1-1-2015

Unsupervised Learning And Image Classification In High Performance Computing Cluster

Itauma Itauma
Wayne State University,

Unsupervised Learning and Image Classification in High Performance Computing Cluster

by

ITAUMA ITAUMA

THESIS

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the

requirements for the degree of

Master of Science

2015

MAJOR: COMPUTER SCIENCE

Approved By:

_____

Advisor Date

DEDICATION

To the memory of my MOTHER Idongesit Affiah, to my darling wife Omosalewa Itauma and to my loving daughter Princess Lily Itauma

ACKNOWLEDGEMENTS

I would not have been able to complete this work without the encouragement and support of many people. My first appreciation goes to my advisor, Dr. Xuewen Chen, for his direction, and invaluable advice during this study.

I would like to thank the members of the Big Data lab for their helpful and constructive feedback over the course of this study. I would like to specially thank Dr. Melih Aslan for his constant check in providing me with guidance and assistance.

I would like to acknowledge the support from National Science Foundation awards OIA-1028098.

Also, this thesis would not have been completed without the support from my wife Dr. Omosalewa Itauma and daughter Miss Lily Itauma.

And finally, I extend my special and profound thanks to other members of my family for their love and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1: Introduction

1.1 Objective

The purpose of this study is the efficient implementation of unsupervised learning and classification algorithms in High Performance Computing Cluster (HPCC) Systems. HPCC is a massively parallel processing computing platform used for solving Big Data problems. A multi-node system leverages the full power of massively parallel processing (MPP). While the single-node system is fully functional, it does not take advantage of the true power of an HPCC which has the ability to perform operations using MPP. Algorithms are implemented in HPCC with a language called Enterprise Control Language (ECL). ECL compiler generates highly optimized C++ for execution.

This study proposes a new idea to use the HPCC platform that i) can lower the execution time of the unsupervised learning process if the required hardware system is equipped, ii) can handle identity recognition using image and speech data, and iii) lowers the budget cost by using existing computers instead of designing an expensive system with GPUs.

In addition to the advantages described above, this proposed method has several contributions:

- A novel facial representation using multimodal learning strategy. By dividing a face image into several subunits, intra-class and intra-region variance can be effectively dealt with.

- Several classification methods are investigated and compared. The C4.5 decision tree is found to be superior than other alternatives in the most of the cases. Also,

the C4.5 decision tree is less variant to the parameter and structure selection than some of the classification methods such as deep neural networks.

1.2 Background

Feature learning and object classification have been very active research areas in machine learning in the past few decades. There are two major stages in this study:

- feature representation from face images and sound data and
- classification over the representations.

The performance of classification depends on the extracted features, classification models, and their corresponding parameters.

In addition to these stages and factors, there is a big demand for new ideas to deal with the feature learning and classification stages on high dimensional data. The high dimensionality and the sheer size of unlabeled data available today demand new developments in learning methods. In spite of recent advances in representation learning, most of the current methods are limited when dealing with large scale unlabeled data. Complex deep architecture and expensive training time are mostly responsible for lack of good feature representations for large scale data. In some of the cases, researchers reduce the sizes of data sets and models in order to train networks in a practical amount of time. However, these reductions undermine the learning of high-level features. As another solution to deal with high dimensional data, various researchers in the machine learning community have adopted the use of GPUs and parallel programming techniques to speed up computationally intensive algorithms. Furthermore, important studies have been carried out to propose more efficient optimization methods to speed up the convergency (such as Bristow et al. [2013]). In

addition to these various ideas and platforms, the goal of this study is to investigate a new environment (High Performance Computing Cluster) to assess the proposed framework's effectiveness in terms of computation time and classification accuracy.

The use of HPCC Systems is proposed because it enables researchers to leverage a multi-cluster environment to speed up the running time of any computationally intensive algorithm. It is open source and easy to setup. Figure 1.1 shows an HPCC System multi-cluster setup. The figure shows a THOR processing cluster which is similar to Google and Hadoop MapReduce platforms with respect to its function, filesystem, execution, and capabilities but offers higher performance Middleton [2011].

HPCC systems is also preferred because of its scalability in respect to code reuse irrespective of the size of dataset and number of clusters. It provides programming abstraction and parallel runtime to hide complexities of fault tolerance and data parallelism. In addition, machine learning algorithms can be implemented in HPCC which already consists of several unsupervised and supervised learning algorithms.

The scope of this study is to show that the HPCC system is able to run image classification problems even using a single core computer. A faster training time is expected if the algorithms are tested on a multinode HPCC cluster. The use of a system combining multiple computers is left for future studies.

1.2.1 HPCC Systems Platform

HPCC moves the algorithm to the data. The platform optimizes codes for efficient parallelizations and minimization of data exchange. ECL which is the core language used by HPCC systems is designed to expressively code complex data manipulation problems.

There are three main components as shown in Figure 1.2:

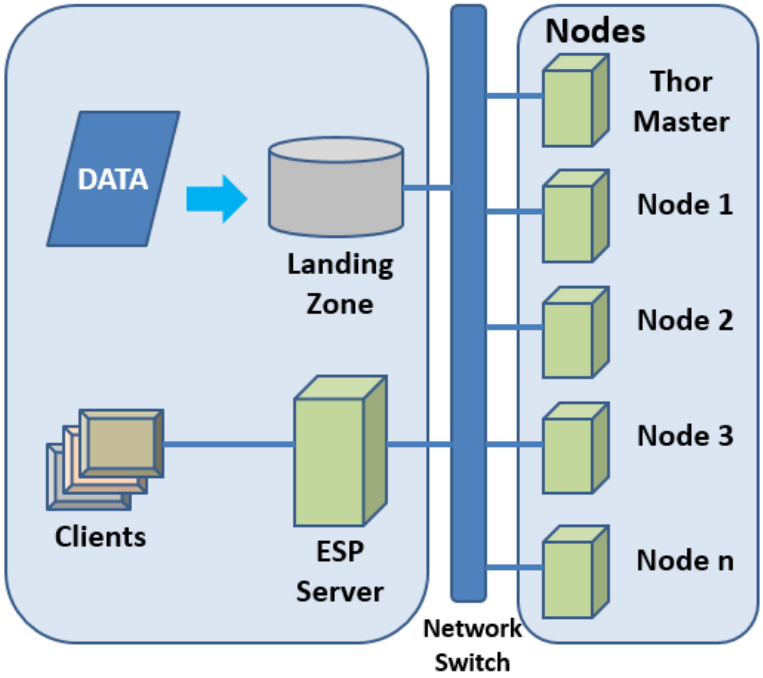


Figure 1.1: *HPCC* THOR Cluster.

- HPCC Data Refinery (THOR)
- HPCC Rapid Data Delivery Engine (Roxie) and
- Enterprise Control Language (ECL)

The THOR cluster as shown in Figure 1.1 performs the ETL (Extract, Transform, Load) of Big Data processing. Extract involves importing and cleaning raw data from multiple sources. Transform involves combining and collating information from multiple sources. Load involves producing the indexes for data delivery to be used on Roxie. The Roxie cluster runs search on massive datasets.

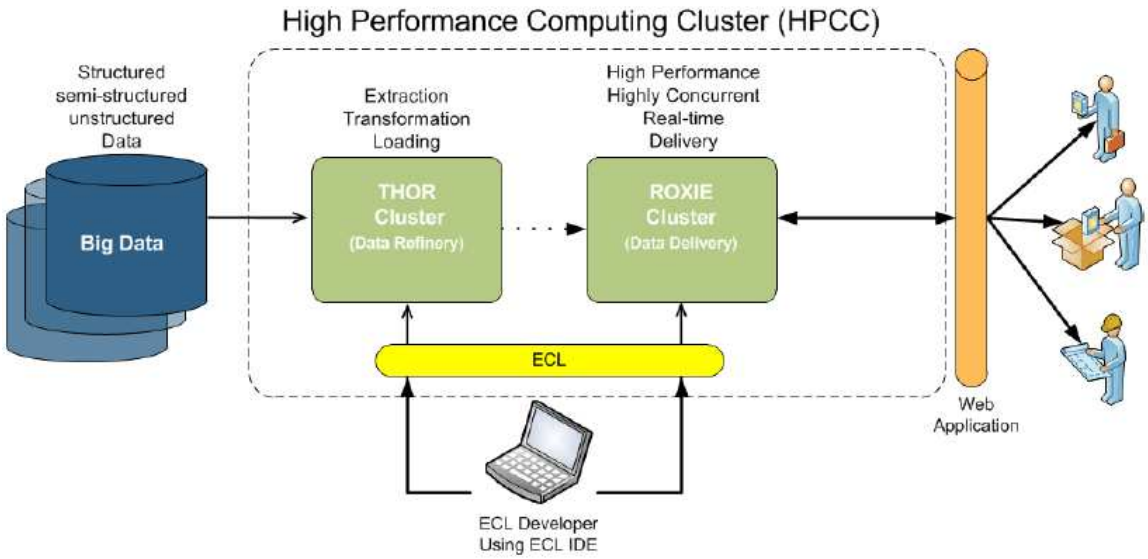


Figure 1.2: *HPCC* Systems Platform Middleton [2011].

1.3 Contributions of this Thesis

This thesis presents the use of HPCC Systems for unsupervised feature learning from large datasets and object classification. K-Means in HPCC is compared with the method in Coates et al. [2011]. These methods are unsupervised learning algorithms to learn features from unlabeled data as shown in Chapter 1.3. Various feature extraction processes for each data and classification methods are followed. The proposed outcome of this step is that once good features are learnt, a supervised learning classification algorithm should perform well as described in Chapter 2.3.

Other contributions are:

- A proposed new feature learning and recognition framework using a multimodal strategy. The new idea is to use the HPCC platform to handle identity recognition using multimedia data (i.e., image and speech information) in a multimodal framework with high recognition accuracy rates. For instance, by dividing a

face image into several subunits, we can extract intra-region information more precisely.

- The use of HPCC systems in the implementation of the feature learning and object classification tasks is proposed.

HPCC systems enables researchers to leverage a multi-cluster environment to speed up the running time of any computationally intensive algorithm. It lowers the budget cost by using existing computers instead of designing an expensive system with GPUs. Also, it is scalable in respect to code reuse irrespective of the size of dataset and number of clusters.

Chapter 2: Methods

The project framework consists of image reading in HPCC platform, feature learning from unlabeled data, feature extraction from labeled data using the learnt coefficients, and classification stages. The implementation is carried out based on the following stages:

- Generating random patches of image data.
- Reading the image dataset into HPCC systems cluster.
- Learning feature representation using K-Means clustering algorithm from unlabeled input dataset.
- Performing feature extraction by projecting the input dataset with respect to the feature representation resulting in a low dimension dataset.
- Training various classifiers on the low dimensional training dataset.
- Using a testing dataset to predict the labels based on the model generated above.

Details for each stage is given in the following sections.

2.1 Image Reading In HPCC

This is the first study on image classification using HPCC systems, to the best of my knowledge. First, the databases are integrated into the HPCC system. In HPCC, images are represented as Binary Large OBject (BLOB). BLOB support in ECL begins with the DATA value type which makes it perfect for housing BLOB data.

There are essentially three issues around working with BLOB data:

- How to get the data into the HPCC THOR Cluster (Spraying).
- How to work with the data, once it is in HPCC.
- How to get the data back out of HPCC THOR Cluster (Despraying).

The BLOB spray is described in ECL [2015] and HPC [2014]. The image dataset should be sprayed in BLOB format. There are different formats for spraying data such as delimited for CSV, fixed for texts and blob for images. The BLOB spray option is used which will result in a dataset on the cluster where each record is one of the image datasets. Typically, a prefix of both the name and length is used to define the record structure of the image dataset. The record would look like:

```
ImageRecord := RECORD
STRING fileName;
DATA image;
END;
```

The data string "image" is the file content. The first 4 bytes contain the length of the image data. All .jpg (or any image format) files in the directory are sprayed to a single logical file such as (BIGDATA::II::IMAGE). After the data is sprayed into the HPCC THOR cluster, the RECORD structure and DATASET are defined in order to read it in ECL. The following RECORD structure defines the result of the sprayed data.

```
imageRecord := RECORD
STRING filename;
DATA image;
END;
```

```
imageData := DATASET('~BIGDATA::II::IMAGE',
    imageRecord,FLAT);
```

Since only grayscale images are used, all grayscale images are converted to Comma Separable Value (CSV). A CSV is used to store gray scale images since a gray scale image can be treated as a data set of pixel values. Each row in the CSV contains the pixel values of each image respectively.

The following steps are followed to use the image database:

- Extract patches from images.
- Normalize the patches.
- Convert these patches to CSV.
- Spray the CSV to HPCC.

2.2 Feature Learning Representation

Most applications in image processing involve the use of high dimensional data. The features that represent our data is a lower dimension in a high dimensional space. The goal of unsupervised learning is to find a lower dimensional projection of the unlabeled data that preserves all the information in the data while reducing redundant dimension. The problem in unsupervised learning is trying to find hidden structures in unlabeled data.

After reading the image information in ECL, the feature learning and classification algorithms are run respectively. In this work, K-Means implemented in ECL and MATLAB is applied for feature learning. Coates et al. [2011] proved that the K-means method can achieve superior results than other possible unsupervised learning

methods. The algorithm takes the dataset X and outputs a function $f : R^n \rightarrow R^k$ that maps an input vector $x^{(i)}$ to a new feature vector of k features. To extract high quality features in order to obtain a high classification accuracy, the methods are run with respect to the key points of this stage such as using:

- a good number of samples,
- choice of parameters, and
- number of weights.

Section 2.2.1 gives a brief description of K-Means algorithms.

2.2.1 K-Means on ECL

K-Means is a partitioning algorithm in which various bases or centroids are constructed and evaluated based on specific criteria. It is an unsupervised clustering method and partitioning algorithm where data are assigned in clusters defined by their centroid, based on their features and distance from the centroids. The goal is to minimize the sum of the square errors (SSE) as can be seen in Eq. (2.1). The SSE is used to make partitions. It is the sum of squared differences between each observation in its cluster as a centroid over all the k clusters. If a data point is on a centroid then the SSE will be equal to zero. The SSE strictly decreases after recomputing new centers in the K-Means algorithm. The new center of a cluster comes from the average of all data points in this cluster, which minimizes the SSE kme; as follows:

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - m_i\|^2, \tag{2.1}$$

where m_i is the mean of points in C_i and x is the data point in cluster C_i. Given two partitions, the one with the smallest error is chosen. Each cluster is represented

Figure 2.1: K-Means Clustering Algorithm

Select k points as initial centroids
repeat
  Form k clusters by assigning each point to its closest centroid
  Re-compute the centroids of each cluster
until convergence criterion is satisfied

by the center of the cluster which is the centroid. Points are assigned to the cluster with the nearest centroid. The distance between clusters is based on their centroids:

$$dis(K_i, K_j) = dis(C_i, C_j), \tag{2.2}$$

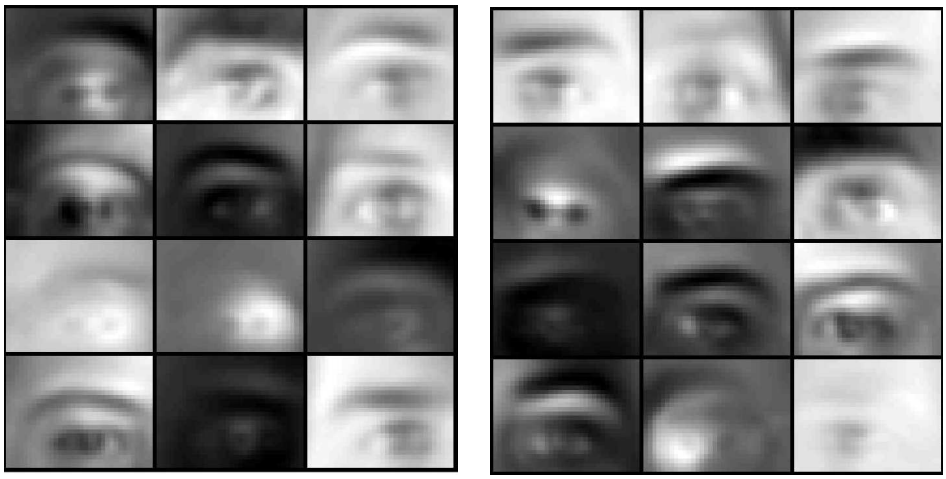
where K_i and K_j are two groups of points or information, and C_i and C_j are the corresponding centroids. Given k, the number of clusters, the K-Means clustering algorithm is outlined as in Figure 2.1.

In order to specify the best k, a range of values are run. The computational complexity is $\mathcal{O}(tkn)$ where n is the number of data points, k is the number of clusters and t is the number of iterations. It is an efficient method since usually, $k, t << n$.

The work Coates [2012] summarizes recent results and technical points that are needed to make effective use of K-Means clustering for learning large-scale representations of images. Figure 2.2 shows the centroids learned by K-Means implemented in ECL from the AR dataset without whitening.

2.3 Classification

After learning the features from the unlabeled data, the dimension of the labeled data is reduced by a specially designed feature extraction procedure that is described for each data in Chapter 2.3.



"(AR Left eye Weight)" "(AR Right eye Weight)"

Figure 2.2: Selected bases (or centroids) trained on AR images using K-Means in HPCC.

Classification is a supervised learning process that aims at accurately predicting some value or attribute of an object based on known facts about the object. It involves deriving a rule or model from a training set which is then used to predict a test set. In machine learning, all classification algorithms follow three logical steps. Learning the model from a training set, testing with respect to obtaining measures of how well the classifier fits, and classifying which involves testing the model on new data in order to compute a classification accuracy. Several classification methods and configurations are used in these experiments that I will give more details in the next section.

Chapter 3: Experiments and Results

In this study, feature learning and classification algorithms are applied on a subset of Caltech-101 based on Fei-Fei et al. [2007], AR Martınez and Benavente [1998], and a subset of PubFig83 database Becker and Ortiz [2013] to which speech content has been added in addition to face images. The major contribution of this study is pioneering the exploration of image/identity classification in HPCC. Note that all images in these experiments are locally normalized to have the Gaussian distribution. For the classification, various classifiers such as Naive Bayes, Random Forest, and C4.5 Decision Tree are used.

The experiments are run on a single-node HPCC system platform. The single-node HPCC system virtual machine running Ubuntu 64-bit is setup as a guest machine on Windows. The host machine has a usable RAM size of 3.89GB of which 1.5GB is allocated for the guest machine.

3.1 Evaluation on Caltech-101

The Caltech-101 database consists of 102 categories. As a subset of Caltech-101, ten classes are used (which have more than 60 images per class) for both unsupervised and supervised learning steps. Up to 60 images per class are randomly selected and pre-processed as in Kavukcuoglu et al. [2009]: The images are converted to gray-scale, then down-sampled and zero padded to 143×143 pixels. Finally, the images are normalized to have the standard Gaussian distribution.

In this experiment, the performance of unsupervised and supervised learning methods are assessed in MATLAB and HPCC. Unsupervised learning methods are performed using $3,000$ randomly selected patches of 16×16 dimensional pixels. In the unsupervised learning part, the entire unlabeled training set of images is trained

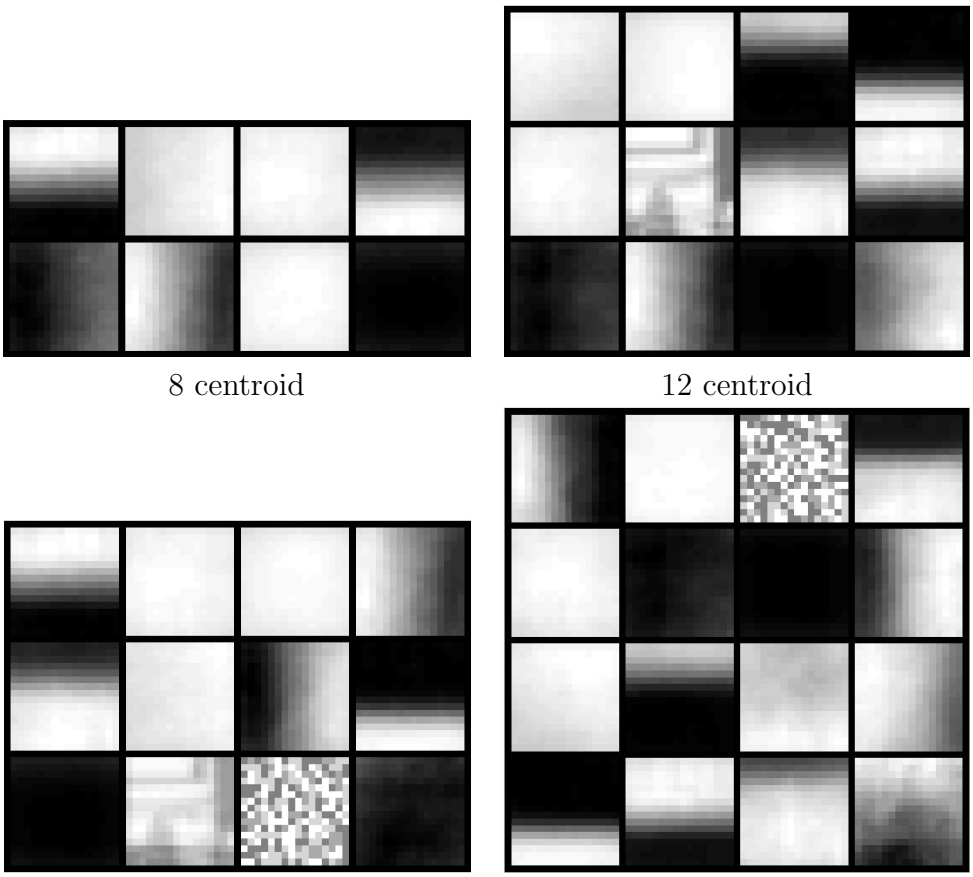


Figure 3.1: Selected bases (or centroids) trained on Caltech101 images using K-Means in HPCC

before the classification step. I learn 32 weights in the unsupervised learning of all methods in two platforms.

For the supervised learning, 30 training and 30 testing images are used for each category. To extract features from the labeled training samples, I follow the convolutional extraction process of Coates et al. [2011]. I use stride 1 with 16×16 patches to obtain a dense feature extraction. The non-linear mapping transforms the input patches into a new representation with 32 features using the learned weights. Then pooling is used for dimensionality reduction. 132 pooled features are used to train the classifiers.

Table 3.1: Precision results for the Caltech-101 database

	Acc. (%)
Methods	*32 Weight*
Methods	**Acc. (%)**
K-means (MATLAB) + Linear SVM (MATLAB)Coates et al. [2011]	80.7
K-means (HPCC) + Random Forest (HPCC)	81.33
K-means (HPCC) + Naive Bayes (HPCC)	82.0
K-means (HPCC) + C4.5 Decision Tree (HPCC)	**83.5**

The visualizations of the learned feature representations are shown. The bases (or centroids) learned by K-Means are shown in Figure 3.1. The result using this data configuration is reported in Table 3.1. The important point of this result is that methods in the HPCC platform are able to obtain comparative or better results than the methods in the MATLAB environment. Three classification methods in HPCC are compared with the same features obtained by the K-means learning. It is observed that the Decision tree method achieves the best results for this data.

3.2 Evaluation on AR Face Database

The classification quality is also measured on AR Martınez and Benavente [1998] face database. The aligned AR database contains 100 subjects (50 men and 50 women), with 26 different images per subject which totals $2,600$ images taken in two sessions. In this database, there are facial expression (neutral, smile, anger, scream), illumination and occlusion (sunglass, scarf) challenges. In this experiment, images without the occlusion challenges are used that totals to $1,400$ images for both the unsupervised learning and classification steps. Figure 3.2 shows some example images from a subject.

Figure 3.2: Example images from one subject in AR database with various facial expressions, illumination, and occlusion.

A framework is proposed that is shown in Fig. 3.3 for the face databases in the experiments. First, the weight for each facial region is learnt separately. Four essential facial regions are segmented with sizes of 39 x 51 (left eye and right eye), 30 x 60 (mouth), and 45 x 42 (nose). It is believed that better representations are obtained by running unsupervised learning for each region. I also obtain the features of the labeled facial regions using the corresponding learned weights separately. To do this, I calculate the correlation between each labeled sample and each center vector (weight) to get a vector of features. I then combine the features extracted from the four facial regions (and other possible modalities), and train the classifiers.

For the AR database, I follow a scenario described in Zang et al. [2012] which reported one of the state-of-the-art recognition rates. Each subject has 14 images with facial expression and illumination changes. Various train-test image partitions are tested. I conduct 10 runs for train-test procedure to get the average recognition rate for each partition.

Table 3.2 shows the face classification results obtained from MATLAB and HPCC. In this experiment, several configuration of unsupervised and supervised

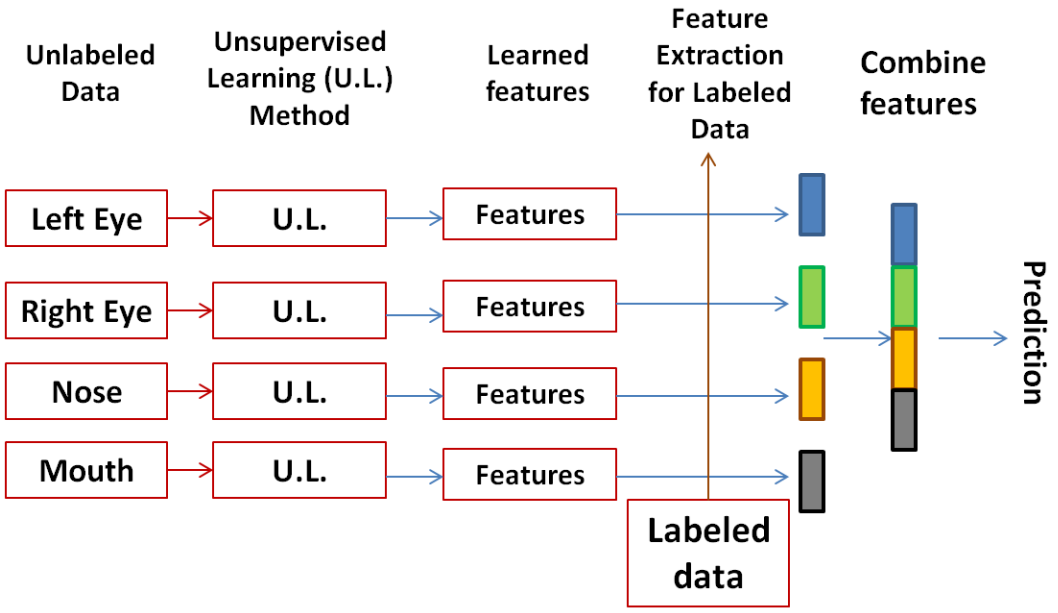


Figure 3.3: The framework that is followed for the classification of AR data.

Table 3.2: Comparison of face recognition rates on *AR* database.

Methods	**Acc. (%) with Train** *5 Train*
K-means (HPCC) + Linear SVM (MATLAB)	69.3
K-means (MATLAB) + Linear SVM (MATLAB)Coates et al. [2011]	74.3
K-means (HPCC) + C4.5 Decision tree (HPCC)	78.9
K-means (MATLAB) + C4.5 Decision tree (HPCC)	**85.2**

learning methods are compared in the HPCC and MATLAB enviromnents. From the table, one can observe that Coates et al. [2011] performs better in obtaining good features for classification but a higher classification accuracy is obtained by using decision tree C4.5 classifier in HPCC. I improve the classification results of Coates et al. [2011] between 4.6% and 10.9% using the Decision tree classification in ECL.

3.3 Identity Recognition on the Wild and Multimedia Database

In recent years, several unconstrained databases have emerged in the literature for face identification or verification. Unlike the traditional face databases which are composed of images taken in controlled environments, face images in unconstrained databases are generally collected from Internet sources. In particular, these images contain unrestricted varieties of expression, pose, lighting, occlusion, resolution, etc. Therefore, unconstrained face recognition is a very challenging task.

A data set is prepared from an aligned version of wild PubFig83 database Becker and Ortiz [2013]. I select 10 subjects which totals to $1,000$ face images. Some example images are shown in Figure 3.4. For the images, I randomly select 50 images per subject as the training set, and the rest of the images are used as the testing set in the supervised learning step. Four essential facial regions are used for facial representation learning. I segment four essential facial regions with sizes of 32 x 52 (left eye and right eye), 48 x 76 (mouth), and 60 x 48 (nose), which are further reduced by half using bicubic interpolation.

Table 3.3 shows the classification results across 10 runs using various classification configurations. K-means method in HPCC environment is used to obtain features for all classification in this experiment. One observes that the linear SVM in MATLAB obtains 71.6% classification rate. Random forest and softmax classifi-

cation methods in HPCC achieves 75.5% and 83.0% rates, respectively. As observed from the two databases reported above, the Decision tree achieves the best and much superior classification rate, 91.5%.

To assess the feasibility of the proposed framework on the multimedia data, several videos are downloaded for 10 subjects from YouTube to extract around 5 minute speech information. There have been research efforts to show that it is beneficial to leverage the knowledge from multimedia data Yang et al. [2013]. For instance, multimedia entertainment companies have started to offer information on cast and characters for movies and shows during playback, presumably via a combination of visual and sound contents Bauml et al. [2013]. Also, it is beneficial to borrow knowledge from some other related tasks for feature extraction.

The Mel frequency cepstral coefficients (MFCCs) Davis and Mermelstein [1980], and their first and second derivatives are employed to represent the acoustic features. Note that the content and quality of the speech data are heterogenous. These features are calculated every 10 milliseconds using 25 milliseconds Hamming-window[1], and their first 12 elements are selected to form a 36-dimensional feature vector for each frame. In the experiment, features extracted from every 40 consecutive frames are concatenated to be a 1440-dimensional feature vector which is considered as one training/test example. The unsupervised generic features are learned over 5000 examples (500 per subject). Half of the samples are randomly selected to train the classifiers and the rest is used for the testing.

In addition to the experiments on the visual content, Table 3.3 shows the identity recognition across 10 runs using speech only and multimedia representation with visual and speech contents. For these two cases, the Decision tree classification

[1]Dan Ellis' implementation for MFCC is used which is available at `http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/`

Figure 3.4: Example images of 10 celebrities with various real-world changes on facial expression, pose, illumination, occlusion, resolution, etc.

method is run. I achieved 91.6% recognition rate using only the speech information. When the visual and speech information together are combined in a multimodal way, a 94.0% recognition accuracy was achieved. By combining different information such as features of different facial regions and corresponding speech contents, the classification accuracy is increased by 2.5%.

Table 3.3: Identity recognition results using visual and/or speech contents on multimedia database. Note that K-means in the HPCC environment is used for all cases.

	Acc. (%)
Methods	*50 Train*
Only Visual Content	
Naive Bayes (HPCC)	55.5
Linear SVM (MATLAB)Coates et al. [2011]	71.6
Random Forest (HPCC) - maxLevel=10	74.2
Random Forest (HPCC) - maxLevel=15	75.5
Softmax Classification (HPCC)	83.0
C4.5 Decision tree (HPCC) - maxLevel=25	**91.5**
Only Speech Content	
C4.5 Decision tree (HPCC)	**91.6**
Multimedia (Visual + Speech)	
C4.5 Decision tree (HPCC)	**94.0**

Chapter 4: Discussion

Image classification in HPCC has been successfully performed in this study. It is observed that as the depth of the C4.5 decision tree algorithm increases, one obtains a higher classification accuracy. Hence, the deeper the tree, the more complex the decision rules and the fitter the model.

Furthermore, clustering can be used to help with classification. K-Means can be used for engineering complex features. A potential application of this project would be in the area of medical imaging. It can be used in finding distinct features that could lead to improved diagnostic accuracy. Considering medical databases, patients may have a unique real-value measure for certain tests such as glucose or cholesterol. Clustering patients first would help us understand how binning should be done on real-value features to improve accuracy on classification.

Since a single core computer was used on the HPCC platform, I had a slightly slower training time of the unsupervised learning algorithms compared to MATLAB on Windows. Therefore considerably smaller dimensional features were used, such as 16 for AR and 32 for Caltech-101 databases. A faster training time is absolutely expected when a multinode HPCC cluster is used. I leave the use of a system combining multiple computers for future studies. Further evaluations with higher dimensional features would be straightforward using an HPCC system with multiple CPUs.

In addition to visual information, the proposed multimodal learning framework is also capable of integrating other multimedia content, e.g., speech, by treating individual sources as a unique modality. Representations of multimedia data are learned in a similar way to face images and then fused together with face descriptors to feed to the identity recognition step.

Chapter 5: Conclusions and Future Work

In this study, I have presented an interesting and novel idea to run image classification problems in a relatively new platform (HPCC) that can lead to faster optimization/-calculation of algorithms and low cost of hardware designs. CALTECH-101, AR databases, and a subset of wild PubFig83 data with multimedia content, have been tested to extract features using K-Means, which is then used to improve the classification accuracy based on different classifiers in HPCC. This framework developed in ECL programming language in HPCC system is compared with MATLAB implementation on Windows system. It is observed that the results obtained using the HPCC platform are at least comparable with the outcome from MATLAB. A novel face recognition algorithm was proposed that can lead to further exploration of face recognition problems. The classification accuracy of AR database was increased to 85.2% when compared with the original method described in Coates et al. [2011]. The C4.5 decision tree classification results shows that K-Means selects good features. Also, this learning framework has been proven to leverage new representations that are learned over multimedia data automatically.

Future work will include integrating an improved method to select the feature or cluster number automatically to obtain a higher accuracy. I also plan to integrate multiple CPUs in the HPCC system to speed up the process, compete with GPU hardware and handle larger dimensional calculations

APPENDIX A: K-means Implementation in ECL

The code below shows K-means implementation in ECL for unsupervised feature learning.

```
IMPORT ML;
IMPORT $;

//Input Data of 750 unlabeled attributes

value_record := RECORD
REAL f1;
REAL f2;
REAL f3;
REAL f4;
REAL f5;
REAL f6;
REAL f7;
REAL f8;
REAL f9;
REAL f10;
REAL f11;
REAL f12;
// Other attributes deleted for space
REAL f739;
REAL f740;
REAL f741;
```

```
REAL f742;
REAL f743;
REAL f744;
REAL f745;
REAL f746;
REAL f747;
REAL f748;
REAL f749;
REAL f750;
END;

//Reading data from HPCC Cluster
input_data_tmp := DATASET('~thesis::ii::bigdata::
data_pubfig_speech1000x750samples', value_record, CSV);
ML.AppendID(input_data_tmp, id, input_data);
OUTPUT(input_data(id < 99), NAMED ('input_data'));

ML.ToField(input_data, indepDataC);
OUTPUT(indepDataC, NAMED('indepDataC'));

//Centroids Random Generation
minimum := MIN(indepDataC,indepDataC.value)+2;
maximum:= MAX(indepDataC,indepDataC.value)/2;
K := 16;
dCentroids := ML.Types.FromMatrix($.RandMatR(K,750,minimum,maximum));
```

```
OUTPUT(dCentroids, NAMED('dCentroids'));
/*
## First Task Experiment
- Experiment 1
 Sample #:            1000
 Iteration #:         50
 Centroid # :          12
Total Thor Time: 29:38.584
Convergence: 18 iteration out of 50

*/

//Kmeans Training
MyKMeans:=ML.Cluster.KMeans(indepDataC,dCentroids,50,.0003);

//Results
MyKMeans.AllResults;

MyKMeans.Convergence;

MyKMeans.Result();

//Despray extracted weight feature in CSV format
OUTPUT(MyKMeans.Result(),, '~online::ii::bigdata::
```

```
extractedkmeansweight_speech', OVERWRITE, CSV);
```

APPENDIX B: C4.5 Decision Tree Classification Method in ECL

```
//C4.5 Decision Tree Classification

IMPORT * FROM ML;
IMPORT * FROM $;
IMPORT PBblas;
Layout_Cell := PBblas.Types.Layout_Cell;

//Train and Test data layout
value_record := RECORD
REAL f1;
REAL f2;
REAL f3;
REAL f4;
REAL f5;
REAL f6;
REAL f7;
REAL f8;
REAL f9;
REAL f10;
REAL f11;
REAL f12;
REAL f13;
REAL f14;
```

```
REAL f15;
REAL f16;
REAL f17;
INTEGER Label;
END;

//reading train and test data
train_data_tmp := DATASET('~thesis::ii::bigdata::
supervised_speechdata_train', value_record, CSV);
test_data_tmp := DATASET('~thesis::ii::bigdata::
supervised_speechdata_test', value_record, CSV);
//Train
ML.AppendID(train_data_tmp, id, train_data);
OUTPUT (train_data, NAMED('train_data'));

//Test
ML.AppendID(test_data_tmp, id, test_data);
OUTPUT (train_data, NAMED('test_data'));

//convert train data to two datset:
//samples dataset and labels dataset
Traindata_Format := RECORD
train_data.id;
train_data.f1   ;
train_data.f2   ;
```

```
train_data.f3    ;
train_data.f4    ;
train_data.f5    ;
train_data.f6    ;
train_data.f7    ;
train_data.f8    ;
train_data.f9    ;
train_data.f10   ;
train_data.f11   ;
train_data.f12   ;
END;

train_table := TABLE(train_data,Traindata_Format);
OUTPUT  (train_table, NAMED ('train_table'));

labelTraindata_Format := RECORD
  train_data.id;
  train_data.label;
END;

trainlabel_table := TABLE(train_data,labelTraindata_Format);
OUTPUT  (trainlabel_table, NAMED ('trainlabel_table'));

ML.ToField(train_table, indepTrainDataC);
OUTPUT  (indepTrainDataC, NAMED ('indepTrainDataC'));
```

```
ML.ToField(trainlabel_table, depTrainDataC);
OUTPUT  (depTrainDataC, NAMED ('depTrainDataC'));
trainlabel := PROJECT(depTrainDataC,Types.DiscreteField);
OUTPUT  (trainlabel, NAMED ('Trainlabel'));

//convert test data to two datset:
//samples dataset and labels dataset
Testdata_Format := RECORD
test_data.id;
test_data.f1    ;
test_data.f2    ;
test_data.f3    ;
test_data.f4    ;
test_data.f5    ;
test_data.f6    ;
test_data.f7    ;
test_data.f8    ;
test_data.f9    ;
test_data.f10   ;
test_data.f11   ;
test_data.f12   ;
END;

test_table := TABLE(test_data,Testdata_Format);
OUTPUT  (test_table, NAMED ('test_table'));
```

```
labelTestdata_Format := RECORD
  test_data.id;
  test_data.label;
END;

testlabel_table := TABLE(test_data,labelTestdata_Format);
OUTPUT  (testlabel_table, NAMED ('testlabel_table'));

ML.ToField(test_table, indepTestDataC);
OUTPUT  (indepTestDataC, NAMED ('indepTestDataC'));
ML.ToField(testlabel_table, depTestDataC);
OUTPUT  (depTestDataC, NAMED ('depTestDataC'));
testlabel := PROJECT(depTestDataC,Types.DiscreteField);
OUTPUT  (testlabel, NAMED ('Testlabel'));

//Decision Tree Classification
//Learning Phase
MinNumObj := 2;
MaxLevel := 20;    //Decision Tree parameters
learner1 := ML.Classify.DecisionTree.C45binary(MinNumObj, MaxLevel);
dtModel := learner1.LearnC(indepTrainDataC, trainlabel);  //Decision Tree
//Classification Phase
results1 := learner1.ClassifyC(indepTestDataC,dtModel);
//Comparing to Original
```

```
compare1 := ML.Classify.Compare(testlabel, results1);
OUTPUT(compare1.CrossAssignments,ALL);  //CrossAssignment results


//Computing the accuracy of all the classes
//Append unique ID to crossAssignments
ML.AppendID(compare1.CrossAssignments, id, crossAssignments);
//A dataset of crossAssignments
OUTPUT(crossAssignments, NAMED('crossAssignments'));

diagVec := crossAssignments(c_actual = c_modeled);
//diagVec;
totalAccuratePrediction := SUM(diagVec,cnt);
totalSamples := SUM(crossAssignments,cnt);
accuracy := (totalAccuratePrediction / totalSamples)*100;
OUTPUT(accuracy, NAMED('Accuracy'));
```

REFERENCES

A. Middleton, "HPCC Systems: Introduction to HPCC (high performance computing cluster)," 2011.

H. Bristow, A. Eriksson, and S. Lucey, "Fast convolutional sparse coding," *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pp. 391–398, 2013.

A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," *International Conference on Artificial Intelligence and Statistics*, pp. 215–223, 2011.

"HPCC Systems: ECL Programmers Guide. Boca Raton Documentation Team," 2015.

"HPCC Systems: HPCC Client Tools. Boca Raton Documentation Team," 2014.

"K-Means Clustering Algorithm," *http://en.wikipedia.org/wiki/K-means_clustering Last accessed: 06.15.2015.*

A. Y. Coates, Adam; Ng, "Learning feature representations with k-means," *In G. Montavon, G. B. Orr, K.-R. Mller. Neural Networks: Tricks of the Trade. Springer*, 2012.

L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *Computer Vision and Image Understanding*, vol. 106, pp. 59–70, 2007.

A. Martınez and R. Benavente, "The AR Face Database," *Computer Vision Center, Technical Report*, vol. 24, 1998.

B. C. Becker and E. G. Ortiz, "Evaluating Open-Universe Face Identification on the Web," *Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshops*, pp. 904–911, 2013.

K. Kavukcuoglu, M. A. Ranzato, R. Fergus, and Y. Le-Cun, "Learning invariant features through topographic filter maps, Computer Vision and Pattern Recognition, 2009," *CVPR 2009. IEEE Conference on*, vol. 1, pp. 1605–1612, 2009.

F. Zang, J. Zhang, and J. Pan, "Face recognition using Elasticfaces," *Pattern Recognition*, vol. 45, pp. 3866–3876, 2012.

Y. Yang, Z. Ma, A. G. Hauptmann, and N. Sebe, "Feature selection for multimedia analysis by sharing information among multiple tasks," *IEEE Transactions on Multimedia*, vol. 15, pp. 661–669, 2013.

M. Bauml, M. Tapaswi, and R. Stiefelhagen, "Semi-supervised learning with constraints for person identification in multimedia data," *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 3602–3609, 2013.

S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoustics, Speech and Signal Processing*, pp. 357–366, 1980.

ABSTRACT

Unsupervised Learning and Image Classification in High Performance Computing Cluster

by

ITAUMA ITAUMA

August 2015

Advisor: Dr. Xuewen Chen

Major: Computer Science

Degree: Master of Science

Feature learning and object classification in machine learning have become very active research areas in recent decades. Identifying good features has various benefits for object classification in respect to reducing the computational cost and increasing the classification accuracy. In addition, many research studies have focused on the use of Graphics Processing Units (GPUs) to improve the training time for machine learning algorithms. In this study, the use of an alternative platform, called High Performance Computing Cluster (HPCC), to handle unsupervised feature learning, image and speech classification and improve the computational cost is proposed.

HPCC is a Big Data processing and massively parallel processing (MPP) computing platform used for solving Big Data problems. Algorithms are implemented in HPCC with a language called Enterprise Control Language (ECL) which is a declarative, data-centric programming language. It is a powerful, high-level, parallel programming language ideal for Big Data intensive applications.

In this study, various databases are explored, such as the CALTECH-101 and AR databases, and a subset of wild PubFig83 data to which multimedia content

is added. Unsupervised learning algorithms are applied to extract low-level image features from unlabeled data using HPCC. A new object identification framework that works in a multimodal learning and classification process is proposed.

Coates et al. [2011] discovered that K-Means clustering method out-performed various deep learning methods such as sparse autoencoder for image classification. K-Means implemented in HPCC with various classifiers is compared with Coates et al. [2011] classification results.

Detailed results on image classification in HPCC using Naive Bayes, Random Forest, and C4.5 Decision Tree are performed and presented. The highest recognition rates are achieved using C4.5 Decision Tree classifier in HPCC systems. For example, the classification accuracy result of Coates et al. [2011] is improved from 74.3% to 85.2% using C4.5 Decision Tree classifier in HPCC. It is observed that the deeper the decision tree, the fitter the model, resulting in a higher accuracy. The most important contribution of this study is the exploration of image classification problems in HPCC platform.

AUTOBIOGRAPHICAL STATEMENT

Itauma was born in Calabar, Nigeria. He has an undergraduate degree in Electrical Engineering from University of Ilorin. He also received an MSc. in Computer Engineering from Istanbul Technical University. He is currently completing an MSc in Computer Science at Wayne State University with a concentration in leveraging High Performance Computing Clusters (HPCC) for Large-Scale Image Analysis to solve Big Data problems. During his curriculum at Wayne State University, Itauma worked as a Graduate Teaching Assistant (Winter 2013 - Winter 2014) and Graduate Research Assistant (Winter 2015) in Computer Science at Wayne State University. He also worked as a Supplemental Instruction Leader / Tutor at the Wayne State Academic Success Center (Fall 2014) where he was awarded the Leader's Choice Award first place among the Tutors in the Academic Success Center. Itauma was also awarded the Supplemental Instruction (SI) Leaders second place for Fall 2014 semester. His interests lie in Robotics and Big Data. He wishes to become a professional educator.