CLASSIFICATION OF INFECTED RBCs

INTERNSHIP PROJECT REPORT

by

P THAISEER
(SC13B175)

DEPARTMENT OF EARTH AND SPACE SCIENCES
INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY
THIRUVANATHAPURAM
JULY 2016

CLASSIFICATION OF INFECTED RBCs

INTERNSHIP PROJECT REPORT

by

P THAISEER
(SC13B175)

Under the guidance of
Dr. Gorthi R K S Subrahmanyam
Associate Professor

DEPARTMENT OF EARTH AND SPACE SCIENCES
INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY
THIRUVANATHAPURAM
JULY 2016

BONAFIDE CERTIFICATE

This is to certify that this Project Report entitled “ **CLASSIFICATION OF INFECTED RBCs**” submitted to **Indian Institute of Space Science and Technology, Thiruvanathapuram**, is a bonafide record of work done by **P THAISEER** under my supervision from **05-06-2016** to **15-07-2016**.

Dr. Gorthi R K S Subrahmanyam
Associate Professor

Dr. Anandmayee Tej
Head of Department
Department Of Earth and Space Sciences

Place
Date

Declaration by Author

This is to declare that this report has been written by me.No part of the report is plagiarized from other sources.All information included from other sources have been duly acknowledged.I aver that if any part of the report is found to be plagiarized I shall take full responsibility for it.

P THAISEER
SC13B175

Place
Date

ACKNOWLEDGMENTS

Words would not suffice to explain the support and advices I received from my guide **Dr. Gorthi R K S Subrahmanyam**, Department of Earth and Space Sciences, for giving me an opportunity to work under his guidance which is invaluable. His unflinching support, suggestions, directions have helped in smooth progress of this internship. He has been a constant source of inspiration in all possible ways for successful completion of my project.I thank him for all the care and concern he has given to me.

I express my heartfelt thanks to **Mr. Gopakumar G**, Research scholar, Indian Institute of Space Science and Technology, for spending his valuable time during discussions and mentoring me throughout the internship. Without his immense help, completion of this internship will still be a mile far.

Its my pleasure to thank **Fahd Bin Abdul Hasis** and **Muhammed Rafi KM**, B.Tech Students, Indian Institute of Space Science and Technology for helping me in High Performance Computing Lab.

Abstract

Diagnosis of many diseases to a great extent depends on detection of infected cells. Here we will look into classification of infected RBCs from cell images and patches of cells using their features. In most of the cases, algorithm used for classification is Support Vector Machine (SVM) and in some cases Neural Network were also used for comparison of accuracy in different classification algorithm. Different methods were incorporated in extracting features for classification: **1) 1D PCA 2) 2D PCA. 3) PCANet 4) Statistical Features.** After testing with different percentage of testing and training on 1D PCA, 2D PCA and Statistical Features, it's found that classification using Statistical Features (98.78 %) gave maximum accuracy followed by 2D PCA (88.05 %) and 1D PCA (84.90 %). A different data set was used for classification using PCANet (96.43%) which gave fairly good accuracy.

Contents

Abstract **i**

List Of Figures **v**

List Of Tables **vi**

List Of Symbols **viii**

1 Introduction **1**

2 Methods Used for Classification **3**

2.1 Support Vector Machine (SVM) 3

How to optimize our problem? 4

2.2 Neural Network 6

3 1D PCA **7**

3.1 How to find the Principal Component? 7

3.2 Observations 9

4 2D PCA **11**

4.1 Finding the Principal Component 11

4.2 Observations 13

5 Statistical Features **15**

5.1 Features Used 15

Contrast 15

Correlation 15

Energy 16

Homogeneity 16

Min. and Max. Mean, Variance and Intensity 16

5.2 Observation 16

6 PCANet **17**
6.1 Structure Of PCANet 17
First Stage (PCA) 18
Second Stage (PCA) 19
Output Stage (Hashing and Histogram) 19
6.2 Observations 20

7 Results, Comparison and Conclusions **21**

Appendices **22**

A Statistical Analysis[8] **23**
A.1 Sensitivity 23
A.2 Specificity 23

B 1D PCA: MatLab Code **25**

C 2D PCA: MatLab Code **27**

D Statistical Feature: MatLab Code **29**

E PCANet: MatLab Code **31**

References **33**

List of Figures

1.1 Infected RBCs 1
1.2 Non Infected RBCs 1

2.1 Hyper-Planes separating two classes 3
2.2 Support Vector Machine: Optimization 4
2.3 Support Vectors and Margin 5
2.4 Single Neuron 6
2.5 Single hidden layer Feed Forward Neural Network 6

3.1 First three Eigen Cells 9

4.1 Shows 2D data from 2 classes, principal component (PC) 11

6.1 Detailed block diagram of PCANet [2] 17
6.2 An input Image 18

List of Tables

3.1 Patches Used - 1D PCA 9
3.2 Confusion Matrix and Accuracy - 1D PCA 9

4.1 Patches Used - 2D PCA 13
4.2 Confusion Matrix and Accuracy - 2D PCA 13

5.1 Patches Used - Statistical Features 16
5.2 Confusion Matrix and Accuracy - Stat. Feat. 16

6.1 Cells Used - PCANet 20
6.2 Confusion Matrix and Accuracy - PCANet 20

7.1 Results for 1D PCA, 2D PCA and Stat. Feat. 21
7.2 Results for PCANet. 21

A.1 Confusion Matrix 23

Chapter 1

Introduction

Diagnosis of many diseases to a great extent depends on how accurately you can detect infected cells in blood. Here we are discussing about the detection of Malaria infected Red Blood Cells from the segmented slide images or patches of full blood. In order for classification, we extract features from cell images or patches through different methods and classify them using these features based on Support Vector Machine (SVM). In order for comparison of classification accuracy with different algorithms, classification was also carried out using Neural Network (NN) and Naive Bayes in some of the cases.

Accuracy and effectiveness of classification depends on how effectively you can extract the features from the images. Effective feature extraction requires accurate segmentation of cell images before feature extraction. Since blood cells will be in clumps in slide images, it is always a difficult task to segment and separate them. For this project, I used already segmented images of RBCs and patch images of RBCs from full blood slides.

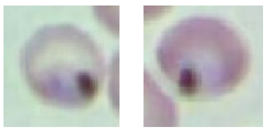

Figure 1.1: Infected RBCs

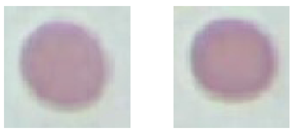

Figure 1.2: Non Infected RBCs

Four different methods for feature extraction on these images and classification using those features is discussed in this report. Methods of feature extraction includes:

1. 1D PCA
2. 2D PCA[1]
3. PCANet[2]
4. Statistical Features

In 1D PCA, the pixel values of each training image is considered as a 1D vector and is used for finding covariance matrix from which the pricipal components are found. These principal components are used to extract features from training and testing images. In 2D PCA, 2D image matrix is directly used to calculate image covariance matrix rather than converting it into 1D vector. From this image covariance matrix, principal components are found and is used for extraction of features from training and testing images. PCANet[2] is a simple deep learning method for feature extraction based on basic data processing components like: PCA, Binary Hashing, Block wise histogram etc. In Statistical Feature extraction, in order to distinguish between the infected and healthy cells, we extract features that could distinguish them based on cell area, perimeter, energy, contrast, correlation, homogeneity etc.

In all above methods except in statistical feature extraction, we try to represent the image in a different space using PCA expecting these new features can distinguish infected and healthy cells in this new space.

Chapter 2

Methods Used for Classification

Features extracted using 1D PCA, 2D PCA, PCANet and Statistical Features are used to classify the cells into infected and healthy. The method used for classification in all cases is Support Vector Machine (SVM). In 1D and 2D PCA, Neural Network and Naive Bayes methods were also used for comparing accuracy of different methods.

2.1 Support Vector Machine (SVM)

According to wikipedia,

"Support Vector Machines (SVMs) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis."

However, it is mostly used in classification problems. Let us look briefly into the SVM algorithm for classification: Initially, all data is plotted in a n-dimensional space (n is the number of different features we have). Then we perform classification by finding the separating hyper-plane that can differentiate the two classes effectively.

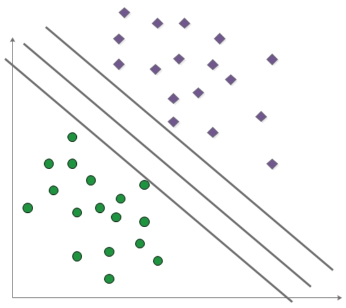

Figure 2.1: Hyper-Planes separating two classes

Maximizing the distances between nearest data point from either classes and hyper-plane will help us to decide the right hyper-plane. This distance is called **Margin**

How to optimize our problem?

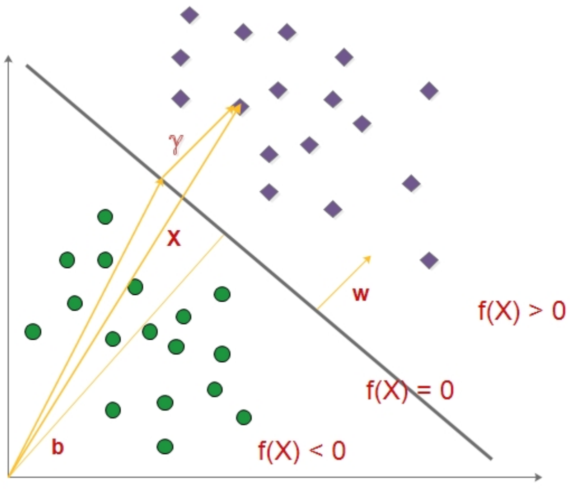

Figure 2.2: Support Vector Machine: Optimization

Let w be any perpendicular vector to hyper-plane, b be the perpendicular distance from origin to hyper-plane, X_n and γ be position vector and margin of any sample and $X_{\perp}$ be the position vector to margin (of same sample) on hyper-plane. Then X can be expressed as,

$$X_n = X_{\perp} + \gamma \frac{w}{\|w\|} \tag{2.1}$$

For X on hyper-plane, we have

$$w^T X + b = 0 \qquad (f(X) = 0) \tag{2.2}$$

$w^T * (2.1) + b \implies$

$$w^T X_n + b = w^T X_{\perp} + b + \gamma \frac{w^T w}{\|w\|} \tag{2.3}$$

$$\implies f(X_n) = 0 + \gamma \frac{w^T w}{\|w\|} \tag{2.4}$$

$$\implies \gamma = \frac{f(X_n)}{\|w\|} \tag{2.5}$$

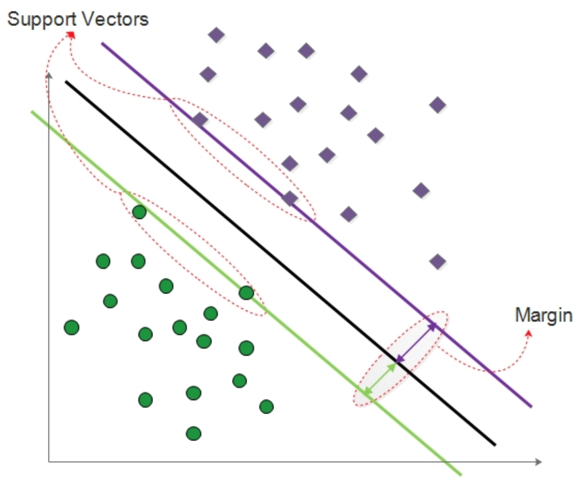

Figure 2.3: Support Vectors and Margin

Define t_n as

$$t_n = \begin{cases} \text{1 for X in class 1} \\ \text{-1 for X in class 2} \end{cases} \tag{2.6}$$

$$\implies \gamma = \frac{t_n f(X_n)}{\|w\|} \tag{2.7}$$

We can scale w and b by same amount according to our wish (this won't affect the margin) and in order to maximize the margin we can minimize:

$$\frac{1}{2}\|w\|^2 \quad s.t. \quad t_n(w^T X_n + b) \geq 1 \tag{2.8}$$

This constrained optimization problem can be solved using Lagrange multipliers. When we construct our Lagrangian for our optimization problem we have:

$$J(w, b, a_n) = \frac{1}{2}\|w\|^2 - \sum_{n=1}^{N} a_n[t_n(w^T X_n + b)] \tag{2.9}$$

Minimizing (2.9) and using KKT conditions, we get:

$$w = \sum_n a_n t_n X_n \tag{2.10}$$

$$b = \frac{1}{N_s} \sum_n [t_n - \sum_{X_m \epsilon S} a_m t_m X_m^T X_n] \tag{2.11}$$

From the values of w and b we can decide the right hyper-plane.

2.2 Neural Network

Neural Network is a classification method motivated or inspired from biological neurons. It process information in a similar way human brain does. Neural Network is composed of large number of highly inter-connected processing elements called neurons. They are organized in layers, typically consists of 3 layers. The first *Input layer* has neurons which sends data to the second *Hidden layer* of neurons and then to the third *Output layer* neurons. These layers are made up of number of inter-connected nodes, which contain **activation function**. They are some predefined function, such as *hyperbolic tangent* or *sigmoid* etc.

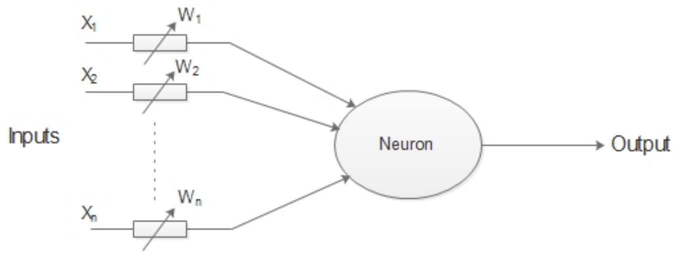

Figure 2.4: Single Neuron

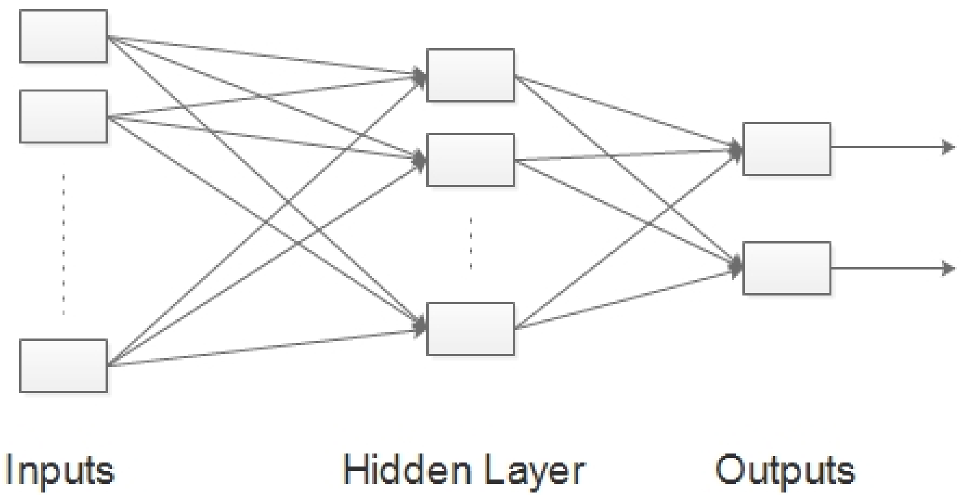

Figure 2.5: Single hidden layer Feed Forward Neural Network

Chapter 3

1D PCA

Direct classification of images based on visual content is very difficult task to achieve due to the intra-class variability and inter-class similarities. In order to overcome these factors we will use a method widely used in data representation and feature extraction technique - *One Dimensional Principal Component Analysis* or *1D PCA*.PCA was first used by Sirovich and Kirby [3][4] to efficiently represent images of human faces. They argued that any image can be reconstructed from a small set of images that define a image basis (Eigenimages). In Principal Component Analysis (PCA), we will find the principal component of the data which is the direction of maximum intra-class variability. Then we will project our data onto this direction. This new projected data is taken as features for classification. In our case, each image is taken as a single one dimensional vector and the principal component for the training images is found.

3.1 How to find the Principal Component?

Let X_k be the k^{th} sample (k^{th} image converted into 1D vector), a_k be its projection length and e be the projection direction. Then, projected sample can be represented as

$$\hat{X}_k = a_k e + m \tag{3.1}$$

where m is the mean of whole data.
In order to find a_k and e, the cause function can be represented as

$$J(a_k, e) = \sum_{k=1}^{n} \|\hat{X}_k - X_k\|^2 \tag{3.2}$$

Substituting (3.1)

$$J(a_k, e) = \sum_{k=1}^{n} \|a_k e + m - X_k\|^2 \tag{3.3}$$

Differentiating (3.3) w.r. to a_k and equating to zero, we get

$$a_k = e^t(X_k - m) \tag{3.4}$$

On expanding (3.3) and Substituting (3.4), we get

$$J(a_k, e) = -e^t \sum_{k=1}^{n} (X_k - m)(X_k - m)^t e + \alpha \tag{3.5}$$

Where α is terms independent of a_k and e

$$J(a_k, e) = -e^t S e + \alpha \tag{3.6}$$

Where $S = \sum_{k=1}^{n} (X_k - m)(X_k - m)^t$ is Scatter Matrix

Rewrite (3.6) as

$$J(a_k, e) = -e^t S e + \lambda(1 - e^t e) + \alpha \tag{3.7}$$

Differentiating (3.7) w.r. to e and equating to zero, we get

$$Se = \lambda e \tag{3.8}$$

$$e^t S e = \lambda \tag{3.9}$$

Maximizing (3.5) $\implies$ maximum λ $\implies$ e is in the direction of eigen vector of S with maximum eigen value.

For d dimension (d dimension reducing into d' dimension)

$$X_k = \sum_{i=1}^{d'} a_{ki} e_i + m \tag{3.10}$$

Where e_i's are the principal eigen vectors corresponding to maximum eigen values in decreasing order.

3.2 Observations

The slide images of blood were converted into 32×32 patches. Out of the total 337409 patches 5600 were marked as positive and remaining 331809 as negative. 1 D PCA was applied on these 32×32 patches to obtain first 10 principal components of the 1024 dimension.

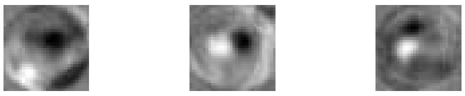

Figure 3.1: First three Eigen Cells

Then SVM and Neural Network were applied on these features for classification. Out of the 5600 positives - 3360 (60%), 1120 (20%) and 1120 (20%) were taken as training, validation and testing sets. Out of the 331809 negatives 3300, 1100 and 327409 were taken as training, validation and testing sets (Almost equal no. of positive and negative training samples were taken).

Type	No.of Samples	Training	Validation	Testing
Positive	5600	3360 (60%)	1120 (20%)	1120 (20%)
Negative	331809	3300	1100	327409

Table 3.1: Patches Used - 1D PCA

On applying SVM and Neural Network, the results were obtained as

Method	Training		Validation		Testing		Accuracy (Testing)
SVM	3108	252	932	188	882	238	84.90%
	197	3103	174	926	49361	278048	
NN	3134	226	976	144	902	218	84.94%
	146	3154	170	930	49247	278162	

Table 3.2: Confusion Matrix and Accuracy - 1D PCA

Sensitivity or True Positive Rate for testing is found from the confusion matrix as 0.79. Specificity or True Negative Rate for testing is found from the confusion matrix as 0.85
For more details on Statistical analysis, refer appendix A and for program code refer appendix B.

Chapter 4

2D PCA

As explained in the previous chapter, Principal Component Analysis (PCA) is a data representation and classical feature extraction technique in the areas of computer vision and pattern recognition. 2D PCA [1] works same as 1D PCA except that the inputs are given as 2D array of image instead of 1 D vector. Comparing with the covariance matrix calculated using PCA, 2D PCA have small size as a result it is easier to compute and lesser time is required for determining the eigenvectors.

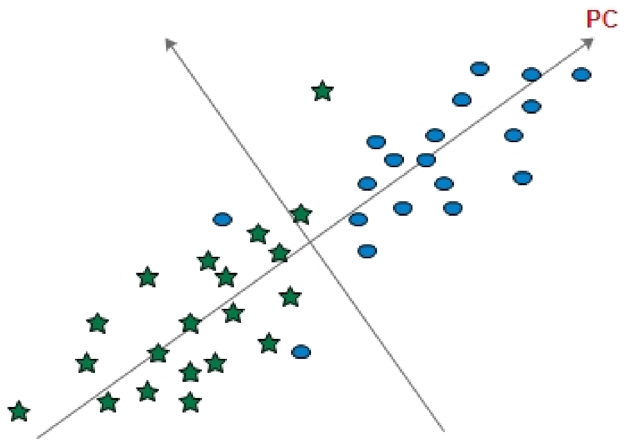

Figure 4.1: Shows 2D data from 2 classes, principal component (PC)

4.1 Finding the Principal Component

Let X be an n-dimensional unitary column vector (Projection vector). Our aim is to project image A (mxn dimension) on to X through linear transformation:

$$Y = AX \tag{4.1}$$

Where Y is m - dimensional feature vector. Now the question arises - How do we find a good X. The total scatter of the samples can be used to find a good projection vector, in fact total scatter is characterized by covariance matrix of the projected feature vectors. Thus we obtain the cause function as:

$$J(Y) = tr(S_y) \tag{4.2}$$

where S_y is the covariance matrix of projected features and $tr(S_y)$ is its trace. Covariance matrix, S_y can be represented as:

$$S_y = E[(Y - E(Y))(Y - E(Y))^T] \tag{4.3}$$

Substituting 4.1

$$S_y = E[(AX - E(AX))(AX - E(AX))^T] \tag{4.4}$$

$$S_y = E([(A - E\mathbf{A})X][(A - E\mathbf{A})X]^T) \tag{4.5}$$

Then we have,

$$tr(S_y) = X^T[E(\mathbf{A} - E\mathbf{A})^T(\mathbf{A} - E\mathbf{A})]X \tag{4.6}$$

Let us define *image covariance matrix* G_t (nxn non-negative definite matrix from its definition) as:

$$G_t = E(\mathbf{A} - E\mathbf{A})^T(\mathbf{A} - E\mathbf{A}) \tag{4.7}$$

G_t can be evaluated from the training images as,

$$G_t = \frac{1}{M}\sum_{j=1}^{m}(A_j - \mathbf{A})^T(A_j - \mathbf{A}) \tag{4.8}$$

Substituting in 4.2, we get

$$J(Y) = X^T G_t X \tag{4.9}$$

In order for maximizing the scatter, we maximize 4.9 (Same as 4.2). Same as we seen in previous chapter eq. 3.6, maximizing 4.9 will give optimum X as the eigenvector of G_t corresponding to largest eigenvalue. Usually, instead of taking only largest eigenvector, we take a set of first few eigenvectors.
Projecting A on to X_k we get,

$$Y_k = AX_k, k = 1, 2, .., d. \tag{4.10}$$

4.2 Observations

The slide images of blood were converted into 32×32 patches. Out of the total 337409 patches 5600 were marked as positive and remaining 331809 as negative. 2D PCA was applied on these 32×32 patches to obtain first 10 principal components. Then SVM and Neural Network were applied on these features for classification. Out of the 5600 positives - 3360 (60%), 1120 (20%) and 1120 (20%) were taken as training, validation and testing sets. Out of the 331809 negatives 3300, 1100 and 327409 were taken as training, validation and testing sets (Almost equal no. of positive and negative training samples were taken).

Type	No.of Samples	Training	Validation	Testing
Positive	5600	3360 (60%)	1120 (20%)	1120 (20%)
Negative	331809	3300	1100	327409

Table 4.1: Patches Used - 2D PCA

On applying SVM and Neural Network, the results were obtained as

Method	Training		Validation		Testing		Accuracy (Testing)
SVM	3129	231	959	161	969	151	88.05%
	146	3154	130	970	39078	288331	
NN	2965	395	978	142	954	166	89.84%
	316	2984	107	993	33225	294184	

Table 4.2: Confusion Matrix and Accuracy - 2D PCA

Sensitivity or True Positive Rate for testing is found from the confusion matrix as 0.86. Specificity or True Negative Rate for testing is found from the confusion matrix as 0.88
For more details on Statistical analysis, refer appendix A and for program code refer appendix C.

Chapter 5

Statistical Features

We have seen two different methods for feature extraction based on PCA and its results on image classification. Now we will see a more basic method of feature extraction, which is using statistical features of the images and do classification based on those features. Statistical features include statistical data like Mean, Median etc. of pixels and Textural features like Energy, Contrast, Correlation and Homogeneity etc. Instead of extracting all these statistical features, a dataset [7] which has already extracted statistical features (20) of the Malarial cells were used and classification was done on that. In order to avoid domination of one feature, all data were normalized.

5.1 Features Used

The dataset[7] used has 20 features for a total of 330701 samples. These 20 features are 10 features each on cells masked by the cell binary image and non-masked slide images of blood plasma. The 10 features include:

Contrast

Contrast is the difference in luminance or colour that makes an object (or its representation in an image or display) distinguishable[8]. Contrast is measured from Gray-Level Co-Occurrence Matrix (GLCM) [9]. Contrast in GLCM measures the local variations in the gray-level co-occurrence matrix.

Correlation

Same as Contrast, Correlation is also measured from GLCM. Correlation in GLCM measures the joint probability occurrence of the specified pixel pairs.

Energy

Energy is obtained as the sum of squared elements in the GLCM. It is also known as uniformity or the angular second moment.

Homogeneity

Homogeneity is measured as closeness of the distribution of elements in the GLCM to the GLCM diagonal.

Min. and Max. Mean, Variance and Intensity

These are basic Statistical analysis applied on the image matrix with a window of 3x3 min. and max. for mean and variance. The min. and max. Intensity are found using a 15x15 window.
The above 10 features on image masked by a window of 15x15 cell binary image and image which is not masked will give us 20 feature, which are used for classification.

5.2 Observation

Total 330701 patches of cells were there, out of which 5600 were labeled as positive, remaining 325101 as negative. From these 60% of positive samples (3360) were taken for training with same number of negative samples. The remaining were used for testing. On applying SVM on these, we obtain the results as in Table 6.2. Sensitivity

Type	No.of Samples	Training	Testing
Positive	5600	3360 (60%)	2240 (40%)
Negative	325101	3360	321741

Table 5.1: Patches Used - Statistical Features

Method	Training		Testing		Accuracy (Testing)
SVM	3355	5	1499	741	98.78%
	5	3328	3184	318557	

Table 5.2: Confusion Matrix and Accuracy - Stat. Feat.

or True Positive Rate for testing is found from the confusion matrix as 0.66. Specificity or True Negative Rate for testing is found from the confusion matrix as 0.99
For more details on Statistical analysis, refer appendix A and for program code refer appendix D.

Chapter 6

PCANet

We have seen the effectiveness of 1D PCA and 2D PCA and Statistical features for feature extraction in the previous chapters. Now let us see another, a very simple deep learning algorithm - *PCANet*[2] for the feature extraction. The idea of deep learning is to represent same data in multiple levels with the hope that higher level features can represent more abstract semantics of data. One of such example is Convolutional Neural Network (CNN)[5][6]. In all deep networks, learning a network for useful classification critically depends on parameter tuning which will take lot of time. This simple deep learning network is based on three basic data processing components: 1) Cascaded PCA, 2) binary hashing and 3) block-wise histogram. PCA filters are used as the convolutional filter banks. The non linear layer is the binary hashing and block-wise histogram for final feature output.
Considering the large dataset used for previous methods we use comparatively small dataset for PCANet as per convenience. So comparing the results are not possible.

6.1 Structure Of PCANet

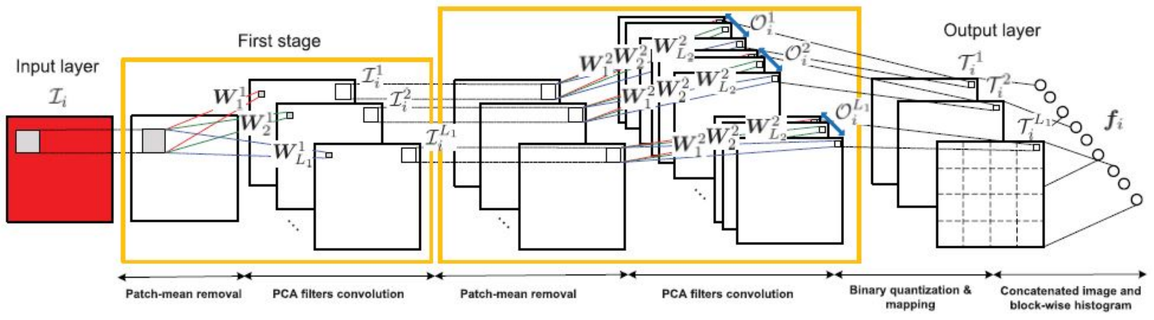

Figure 6.1: Detailed block diagram of PCANet [2]

The PCANet parameters which we can tune are:

- Patch Size - k_1xk_2.
- No. of filters in first (L_1) and second stage (L_2).
- Bin size for block-wise histogram in output layer.

Let we have N input (training) images $[I_i]_{i=1}^{N}$ of size m x n.

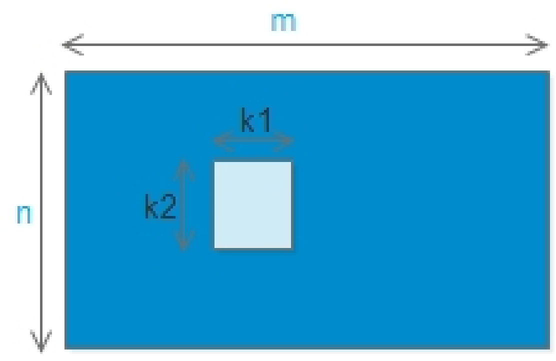

Figure 6.2: An input Image

First Stage (PCA)

For each pixel of image, we take a k_1xk_2 patch, and collect all overlapping patches of i_{th} image as $x_{i,1}, x_{i,2}, ..., x_{i,\tilde{m}\tilde{n}}$, where each $x_{i,j}$ is the j^{th} vectorized patch in I_i and $\tilde{m}$,$\tilde{n}$ are $m - \lceil \frac{k_1}{2} \rceil$ and $n - \lceil \frac{k_2}{2} \rceil$ respectively. Where $\lceil x \rceil$ is the smallest integer greater than or equal to x. Then we subtract mean of all patches from each patch to obtain:

$$\bar{X}_i = [x'_{i,1}, x'_{i,2}, ..., x'_{i,\tilde{m}\tilde{n}}] \tag{6.1}$$

Where $x'_{i,j}$ is the patch mean removed patches. By constructing $\bar{X}$ for all input images, we get

$$\bar{X} = [X_1, X_2, .., X_N] \in \mathbb{R}^{k_1 k_2 N \tilde{m}\tilde{n}} \tag{6.2}$$

Perform PCA over these N images and obtain L_1 principal eigenvectors of XX^T, the PCA filters are then expressed as:

$$W_l^1 = mat_{k_1,k_2}(q_l(XX^T)) \in \mathbb{R}^{k_1 \times k_2}, l = 1, 2, .., L_1 \tag{6.3}$$

Here $mat_{k_1,k_2}(x))$ will convert $x \in \mathbb{R}^{k_1 k_2}$ into $W \in \mathbb{R}^{k_1 \times k_2}$ and $q_l(x)$ denotes the l^{th} principal eigenvector of x.

Second Stage (PCA)

Second stage is repeating the first stage on input images convolved with PCA filters obtained in first stage. Let l^{th} filter output of first stage be

$$I_i^l = I_i * W_l^1, i = 1, 2, .., N \tag{6.4}$$

here $*$ denotes 2D convolution, I_i is zero padded before convolving to get same size of input image.
Repeating the steps mentioned in the first stage, we obtain L_2 PCA filters in the second stage as

$$W_l^2 = mat_{k_1,k_2}(q_l(YY^T)) \in \mathbb{R}^{k_1 \times k_2}, l = 1, 2, .., L_2 \tag{6.5}$$

where Y is the concatenated mean removed patches of all output images of first stage. Then we get output of second stage as

$$O_i^l = \{I_i^l * W_l^2\}_{l=1}^{L_2} \tag{6.6}$$

Where $*$ denotes 2D convolution.

Output Stage (Hashing and Histogram)

For each output image in the first stage (L_1), we have L_2 output images after second stage. These output images are binarized Heaviside step like function whose value is one for positive and zero otherwise. Now around each pixel, there is L_2 binary bits, which we sum to a decimal number. This converts L_2 outputs at second stage into single image. Thus we obtain:

$$T_i^k = \sum_{l=1}^{L_2} 2^{l-1} H(I_i^l * W_l^2), \quad where \quad k = 1, 2, .., L_1 \tag{6.7}$$

Value of T_i^k will be in the range $[0, 2^{L_2} - 1]$, each of these is partitioned into B blocks. Then we compute histogram (with 2^{L_2} bins) in each block and concatenate all histogram into one vector - $Bhist(T_i^k)$. These $Bhists$ concatenated for $k = 1, 2, .., L_1$, gives feature for i^{th} training image as

$$f_i = [Bhist(T_i^1), Bhist(T_i^2), .., Bhist(T_i^{L_1})]^T \in \mathbb{R}^{2^{L_2} L_1 B} \tag{6.8}$$

The number of features for a single training image can be changed by tuning K_1, K_2, L_1, L_2 and size of Block B

6.2 Observations

The parameters used for PCANet after fine tuning are:

- Patch Size - k_1xk_2 is 7x7.
- No. of filters in first - 8 (L_1) and second stage - 8 (L_2).
- Block size, B = 9

Type	No.of Samples	Training	Testing
Positive	2492	1300	1192
Negative	56300	1300	55000

Table 6.1: Cells Used - PCANet

On applying SVM, the results were obtained as

Method	Training		Testing		Accuracy (Testing)
SVM	1300	0	972	220	96.43%
	18	1282	1785	53215	

Table 6.2: Confusion Matrix and Accuracy - PCANet

The results heavily depends on the parameters tuned (patch size, no. of filters etc.) and these are tuned so as to get best result for these images. Sensitivity or True Positive Rate for testing is found from the confusion matrix as 0.81. Specificity or True Negative Rate for testing is found from the confusion matrix as 0.97
For more details on Statistical analysis, refer appendix A and for program code refer appendix E.

Chapter 7

Results, Comparison and Conclusions

The dataset used for 1D PCA, 2D PCA and Statistical Features were the same. Their results after classification are in Table 7.2. Statistical feature shows an higher accuracy of 97.78% compared with the 2D PCA (88.90%) and 1D PCA (84.90%). Although the accuracy is very high, it shows very low True Positive Rate. 2D PCA gives best TPR - 0.86. The high accuracy of Stat. Feat. is due to its very high value in True Negative Rate which is 0.99. But it has the least TPR

Method	Accuracy	True Positive Rate	True Negative Rate
1D PCA	84.90%	0.79	0.85
2D PCA	88.90%	0.86	0.88
Stat. Feat.	97.78%	0.66	0.99

Table 7.1: Results for 1D PCA, 2D PCA and Stat. Feat.

As per convenience a different dataset (which has lower number of samples) were used for classification using PCANet due to its time consumption in classification.

Method	Accuracy	True Positive Rate	True Negative Rate
PCANet	96.43%	0.81	0.97

Table 7.2: Results for PCANet.

If we put a trade off for TPR and TNR, PCANet gives very good value for TPR and TNR which is 0.81 and 0.97 respectively. Overall we can say that PCANet performs very effectively in feature extraction, it is obviously a good, simple and effective deep learning network.

Appendices

Appendix A

Statistical Analysis[8]

To make useful conclusions out of experimental results, we do Statistical analysis on our result. One of them is based on Confusion Matrix (**CM**) (Table A.1). Sensitivity

True Positive (TP)	False Negative (FN)
False Positive (FP)	True Negative (TN)

Table A.1: Confusion Matrix

(True Positive Rate) and specificity (True Negative Rate) are two statistical measures of the performance of a binary classification test calculated from confusion matrix.

A.1 Sensitivity

Sensitivity or True Positive Rate (TPR) measures the probability that how often the result will be positive. Or in other words it can be explained as - how effectively all the positives are identified as positives.

$$True\ Positive\ Rate\ (TPR) = \frac{TP}{TP + FN} \tag{A.1}$$

A.2 Specificity

Specificity or True Negative Rate (TNR) measures the probability that how often a negative sample will be identified as negative.

$$True\ Negative\ Rate\ (TNR) = \frac{TN}{TN + FP} \tag{A.2}$$

Appendix B

1D PCA: MatLab Code

```
% File Name: OneDPCA.m
% Function: OneDPCA()

function [EigenVectors] = OneDPCA(data)
    z = zeros(size(data,1),size(data,2));
    s = zeros(size(data,2));
    for i = 1:size(data,1)
        z(i,:) = mean(data) - data(i,:);
        s = s + z(i,:)'*z(i,:);
    end
    [EigenVec,EigenVal] = eig(s);
    [~,indx] = sort(max(EigenVal));
    indx = flip(indx);

    EigenVectors = zeros(size(EigenVec,1));
    for i = 1:size(indx,2)
    EigenVectors(:,i) = EigenVec(:,indx(1,i));
    end
end
```

Appendix C

2D PCA: MatLab Code

```
% File Name: TwoDPCA.m
% Function: TwoDPCA()

function [EigenVectors] = TwoDPCA(data)
    ImageSum = zeros(size(data,1),size(data,2));
    for i=1:size(data,3)
        ImageSum = ImageSum + data(:,:,i);
    end
    ImageAvg = ImageSum/size(data,3);

    G = zeros(size(data,2));
    for i=1:size(data,3)
        G = G + (data(:,:,i)-ImageAvg)'*(data(:,:,i)-ImageAvg);
    end
    G = G/size(data,3);
    [EigenVec,EigenVal] = eig(G);
    [~,indx] = sort(diag(EigenVal));
    indx = flip(indx');

    EigenVectors = zeros(size(EigenVec,1));
    for i = 1:size(indx,2)
    EigenVectors(:,i) = EigenVec(:,indx(1,i));
    end
end
```

Appendix D

Statistical Feature: MatLab Code

```
% File: StatFeatClassification.m

load('MalFeat');
Feat = MalFeat.Feat;
ClsLbl = MalFeat.label;
Set = MalFeat.set;

MalImdb = [Samples,ClsLbl,Set];
ConfMatSVM = SVMClassification(MalImdb);

function ConfnMatSVM = SVMClassification(MalImdb)
    DvdData = DivideDataset(MalImdb);
    SVMModel   = svmtrain(DvdData.TrFeat, DvdData.TrLbl, 'method', 'SMO',
        'kernel_function', 'rbf');

    PrTrLbl      = svmclassify(SVMModel, DvdData.TrFeat);
    ConfnMatTr = confusionmat(DvdData.TrLbl,PrTrLbl);

    PrValLbl     = svmclassify(SVMModel, DvdData.ValFeat);
    ConfnMatVal = confusionmat(DvdData.ValLbl,PrValLbl);

    PrTsLbl      = getPrTsLbl(SVMModel,DvdData.TsFeat,100);
    ConfnMatTs = confusionmat(DvdData.TsLbl,PrTsLbl);

    ConfnMatSVM = [ConfnMatTr; ConfnMatVal; ConfnMatTs];
end
```

```
function [Imdb] = DivideDataset(Samples)
    Set = Samples(:,end);
    Imdb.TrFeat = Samples(Set(:,end)==1,1:size(Samples,2)-2);
    Imdb.TrLbl = Samples(Set(:,end)==1,size(Samples,2)-1);
    Imdb.ValFeat = Samples(Set(:,end)==2,1:size(Samples,2)-2);
    Imdb.ValLbl = Samples(Set(:,end)==2,size(Samples,2)-1);
    Imdb.TsFeat = Samples(Set(:,end)==3,1:size(Samples,2)-2);
    Imdb.TsLbl = Samples(Set(:,end)==3,size(Samples,2)-1);
end

function PrTsLbl = getPrTsLbl(SVMModel,TsFeat,ChunkSize)
    [numSamp, ~] = size(TsFeat);
    PrTsLbl        = zeros(numSamp, 1);
    btchSz      = floor(numSamp/ChunkSize);
    for i = 1:ChunkSize
        strtInd = (i-1)*btchSz+1;
        endInd = strtInd+btchSz;
        PrTsLbl(strtInd:endInd) = svmclassify(SVMModel,
           TsFeat(strtInd:endInd, :));
    end
    strtInd = endInd+1;
    PrTsLbl(strtInd:end) = svmclassify(SVMModel, TsFeat(strtInd:end, :));
end
```

Appendix E

PCANet: MatLab Code

```
% File: PCANet.m , This only includes the main code. There are sub-codes
    for this file.

load('data.mat'); load('label.mat');

TrnData = data(:,label(:,1)==1);
TrnLabels = label(:,1)==1;
TestLabels = label(:,1)==2;
TestData = data(:,label(:,1)==2);
nTestImg = length(TestLabels);

PCANet.Stages = 2;
PCANet.PatchSize = [9 9];
PCANet.NumFilters = [8 8];
PCANet.HistBlockSize = [9 9];
PCANet.BlkOverLapRatio = 0;

%% PCANet Training

fprintf('\n PCANet Training \n')
TrnData_ImgCell = Img2Cell(TrnData);
clear TrnData;
[FeatureTrain,PCAFilter,BlkIdx] = PCANetTrain(TrnData_ImgCell,PCANet,1);
save('PCAFilter','PCAFilter'); clear PCAFilter;

SVMModelLinear = svmtrain(FeatureTrain', TrnLabels, 'method', 'SMO',
    'kernel_function', 'linear');
```

```
save('SVMModelLinear','SVMModelLinear'); clear SVMModelLinear;

%% PCANet Feature Extraction and Testing

TestData_ImgCell = Img2Cell(TestData);
clear TestData;
load('PCAFilter');
load('SVMModelLinear');

fprintf('\n PCANet Testing \n')

PrTsLbl = zeros(nTestImg,1);

for p = 1:1:nTestImg
    fval = PCANet_FeaExt(TestData_ImgCell(idx),PCAFilter,PCANet);
    PrTsLbl(p,1)     = svmclassify(SVMModelLinear, fval');
end

ConfnMatTest = confusionmat(TestLabels,PrTsLbl);

% Remaining Functions (Functions called within this code) are in separate
    files
```

References

[1] Jian Yang, David Zhang, Alejandro F. Frangi, and Jing-yu Yang, *“Two-Dimensional PCA: A New Approach to Appearance-Based Face Representation and Recognition”*, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 26, no. 1, pp. 131-132, Jan. 2004.

[2] Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, and Yi Ma , *“PCANet: A Simple Deep Learning Baseline for Image Classification?”*, IEEE Trans. Image Processing, vol. 24, no. 12, pp. 5019-5020, Dec. 2015.

[3] L. Sirovich and M. Kirby, *“Low-Dimensional Procedure for Characterization of Human Faces,* J. Optical Soc. Am., vol. 4, pp. 519-524, 1987.

[4] M. Kirby and L. Sirovich, *“Application of the KL Procedure for the Characterization of Human Faces,”* IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 12, no. 1, pp. 103-108, Jan. 1990.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *“ImageNet classification with deep convolutional neural network,”* in Proc. NIPS, 2012, pp. 10971105.

[6] K. Kavukcuoglu, P. Sermanet, Y. Boureau, K. Gregor, M. Mathieu, and Y. LeCun, *“Learning convolutional feature hierarchies for visual recognition,”* in Proc. NIPS, 2010, pp. 10901098.

[7] Data Set extracted by G Gopakumar. Research Scholar, Dept. of Earth and Space Science. IIST

[8] WIKIPEDIA, *“http://www.wikipedia.org”*

[9] MathWorks, Texture Analysis Using the Gray-Level Co-Occurrence Matrix (GLCM) *“http://in.mathworks.com/help/images/gray-level-co-occurrence-matrix-glcm.html”*