CHAPTER 6

NOVEL SEGMENTATION BASED CERVICAL CANCER DETECTION USING DEEP CONVOLUTIONAL-BASED NEURAL NETWORK WITH RELU (CERVI-CYTO-CNN)

6.1 Introduction

The uterine cervix is an important part of female reproductive system that can develop abnormal cell growth resulting in cervical cancer. Notably, cervical cancer procreates a significant danger to women's health, ranking second to breast cancer in terms of prevalence. The timely identification of abnormalities in cervical cells has important implications on early screening for cervical cancer. Different screening methods like Pap smear, colposcopy, and HPV testing are essential for detecting and diagnosing cervical cancer. Out of these techniques, categorization of Pap smear cell highlights potential accuracy and automated early detection. Precise categorization of Pap smear cell images offers additional characteristics of simplifying the diagnostic procedure for cervical cancer. The significance of precise Pap smear image screening cannot be overstated in aiding both early detection and diagnosis of cervical cancer. Although traditional screening techniques such as VIA, Pap smear tests, colposcopy, biopsies, and HPV-DNA detection necessitates medical professionals with expertise, the subjective aspect of cancer diagnosis emphasizes the consideration of a pathologist's knowledge and experience.

The integration of automated and intellectual screening technologies is crucial which can offer significant assistance by enhancing the precision and timeliness of the screening process, minimizing subjective evaluations, and potentially intensifying the outcomes of early diagnosis. In the succeeding parts of this section, the progress and uses of automated methods for classifying cervical cancer is examined, with a special prominence on using intelligent technologies to augment conventional screening techniques. These developments are focused on improving the effectiveness and accessibility of early detection and diagnosis methods for cervical cancer.

6.2 Methodology

The proposed approach for screening cervical cancer highly involves a computer-assisted automatic detection system. The overall architecture for cervical cancer identification and classification is detailed in Figure 6.1. At first, the Anisotropic Diffusion

Filter is applied to pre-process the initial cervical image. Subsequently, Dragonfly optimization is applied to enhance the ADF's weight for noise reduction while preserving the edges. The pre-processed images are utilized to separate the impacted cell through the integration of improved Weighted FCM and Grasshopper Optimization Algorithm (GOA) that aims to effectively differentiate features including shape, GLCM highlights, and other geometrical features. The trained features are processed using trained Deep CNN with ReLU as the activation function to classify the Pap smear images as either benign or invasive. Parameters such as accuracy, precision, recall, and F-measure are implemented so that the performance of cervical image classification is evaluated.

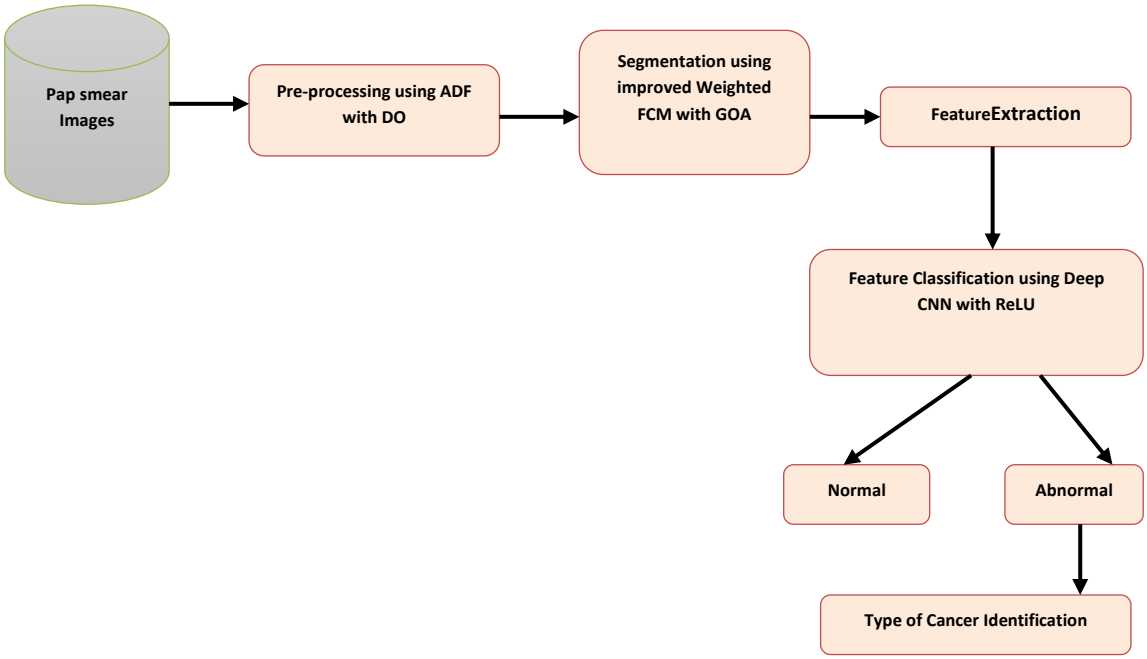

Figure 6.1 Work Flow of Proposed Method

6.2.1 Pre-processing: Anisotropic Diffusion Filter –Histogram Equalization with Dragonfly Optimization

Pre-processing

Pre-processing is a crucial step in diverse image classification as it helps minimize distortions, eliminate noise, and enhances image quality. The procedure of pre-processing contains two broad categories involving radiometric and geometric correction, and is generally incorporated before applying main data analysis and information extraction. This research utilizes techniques such as image resizing, noise removal, improving image quality, and converting RGB to grayscale.

In this proposed work, the process of Pre-processing techniques is mentioned below:

1. The pap-smear colored image is transformed into a grayscale image.

2. To preserve the edges and to reduce the noise in an image, the Anisotropic Diffusion Filter is applied.

3. The best weights for the ADF filter were chosen using Dragonfly Optimization, which aims at effective pre-processing of the image. The pre-processing techniques involved are shown in Figure 6.2.

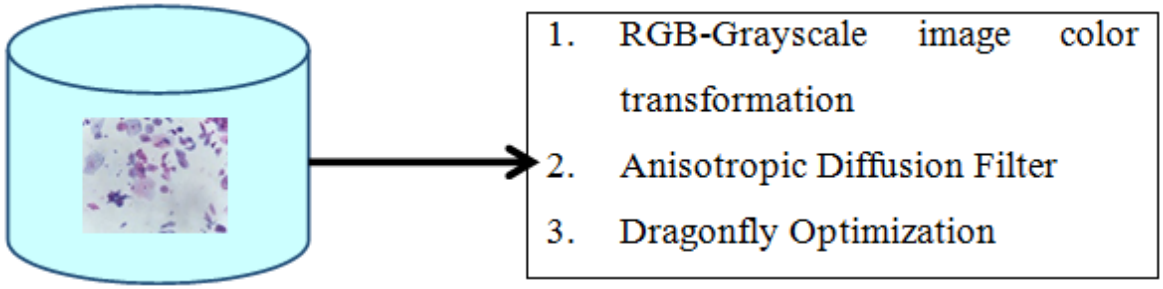

Figure 6.2 Illustration of Pre-processing

Anisotropic Diffusion Filter

The process of anisotropic diffusion filter comprises of generating a scale space by employing it in a series of progressively blurred images of the original image with different parameters. This approach estimates and utilizes the generalized diffusion equation for anisotropic diffusion, which is applied to the initial image for generating each subsequent image within the series. Therefore, the anisotropic diffusion process is iterated until an adequate amount of smoothing is attained. Anisotropic diffusion preserves brightness discontinuities while qualitatively smoothing the original image.

The mentioned formula focuses on regulating the diffusion coefficient depending on the image's characteristics and preserves the image's edge details when removing noise. The diffusion model is represented as in Eq. (6.1).

$$\frac{\partial P}{\partial t} = div(c(x,y,z)\nabla P) = \nabla c.\nabla P + c(x,y,t)\Delta P \quad (6.1)$$

Whereas,

Δis denoted as Laplacian,

∇is the gradient,

$div(...)$represents the divergence operator and

$c(x, y, z)\Delta$is defined as diffusion coefficient

For t > 0, the output is given as $P(., t)$ by the highest t generates the images with blur effect.

$c(x, y, z)$ Normalises, the diffusion rate, and it is measured as an image gradient function to preserve the edges and is calculated using Eq. (6.2).

$$c(\| \nabla P \|) = e^{-\left(\frac{\|\nabla l\|}{k}\right)^2} \quad (6.2)$$

Depending on the individually sampled image, the equation is represented as Eq. (6.3).

$$P_{t+1}^{S} = IP_t^s + \frac{\lambda}{|\eta^s|}\sum_{p\epsilon\eta} g(\left(|\nabla P_{s.p}(t)|\right)\nabla\nabla P_{s.p}(t) \quad (6.3)$$

Whereas t denotes the discrete time, I_t^S remains as the discrete image sample. Moreover, η^S is referred to as the neighborhood pixel s, $|\eta^S|$ provides the number of neighboring pixels. Besides, the image gradient magnitude for the specified direction is given in the Eq. (6.4).

$$\nabla I_{s,p}(t) = I_t^p - I_t^s, p\epsilon\eta^s \quad (6.4)$$

Dragonfly Optimization Algorithm

Dragonfly Algorithm (DO) encompasses the natural actions of dragonflies for creating a trained model. This algorithm was based on dragonflies' natural swarming behaviors, which are observed in both static and dynamic states where both swarming behavior is considered as critical stages of development which is pinpointed below.

(1) Exploitation that helps achieves the global optimum by bringing it closer together.

(2) Discovery is used to pinpoint potentially valuable areas in the search area.

The flowchart for Dragonfly Optimization Algorithm is shown in Figure 6.3.

The process comprises of initial phase and then goes through N iterations for detecting adjacent pixels in an image, alternating between exploration and exploitation. A final decision is made to pinpoint the best image edges.

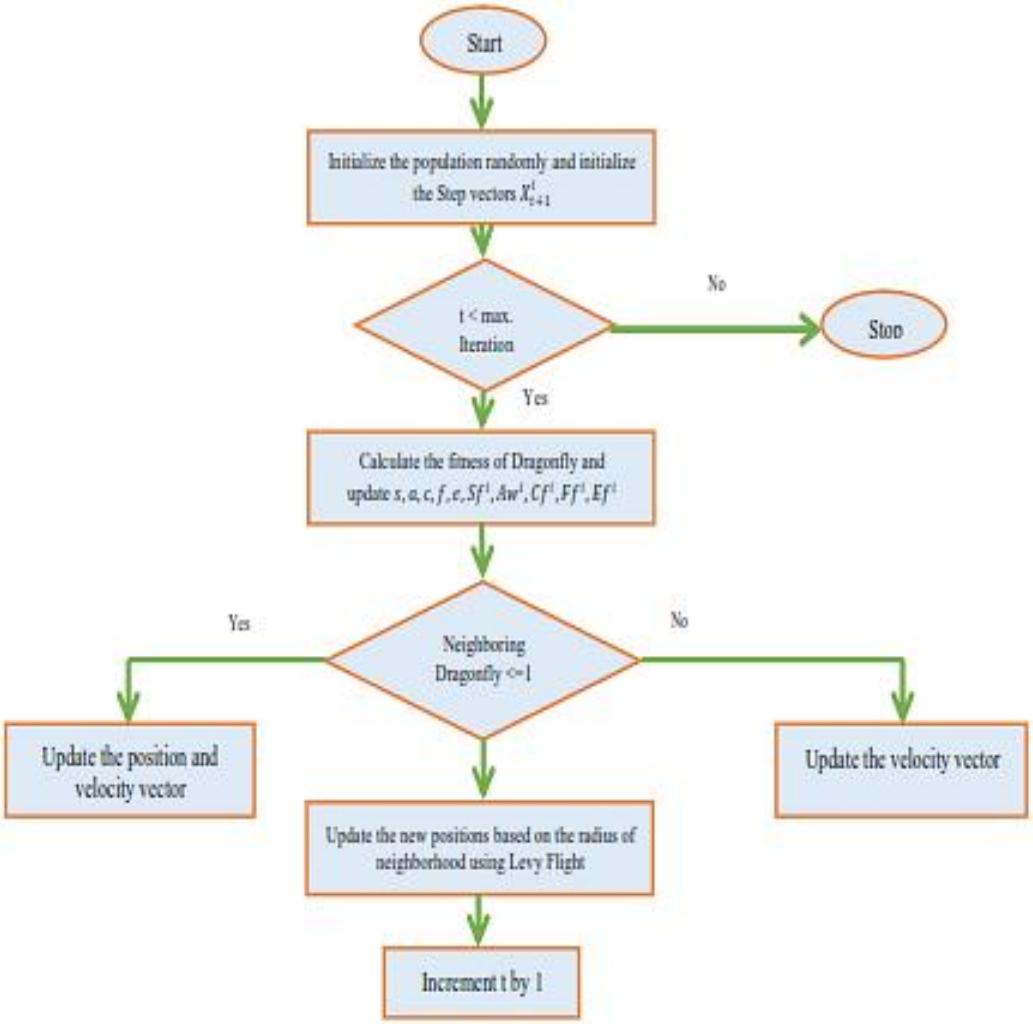

Figure 6.3 Work Flow of Dragonfly Optimization Algorithm

While performing the exploration and exploitation, the dragonflies are influenced by five significant factors including the food factor, enemy factor, separation, alignment, and cohesion which are regulated by the following components:

(i) Separation weight denoted as "s,"

(ii) Alignment weight represented by "a,"

(iii) Cohesion weight indicated as "c,"

(iv) Food factor labeled as "f,"

(v) Enemy factor denoted by "e," and

(vi) Inertia weight identified as "w".

In order to guarantee the survival of the swarm, the food sources considered for drawing while minimizing it clear of potential dangers. During every cycle, the least strong

solution is considered as opponent, and top solution is marked based on the source of sustenance. In the discovery stage, modifying weights encourages significant agreement and deters cohesion. During the exploitation phase, there is a focus on strong alignment but weak cohesion. In order to transform from exploration to exploitation, weights are adjusted in the algorithm.

The collision of a single dragonfly with the other in the neighbouring pixels is avoided and it is calculated by the separation factor which is depicted below in the Eq. (6.5).

$$Sf^i = -\sum_{j=1}^{N} X^i - X^j \tag{6.5}$$

Hence, X^i is the recent location, X^j is the neighbor position, and N denotes the number of dragonflies.

When equating the speed of the dragonfly with another alignment weight, Aw^i is calculated as Eq. (6.6).

$$Aw^i = \frac{\sum_{j=1}^{N} V^j}{N} \tag{6.6}$$

When V^j states the speed of the jth neighbor dragonfly.

However, the cohesion factor is measured by centering the affinity of each dragonfly to the mass center of neighborhood and it is expressed in the Eq. (6.7).

$$Cf^i = \frac{\sum_{j=1}^{N} X^j}{N} - X^i \tag{6.7}$$

Besides, the food factor is estimated by centering the dragonfly's affinity to the food resource, and it is given in Eq. (6.8).

$$Ff^i = X_f - X^i \tag{6.8}$$

Then, X_f indicates the food source location.

The evaluation of the Enemy factor is resolute through the dragonfly against the selected enemy, and it is given by Eq. (6.9).

$$Ef^i = X_e + X^i \tag{6.9}$$

Wherein, X_e represents the position of the enemy.

The step vector ΔX_{i+1}^i is estimated by imitating the movements and updating the position of the dragonflies in the search space as Eq. (6.10).

$$\Delta X_{i+1}^i = \left(swSf^i + awAw^i + cwCf^i + fwFf^i + ewEf^i\right) + \omega \Delta X_{i+1}^i \quad (6.10)$$

Whereas the symbols denote the distinct weights and factors, such as

sw - separation weight

aw - alignment weight

cw - cohesion weight

fw- food factor weight

ew- enemy factor weight

ω- weight of inertia

t- counters

The position of dragonflies is enhanced by the succeeding estimation of step vector ΔX_{i+1}^i by the Eq. (6.11).

$$X_{i+1}^i = X_t^i + \Delta X_{i+1}^i \quad (6.11)$$

In the last optimization stage, all dragonflies can come together to form one energetic swarm, that aim towards finding the best global solution. In case of null artificial dragonflies in the search space, the Lévy flight mechanism is employed for movement. Dragonflies that do not have neighbors are assigned a random position through a random walk. The Eq. (6.12) can be employed to update the position.

$$X_{i+1}^i = X_t^i + Levy\ (d) \times X_t^i \quad (6.12)$$

In this case, t represents the current number of iterations, and d indicates the dimensions of the position vectors. Finally, the best pixel intensity value can be determined to recognize the optimized edges of an image by utilizing an anisotropic diffusion filter on the enhanced image.

6.2.2 Segmentation: improved - Weighted Fuzzy C-Means Algorithm with Grasshopper Optimization Algorithm

Segmentation using improved Weighted FCM

Image segmentation in computer vision comprise of multiple applications in different real-world systems. Image segmentation is crucial for various applications including content-based image retrieval, video surveillance, object detection, recognition, medical imaging, and Automatic Traffic Control systems. Substantially, segmentation algorithms have been used in the medical field to excerpt required information from medical images to assist with diagnosis and treatment planning. Segmentation process divides an image into separate sections by grouping pixels with similar attributes such as color, intensity, texture, or volume. Existing research suggested a variety of methods for segmentation to enhance accuracy and efficiency. Image segmentation methods are extensively incorporated in medical imaging to separate input images and extract key data regarding the regions of interest. Lesions, cancerous cells, or tumor tissues are potential ROI attributes. The importance of segmentation is extremely crucial and imposes a high level of accuracy in order to conclude that the expert advises the patients.

Improved Weighted Fuzzy C-Means (i-WFCM) with Grasshopper Optimization Algorithm

The weighting factor has been employed in this method FCM for enhancing the cluster center in the FCM which is widely incorporated in past studies. However, a method to enhance the weight in i-WFCM to achieve an optimal weight is suggested using the Grasshopper Optimization Algorithm (GOA).

The suggested i-WFCM method includes transformation of input data into a higher-dimensional feature space using nonlinear mapping. Following, Fuzzy C-means (FCM) are exploited in this particular feature space. The method of kernel-weighted fuzzy C-means is employed to reduce the objective function using the following Eq. (6.13).

$$Y\omega = \sum_{i=1}^{k} \sum_{j=1}^{n} b_{ij}^{l} \parallel \omega(x_j) - \omega(v_i) \parallel^2 \quad (6.13)$$

Where ω(vi) is the feature space centre of cluster i, b_{ij} indicates if x_j is a member of cluster i, and ω is used to map input space X to feature space F.

Grasshopper Optimization Algorithm

The inspiration of grasshopper optimization algorithm is innate swarming behavior of grasshoppers. The objective of grasshopper is influenced by three key characteristics: interpersonal connections, gravity's pull, and wind direction. Below equation Eq. (6.14) shows the mathematical representation of how they behave as a group:

$$X^i = r_1 S^i + r_2 G^i + r_3 A^i \tag{6.14}$$

Here, $r_1, r_2 and r_3$ are the random numbers.

S^i – social relationship

G^i – gravity force

A^i-wind advection

To improve the equation (14), the best optimization problems is solved using Eq. (6.15).

$$X_d^i = c\left(\sum_{j=1, j\neq i}^{n} c\frac{UB_d - L_d}{2} S\left(|x_d^j - x_d^i|\right)\frac{x^j - x^i}{d^{ij}}\right) + best \tag{6.15}$$

Moreover, the upper bound (UB_d) and lower bound (LB_d) denote the limitations in d^{th} dimension, where "best" considers the present optimum value in the certain dimension in the solution space, where $d^{ij} = |x^j - x^i|$ represents the space among the i^{th} and j^{th} grasshoppers(r) and is evaluated by $s(r) = \frac{fe^{-r}}{i} - e^{-r}$, where fe and i are two constants.

Moreover, c remains a factor in GOA that can be implemented to associate the utilization and assessment. In addition, if the coefficient of c decreases in accord with the amount of repetitions. Then, the value of c is calculated through the equation Eq. (6.16), and it is as follows.

$$c = c^{max} - u\frac{c^{max} - c^{min}}{u^{max}} \tag{6.16}$$

Whereas c^{min} and c^{max} are the minimum and maximum amount of repetitions, u denotes the recent repetitions. Table 6.1 shows the algorithm for the improved Weighted FCM with Grasshopper Optimization Algorithm (i-WFCM-GOA), and it is portrayed.

Table 6.1 Algorithm of i-WFCM-GOA

Algorithm- improved Weighted FCM with Grasshopper Optimization Algorithm (i-WFCM-GOA)
1. Primary class sample $\{V_i\}^c$ is chosen. 2. b_{ij} memberships are reorganized by the weighted average of cluster center through $u_{ik} = \frac{(1-B(x_k,v_i)^{\frac{1}{(m-1)}}}{\sum_{j=1}^{c}(1-B(x_k x_j)^{-\frac{1}{(m-1)}}}$ (6.17) 3. The clusters sample is achieved as a weighted average by $v_i = \frac{\sum_{k=1}^{n} u_{ik}^m B(x_k,v_i)x_k}{\sum_{k=1}^{n} u_{ik}^m B(x_k, v_i)}$ (6.18) 4. Set the factorsc^{min},c^{max}, u, f and l. 5. Set the random population in the search space wherein $X^j, \{j = 1,2,3 \ldots N\}$ also set the search agent "best". 6. While $u < u^{max}$ a. Calculate c through the Equation (6.16) b. For= i=I: n begin i. Update best ii. Standardize the space among the grasshoppers within the range[1,4] iii. Update the location of the grasshopper X_d^i by the equation (6.15) iv. Verify the recent grasshopper location in case it moves beyond the boundaries v. End for c. U=u+1; 7. End while 8. Return the best solution for updating the weights in WFCM in equation (6.17) and (6.18) 9. Remain steps 2-3 till the preventing requirement is achieved. 10. The termination condition is given by $\lvert V_{new} - V_{old} \rvert \leq \varepsilon$ 11. The Euclidean norm is l.l. The cluster center vector is given by V. ε indicates a user-adjustable small number (where $\varepsilon = 0.01$).

The i-WFCM-GOA algorithm is employed in proposed method to elevate the weighted factor for the clustering center without making computation more difficult. Typically, the intention of population is to evaluate the collective performance of search agents within the current generation. An individual's fitness function value should be lower than the average in case of minimal problem. As a result, a plan to improve local search should be implemented.

Additionally, the population is dispersed more randomly. Especially during the later stages of evolution, this property can help the population maintain its diversity more effectively.

6.2.3 Feature Extraction

Feature extraction is the most important method in image processing and it has a crucial role in cancer detection. To identify cancer, features are taken from the image after segmentation. Feature extraction is a crucial stage in recognizing cancer and non-cancer in images. Identifying and representing specific image features of interest for future processing is known as a vital process of feature extraction.

To demonstrate the state and aid in classifier training, Feature Extraction diminishes particular features from Pap smear images. Current research has focused on the integration of modern feature extraction techniques for the identification of CC in Pap smear images. The extracted characteristics comprise geometric properties including asymmetry, concavity, area, diameter, perimeter, and eccentricity. Furthermore, GLCM and Haralick features are utilized to analyze characteristics such as texture, size, and shape identification. This method has been implemented at present to identify images of skin lesions.

Moreover, the estimation of asymmetry is attained by the equation Eq. (6.19).

$$A_1 = \frac{\Delta A}{A} X\ 100 \tag{6.19}$$

a. The eccentricity is determined by the equation Eq. (6.20).

$$Eccen = \frac{Length_{major}}{Length_{minor}} \tag{6.20}$$

b. The area of the nucleus is given by the equation Eq. (6.21).

$$Area_N = \sum_{i=1}^{n} \sum_{j=1}^{m} I(i,j) \tag{6.21}$$

c. The formula for finding the perimeter is given by the equation Eq. (6.22) and it is represented.

$$Perimeter_N = Even_{count} + \sqrt{2}(odd_{count}) \quad (6.22)$$

Besides, the nucleus diameter is evaluated among the space between two points of both the minor and major axis.

6.2.4 Classification: Deep CNN with ReLU

The feature classification is performed by utilizing Deep Convolutional Network (Deep CNN) with ReLU as an activation function. In the Deep CNN, the network comprises an input layer followed by 2X2 Convolution layers, a Pooling layer, and a 2X2 Convolution layer, which is used for learning the extracted features. Deep CNN uses the fully connected and softmax layers for its classification stage, which is shown in Figure 6.4.

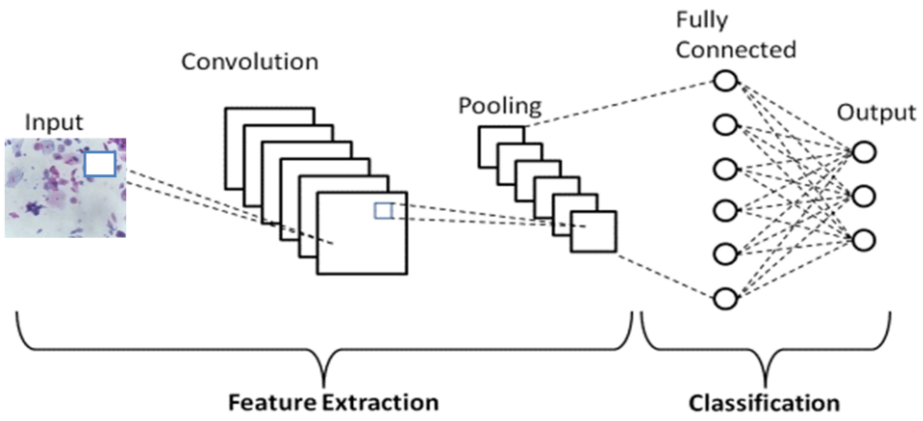

Figure 6.4 Deep CNN with ReLU for Identification of Cervical Cancer

6.2.4.1 Convolutional layer

Every neuron function represents a single convolutional kernel within the convolutional layer. Convolutional kernels process the division of the image into small segments considered as receptive fields. Segmenting an image into smaller pieces facilitates the extraction of features. The kernel interacts with images with the help of a unique set of weights. This is achieved by multiplying the kernel elements with the respective elements in the receptive field. The operators in convolutional network possess weight-sharing capability and outperforms fully connected networks in terms of efficiency for CNN parameters. Sliding a kernel with identical weights across the image allows extraction of unique feature sets from the image.

The Rectified Linear Activation Function, also called ReLU, is a piecewise linear function that directly produces the input if it is positive; otherwise, it returns zero. The utilization of ReLU in a CNN model shortens the training process and often leads to enhanced performance. Besides, the ReLU function is calculated by the equation Eq. (6.23).

$$R_f = max(0.0, y) \tag{6.23}$$

When the value of R_f becomes zero, the gradient descent approach ceases learning.

Additionally, Convolutional Neural Networks (CNN) is a well-known Deep Learning (DL) technique that has gained popularity in computer vision assignments, such as classifying images, detecting objects, and segmenting semantics. CNNs offer multiple benefits compared to conventional computer vision methods.

CNNs can acquire features that are impervious to minimal alternations in object position and angle in images, making them ideal for applications such as object detection and semantic segmentation due to their robustness to translation and rotation.

Dealing with diverse quantities of data: CNNs can process large amounts of data, making them suitable for training on extensive datasets including ImageNet.

Transfer learning involves fine-tuning CNNs that have been trained on large datasets namely ImageNet on smaller dataset decreases the data and computational resources needed to train a model for a particular task.

6.2.4.2 Pooling Layer

The next layer after a Convolution Layer is a Pooling Layer. It operates on each map separately to generate a new arrangement with the same number of pooled input maps. The main aim of this layer is to condense computational expenses by diminishing the size of the convolved feature map. Reducing the connections between layers is done independently on each feature map. Max pooling was selected due to its superior performance compared to other pooling techniques.

Pooling Layers, also identified as down-sample layers, are necessary elements of CNNs evolved in DL. They remain accountable for decreasing the three-dimensional sizes of the input data, in accordance with width and height, while retaining the furthermost significant data.

However, pooling layers distribute the input data into minor sections, named pooling frames or accessible fields, and then achieve an accumulation process involves compelling the determined or typical rate, in every frame. Hence, accumulation decreases the dimension of the feature records, which is consequential in a compacted demonstration of the input data.

The procedure of Pooling Layers includes the subsequent stages:

1. Separate the input data to non-overlapping sections (or frames).
2. Relate an accumulation function, such as max pooling or average pooling, that proceeds every frame towards acquiring a distinct value.
3. Organize the standards attained against every frame to generate a down-sampled demonstration of the input data.

Pooling Layers can be used repeatedly in a deep learning model to moderately decrease the spatial dimensions of the feature maps.

Pooling Layers provide multiple advantages in deep learning models:

- Dimensionality reduction is accomplished by Pooling Layers with reduced spatial dimensions of input data, which leads to a decrease in the number of parameters and computational complexity of the following layers in the model.
- Pooling layers suggest a type of translation invariance by picking critical features from several spatial positions, and enhances the model's resilience to changes in feature positioning.
- Pooling Layers aid in establishing a feature hierarchy by merging lower-level features to generate higher-level features, capturing abstract representations within the input data.

Pooling Layers can serve as a type of regularization by declining overfitting. By combining data from nearby areas, the model concentrates on the most important information and disregards minor differences.

6.2.4.3 Fully connected Layer

In most cases, for classification at the end of the network the fully connected layer is applied. It is a worldwide process, as opposed to pooling and convolution where selected

characteristics are amalgamated in a nonlinear fashion to categorize information. It takes input from the feature extraction stage and conducts a comprehensive examination of the output from all preceding layers.

In fully connected layers every neuron in one layer is connected with the corresponding neuron in another layer, thus earning the name "fully connected." These layers make the final decisions by utilizing the features learned by CNN. In an image classification task, the fully connected layers are responsible for identifying the class to which the input image is assigned.

Understanding Flattening in CNNs

In order to link the pooling layers' output to the fully connected layers, the data needs to be converted from a 2D layout to a 1D layout. This procedure is referred to as flattening.

Flattening aims to transform the acquired features from earlier convolutional and pooling layers into a suitable format for the fully connected layers. Fully connected layers necessitate the use of vector inputs, not multi-dimensional data, thus making this requirement necessary.

Connection between Convolutional, Pooling, and Fully Connected Layers

Every layer in a CNN has a distinct function. The major functionality of convolutional layer is to acquire the input's local characteristics. This is achieved by using the convolution operation, which includes filtering the input data.

- The pooling layers condense the features, reduces computational complexity for the network and providing translational invariance.
- In the end, the fully connected layers utilize these flattened, advanced features to generate a final prediction.
- Flattening serves as the link between the pooling and fully connected layers, converting the 2D data from pooling into a 1D format well-suited for fully connected layers.

6.2.4.4 Softmax Layer

The typical final layer in a Deep Convolutional Neural Network (CNN) is the Softmax layer. Softmax functions as an activation function, converting numeric values into probabilities by generating a vector that shows the likelihood of each possible result. As a result, it assures the total probabilities of all categories in the vector add up to one.

The softmax function converts K values into real values whose sum equals 1. The softmax function converts a range of values- including negative, zero, positive, or greater than one - into a range between 0 and 1, enabling them to be interpreted as probabilities.

If one of the inputs is small or negative, the softmax will transform it into a low probability; if one of the inputs is large, it will be transformed into a high probability; however, the result will always fall within the range of 0 to 1.

Softmax is also known as multi-class logistic regression. It is an augmentation of logistic regression in multi-class classification, with a formula closely resembling the sigmoid function in logistic regression. This function can only be used in a classifier if the classes are entirely separate from each other.

A lot of complex neural networks end with a second-to-last layer that generates scores in real numbers which are hard to adjust and handle. The softmax function is very useful in the above-mentioned scenario since it transforms the scores into a normalized probability distribution that can be shown to users or utilized by other systems. Therefore, it is necessary to include the softmax classification layer as the final layer of the neural network.

6.3 Summary

The chapter 6 explains in detail about the pre-processing, segmentation of feature in the identification and classification of cervical cancer at the earlier stage. In addition, it explains the working principles of the pre-processing using ADF with DO, segmentation by improved weighted FCM with GOA and the classification using deep CNN with ReLU and its implementation, integration. The overall work flow used to detect the image for cervical cancer cells accurately is described in simple manner.