperplane.**

Instead of doing feature extraction and performing recognition on images regarded as points of a space of high dimension, the potential of the SVM is illustrated on a 3D object recognition task using the Coil database and on an image classification task using the Corel database. **The images are either represented by a matrix of their pixel values (bitmap representation) or by a color histogram**.

Among the separating hyperplanes, the one for which the distance to the closest point is maximal is called the **optimal separating hyperplane (OSH)**. SVM is trying to maximize the margin between two closest different labeling points. The saddle point of the Lagrange function should be found to minimize the weight (we are trying to minimize ½ * w^2) and give better generalization. **The closest points to the optimal hyperplane are called support vectors. The support vectors are the only points needed in the expression of the OSH.**

Key Points

(1) **How do they handle noise, outliers, and non-linearities?**

- If the data are not linearly separable, the problem of finding the OSH becomes meaningless. **The idea of handling this problem is to map the input data into a high-dimensional feature** space through some nonlinear mapping (kernel tricks).
-

(2) **How did the author handle multi-class and dimensionality?**

Since Support Vector Machines are designed for binary classification. When dealing with several classes, as in object recognition and image classification, one needs an appropriate method.

- Modify the design of the SVM to incorporate multi-class learning directly in quadratic solving.
- Combine several binary classifiers (one against one or one against the others)

The author eventually used "**one against the others"** since it has **the lowest complexity**. In the "one against the others" algorithm, n hyperplanes are constructed, where n is the number of classes. Each hyperplane separates one class from the other classes. The class of a new point x is given by the class with the largest output of the decision function.

The author resized the size of all images to h for the height and w for the width and represent them as bitmap representations. This will help increase the efficiency of the SVM prediction since the objects in their images are centered. **The author also used 3-dimensional histograms to represent the color information (HSV-based since it is less sensitive to illumination changes)** because the color histogram technique is a very simple and low-level method to consider the introspect features of classes. **The author further stated that the prediction could improve by considering the derivatives of the images. Therefore, the author computed the horizontal and vertical derivatives of the image and construct for each one a similar histogram to the one used for the pixel values**. In this way, the input data for the SVM is not one histogram anymore, but three.

One of the main issues in image learning is the curse of dimensionality, especially for non-linear SVM. If the number of training examples is not big enough in front of the number of dimensions of the input data, the learning machine tends to learn by heart and has difficulty generalizing. Therefore, **the authors use Principal Component Analysis (PCA) to perform a dimensionality reduction.** This "black box" has been implemented straightforwardly on the various input data. According to choosing different kernels, there are four kinds of kernels presented: Linear, Polynomial, Radial Basis Function (RBF), and Neural Network. **The authors chose the Radial Basis Function (RBF) kernel to perform SVM for image classification** since it provides good performance on their input bitmap images.

Based on the paper, I will implement my SVM as the procedure shown below:

1. Collected all training images' paths.
2. Resize all training images to the same height and width and store them based on their category (since function imageDatastore(), which is a required input for bagOfFeatures(), can only read paths).
3. Use the bag of features with 1000 vocabulary size to find the bag of visual words instead of using transition histograms.
4. Use polynomial kernel and Sequential Minimal Optimization (SMO) as the optimization route to perform SVM for image classification.
5. Collected all testing images' paths.
6. Resize all testing images to the same height and width and store them based on their category (since function imageDatastore(), which is a required input for bagOfFeatures(), can only read paths).
7. Test the SVM image classification model with the reduced test dataset
8. Print the confusion matrix.

3.2 Train SVM on 3 class datasets

Train an SVM on 3 class datasets. You can use MATLAB's inbuilt functions or other packages available online. Cite them appropriately though. You can choose your feature representation, kernel, and the procedure used to extend SVMs for multiple classes. However, explain the reasoning behind the choices. Provide a technical writeup similar to the one in the previous questions and also provide a confusion matrix.

(1) **Resize the training and testing images**

⇨ Since most of the Machine Learning techniques normalized the size of the images into the same shape (and the SVM paper also implemented the normalization procedure), I perform normalization on the training and testing data.

(2) **Using polynomial as the SVM kernel function**

⇨ Based on the MATLAB document, ‘gaussian’ and ‘RBF’ kernel functions are used for one-class learning, ‘linear’ kernel function is used for two-class learning, and ‘polynomial’ kernel function is used for other different class orders of learning. That is, since we are classifying three different classes in total, I chose the polynomial function as my SVM kernel function.

(3) **Using Sequential Minimal Optimization (SMO) as the optimization route in SVM**

⇨ Sequential Minimal Optimization (SMO) is an algorithm for solving the quadratic programming (QP) problem that arises during the training of support vector machines (SVM). I use this instead of the default Iterative Single Data Algorithm (ISDA) simply because ISDA is for two-class learning so I change the optimization function to SMO. Moreover, SMO did give me some better results after some experiments.

(4) **Setting kernelScale to ‘auto’**

⇨ The default value of kernelScale is 1. Setting kernelScale to auto enables the software to select an appropriate scale factor using a heuristic procedure. This heuristic procedure uses subsampling, so estimates can vary from one call to another. Therefore, to reproduce results, set a random number seed before training.

3.3 Competition

Following the same procedure stated in the first question and report the accuracy of your classifier. This part of the grade is based on your classifier’s performance compared to the classifiers trained by your peers in the class.

Note: Failure to report this number will automatically award you zero points and reporting wrong numbers is against the honor code.

	PREDICTED		
KNOWN	024	051	251
024	1.00	0.00	0.00
051	0.20	0.80	0.00
251	0.00	0.06	0.94

* Average Accuracy is 0.91.

4 Conclusion

(1) **Issue**

The biggest issue I encountered was refactoring my code for the expanded dataset since the format of reading the inputs in this assignment and the previous one is different. There is another issue like the training time is too slow; however, I think this will not be my major concern since the result could still be displayed after waiting. This will only affect my debugging process since I have to wait for a long time. Therefore, I implemented save inside my MATLAB code to help debug my code so that I will not have to run the previous loading process again each time I modified my code.

(2) **Surprise**

I think this assignment is like an extension of the previous assignment. I learned more MATLAB syntaxes after this assignment and realize the importance of saving variables in MATLAB.