CHAPTER 6

CLASSIFICATION USING DEEP LEARNING CLASSIFIER

Phase II of the research methodology focused on proposing an ALL-C system that used machine learning classifier. Phase III, on the other hand, proposes techniques that use deep learning classifier to perform ALL-C. This chapter presents details regarding the concepts of deep learning classifier, followed by the methods proposed to enhance the working of CNN-based deep learning classifier.

6.1. DEEP LEARNING CLASSIFIER

Deep learning can be defined as a model that can learn and improve automatically by analysing algorithms. It uses neural networks to simulate human brain along with input data, weights and bias, for making decisions. The popular description of DL is that it is a type ML that is motivated by the structure of human brain (Sundaram *et al.*, 2021). To accomplish this, the DL uses multi-layered structures of algorithms, called Neural Network (NN). A conventional NN is presented in Figure 6.1.

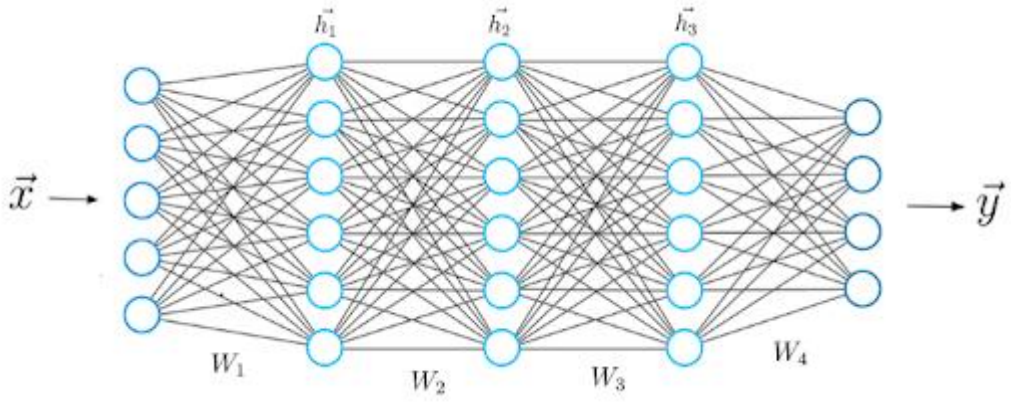

Figure 6.1 : Neural Network

The design of NN involves a learning algorithm that can identify and classify patterns and different kinds of data, just like a human brain. The layers of NN works like a filter that converts raw data to a more refined form, with the motive of increasing the classification accuracy. The NNs have unique capabilities that enable DL to solve tasks that is challenging for a ML to perform.

As mentioned earlier, the NN is inspired by the biological neurons of human brains. The NNs simulate the basic functionalities of the brain neurons (Mirjankar and Patil *et al.*, 2018). The biological neuron has several neurons (Figure 6.2), where each neuron has a cell body, dendrites and an axon.

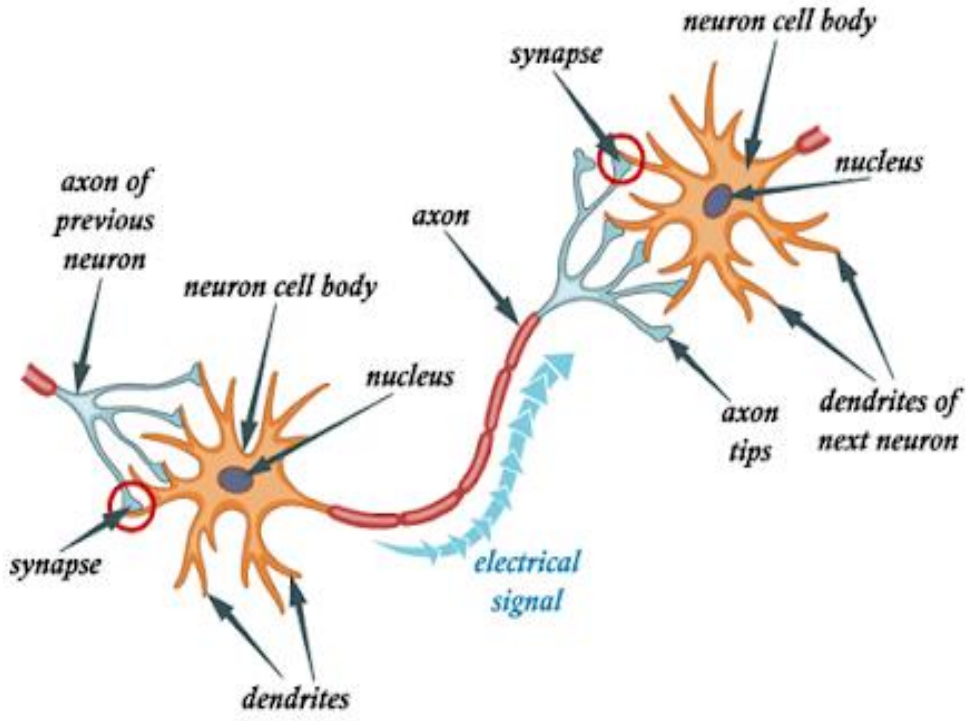

Figure 6.2 : Biological Neural Network

Both dendrites and axon emerge from cell body. Dendrites are thin structures, while axon are cellular extensions. The neuron in the brain receive signals through the dendrites and sent out using axon. The signals more from axon of one neuron to the dendrite of another through synapses. The membrane of each neurons maintains voltage gradients and hence all neurons can be electrically charged. If this voltage change is high in a short time gap, then the neuron generates an action potential, which is an electrochemical pulse. This potential travels quickly through the axon and activates synaptic connections.

The ANN is build using the above concept of biological NN. An ANN consist of a set of connected nodes, which are called neurons, which loosely model the biological NN (https://builtin.com/machine-learning/deep-learning) (Figure 6.3).

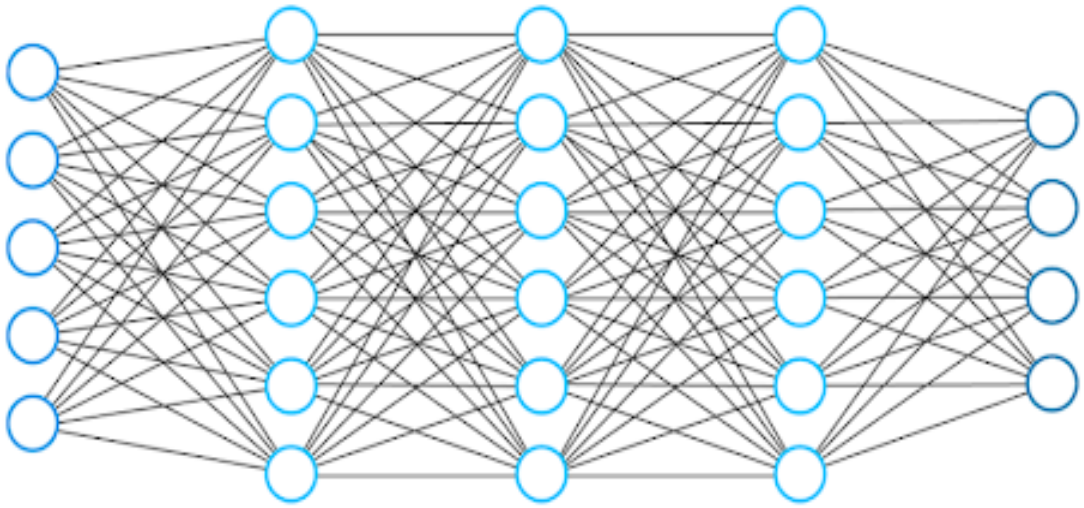

Figure 6.3 : Artificial Neural Network

In this architecture, the neuron can be described as a graphical representation of a numeric value, and the connection between two ANNs is considered as an axon. The connections between the neurons are realized using numerical values, called weights. During the learning process, the weights between neurons vary, which strengthens the connections. That is, given a training set along with classification task, the ANN searches for a set of weights that allow the NN to perform the classification task. The set of weights vary from application to application and even data sets. The weights cannot be pre-determined but has to be determined dynamically by the learning part of NN. This process is called training.

The ANN has three types of layers, namely, input layer, hidden layer and output layer. These multiple layers of interconnected nodes is built upon previous layers to refine and optimize the classification task. This progression of computations through the NN is called as forward propagation and the architecture is named as feed forward NN (Figure 6.1). The input and output layers are also called as visible layers. The input layer receives the input vector that is used by NN to learn. The number of neurons in input layer is equal to the entries in the input vector. In other words, each input neuron is used to represent an element in the input vector. The output layer is the place where the final classification results are obtained. This neurons of the output layer has the values of the output neurons. With the task of classification, each neuron in the output class represents the different target classes.

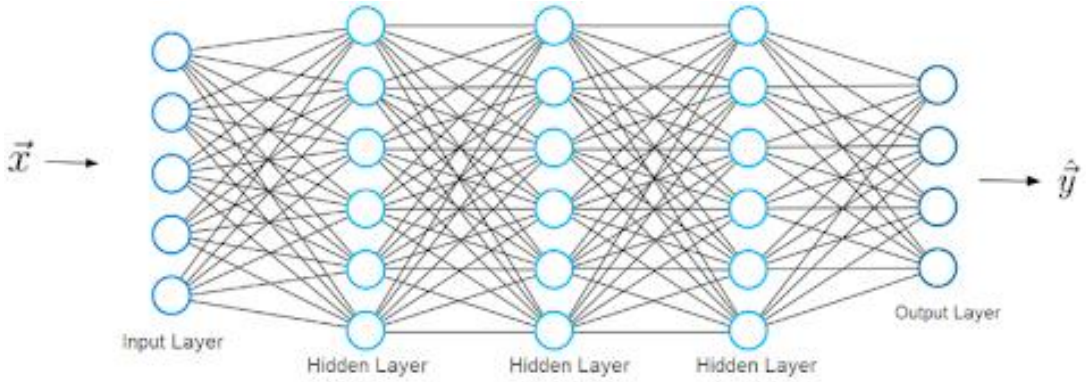

Figure 6.4 : Feed Forward Neural Network

In order to obtain an output using the input, the NN performs several mathematical operation that are perform by layers in-between input and output layers. These layers are called as hidden layers. The hidden layer has neurons that process the input and transfer the result to the next layer. This is repeated until the results reach the output layer.

As previously mentioned, the connection between neurons is represented using a weight (W), which has an index associated with it. The index is represented using a pair of values, where the first presents details regarding the number of neurons in the layer from where the connection starts and the second provides details regarding the number of neurons in the layer to which the connection moves towards. All weights between two NN layers are denoted as a 2-dimensional matrix called Weight Matrix (WM) (Equation 6.1).

$$WM = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix} \tag{6.1}$$

The number of entries in WM is equal to the number of connections between neurons. Thus, the size of WN is obtained from the size of the layers that are connected using this WM. In the matrix, the number of rows matches to the number of neurons in the layer from where the connection originates and number of columns is equal to the number of neurons in the layer to which the connection moves towards.

The most vital part of DL is the learning task, also called as forward propagation. Let x be the input feature set and h be the classification output. Using x and WM connecting two neuron layers, a dot product between x and WM is computed (Equation 6.2).

$$\vec{x}^{T} \bullet WM = (x_1, x_2) \bullet \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix}$$

$$= (x_1 w_{11} + x_2 w_{21}, x_1 w_{12} + x_2 w_{22}, x_1 w_{13} + x_2 w_{23}) \quad (6.2)$$

$$= (z_1, z_2, z_3) = \vec{z}, \vec{h} = \sigma(\vec{z})$$

Let the result be represented as z. The final output h is obtained by using activation function on z. The activation function is represented by the word 'sigma'. An activation function is a non-linear function that performs a non-linear mapping from z to the output h. The activation function in DL can be tanh, sigmoid and ReLu. Each element of z consists of x. A value of a neuron in a layer consists of a linear combination of neuron values of the previous layer weighted using numeric values. As mentioned earlier, these numerical values express the strength of neurons connected with each other. The weights are adjusted during the training phase, due to which some neurons become more strongly connected, while some lose connection. Change in weight changes the value of z and h, which in turn, changes the final output vector.

6.2. DLC USED IN THE RESEARCH WORK

Researchers have used several types of deep learning algorithms to design a classifier. Out of them, the CNN is more popular, as it decreases the need for human interference during preprocessing and development of its functionality, easy to understand and can be implemented quickly. Moreover, CNN has been proved to produce high accuracy among several deep learning classifiers (Effati and Nejat *et al.*, 2023). Due to these advantages, the final phase of the research work adopts CNN to perform ALL-C.

Phase III begins by building a base model, which is then enhanced by combining it with ensemble SVM classifier. The methodology used in Phase III is presented in Figure 6.5. The first step, preprocessing, consisting of two tasks, namely, enhancement (contrast enhancement, noise removal, edge enhancement) and segmentation, were performed using the algorithms proposed in Phase I of the research methodology (Chapter 4, Design of Preprocessing Algorithms). The rest of the steps are described in the following sections.

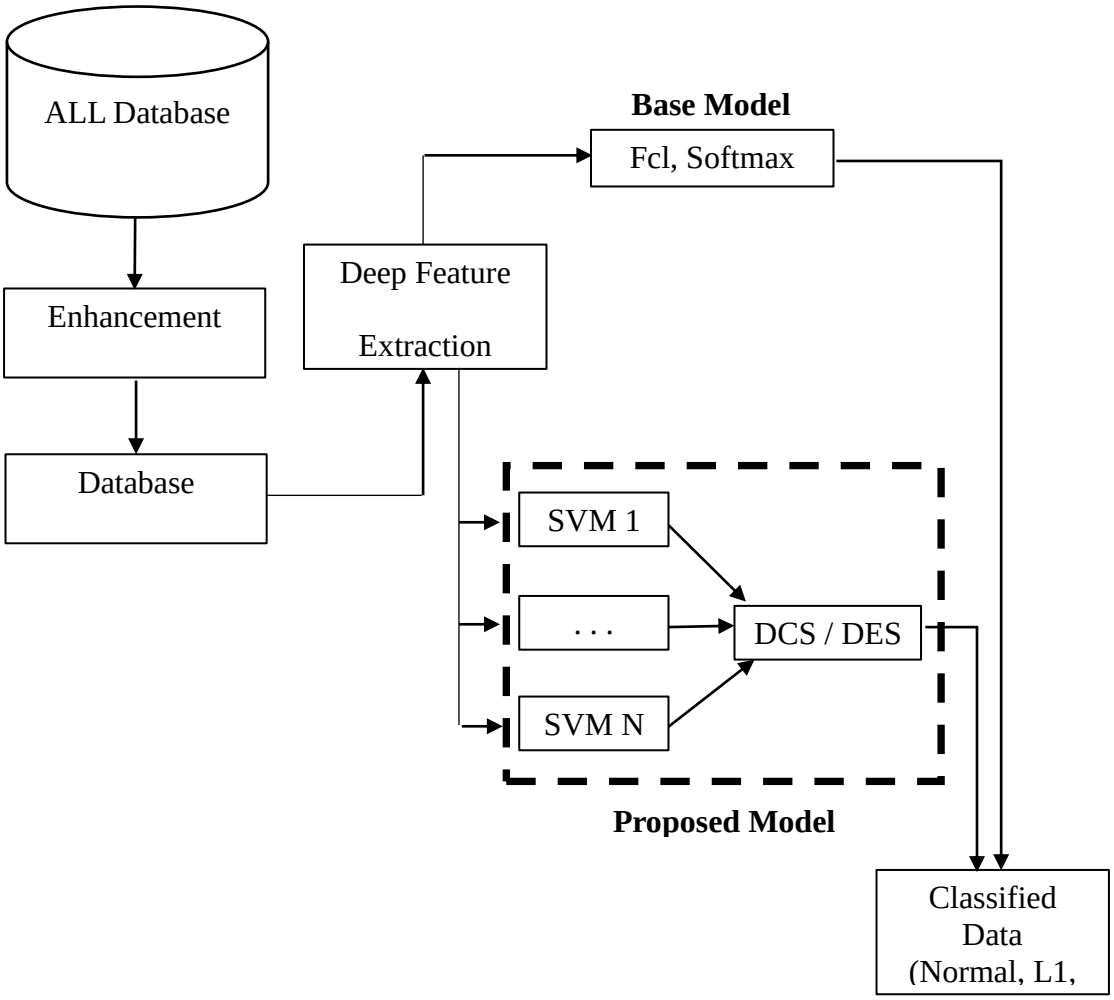

Figure 6.5 : Phase III Methodology

6.3. DATABASE PREPARATION

As mentioned in Chapter 3 (Methodology), one main drawback of using CNN is the overfitting that occurs when the training set size is small. Augmentation method is used to solve this problem. Augmentation methods are frequently used to increase the size of the dataset and avoid memorization, and are more popular with deep learning classifiers. The augmentation method uses image transformation methods to obtain several variants of the same image, thus increasing the dataset size. Several researchers have proved that the CNN classifier has decreased error rates while using augmentation methods by providing better generalization (Mikolajczyk and Grochowski, 2018; Shijie *et al.*, 2017). As suggested by Ahmed *et al.* (2019), seven transformation methods were used in this research work. They are, rotation, height shift, width shift, zoom, horizontal flip, vertical flip and shearing (Table 6.1).

TABLE 6.1

TRANSFORMATION METHODS

Method	Description	Value Used
Rotation	Rotating the image in random direction (left or right) according to the degree value specified between 0–180. In this study, the degree value was selected at 40 to obtain various images.	40°
Height Shift	Randomly shift image pixels up or down.	40%
Width Shift	Randomly shift image pixels left or right.	40%
Zoom	Done by adding new pixel values to the original image, in order to make the image closer. When adding the values, the original pixel values were examined and the nearest value was determined.	30%
Horizontal Flip	Image pixels were moved horizontally from one half of the image to the other half.	
Vertical Flip	Image pixels were moved vertically from the centre of the image.	
Shearing	Done by shifting the image pixels counter-clockwise according to the specified angle in degree.	20°

These methods are applied on the database microscopic images, which originally had a total of 260 images (130 normal and 130 having leukaemia). Additionally, 50 more microscopic blood images for typical L1, L2 and L3 type cells were downloaded through Google image search and added to the dataset. Thus, the original dataset had 130 normal images and 180 images having leukemia. After augmentation, the size of the dataset increased to 1,760 images (260 normal and 1,500 leukemia affected). Out of the 500 affected images, 666 belonged to L1, 453 belonged to L2 and 381 belonged to L3 types. One side effect of augmentation was that the each category in the dataset became unbalanced. This was solved by equalizing the training set by determining the smallest amount of images in each category. The above step makes each category in training set to have exactly the same number. An example of a microscopic image after augmentation is given in Figure 6.6.

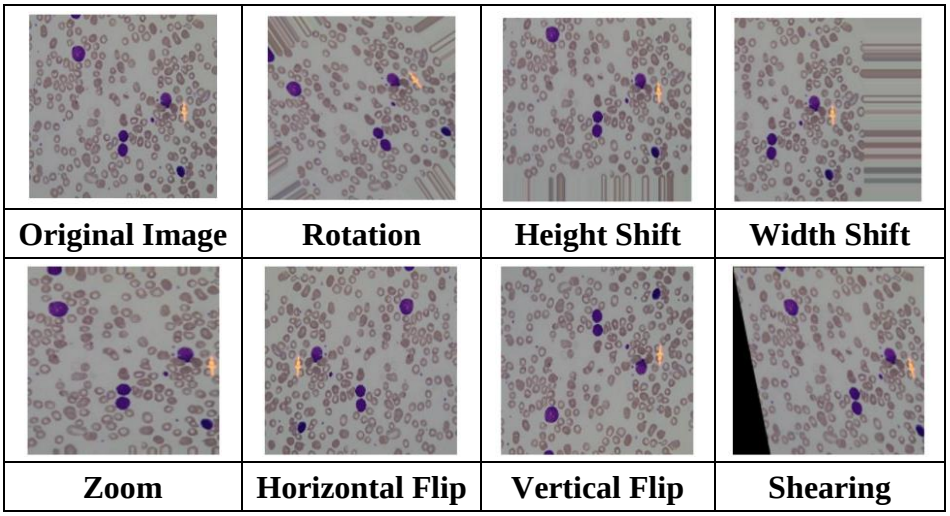

Figure 6.6 : Microscopic Image After Augmentation - An Example

6.4. CNN BASE MODEL

The CNN classification model consist of six layers, namely, input layer, hidden layers (convolution layers, pooling layers, flattening and MLP or multilayer perceptron) and output layer (Simonyan and Zisserman, 2015). The back bone of the CNN model are the filters also called as kernels. The filters provide the CNN model the ability to perform automatic feature extraction and results with a feature map. The feature map thus created is a low dimensional version of the original image with all its original important characteristics intact. Feature extraction is performed by the convolution layer using convolutional operations and pooling layer. Each layer has a set of non-linear functions of weighted sums at different coordinates of spatially nearby subsets of outputs from the previous layer, which allows the weights to be reused. After applying filters on the image in these layers, the features are obtained and then the second step, classification is performed. The base CNN model used in this research is presented in Figure 6.12.

The raw pixels of the input microscopic image represented as a 3-dimensional matrix is first presented to the input layer. The three dimensions W x H X D indicates the width, height and number of colour components of the input image.

The second type of layers are the hidden layers, whose values are not observable in the training set. There are four types of hidden layers, namely, convolutional layers, ReLu (REctified Linear Units) layers, pooling layers and fully connected layer. Each layer can

occur any number of times and the order of these layers are not fixed. However, the layers need to follow the rules given below (https://levity.ai/blog/how-do-image-classifiers-work).

1. The neural network should start with the convolutional layer and end with the fully connected layer, and
2. The fully connected layer should be preceded by the final pooling layer.

An example of the different layers in a CNN model can be similar to the one given below.

Input→ Convolution → ReLu → Convolution → ReLu → Pooling → ReLu → Convolutional → ReLu → Pooling → fully connected → Output

The CNN classifier begins with accepting the numerical pixel values of a microscopic image as input and passes it through the CNN, to obtain the final result, which can be a single class or a probability of classes the best describes the image. Each layer has a specific task associated to it.

The convolution layers are the most important part of the CNN classifier. In mathematical terms, convolution is the combination of two functions to produce a third function. This layer accepts an input, applies a filter and results with a feature map. The feature map is a combination of input and the filters and thus, derived the name convolution layer. Each feature map corresponds to a visual feature seen at specific locations within the input image. When used with classification model, the convolution layer is used to extract features of the microscopic image. The features extracted a stored in a matrix format having numerical pixel values. This matrix serves as a feature detector. An example of an image having a vertical line is shown in Figure 6.7.

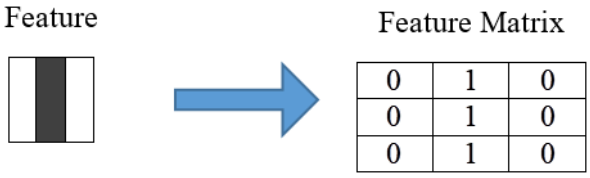

Figure 6.7 : Filter Matrix - An Example

To scan the image, the filter matrix traverses across the image, pixel block by block and for each block, a value that is based on how goodness fit between filter and

image, is estimated. A simple multiplication of the two matrices is used during estimation (Equation 6.3). This means, a better match results with a higher value in the resulting feature map. Figure 6.8 shows the image transformed to a feature map by using product of image matrix with filter matrix. For example, the value 1 obtained in the top left element of the feature map is calculated as 0*0 + 0*1 + 0*0 +0*0 + 0*1 + 0*0 + 0*0 + 1*1 + 0*0 =1.

Feature Map = Image matrix x Filter Matrix (6.3)

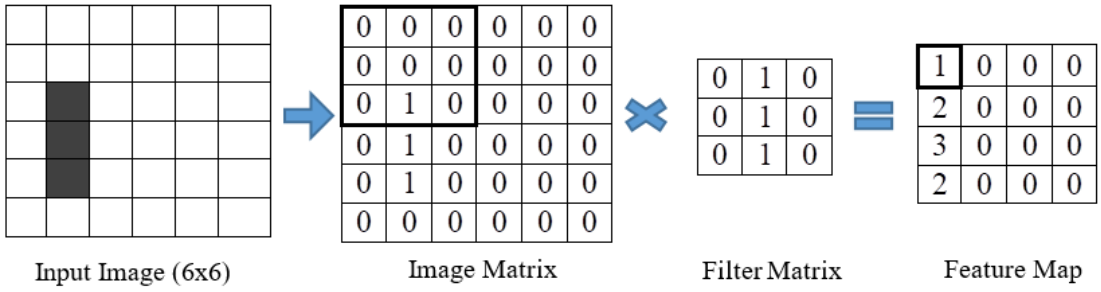

Figure 6.8 : Image Transformed to Feature Map - An Example

Filter use two hyper-parameters that will influence the size of the output volume of each filter (https://in.mathworks.com/discovery/deep-learning.html). They are stride and padding. Stride refers to the distance that the filter moves at a time. For example, if stride = 1, then the filter moves one pixel at a time, over the input microscopic image. A zero-padding scheme 'pads' the edges of the output with zeroes in order to preserve spatial details of the microscopic images.

The ReLU, also called as the activation function, is used to add non-linearity to the computations. It accepts the feature map as input and rectifies any negative values to zero, leaving the positive values the same (Equation 6.4).

ReLU '(x) = {0, x < 0 | 1, x > 0} (6.4)

Activation functions determine the relevancy of a given node in a neural network. An example is shown in Figure 6.9. Non-linearity, in this scenario, means that the slope is not a constant and the ReLU layer is non-linear as the function's slope is either 0 (for x <0) or 1 (x>0) (Figure 6.10). An important advantage of the ReLU is that it has a derivative

function that allows for backpropagation and network convergence happens very quickly. It is computationally efficient and incredibly simple.

15	20	-10	35
18	-110	25	100
20	-15	25	-10
101	75	18	23

Transfer Function

15	20	0	35
18	0	25	100
20	0	25	0
101	75	18	23

Figure 6.9 : ReLu Layer

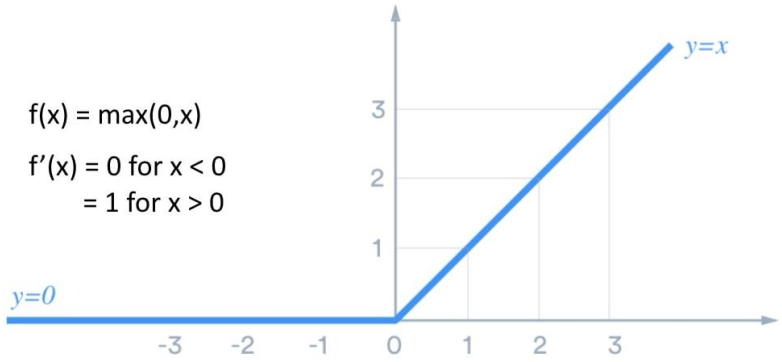

Figure 6.10 : Slope of Activation Function

Next is the pooling layer, where a filter is used to traverse the input matrix and assign a single unique value to each subregion to a new output matrix. As the number of feature map increases, the dimensionality and parameters used also increased, this may result in overfitting. This situation is solved using pooling. The main goal here is to downsize the image. Further, pooling layers also help to reduce the computational complexity, thus increase speed of classification. Pooling layers accepts as input the feature maps and performs down-sampling. This can be performed in two manners. The first is max-pooling and global average pooling. In this research work, max pooling is used.

The max pooling layers uses the kernel width, height and stride as input. It starts at the top-left corner of the feature map and traverse over the pixels to the right using the stride value. The pixel that has the maximum value contained within the kernel window is used as the value for the corresponding region in the pooling layer. In simple terms, the max pooling filter results with the maximum value of subregion of the input (Figure 6.11).

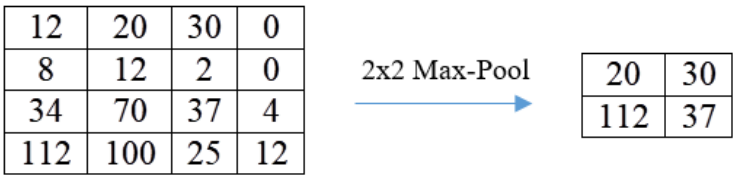

Figure 6.11 : Max-Pooling

A 2 x 2 max pooling filter runs over a 4x4 matrix and takes the maximum value of the specific region and results with a 1x1 value. For example, in the above Figure, the highest value among the four boxes in the top left is 20. This number is assigned as a single field in the output matrix. This means that four boxes are replaced by one box, thus achieving a data reduction of 75%. The filter then moves to the next block of four boxes, without overlapping. This process results with an output matrix where each value corresponds the maximum value in each associated block or sub-region.

Finally, the fully connected layer, concludes the CNN classification model. This layer uses the output of the last pooling layer as input and combines all results to generate the final classification result. Here, each value of the input vector represents the likelihood of a feature belonging to a specific class. For example, one value might suggest that a sub-region might belong to Leukemia region at a 90% likelihood, while another value might correspond to L1 Leukaemia. The fully connected layer accepts all these information, applies it to a weight and presents the final classification result. This trained network is now ready to perform ALL-C.

Thus, the CNN architecture is a specific arrangement of the above-described layers. Several variations of such arrangement are used to design different CNN architectures. The most popular ones include LeNet-5, AlexNet, ZFNet, GoogleNet, Inception, VGGNet and ResNet. Out of these, this research work uses AlexNet CNN architecture. The AlexNet model, the first convolutional network designed by Alex Krizhevsky won the ImageNet Challenge in 2012 and proved that the performance of image classification can be improved (Krizhevsky *et al.*, 2012). It has five convolutional layers, three max-pooling layers, two normalization layers, 2 fully connected layers and one softmax layer. Each convolutional layer has convolutional filters along with the non-linear activation function ReLU. The pooling layer are used to perform max pooling. Due

to the presence of the fully connected layer, the input size is always fixed with AlexNet. The popularly used input size is 227x227x3.

The input for the proposed AlexNet CNN model are the RGB coloured images with 227x227 pixel resolution. The proposed AlexNet is designed with 5 convolutional layers with 3 max polling layers. Each convolutional layer is followed by ReLU. For transfer learning, the last 3 layers (fully connected layer, softmax layer and classification layer) of the pre-trained AlexNet are used.

Next, the proposed AlexNet architecture was fine-tuned to perform ALL detection and classification, where the data is first classified into normal or abnormal. For this, another fully connected layer with 1024 neurons was added followed by ReLU layer in order to reduce the feature vector size. All units in this layer are fully connected to 2-class output (normal, abnormal) probability using softmax function. In the next step, the abnormal category is further classified into L1, L2 and L3 subtype categories. For this, the last fully connected layer is modified from 2 class to 4 class output probability, while keeping the rest of the layers the same. The rate of network change is dependent on the learning rate. The default learning rate of 0.01 was used.

In the proposed base CNN model, the learning rate of the previous layers were not changed, but the learning rate of the newly added three layers were increased, so as to make them perform the updation faster than the original previous layers and consequently, learn the new weights of these three layers in a fast manner. In this research work, the batch sized used was 28 with max epochs set to 100. All the parameters including the filter size, the number of filters, and stride for each layer used in the base CNN model are presented in Figure 6.12.

6.5. PROPOSED HYBRID CNN-ENSEMBLE SVM CLASSIFICATION MODEL

Through experiments, it was found that the CNN base model, described in the previous section, outperforms the conventional SVM-based model in terms of accuracy and its performance is in par with enhanced ensemble SVM from Phase II. However, CNN has more steps, so, the time needed to run it is longer than SVM. The CNN models are very good at automatically learning the optimal features. However, its performance is not

always the best during classification, as the fully connected layers use parameters that have to be fine-tuned manually. To solve this issue, hybrid model that combine CNN and SVM have been proposed by several researchers (Kang *et al.*, 2018; Liu *et al.*, 2018). These systems use CNN for feature extraction and SVM for classification. This research work, moving in the same path, also combines CNN and SVM and includes optimization procedures that can improve its performance. The proposed hybrid model combines the best qualities of CNN and SVM to improve ALL-C.

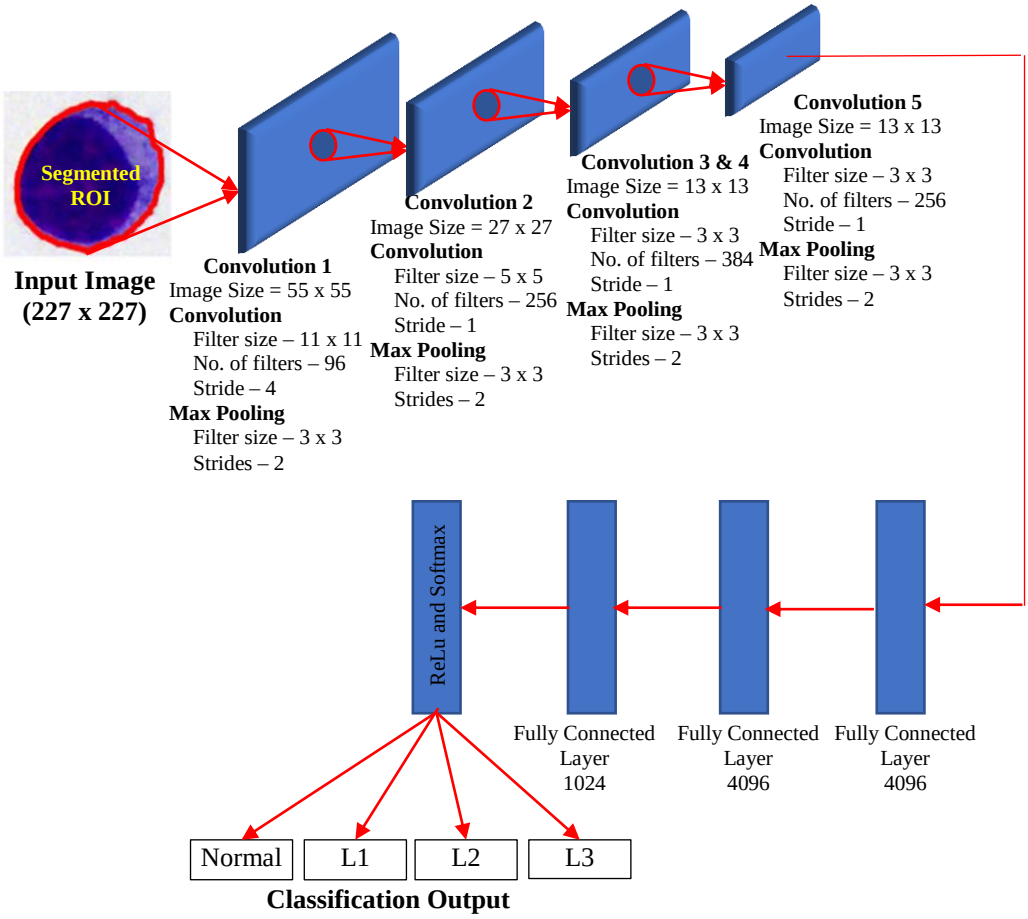

Figure 6.12 : CNN Base Model

6.5.1. Conventional Hybrid CNN-SVM Model

The conventional hybrid model begins by sending the segmented image to the input layer. The CNN with the output layer is then trained with several epochs until the training process converges. At this point, the hybrid model deviates from the CNN classifier. Here, the SVM with RBF (Radial Basis Function) replaces the output layer. The

SVM classifier treats the output of the hidden layer as a new feature vector and uses it for training. After training the SVM classifier, the ALL classification can be performed on test images with automatically extracted features.

Thus, in the hybrid model, all the layers of Figure 6.12 are used as it is, except for the last three layers, which are replaced by the training and testing of SVM classifier. The CNN performs feature extraction, and the input to any fully connected layer could just as well be used as the input to a SVM classifier. The conventional manner of combining CNN and SVM is shown in Figure 6.13.

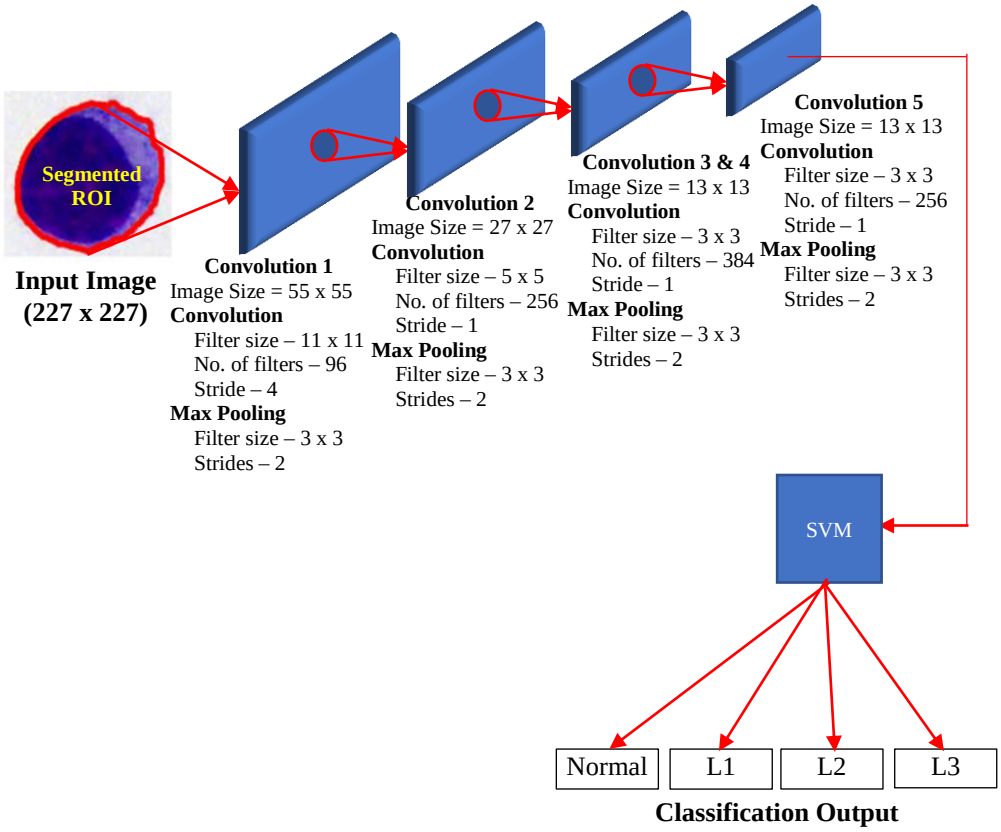

Figure 6.13 : Conventional Hybrid CNN-SVM Classifier

6.5.2. Enhanced Hybrid CNN-Ensemble SVM Classifier

As mentioned in Phase II of the research methodology, ensemble systems perform better than a single classification system. Therefore, the proposed hybrid system is designed to combine CNN with ensemble SVM. It was noted by several researchers that combining different kernel types can make the ensemble classifier stronger than using a

homogeneous SVM trained with different feature vectors but using the same kernel function (Stork *et al.*, 2015; Krishnaveni and Radha, 2021). Using this finding, the ensemble SVM component of the proposed hybrid system is designed to use several SVMs, each differing with the type of kernel function used.

The kernel functions used to design the heterogeneous SVM classifier component of the proposed hybrid system are Linear Kernel, Polynomial Kernel, Gaussian RBF Kernel, Exponential Kernel, Laplacian Kernel, Bessel Function Kernel, ANOVA RBF Kernel and Hyperbolic or Sigmoid Kernel**.** Another point from Phase II is that not all base classifiers in the classification system help with classification. For this purpose, the hybrid CNN-ensemble SVM classifier is enhanced to include the selection step described in Phase II. The general methodology used by the proposed hybrid CNN-Ensemble SVM classifier is shown in Figure 6.14.

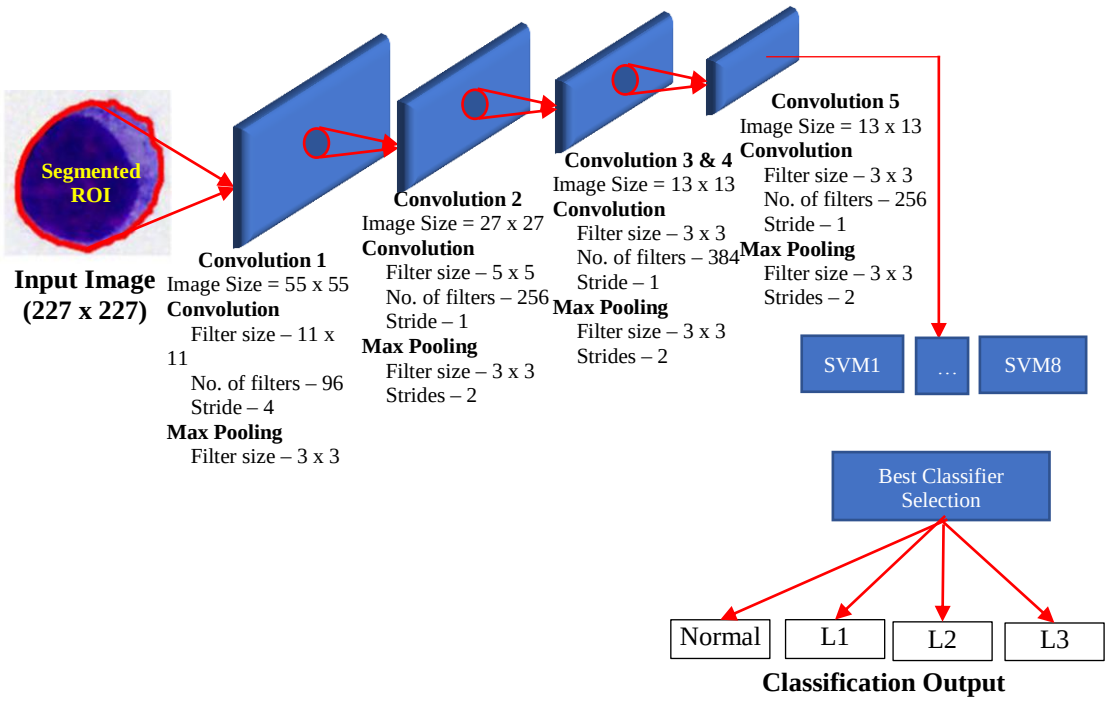

Figure 6.14 : Enhanced Hybrid CNN-Ensemble SVM Classifier

The proposed classifiers works in two stages. In the first stage, the AlexNet CNN extracts the deep features and feeds them to the second stage. The second stage then uses the ensemble SVM classifier to classify the deep feature maps extracted from AlexNet CNNs. The hybrid model proposed replaces the last three fully connected layers of base

model by the ensemble SVM classifier. In this stage, dynamic selection method, DCS or DES, is used, so that only optimal base classifiers are used during classification. Thus, the hybrid model is enhanced by combining CNN with ensemble SVM with DCS/DES method. The DES method selects five optimal classifiers during the construction of the ensemble system. During aggregation of results, while using ensemble with DES, a weighted ensemble of networks method is used. The main advantages of the proposed hybrid system are to reduce the chance of overfitting, number of parameters and the time & process complexities.

The main part of the proposed hybrid CNN-Ensemble SVM classifier is the kernel functions used to create different SVM base classifiers. A kernel in SVM is defined as functions that transform the feature set to find an optimal boundary. Different variants of SVM classifier can be constructed using different kernel functions. The kernel is a similarity function, which accepts two inputs and finds out how similarly they are matched. The kernel function returns the inner dot product between two features. The kernel or a window function is defined as in Equation (6.5).

$$K(\bar{x}) = \begin{cases} 1 & \text{if } \| \bar{x} \| \leq 1 \\ 0 & \text{Otherwise} \end{cases} \tag{6.5}$$

While using the above equation, the value of K is one inside closed circle of radius one centered at the original and zero otherwise. This is illustrated in Figure 6.15a. For a fixed x_i, the function is $K(z-x_i)/h) = 1$ inside the closed circle of radius h centered at x_i and 0 otherwise (Figure 6.15b) (https://medium.com/geekculture/kernel-methods-in-support-vector-machines-bb9409342c49; https://data-flair.training/blogs/svm-kernel-functions). Using the above definition of kernel, the eight functions used to construct the ensemble SVM classifier used in the proposed hybrid models are described in Table 6.2, where x_i and x_j are two feature vectors, d is the degree of polynomial, n is the number of instances in the feature set, c is the cost function, γ is the influence parameter and J_v is the first kind of Bessel objective.

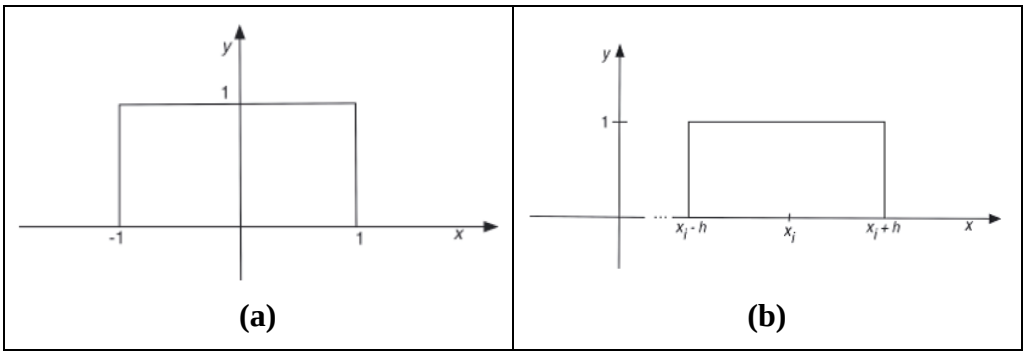

Figure 6.15 : Kernel Function Map

TABLE 6.2

KERNEL FUNCTIONS

Code	Kernel Function	Description
K1	Linear Kernel	$K(x_i, x_j) = x_i^T x_j$
K2	Polynomial Kernel	$K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$, $\gamma > 0$.
K3	Gaussian RBF	$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$, $\gamma > 0$
K4	Exponential Kernel	$K(x_i, x_j) = \exp\left(-\frac{\| x_i - x_j \|^2}{2\gamma^2}\right)$
K5	Laplacian Kernel	$K(x_i, x_j) = \exp\left(-\frac{\| x_i - x_j \|}{\gamma}\right)$
K6	Bessel Function Kernel	$K(x_i, x_j) = -\frac{J_{v+1}(\sigma \| x_i - x_j \|)}{\| x_i - x_j \|^{-n(v+1)}}$
K7	ANOVA RBF Kernel	$K(x_i, x_j) = \sum_{k=1}^{n} \exp(-\gamma(x_i^k, x_j^k)^2)^d$
K8	Hyperbolic or Sigmoid Kernel	$K(x_i, x_j) = \tanh(x_i^T x_j + c)$

6.5.3. Weighted Ensemble of Networks Method

While using weighted ensemble of networks method, the proposed model considers the performance of each CNN using a weight which denotes its contribution to final classification. The main aim of this method is to enhance CNN model performance by using individual CNN's importance, which depends on the individual classifier performance (Mondal *et al.*, 2021). The weights are estimated using the following method in this research. Let P'_j denote the probability of each class and is estimated using Equation (6.6).

$$P'_j = \frac{\sum_{k=1}^{5} W_k P_j^{'}}{\sum_{k=1}^{5} W_k} \tag{6.6}$$

In the above equation, w_k is the weighted value of each base CNN_k (k = 1..5), denominator is used to normalize P'_j and j denotes the target classes (1-4 indicating normal, L1, L2 and L2). Classification accuracy is used as evaluation score to estimate weighted values (W_k). Using classification accuracy, the probability is calculated using Equation (6.7).

$$P'_{jf1} = \frac{\sum_{k=1}^{m} W_k^{f1} P_{jk}}{\sum_{k=1}^{m} W_k^{f1}}, \quad \forall_j = 1,2,...,4 \tag{6.7}$$

where w_k^{f1} are the weights obtained while using accuracy as the evaluation score of the individual CNN_k when tested with test images individually. The weight are assigned using a simple assignment concept, using the Equation (6.8) .

$$W_i^{f1} = \text{Accuracy of SVM using } K_i \tag{6.8}$$

where i is the kernel code as in Table 6.2. Thus, Phase III of the research methodology proposes three systems as listed below:

1. Hybrid CNN-ensemble SVM classifier,
2. Hybrid CNN-ensemble SVM using DCS, and
3. Hybrid CNN-ensemble SVM using DES.

6.6. CHAPTER SUMMARY

This chapter proposed the use of deep learning classification to design ALL-C system. For this, the most popular CNN model was used. A base CNN model using AlexNet was built, which was then enhanced by combining it with the enhanced ensemble SVM classifier. The SVM was designed in a heterogeneous fashion using different kernel functions. The dynamic selection algorithms (Chapter 5, ALL-C Using Machine Learning Classifier) were used to improve the ensemble SVM classifier. Several experiments were conducted to evaluate the performance of the algorithms proposed in the three phases of the research methodology. The results obtained are tabulated and discussed in the following chapter, Chapter 7, **Results and Discussion**.