CHAPTER 6

Cytopathological Image Analysis using Deep Learning Networks in Microfluidic Microscopy

In Chapter 4, we have introduced a low-cost high-throughput microfluidic microscopy instrumentation together with a signature based cell classification for point-of-care diagnosis/screening. Chapter 5 discussed a complete automated framework for processing cells captured using the prototype device. Basically, this framework was a hand-engineered feature based cell classification system and it improved the classification accuracy as reported in chapter 5 for the leukaemia cell-line classification. In this chapter, we explore the possibility of using deep learning networks for cytopathology to improve the results further. We demonstrate that without any conventional fine segmentation and explicit feature extraction, the proposed deep learning algorithms effectively classify the coarsely localised cell lines. The designed deep belief network as well as the deeply pre-trained CNN outperforms the conventionally used decision systems and are important in medical domain where the availability of labeled data is limited for training. We hope that our work enables the development of a clinically significant high-throughput microfluidic microscopy based tool for disease screening/triaging especially in resource limited settings.

6.1 Introduction

Biologists have identified that the power of human brain comes from large number of massively interconnected neurons capable of parallel computation (Deco et al. (2008)). Several Artificial Neural Network (ANN) architectures were studied (SHI and He (2011)) to provide the expertise to the machines, some of them bypassing the need for hand engineered features. A review on the use of artificial neural network in cytopathology can be found in Pouliakis et al. (2016). Traditional

classification systems are typically modelled to contain steps such as cell segmentation, feature extraction, and classification using SVM (Irshad et al. (2014)). For large datasets, SVM takes more time for learning and results in large number of support vectors, particularly when the decision problem is hard. Fully connected normal artificial neural network is also not that feasible to learn such complex decision problems from big dataset. Such problems often requires large number of layers and neurons and hence large number of parameters need to be learned making the learning process slower. Also, the problem of vanishing of gradients in lower layers of ANN using back propagation makes the learning problem further difficult, if not impossible. Another difficulty with ANNs and SVMs is that they need labeled data and the amount of information that the system can learn is restricted by labels. Recently deep learning systems are emerging as reliable and de-facto model for image analysis and many groups across the globe are quickly entering the field and applying these techniques to variety of applications. A recent review on deep learning in medical domain can be found in Greenspan et al. (2016).

As discussed in earlier chapters, mIFC (Barteneva et al. (2012); Lisa et al. (2014); Beers et al. (2014)) is a nascent technology that combines the statistical power of flow cytometry with spatial and quantitative morphology of digital microscopy. We have a low-cost, high-throughput prototype microfluic microscopy system suitable for disease screening in resource limited settings and the proof of concept of the system is presented in chapter 4, where we have used advanced PCA signatures for the classification of leukaemia cell-lines. Further in chapter 5, a general framework for processing cells in mIFC is developed and cell classification based on hand-engineered morphological features of the cells have been carried out. Though the system offered high-throughput at low cost, the reported accuracy had to be improved so as to make it clinically usable. In this chapter, we explore and propose the possibility of using deep learning for cytopathology for a reliable classification system. Though we demonstrate the performance of mIFC based classification system only on three important leukaemia cell lines K562, MOLT and HL60, the approach is quite general as it does not require explicit segmentation and feature extraction. When augmented with other diagnostic modalities, such systems enable early detection of diseases such as cancer.

In real scenarios, more often we will be having large amount of data for training but having a small percentage of them labeled. In order to overcome the difficulty of learning both from labeled as well as unlabeled data, a system that model the structure of the data is needed. We introduce one such model where the idea is to tune the weights to learn a general abstract representation of the structure of the data without considering the labels. This is done by building a Deep Belief Network (DBN) using Restricted Boltzmann Machines (RBM) and capturing the distribution of the training vectors using the parameters of the RBM; the weights and biases. The model keeps the efficiency and simplicity of the gradient method for learning. Also we discuss the use of CNN for cytopathology analysis using a network deeply trained on the popular imaging database ImageNet (Deng et al. (2009); Chatfield et al. (2014)). This also overcomes the need for large labeled data and produce discriminative features for an accurate classification.

The major contributions of this chapter are 1) the proposal of a highly accurate classification framework based on deep learning for unstained, unlabeled IFC data for the first time and is an improvement over the methods/frameworks proposed in chapters 4 and 5, there by moving in a direction one step ahead for a reliable screening/triaging tool for cost-effective disease diagnosis. The classification framework proposed in this chapter does not require fine segmentation and explicit feature extraction unlike the method described in earlier chapters, still producing better classification accuracy. 2) the proposal of a DBN based cell classifier for better accuracy and faster response particularly when the availability of labeled medical data is limited and 3) the finding that the CNN pre-trained on ImageNet database can generate discriminant features leading to very good classification accuracy for the leukaemia cell-lines K562, MOLT and HL60. Just like the DBN based classifier, CNN pre-trained on ImageNet is also very useful in medical field when there is only limited labeled medical data for training (to come up with a trained deep CNN classifier).

This chapter is organised such that section 6.2 proposes a new and simple framework for cytopathological analysis and classification of IFC data using deep learning. Section 6.3 discusses the designed DBN for learning the structure of the data and section 6.4 introduces the CNN ImageNet model to extract discriminative features for classification. Results and discussion are provided in section

6.5 followed by publications in section 6.6. The chapter is concluded with the summary report in section 6.7.

6.2 Framework for Analysis of Cells in Microfluidics based IFC

The image dataset employed for the proposed deep learning based classification is the leukaemia cell-line dataset, and is the one and the same that we had presented in chapter 4. The dataset was captured using the relatively new flow imaging modality microfluidic microscopy (Mf-Ms). As opposed to conventional IFCs, the method leverages unconventional optics and microuidics based sample handling to meet the required imaging throughput and fidelity specifications for cytopathology. As noted in earlier chapters, the dataset contains 618 cells and are localised from the raw video stream of the leukaemia cell lines K562, MOLT and HL60.

The basic steps in proposed framework for making cytopathology decision are pre-processing, rough localisation and classification and are shown in the block diagram in Fig. 6.1. The first major step in automating the cytopathology analysis is segmentation of the cells. Often, segmentation is difficult and computationally intensive. In the proposed approach rather than going for an accurate segmentation, we look for good classification with the features extracted using deep learning networks from the roughly localised cells. The rough segmentation is achieved by finding a rectangular bounding box containing the cell.

Figure 6.1: Block diagram showing overview of the system.

In order to enhance features of cells, a simple preprocessing is done by subtracting the background. A background frame for this purpose can be readily captured by keeping only the sheath fluid in the channel but before pumping the cells. Before subtracting the background both foreground and background frames are filtered using an average mask of size 5×5 to reduce the effect of noise. The

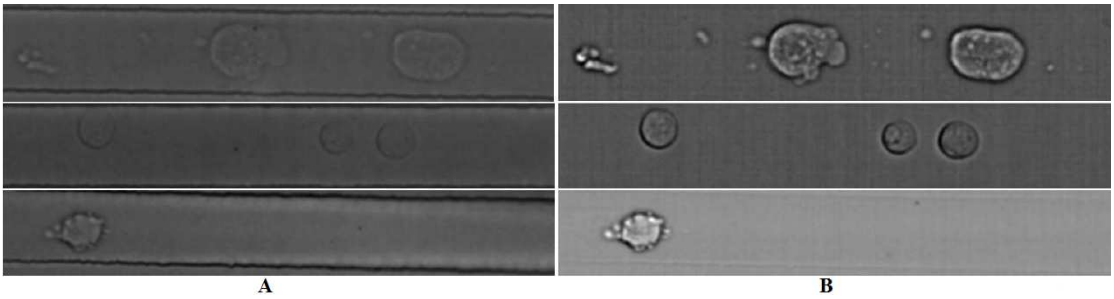

Figure 6.2: a) Frames containing K562 cells (first row), MOLT cells (second row) and HL60 cells (third row); b) the corresponding background subtracted enhanced frames.

Figure 6.3: a) Roughly localised cells from K562 frame in Fig. 6.2; b) bounding box containing left most object from (a), and left most cells from background subtracted frames in Fig. 6.2

effect of the operation is shown in Fig. 6.2. We have roughly localised the cells as discussed in section 5.3 and identified the cell region constrained by a rectangular bounding box. The roughly localised, left most cells from the frames in Fig. 6.2 are shown in Fig. 6.3 **b**.

Recently, deep learning networks based on RBM and CNN are found to be effective in learning complex features for higher level visual recognition task. In subsequent sections we introduce these networks that operate on cell images. The effectiveness in cell classification by these networks is going to be demonstrated with the good results achieved in classifying the leukaemia cell lines HL60, K562 and MOLT. Note that, unlike the framework discussed in chapter 5, the updated framework shown in Fig. 6.1 does not need the fine segmentation and explicit feature extraction and it uses deep learning based classifiers.

6.3 Deep Belief Network for Classification

We design a Deep Belief Network (DBN) for the classification of leukaemia cell lines. A DBN can be thought of as multiple layers of hidden units with connections

between the layers but not between the units within each layer. In DBN, every two adjacent layers except at the fully connected final layer can be treated as Restricted Boltzmann Machine (RBM). Typically DBN is trained in greedy way by training each of the RBM (Hinton (2012)), one at a time starting from the lowest layer. This is an unsupervised training and is going to be discussed in the following subsection. Finally, the weights learned are fine tuned by back-propagation using the available labeled data.

6.3.1 Restricted Boltzmann Machine

Restricted Boltzmann Machine is a popular generative model. A pictorial representation of typical RBM architecture is shown in Fig. 6.4. Note that the connections are undirected and hence bidirectional. In RBM, no connections are made between the hidden units in the hidden layer and no connections are made between the visible units in the visible layer. This helps to reasonably assume that *i*) for a given input at the visible layer, the output at different hidden units are independent and *ii*) for a given output observed at the hidden layer, the output that can be induced at different visible units are also independent. Though RBM can be used as a stand alone classifier (Larochelle et al. (2012)), typically they are used in DBN where a discriminative fine tuning is applied on top of the structure of the data learned by the RBM. It has been shown that generative model learning with RBM improved the discriminative classification result in studying fMRI images of patients recovering from stroke (Schmah et al. (2009)). In Nayak et al. (2013), a variant of RBM is used to classify tumour histopathology images. In Hinton and Salakhutdinov (2006), RBM is used to learn low dimensional effective features and in Brosch and Tam (2013), RBM is used to learn low dimensional effective features in manifold learning of brain MRI. The applicability of RBM in medical data is discussed in Aalto (2014).

Consider the architecture provided in Fig. 6.4. In this particular configuration, there are m visible nodes (v_i) and n hidden nodes (h_j). The visible node biases are labeled as $\{b_i\}_{i=1}^{m}$ while the hidden node biases are $\{c_j\}_{j=1}^{n}$. The energy of this network at any instant can be defined as

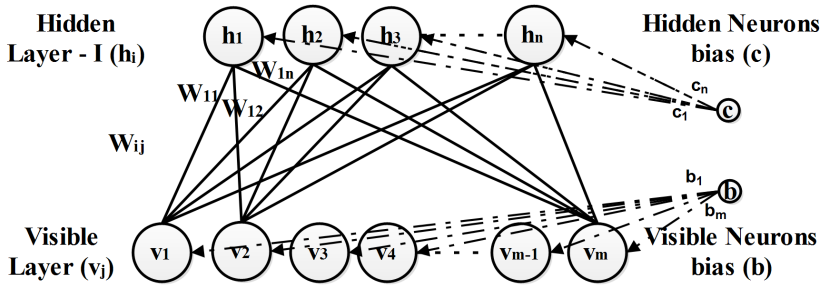

Figure 6.4: Architecture of Restricted Boltzmann Machine

$$E(v,h) = -\left[h^T W v + b^T v + c^T h\right]; \tag{6.1}$$

Where,

$$W = \begin{bmatrix} W_{11} & . & . & W_{1m} \\ . & . & . & . \\ . & . & . & . \\ W_{n1} & . & . & W_{nm} \end{bmatrix}; \tag{6.2}$$

$$b = \begin{bmatrix} b_1 \\ . \\ . \\ b_m \end{bmatrix}; c = \begin{bmatrix} c_1 \\ . \\ . \\ c_n \end{bmatrix}; v = \begin{bmatrix} v_1 \\ . \\ . \\ v_m \end{bmatrix}; h = \begin{bmatrix} h_1 \\ . \\ . \\ h_n \end{bmatrix} \tag{6.3}$$

With a given configuration (of parameters), the RBM can be thought of representing a joint probability distribution for observing v and h together.

$$p(v,h) = \frac{\exp^{-E(v,h)}}{\Sigma_{v',h'} \exp^{-E(v',h')}} \tag{6.4}$$

If there is an algorithm to train a RBM to capture (different aspects of) the probability distribution of the training data (Eq. 6.5), it can be very useful in classification (Fischer and Igel (2014)). With training vectors v and target labels ω to predict, a subset of the factors explaining the distribution of the training vectors explain much of ω, given v. Hence representations that are useful for $p(v)$ tend to be useful when learning $p(\omega|v)$. Learning RBM corresponds to fitting

its parameters such that the distribution represented by the RBM models the distribution underlying the training data, $p(v)$. More specifically, the aim is to find weights and biases that define a Boltzmann distribution (Hinton (2007)) in which the training vectors have high probability.

$$p(v) = \frac{\Sigma_h \exp^{-E(v,h)}}{\Sigma_{v',h'} \exp^{-E(v',h')}} \tag{6.5}$$

As noted, the visible units constitute the first layer of RBM and correspond to the components of an observation (e.g., one visible unit for each pixel of a digital input image). The hidden units model dependencies between the components of observations (e.g., dependencies between the pixels in the images) and can be viewed as non-linear feature detectors (Hinton (2007)). Several such RBM layers can be then stacked up by treating the hidden layer of the lower RBM as input layer to the RBM in the immediate upper layer and can be trained one after the other starting from the lower RBM. By stacking RBMs in this way, one can learn features from features in the hope of arriving at a high-level representation. It was empirically shown that this has produced better feature representations both in terms of classification error (Larochelle et al. (2009)) and in terms of the invariance properties of the learned features (Goodfellow et al. (2009)).

In order to train RBM, we need to maximise the chance of observing the training vectors in the underlying data distribution. This can be done by maximising $p(v)$ which is equivalent to minimising the average negative log likelihood using gradient descent. Let the parameter that we want to learn is θ. The parameter θ can be the weight of the connections W_{ij}, bias of visible nodes b_j or bias of the hidden nodes c_i of RBM.

It turns out that (Bengio (2009))

$$
\begin{aligned}
\frac{\partial \log p(v)}{\partial \theta} &= \frac{\partial}{\partial \theta} \log \left(\frac{\sum_h \exp^{-E(v,h)}}{\sum_{v,h} \exp^{-E(v,h)}} \right) \\
&= \frac{\sum_{v,h} \exp^{-E(v,h)}}{\sum_h \exp^{-E(v,h)}} \left(\frac{\sum_{v,h} \exp^{-E(v,h)} \frac{\partial}{\partial \theta} \sum_h \exp^{-E(v,h)}}{\left(\sum_{v,h} \exp^{-E(v,h)} \right)^2} \right) \\
&- \frac{\sum_{v,h} \exp^{-E(v,h)}}{\sum_h \exp^{-E(v,h)}} \left(\frac{\sum_h \exp^{-E(v,h)} \frac{\partial}{\partial \theta} \sum_{v,h} \exp^{-E(v,h)}}{\left(\sum_{v,h} \exp^{-E(v,h)} \right)^2} \right) \qquad (6.6) \\
&= -\frac{1}{\sum_h \exp^{-E(v,h)}} \sum_h \exp^{-E(v,h)} \frac{\partial E\left(v,h\right)}{\partial \theta} \\
&+ \frac{1}{\sum_{v,h} \exp^{-E(v,h)}} \sum_{v,h} \exp^{-E(v,h)} \frac{\partial E\left(v,h\right)}{\partial \theta} \\
&= -\sum_h p(h|v) \frac{\partial E(v,h)}{\partial \theta} + \sum_{v,h} p(v,h) \frac{\partial E(v,h)}{\partial \theta}
\end{aligned}
$$

$$
\frac{\partial logp(v)}{\partial \theta} = -\mathbb{E}_{h|v} \left[\frac{\partial E(v,h)}{\partial \theta} \right] + \mathbb{E}_{(v,h)} \left[\frac{\partial E(v,h)}{\partial \theta} \right] \qquad (6.7)
$$

Where $\mathbb{E}$ is the expectation operator. It directly follows from Eq. 6.1 that

$$
\frac{\partial E(v,h)}{\partial W_{ij}} = -h_i v_j; \frac{\partial E(v,h)}{\partial b_j} = -v_j; \frac{\partial E(v,h)}{\partial c_i} = -h_i \qquad (6.8)
$$

The two unknown quantities left out in computing the gradient using Eq. 6.6 are $p(h|v)$ and $p(v,h)$.

$$
p(h|v) = \frac{p(v,h)}{\sum_{h'} p(v,h')} \qquad (6.9)
$$

Considering stochastic binary inputs and outputs,

$$
\begin{aligned}
p(h|v) &= \frac{\exp^{\left(h^T W v + b^T v + c^T h\right)} / \Sigma_{v',h'} \exp^{\left(h'^T W v' + b^T v' + c^T h'\right)}}{\sum_{h' \in \{0,1\}^H} \exp^{\left(h'^T W v + b^T v + c^T h'\right)} / \Sigma_{v',h'} \exp^{\left(h'^T W v' + b^T v' + c^T h'\right)}} \quad (6.10)\\
&= \frac{\exp^{\left(h^T W v + c^T h\right)}}{\sum_{h' \in \{0,1\}^H} \exp^{\left(h'^T W v + c^T h'\right)}}\\
&= \frac{\exp^{\left(\sum_i h_i W_i v + c_i h_i\right)}}{\sum_{h'_1 \in \{0,1\}} \sum_{h'_2 \in \{0,1\}} \cdots \sum_{h'_n \in \{0,1\}} \exp^{\left(\sum_i h'_i W_i v + c_i h'_i\right)}}\\
&= \frac{\prod_i \exp^{(h_i W_i v + c_i h_i)}}{\sum_{h'_1 \in \{0,1\}} \sum_{h'_2 \in \{0,1\}} \cdots \sum_{h'_n \in \{0,1\}} \prod_i \exp^{\left(h'_i W_i v + c_i h'_i\right)}}\\
&= \frac{\prod_i \exp^{(h_i W_i v + c_i h_i)}}{\{\sum_{h'_1 \in \{0,1\}} \exp^{\left(h'_1 W_1 v + c_1 h'_1\right)}\} \ldots \{\sum_{h'_n \in \{0,1\}} \exp^{\left(h'_n W_n v + c_n h'_n\right)}\}}\\
&= \frac{\prod_i \exp^{(h_i W_i v + c_i h_i)}}{\prod_i \sum_{h'_i \in \{0,1\}} \exp^{\left(h'_i W_i v + c_i h'_i\right)}}\\
&= \prod_i \frac{\exp^{(h_i W_i v + c_i h_i)}}{\sum_{h'_i \in \{0,1\}} \exp^{\left(h'_i W_i v + c_i h'_i\right)}}\\
&= \prod_i \frac{\exp^{(h_i W_i v + c_i h_i)}}{1 + \exp^{(W_i v + c_i)}}\\
&= \prod_i p(h_i|v)
\end{aligned}
$$

Thus,

$$
\begin{aligned}
p(h_i = 1|v) &= \frac{\exp^{(W_i v + c_i)}}{1 + \exp^{(W_i v + c_i)}} \quad (6.11)\\
&= \frac{1}{1 + \exp^{-(W_i v + c_i)}}\\
&= Sigmoid(W_i v + c_i)
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
p(v_j = 1|h) &= \frac{1}{1 + \exp^{-(b_j + h^T W_j)}} \quad (6.12)\\
&= Sigmoid(b_j + h^T W_j)
\end{aligned}
$$

Where W_i and W_j denote the i^{th} row and j^{th} column of the weight matrix respectively. Note that based on Eq. 6.11, the first term of Eq. 6.6 can be computed

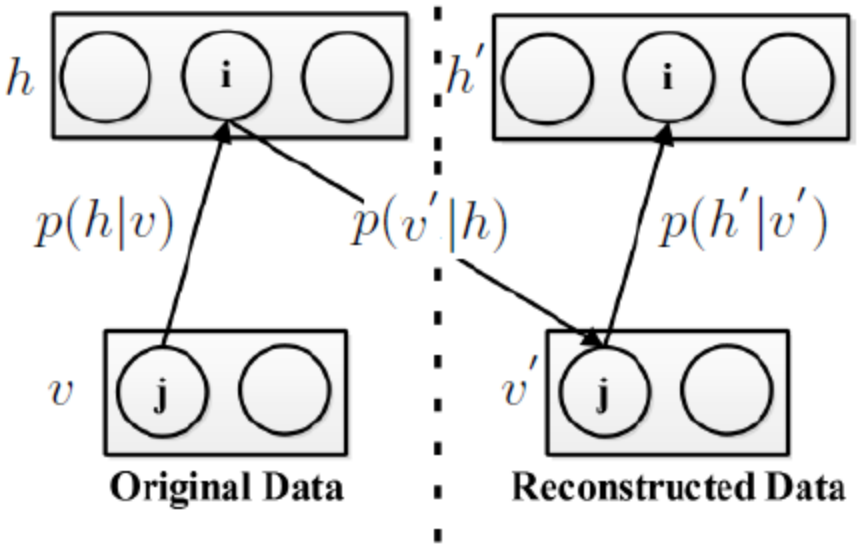

Figure 6.5: Contrastive divergence (CD–1) depiction

analytically. The only term left out in computing the gradient in Eq. 6.6 is the expectation of $\partial E(v,h)/\partial\theta$ over the model distribution $p(v,h)$. Though samples from the model distribution can be generated using Gibbs sampling (Geman and Geman (1984)), computing $p(v,h)$ is intractable due to the partition function at the denominator in Eq. 6.4. Geoffrey Hinton has shown that contrastive divergence (CD) (Hinton (2002)) algorithm can be used to replace this expectation with a point estimate at $p(h'|v')$. Such an estimate captures the direction of the gradient in Eq. 6.6 and works well for all practical applications. A pictorial depiction of the CD algorithm is shown in Fig. 6.5. For every training vector v, compute $p(h|v)$ and get h' by sampling (i.e., For each node in the hidden layer, compute $p(h_i = 1|v)$ using Eq. 6.11 and turn on the node with the computed probability. If the computed probability is greater than a random number selected from a uniformly distributed random variable in the range 0 and 1, make the output value at the corresponding node as 1. Otherwise, set the value as 0. Now compute the value at visible layer and sample v' using the Eq. 6.12. Finally, recompute $p(h' = 1|v')$ using Eq. 6.11). These steps are repeated multiple times (K) and use the final point estimate to replace the model probability $p(v,h)$ in Eq. 6.6. This is K contrastive divergence (CD-K) and Fig. 6.5 shows the depiction of CD for $K = 1$ (CD-1). Thus the main term in computing the gradient in Eq. 6.6 has reduced to $\sum_h p(h|v)\partial E/\partial\theta$, since the $p(h,v)$ in the second term can now be replaced by the point estimate $p(h'|v')$.

$$\begin{aligned}\sum_h p(h|v)\frac{\partial E}{\partial W_{ij}} &= \sum_h p(h|v)(-h_i v_j) \\ &= p(h_i=1|v)(-1\times v_j)+p(h_i=0|v)(-0\times v_j) \\ &= p(h_i=1|v)(-v_j)\end{aligned} \tag{6.13}$$

$$\begin{aligned}\sum_h p(h|v)\frac{\partial E}{\partial b_j} &= \sum_h p(h|v)(-v_j) \\ &= p(h_i=1|v)(-v_j)+p(h_i=0|v)(-v_j) \\ &= \left[p(h_i=1|v)+p(h_i=0|v)\right](-v_j) \\ &= -v_j\end{aligned} \tag{6.14}$$

$$\begin{aligned}\sum_h p(h|v)\frac{\partial E}{\partial c_i} &= \sum_h p(h|v)(-h_i) \\ &= p(h_i=1|v)(-1)+p(h_i=0|v)(-0) \\ &= -p(h_i=1|v)\end{aligned} \tag{6.15}$$

6.3.2 Summary of RBM learning procedure

In the last subsection we have made the necessary derivation needed to train RBM. It turns out from Eq. 6.13 – 6.15 that the gradient for weight (W_{ij}) update is $-[v_j P(h_i|v) - v_j^{'} P(h_i^{'}|v^{'})]$, the gradient for bias update b_j is $-[v_j - v_j^{'}]$ and the gradient for bias c_i is $-[P(h_i|v) - P(h_i^{'}|v^{'})]$. The second term in each of these expressions is due to the expectation over $p(h'|v')$, the point estimate for $p(h, v)$. Also, note that $v^{'}$ is computed using Eq. 6.12 taking vector h computed from Eq. 6.11, for the given input vector v (CD-1). Now, we can use gradient descent to train RBM, where for each epoch we will update based on the average gradient for all training samples. Thus the steps involved in learning RBM are

1. Initialise W_{ij} from samples randomly selected from normal distribution. Also

initialise bias terms (b_j, c_i) to zero.

2. Start with a training vector on the visible units. i.e, the input image of size $m \times n$ is normalised and vectorised. Each component (v_j) is treated as the chance of turning the corresponding visible node ON (set to 1).

3. Update all the hidden units in parallel using Eq. 6.11.

4. Update all the visible units in parallel using Eq. 6.12 to get the reconstruction.

   If the computed probability is greater than a threshold selected from a uniformly distributed random variable which takes values in the range [0, 1], the corresponding node is turned ON. i.e., the nodes are turned ON (set to 1) with the computed probability.

5. Now, try to reproduce h from the reconstructed input $v_j^{'}s$ using Eq. 6.11. Let $h_i^{'}s$ denote these reconstructed nodes.

6. Update the weights and biases (W_{ij}, b_j, c_i) as shown below.

$$\begin{aligned} W_{ij}^{t+1} &= W_{ij}^{t} + \eta \Delta W_{ij}^{t+1} \\ b_j^{t+1} &= b_j^{t} + \eta \Delta b_j^{t+1} \\ c_i^{t+1} &= c_i^{t} + \eta \Delta c_i^{t+1} \\ \Delta W_{ij}^{t+1} &= \mu \Delta W_{ij}^{t} + \alpha [v_j P(h_i|v) - v_j^{'} P(h_i^{'}|v^{'})] \\ \Delta b_j^{t+1} &= \mu \Delta b_j^{t} + \beta (v_j - v_j^{'}) \\ \Delta c_i^{t+1} &= \mu \Delta c_i^{t} + \gamma (P(h_i|v) - P(h_i^{'}|v^{'})) \end{aligned} \tag{6.16}$$

Here η is the learning parameter, μ is the momentum term and α, β, γ are the parameters that decide the weights on the corresponding gradient. In our implementation the parameters are set by empirical estimation and are $\mu = 0.5$, $\alpha = \beta = \gamma = 0.1$

6.3.3 Design and implementation of DBN

The designed DBN for classification of Leukaemia cell lines is shown in Fig. 6.6. The system is trained by considering 3 RBMs and a final fully connected layer.

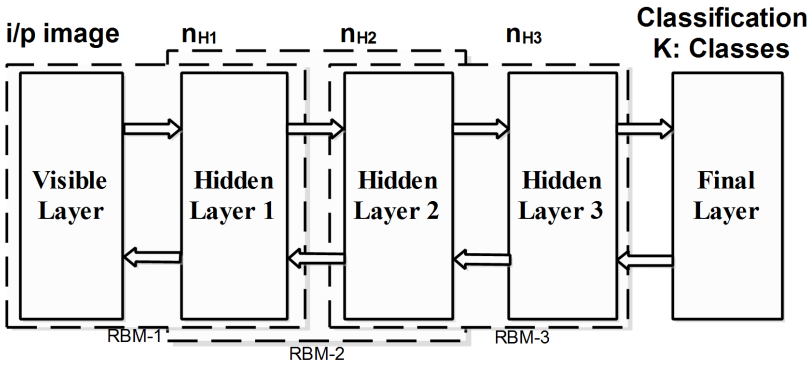

Figure 6.6: The DBN for classification : Depicting the back propagation based fine tuning across the layers

Thus, it has a visible layer, three hidden layers and an output layer. The number of neurons in the visible layer is made same as the number of pixels in the image. Assuming that the input image is of size $m \times n$, the number of neurons is made as mn. In our implementation all roughly segmented cells are made to standard size $m \times n$. The parameters $m = 52$ and $n = 48$ are selected (These are the mean number of rows and columns observed for the roughly segmented cell images). Being a three class classification problem, there are 3 neurons in the output layer. The number of neurons in the hidden layer are $n_{H1} = 100$, $n_{H2} = 300$, and $n_{H3} = 1000$. The number of RBMs in DBN is fixed as 3 for our cell-line dataset after experimenting with 1, 2, 3, and 4 layers of RBMs in the stack. The number of hidden neurons are also fixed in a similar way. This configuration is then trained by choosing different percentage of input data.

Each of the first 3 RBMs are trained independently starting from the first RBM considering all the data (labeled as well as unlabeled) available for training. The algorithm explained in section 6.3.2 is used to train the RBMs. Finally with the labeled data available for training, the parameters are fine tuned. This is done by back propagation algorithm, treating the architecture as a feed forward neural network with 5 layers. As noted earlier, the hidden layer of RBM produces non-linear features and since the subsequent layers operate on these features, they in turn produce high-level feature representations. All the training data is used to learn these features irrespective of whether the samples are labeled or not. It is on top of this pre-trained network, we apply the discriminative fine tuning using the target labels. Since the network parameters are already set during pre-training,

using RBMs in this way overcome some of the problems (such as undesired local minima and slow learning) in learning multi-layer feed forward neural network with backpropagation (Hinton and Salakhutdinov (2006)). The implemented model is tested for classification accuracy, training and testing time. The DBN is also tested for its effectiveness in learning from a mixture of labeled and unlabeled data. The results are discussed in section 6.5.

6.4 CNN as Feature Extractor

As discussed in section 3.3.2, a CNN is a feed-forward artificial neural network mapping an input vector X into an output vector Y. These types of neural networks has proved their greater ability to surpass the skilled human in certain classification task (He et al. (2015)). The discriminative power of CNN is used in a number of medical cases such as in classifying breast tissues (Sahiner et al. (1996); Xu et al. (2014)) and detecting micro calcifications on mammograms (Lo et al. (1995)), classifying interstitial lung disease (Li et al. (2014)), detecting pathologic cases in chest Xrays (Bar et al. (2015)), in thyroid cytopathology (Kim et al. (2016)) and detecting lung nodules in chest radiographs (Lo et al. (1995)). Cireşan has reported the use of deep max-pooling CNN for detecting mitosis in breast histology images (Cireşan et al. (2013)). In all these cases CNN is used as a classifier. We have already discussed about the building blocks of CNN and used it as a classifier in Chapter 3. One of the difficulties in training a CNN for a classification task is that it needs a large dataset. The problem that we are going to address is the Leukemia cell-line classification for which we have only a small dataset. It has only 618 cells (124 K562, 106 MOLT & 388 HL60) which is insufficient to train a CNN classifier. In this section, we discuss the transfer learning capability of the CNN. In transfer learning setting, the knowledge that a CNN has learned for a relatively complex classification task using a large dataset is effectively transferred to a completely different setting. In such setting, CNN is used as a feature extractor and not as a classifier. These features are then used to train a classifier like SVM that needs only training data of moderate size. The transfer learning capability of CNN is studied in Zhang et al. (2015) and utilized in Zeng et al. (2015) to annotate the gene expression patterns in mouse brain.

As noted in section 3.3.2, the main building blocks of a CNN are i) Convolution ii) ReLU and iii) Sub sampling. We will revisit each of these blocks with an emphasis to give intuitive idea on the transfer learning capability of CNN.

Convolutional Layer : For every convolution layer, a number of Kernels are learned during training. A general discussion on CNN learning can be found in Appendix A. Normal convolution is performed between the learned kernels and the input instance but select only valid part of the convolution. This procedure is depicted in Fig. 6.7. We know that the kernels can extract features by convolutions. For example, Sobel kernel find edges, Laplacian kernel detect blobs. Each of these kernel is applied only locally and extract features. Thus depending on the kernel, convolution can extract features like edges, blobs, corners, etc. and are valid features for any images. In CNN, the only learned parameters are the kernel weights and biases, and hence we are learning very local feature detectors rather than the actual features. Suppose that we are learning a CNN for a complex classification task and we have millions of images to learn the classification problem. If we have effectively learned the CNN from such a large training set, it is reasonable to believe that the learned kernels have the capability to extract an exhaustive set of features even to capture the small inter class variability on the original classification problem. Since the kernels learned acts very locally, we can believe that these exhaustive sets of feature detectors are valid for any images irrespective of the classification problem that we are addressing. This fact is the corner stone of transfer learning capability of CNN.

Rectified Linear Unit : ReLU as explained in section 3.3.2 acts as a non linear activation function and it usually follows the convolution block. This will introduce the non-linearity on the output map. Since the subsequent convolutions operate on this non-linear output map, ReLU in turn help to generate non-linear features. It will also enable faster learning by avoiding the problem of vanishing gradients during backpropagation especially at lower layers of CNN (Rohan (2016)).

Sub-sampling : The sub sampling layer helps to learn relevant feature detectors by ignoring small amount of intra class variability in noise, shift and distortion of

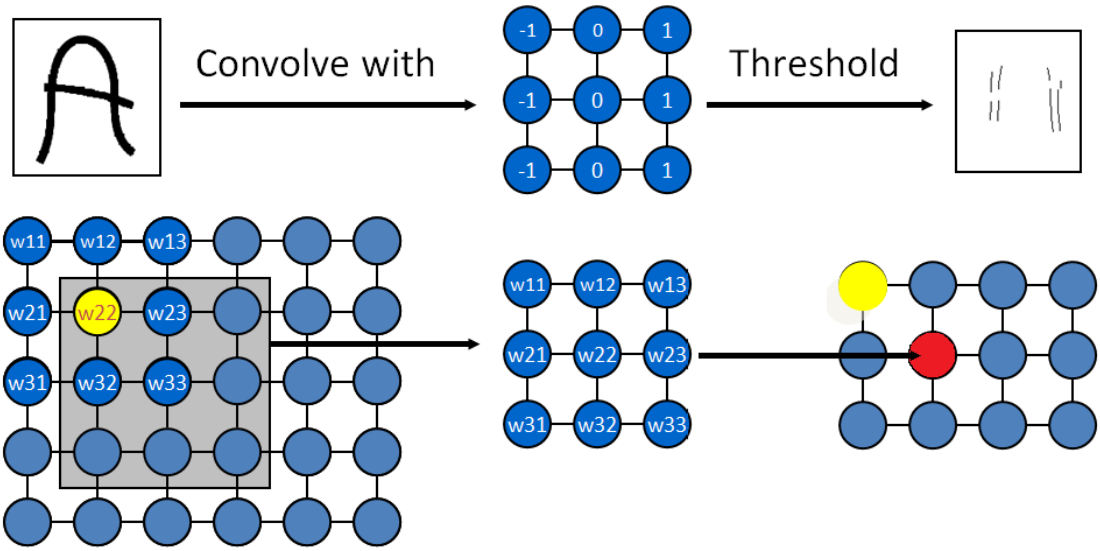

Figure 6.7: Convolution operation: 3×3 kernel on 5×6 image producing an output map of size 3×4

the training samples. The operation is depicted in Fig. 6.8 where sub-sampling by two is shown. It can be seen that the distortion between the two 'A's has reduced as we move for sub-sampling which has reduced the intra class variability between them. Note that these sub-sampling can be done either by taking average of the neighbourhood (average pooling), or by picking max value from the neighbourhood (max pooling).

In a nutshell, a heavily trained CNN for a complex classification task using a large image data set, must have learned an exhaustive set of feature detectors that can capture very local features valid for any images, and have the capability to introduce non-linearity in the detected features due to the architecture involving intermediate ReLU and sub-sampling layers. This enables us to use such a CNN as a feature extractor for our leukaemia cell-line classification problem, and then use these features to build a suitable classifier using the available small training set that we have.

Trained CNN can outperform human in certain classification tasks (He et al. (2015)). As noted, the only bottleneck in using CNN is the need for large amount of labeled data for training, which is often limited in medical domain. In this section, we explore the possibility of doing medical image analysis using a CNN pre-trained on a large scale non medical image database, ImageNet; i.e., we wish to investigate

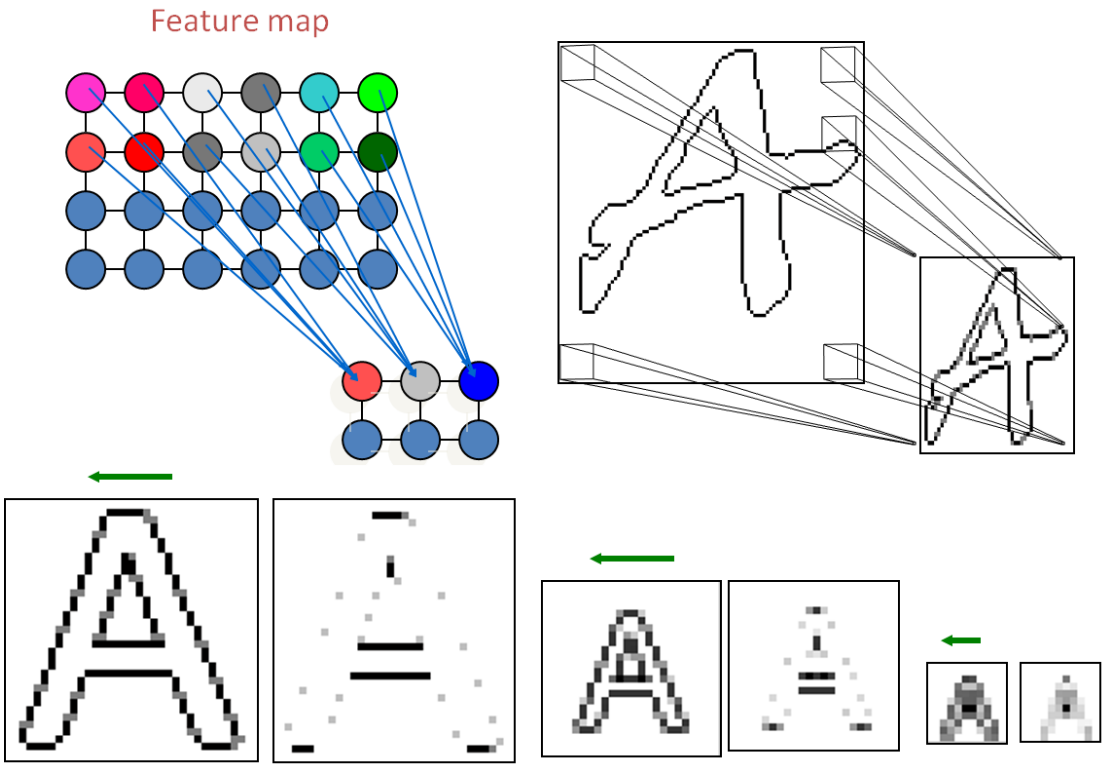

Figure 6.8: Sub Sampling to reduce the output map size by half to reduce the effect of small amount of noise, shift and distortion

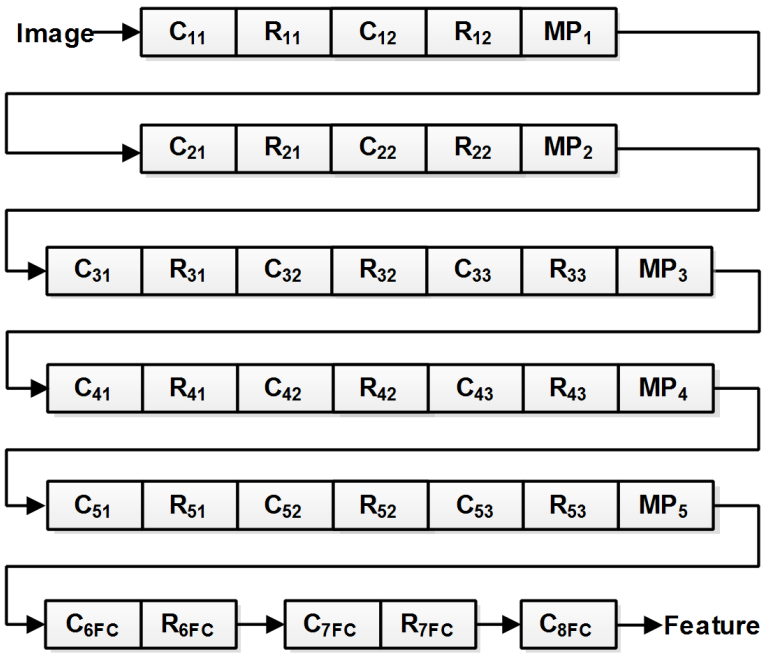

Figure 6.9: CNN architecture pre-trained with ImageNet, used to extract the features for leukaemia cell-line classification

the transfer learning capability of CNN for cell analysis. The ImageNet CNN deep model readily available (Chatfield et al. (2014); Vedaldi and Lenc (2014)) is heavily trained with several hundreds of non medical images (Deng et al. (2009)) for each of the 1000 classes (such as birds, cars, tools, etc.). It has 37 layers. First 36 layers used to generate the cell features for our experiment are shown in Fig. 6.9. The last 3 blocks represent the convolution and ReLU operation in fully connected (represented using the subscript FC) layers (Vedaldi and Lenc (2014)). Note that, the operations in these 36 layers are grouped into 8 blocks where in each block the layer is represented by O_{ij}. The O stands for the operation (Convolution (C), ReLU (R), Max Pooling (MP)), i stands for the block id, and the j stands for the j^{th} instance of the operation O in the block. For each input image, the output of the 36^{th} layer which is a 4096 element vector is taken as the feature. As noted earlier, it is reasonable to believe that these features are generated by an exhaustive set of feature detectors each looking for valid features like edges, blobs, corners etc. very locally in the image. These features are then used to train a classifier that does not require a large training set to determine the decision boundary. We have experimented with Support Vector Machine (SVM), Feed Forward Neural Network (FFN), Naive Bayes (NBS) and K-Nearest Neighbour (KNN) and has shown that the features generated have good discriminative power for the classification of leukaemia cell-lines.

6.4.1 Pre-trained CNN for classification of leukaemia cells

We use the CNN (Fig. 6.9) deeply trained on ImageNet (Deng et al. (2009); Chatfield et al. (2014)) to classify the leukaemia cell-lines. The roughly localised cell image is given as input to the network which will give a 4096 element feature vector at the output. The dimensionality of these vectors is reduced from 4096 to 20 using PCA (Jolliffe (2002)). The feature vectors of the reduced dimensions are used to train the classifiers. The use of CNN which was deeply trained on a specific data set and its use as a general feature extractor in a completely different setting is an example of transfer learning (Bar et al. (2015); Zhang et al. (2015); Zeng et al. (2015)) capability. This can be attributed to be following. The CNN model used was trained on a few million challenging images of 1000 broad categories. In

CNN, the kernel maps are learned by observing local image patches and hence the learned detectors (kernels) look for very local features such as curves, edges, corners and blobs. As most of these features are intrinsic for any image dataset, we can transfer this knowledge for other classification settings. This capability helps us to use deep CNN in classification task even if there is only limited labeled data for training. In such cases CNN is used as a deep feature extractor to find discriminant features. These features are then used with a classifier that does not require large training set in deciding the classification boundary.

6.5 Results and Discussion

In this section we discuss the classification of unlabeled unstained leukaemia cell-lines K562, MOLT and HL60 imaged using the low-cost, high-throughput microscopic imaging paradigm: the microfluidics based imaging flow cytometer. Altogether 618 cells (124 K562, 106 MOLT, and 388 HL60) were used in the experiment. The classification of these cell lines by deep learning networks (DBN and CNN) is compared for training time, testing time, and accuracy. The ability of the system for semi supervised learning is also considered. The classification accuracy achieved is compared with the the SVM based system in chapter 5 where the features reflecting size, shape, texture and complexity of the finely segmented cells are used for classification.

6.5.1 DBN classifier on roughly segmented cell images

In our experiment, we have normalized pixel intensities and treated them as the probability with which the visible nodes are turned ON in RBM. In order to assess the quality of training achieved, cross validation experiments are conducted at different folds ($K = 2, 3, 4, 5, 10$). In one cross validation experiment, training data from each class is randomly divided into K parts, each containing almost the same number of samples. Now, a cross validation test set is constructed by selecting one part from each class. The system is trained by all other samples and tested with this test set. The cross validation testing is repeated K times so that all samples are used for testing in some step. The entire experiment is repeated

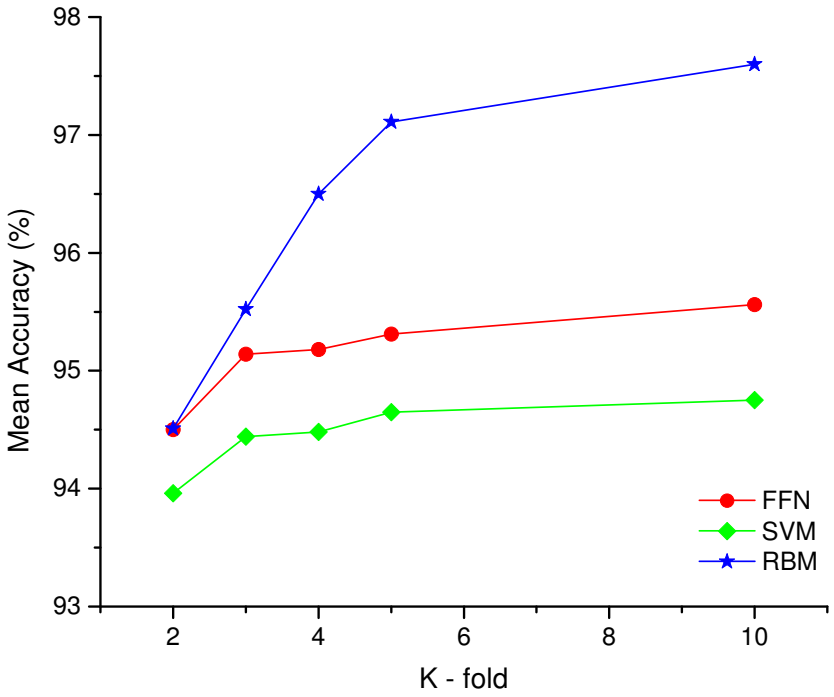

Figure 6.10: Effectiveness of RBM classification on raw cell images

100 times, each time selecting different combinations of samples for training as well as testing. The mean classification accuracy (mean) and standard deviation (std) are computed. The same set of experiments are conducted using SVM with linear kernel and feed forward neural network with back propagation (FFN). Being a three class classification problem, the majority voting based classification strategy is adopted to decide the class label in SVM. The results are shown in Table 6.1. It can be seen that the highest accuracy (with minimum standard deviation) in classifying cell images is achieved by RBM and is shown in Fig. 6.10. For FFN, we have used only one hidden layer. When the number of hidden neurons is varied ($H_n = 5, 10, 25, 50, 100, 250$), the average cross validation accuracy did not improve significantly for $H_n > 10$. So, number of hidden neurons in FFN is fixed as 10 in our implementation.

6.5.2 DBN classifier : ability to learn from unlabeled data

In order to check the effectiveness of RBM based classifier in learning structure of the data rather than learning the labels, the following testing strategy has been adopted. Entire data is divided into 3 sets. (**Case 1**) holding 50%, 30%, and 20% samples respectively and (**Case 2**) holding 67%, 13% and 20% samples

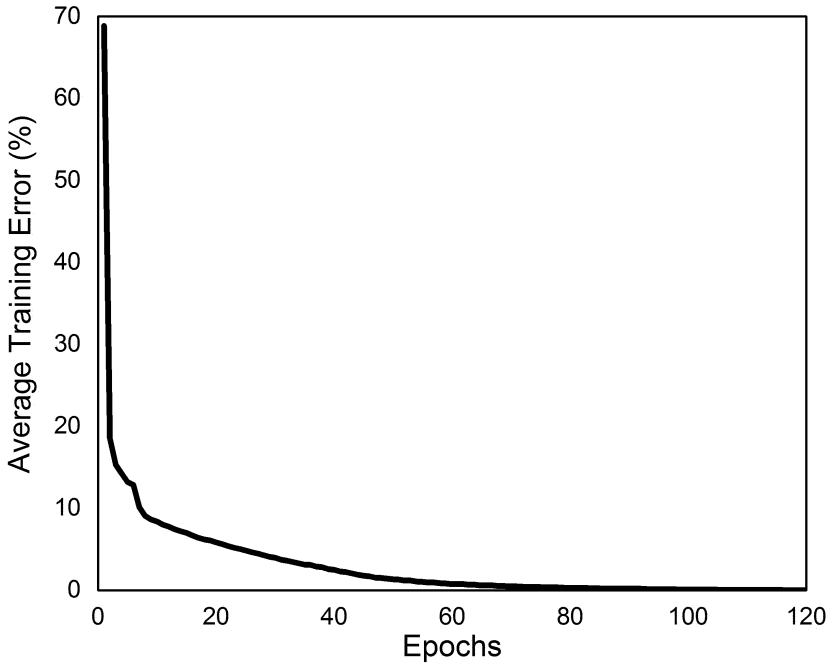

Figure 6.11: Training error of DBN during discriminative fine-tuning

respectively. The first set along with class labels and the second set without class labels are used for training and the system is tested with samples from second and third set. Note that, we have used samples from second set for testing since we did not use their labels during the training process. For both the cases (**Case 1 and Case 2**), the experiment is repeated for 100 times (each time re-initialising the weights) and average classification accuracy (mean) as well as standard deviation (std) is computed. The results are shown in Table 6.2. Note that RBM makes use of samples from set-1 and set-2 for training but uses labeled data (set-1) only during back propagation to fine tune the system. The training error for every pass through the entire training data (an epoch) is computed and the average error for each epoch over 100 iterations is plotted in Fig. 6.11 for **Case 1**. The training error profile is similar to that of the normal feed forward network, and in our case the training converges close to 80 epochs. The result of classification is shown in Table 6.2. The accuracies are slightly better than the results shown in Table 6.1 (2 and 3 fold (i.e., 50% and 67% training data)). Note that the unlabeled data cannot be used for learning FFN and SVM.

Table 6.1: Cross Validation Accuracy in % (mean (std)) - FFN, SVM and RBM on Cell Images

Kfold	**10**	**5**	**4**	**3**	**2**
FFN	95.56 (2.57)	95.31 (1.92)	95.18 (1.73)	95.14 (1.49)	94.50 (1.38)
SVM	94.75 (2.71)	94.65 (1.73)	94.48 (1.64)	94.44 (1.47)	93.96 (1.16)
RBM	**97.60 (0.48)**	**97.11 (0.62)**	**96.50 (0.60)**	**95.52 (0.20)**	**94.51 (0.17)**

Table 6.2: Classification Accuracy — Learn Structure from Data

Cases	Training Accuracy		Testing Accuracy	
	mean	std	mean	std
1	99.98	0.08	95.22	0.71
2	99.99	0.05	95.93	0.86

6.5.3 Classification on features from pre-trained CNN

The architecture of the CNN used to generate discriminative features from the cell images is shown in Fig. 6.9. As noted earlier in section 6.4, output of 36^{th} layer which is a 4096 element feature vector is used for CNN based classification. Before making use of these features, their dimension is reduced to 20 using PCA (Jolliffe (2002)). We have selected the number of principal components starting from 13 (this being the number of features used in chapter 5) and set at 20 since there was no considerable increase in the mean accuracy beyond 20. Once the feature descriptors are generated, the classifiers SVM, FFN, Naive Bayes (NBS), and K nearest neighbour (KNN, $K = 5$) are applied on top of it for 3 class classification. The cross validation experiments are conducted as explained earlier and the results are shown in Table 6.3. These results show that the accuracy is better and consistent across different cross validation experiments by different classifiers. Note that the CNN ImageNet model that we had used to extract features was never trained on the cell images, still producing high classification accuracy. This is a supporting result for the CNN transfer learning capability that we had discussed in section 6.4.

Table 6.3: Cross Validation Accuracy in % (mean (std)) on CNN Features

Kfold	**10**	**5**	**4**	**3**	**2**
SVM	97.80 (1.82)	97.69 (1.38)	97.62 (1.20)	97.47 (1.02)	97.19 (0.87)
FFN	98.26 (1.57)	98.18 (1.16)	98.16 (1.02)	98.05 (0.93)	97.96 (0.72)
NBS	98.40 (1.51)	98.37 (1.08)	98.36 (0.90)	98.40 (0.78)	98.38 (0.53)
KNN	98.42 (1.49)	98.40 (1.04)	98.37 (0.94)	98.35 (0.81)	98.36 (0.59)

Table 6.4: Cross Validation Accuracy in % (mean (std)) on Morphological Features (Chapter 5)

Kfold	**10**	**5**	**4**	**3**	**2**
SVM	95.22 (2.83)	94.47 (1.85)	94.63 (1.52)	94.20 (1.40)	93.57 (1.16)
FFN	92.92 (3.48)	92.74 (2.63)	92.57 (2.35)	92.47 (2.23)	91.76 (3.00)
NBS	90.89 (3.60)	90.77 (2.34)	90.70 (2.18)	90.67 (1.74)	90.35 (1.44)
KNN	80.40 (4.81)	79.93 (3.13)	79.48 (2.90)	78.94 (2.63)	77.75 (0.59)

6.5.4 Comparison with the classification on morphometric features discussed in chapter 5

Table 6.4 shows the result of cross validation experiments performed using cellular features, discussed in chapter 5, by different classifiers. The roughly segmented cells used as input to the RBM are processed further. The cells are localised by finding the cell contour, extracted features and are classified using SVM. The re-

Table 6.5: Run Time Analysis

Method	Rough Loclzn (18ms)	Fine Loclzn chapter 5 (42 ms)	Simple Features chapter 5 (14 ms)	CNN Features Chatfield(2014) (324 ms)	Classifier	Total Time (ms)
Segment & classify (chapter 5)	✓	✓	✓		SVM (1 ms)	75 (ms)
CNN ImageNet (Chatfield(2014))	✓			✓	SVM (1 ms)	342 (ms)
DBN Classifier	✓				RBM (0.05 ms)	18.05 (ms)

sults in Tables 6.1, 6.3 and 6.4 show that the deep learning methods work superior to this method.

Table 6.5 shows the time taken for different processing steps involved in the classification method provided in chapter 5, the ImageNet CNN model (Chatfield et al. (2014)) and the DBN classifier model. It can be seen that the DBN model not only gives good classification accuracy but also a faster response when compared to the other methods.

6.5.5 Performance with small percentage of training data

In order to check the effectiveness of deep learning networks in classification, experiments are carried out, even with less amount of training data and the results are shown in Table 6.6. The entire data set is divided into (S1, S2, S3) such that S1 and S2 together hold the training data (50% of the dataset) while S3 holds the testing data (remaining 50% of the dataset). We assume that the class labels of data contained in S2 are unavailable, and cannot be used for supervised training. Still the data from S2 can be used to train the individual RBMs for initialising the weights. The results of classification i.e., the classification by DBN, SVM (linear) on CNN-ImageNet features, and SVM on morphological features are shown for different training cases; 5%, 10%, 20%, 30%, and 40% labeled training data in S1. The results shown are the mean and standard deviation for 100 runs each time selecting random samples from the set for training. The mean accuracy achieved for the said 100 runs, when used 5, 10, 20, 30, 40, 50, and 75% of the available data for training, is provided in the graph in Fig. 6.12. It can be seen that the classification by deep learning method is effective even for small number of training samples in the case of RBM. Also note that, though the CNN-ImageNet model was heavily pre-trained on non-medical images, the generated features are very good in a classification point of view. These are reflected by the high average accuracy as well as low standard deviation for RBM and CNN in Table 6.6.

Table 6.6: Classification Accuracy: Training with less than 50% Samples

Labelled Data (%)	Morph. Features (chapter 5)	RBM	CNN Features
5	78.29 (3.72)	90.88 (1.18)	91.43 (3.33)
10	85.00 (2.74)	91.40 (1.06)	95.00 (1.40)
20	90.23 (1.49)	93.15 (0.85)	96.36 (0.97)
30	92.06 (1.26)	93.55 (0.61)	96.77 (0.76)
40	93.09 (1.13)	94.01 (0.49)	96.93 (0.90)

6.5.6 Comparison of class-specific accuracy

In order to show that the classification is not too much biased to any class, we report, in Table 6.7, the confusion matrix for each of the system developed. The results shown here are for the 10 fold (K = 10) cross validation experiment. As explained in section 6.5.1, 90% of samples are used for training and remaining 10% of data is used for testing. Such experiment is repeated 10 times such that every samples are used for testing exactly once. Note that Table 6.7 (A) reports the result from SVM running on morphological features (discussed in chapter 5), Table 6.7 (B) reports the results from the classification by RBM and Table 6.7 (C) reports the result from SVM running on CNN features. For example, the first row of 6.7 (A) shows that out of the total 388 (373 + 8 + 7) HL60 cells, 373 are correctly classified as HL60, 8 are wrongly classified as K562, and 7 are wrongly classified as MOLT. We have also presented in Table 6.8, the associated precision and Recall. These measures are based on the number of True Positives (TP), number of False Positives (FP), and number of False Negatives (FN). The precision is defined as $TP/(TP+FP)$ and recall is defined as $TP/(TP+FN)$. As these measures are primarily meant for binary classification, we have considered one-versus-all strategy. Thus we have three cases. In Case - 1, HL60 constitute positive class and K562&MOLT together forms negative class. Similarly, Case - 2 is K562 Vs HL60&MOLT, and Case - 3 is MOLT Vs HL60&K562. These measures tell us how much the system has learned to pick a particular class. The result in Table 6.8 shows that there is significant improvement in both measures with deep networks when compared to the method discussed in chapter 5.

Table 6.7: Comparison of Class Specific Accuracy

A. Morphologic Features (chapter 5)				B. Classification by RBM			C. CNN Features		
	HL60	**K562**	**MOLT**	**HL60**	**K562**	**MOLT**	**HL60**	**K562**	**MOLT**
HL60	373	8	7	382	2	4	385	1	2
K562	4	116	4	1	120	3	1	119	4
MOLT	2	5	99	3	2	101	1	5	100

Table 6.8: Comparison of Precision and Recall in % (One Against All)

	Case - 1		Case - 2		Case - 3	
	Precision	Recall	Precision	Recall	Precision	Recall
Method (chapter 5)	98.42	96.13	89.92	93.55	90.00	93.40
RBM	98.96	98.45	96.77	96.77	93.52	95.28
CNN Features	99.48	99.23	95.20	95.97	94.34	94.34

From the results shown so far, it is understood that RBM and CNN bypass the step of extracting hand-engineered features and performs better job than using a few important features (discussed in chapter 5). RBM based classifier not only classify the data but also extract the structure of the data while CNN extract discriminative features of the data. Further, the proposed deep learning based methods are quite general that it can be used for cell classification studies in microfluidic microscopic setup without the need for precise segmentation as well as explicit feature extraction, still producing an improved level of accuracy. The system could better capture the inter class variability of the cancerous cell-lines compared to the morphometric and textural features explored in chapter 5 and has significantly improved the precision as well as recall of the system when compared to the PCA signature based methods (Jagannadh et al. (2016)) and the SVM trained on morphometric features.

The leukaemia dataset used in this research is prepared by a custom-built, cost-effective microfluidic microscopy system and the accuracy reported on this dataset in this chapter set the benchmark for future studies. The central idea of the proposed method is to employ inexpensive optofluidic instrumentation for automated image acquisition and subsequent deep learning based cell recognition. While the opto-fluidic architecture enables cost-effective automation of image acquisition, the proposed deep learning based system enables implementation of effective de-

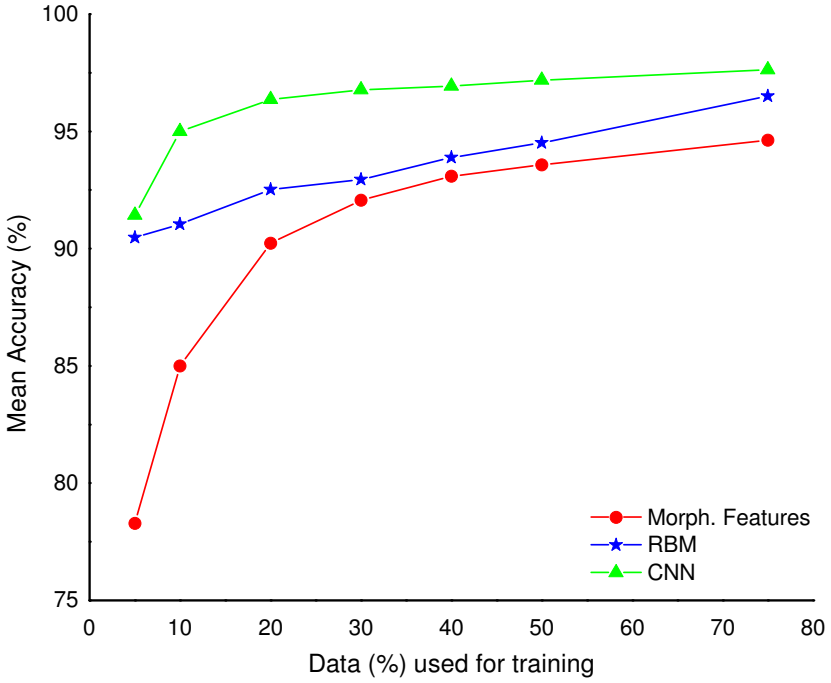

Figure 6.12: Classification accuracy of RBM, and SVM on features generated by CNN–ImgeNet as well as on the morphometric features (chapter 5).

cision making even in resource limited settings, where personnel trained in the art of diagnostic decision making are scarce/unavailable. As demonstrated, the image extraction and identification perform with fair level of accuracy thereby moving a promising step towards implementation of good quality health care even in resource-limited settings.

6.6 Publications

1. **G. Gopakumar**, K. Haribabu, Deepak Mishra, S.S. Gorthi, G.R.K.S. Subrahmanyam. "Cytopathological image analysis using deep learning networks in microfluidic microscopy", *J. Opt. Soc. Am. A*, **34(1)**:111–121, 2017.

6.7 Summary

In this chapter, we have proposed an approach for automatic cytopathologic analysis using deep learning methods. The DBN is found effective compared to the

classifier system proposed in chapter 5. The RBM model not only improved the classification accuracy but also avoided the more demanding accurate segmentation of cells. We have also noted the capability of RBM based system for learning structure of the data rather than learning labels which will be very helpful in medical image domain where often large dataset is available for training but only a small fraction labeled. We have also studied the applicability of CNN for cell analysis and found that a readily available CNN extensively trained on non medical image database ImageNet produces good discriminative features for classifying the leukaemia cell lines K562, MOLT, and HL60. In our investigation, we have found that deep learning methods outperformed the conventional systems in the classification of these cell lines. To the best of our knowledge, such a reporting on cytopathology images is first of its kind and we believe that it holds great promise in terms of enabling cost-effective cancer screening in resource-limited settings.