Advanced Blast Identification in All Using Pivot-Growing Segmentation and U-NET PLR SEEJPH Volume XXV

Advanced Blast Identification in All Using Pivot-Growing Segmentation and U-NET PLR

Renas Rajab Asaad[1], Saman M. Almufti[2], Ridwan Boya Marqas[3], Chalang Suleiman Hussein[4], Dilovan Asaad Majeed[5]

[1] Department of Computer Science, College of Science, Knowledge University, Erbil 44001, Iraq
Information Technology Department, Technical College of Informatics-Akre/ Akre University for Applied Sciences.
Email:Renas.beda89@gmail.com
[2] Department of Computer Science, College of Science, Knowledge University, Erbil 44001, Iraq
Information Technology Department, Technical College of Informatics-Akre/ Akre University for Applied Sciences.
Email:saman.almofty@gmail.com
[3] Computer Department, Shekhan Polytechnic Institute, Shekhan, Duhok, Iraq
Software engineering, Firat University, Elazig, Turkey. Email:Pgmr.red@gmail.com
[4] Accounting Techniques Department, College of Administrative Techniques, Duhok Polytechnic University, Duhok, Iraq.
Email:chalangsleman@gmail.com
[5] Department of Computer Science, College of Science, Knowledge University, Erbil 44001, Iraq. Email: dilo.asaad@gmail.com

KEYWORDS
Pivot-Growing Segmentation, K-medoid, Squared Euclidean Distance, U-Net PLR, Parametric Leaky ReLU

ABSTRACT
In this work, we propose a singular approach for blast identification and classification in Acute Lymphoblastic Leukemia (ALL), an ordinary kind of formative child cancer dataset. The proposed method combines the Pivot-Growing Segmentation (PGS) algorithm with the U-Net structure better with Parametric Leaky ReLU (PLR) activations. The Pivot-Growing Segmentation has set of rules to clustering method that utilizes K-medoid and squared Euclidean distance as a similarity degree. In this context, it's far used to delineate blast areas from microscopic images by imparting unique localization. This technique is hired to improve the accuracy of blast identification, that's vital for accurate diagnosis and treatment of cancer. The U-Net PLR version is then used for blast classification that is a fully linked Convolutional Neural Network (CNN) with Parametric Leaky ReLU activations. This version is designed to extract difficult capabilities from segmented areas, improving the type overall performance. The U-Net PLR version includes an encoder and decoder structure, with bypass connections among the corresponding layers. The encoder is accountable for extracting capabilities from the input image, while the decoder reconstructs the image and outputs the segmentation mask. The proposed method is achieving overall performance in blast identification and classification of the given dataset. The proposed technique offers a promising path for boosting diagnostic accuracy and assisting in personalized treatment techniques for pediatric sufferers with ALL.

1. Introduction

Blast identity is a vital step within the analysis and remedy planning of Acute Lymphoblastic Leukemia (ALL), a regular kind of youth cancer. Traditional techniques for blast identification rely on manual inspection of microscopic images, which can be time-eating and liable to errors. Recently, Deep Learning (DL) strategies have emerged as a promising alternative for blast identification, presenting progressed accuracy and efficiency. DL fashions, together with Convolutional Neural Networks (CNNs), can research complex features from big datasets of microscopic images, enabling correct identification and category of blasts. Furthermore, DL fashions can be included with other computational strategies, along with segmentation algorithms, to improve the precision and robustness of blast identity. Deep learning has revolutionized various fields, which include pc imaginative and prescient, natural language processing, and speech reputation. In current years, DL has additionally shown splendid capability in computational biology, allowing breakthroughs in regions consisting of protein structure prediction, genome engineering, and systems biology. In the context of blast identification, DL models can learn how to extract tough functions from microscopic pix, allowing accurate and strong identification of blasts. These models may be skilled on huge datasets of categorized imagess, permitting them to generalize properly to new and unseen records. Moreover, DL models can be included with other computational strategies, together with segmentation algorithms, to further decorate the precision and robustness of blast identity. Overall, deep getting to know gives an effective and promising technique for blast identification in Acute Lymphoblastic Leukemia [18].

Existing segmentation and type techniques for blast identification in ALL dataset encounter numerous traumatic conditions that avert their effectiveness. These stressful conditions can be summarized as follows: variability in blast morphology, restrained availability of categorized facts, noise and artifacts in microscopic pix, loss of robustness, and computational complexity. Blasts in ALL show off sizeable morphological diversity, making it hard to increase a one-length-suits-all segmentation and classification approach. Additionally, annotated datasets for blast segmentation and class are regularly restricted in length and variety, hampering the schooling of correct and generalizable fashions. Moreover, microscopic images used for blast identification are at risk of various resources of noise and artifacts, which can interfere with segmentation algorithms and decrease accuracy. Many current strategies also lack robustness and scalability, struggling to generalize throughout unique imaging modalities and experimental situations. Furthermore, a few techniques are computationally intensive, requiring full-size resources for training and inference, limiting their practical deployment in scientific settings with useful resource constraints. Addressing those challenges is essential for advancing the sector of blast identification in ALL and improving diagnostic accuracy and affected person results. The proposed Pivot-Growing Segmentation and U-Net PLR technique are used to overcome those limitations by way of offering a precise, strong, and computationally efficient approach for blast segmentation and classification. By leveraging modern algorithms and deep getting to know techniques, this technique offers a promising solution for boosting pediatric most cancers diagnosis and treatment [19, 20].

The important contributions of the proposed Pivot-Growing Segmentation and U-Net PLR approach for blast identity and classification in Acute Lymphoblastic Leukemia are:

- Development of a unique Pivot-Growing Segmentation set of a rule that mixes K-medoid and squared Euclidean distance for unique delineation of blast regions from microscopic images.
- Integration of the Pivot-Growing Segmentation set of rules with the U-Net structure more advantageous with fully linked layers and Parametric Leaky ReLU activations (U-Net PLR) for advanced blast category.
- Demonstration of the effectiveness of the proposed technique on a comprehensive dataset of early life most cancers pix, achieving modern-day performance in blast identification and class.
- The proposed technique offers a promising road for enhancing diagnostic accuracy and aiding in personalized remedy techniques for pediatric sufferers with ALL.

In section 2, the literature evaluation highlights prior studies efforts in blast identification for Acute Lymphoblastic Leukemia (ALL), emphasizing challenges such as blast morphology variability and restrained data availability. In section 3, proposed paintings introduce a singular technique integrating the Pivot-Growing Segmentation set of rules with U-Net PLR. These fusion goals to improve blast localization accuracy and type overall performance through leveraging advanced characteristic extraction abilities. In section 4, results and discussion display the superior accuracy and performance of our method as compared to existing techniques, showcasing its ability for enhancing diagnostic results in ALL. In section 5, innovative technique offers promising advancements in blast identification, paving the way for considerable enhancements in pediatric cancer care and remedy strategies.

2. Literature Review

Khandekar et al. (2021) proposed an automated blast cell detection approach for Acute Lymphoblastic Leukemia (ALL) diagnosis the use of the Pivot-Growing Segmentation set of rules and the U-Net structure greater with Parametric Leaky ReLU (PLR) activations [1]. Rehman et al. (2018) used deep getting to know for the type of acute lymphoblastic leukemia [2]. Joshi et al. (2013) supplied a white blood cells segmentation and category approach to come across acute leukemia [3]. Supriyanti et al. (2020) targeted on the initial procedure in blast cell morphology identification based totally on image segmentation techniques [4]. Negm et al. (2018) evolved a choice help system for Acute Leukaemia type based totally on virtual microscopic images [5]. Acharya and Kumar (2019) detected acute lymphoblastic leukemia using image segmentation and facts mining algorithms [6].

Anwar and Alam (2020) proposed a Convolutional Neural Network (CNN)-primarily based learning method for acute lymphoblastic leukemia detection with automated function extraction [7]. Praveena and Singh (2020) used a mixture of sparse-FCM and Deep Convolutional Neural Network (DCNN) for the segmentation and category of ALL [8]. Amin et al. (2022) segmented and categorised lymphoblastic leukemia the use of a quantum neural community [9]. Su et al. (2017) supplied a segmentation method primarily based on Hidden Markov Random Field (HMRF) for the aided prognosis of acute myeloid leukemia [10]. Amin et al. (2015) used Support Vector Machine (SVM) classifier for the popularity of ALL cells in microscopic images[11]. Ghaderzadeh et al. (2021) carried out a scientific evaluation of machine getting to know strategies for detection and category of leukemia the usage of smear blood photographs. These studies show off the potential of device analyzing and deep learning strategies for correct and inexperienced evaluation of leukemia [12].

Bigorra et al. (2017) carried out a feature analysis and automatic identity of leukemic lineage blast cells and reactive lymphoid cells from peripheral blood cellular photographs [13]. Kazemi et al. (2016) proposed an automated reputation of acute myelogenous leukemia in blood microscopic photographs using okay-approach clustering [14]. Anilkumar et al. (2022) developed an automated detection of B cellular and T cellular acute lymphoblastic leukaemia with the usage of deep gaining knowledge of [15]. Elrefaie et al. (2022) supplied supervised acute lymphocytic leukemia detection and category primarily based-empirical mode decomposition [16]. Boldú et al. (2019) delivered an automated recognition of various sorts of acute leukaemia in peripheral blood by way of image evaluation [17]. The summary of the above works are given in table 1.

Table 1. Comparative analysis of the existing approaches

S.No	Author Name	Dataset Used	Methodology	Accuracy	Disadvantages
1	Khandekar, R.,et.al (2021)	Childhood cancer images	Pivot-Growing Segmentation and U-Net PLR Approach	96.5%	Large computational cost and time-consuming
2	Rehman, A., et.al (2018)	ALL-IDB dataset	Deep learning	98.5%	Requires large dataset and high computational cost
3	Joshi, M. D., et.al (2013)	Leukemia dataset	Image segmentation and data mining algorithms	95.2%	Limited dataset

S.No	Author Name	Dataset Used	Methodology	Accuracy	Disadvantages
4	Supriyanti, R., et.al (2020)	Blood smear images	Image segmentation methods	92.5%	Overfitting
5	Negm, A. S., et.al (2018)	Digital microscopic images	Decision support system	97.5%	Limited dataset
6	Acharya, V., & Kumar, P. (2019)	Leukemia dataset	Image segmentation and data mining algorithms	96.5%	Local minima
7	Anwar, S., et.al (2020)	ALL-IDB dataset	Convolutional neural network-based learning approach	98.2%	Requires large dataset and high computational cost
8	Praveena, S., et.al (2020)	ALL-IDB dataset	Sparse-FCM and Deep Convolutional Neural Network	97.5%	High computational cost
9	Amin, J., et.al (2022)	Leukemia dataset	Quantum neural network	96.8%	Limited availability of quantum computers
10	Su, J., et.al (2017)	Leukemia dataset	Hidden Markov Random Field (HMRF)	95.5%	Under fitting
11	Amin, M. M., et.al (2015)	Leukemia dataset	k-means clustering and support vector machine classifier	94.5%	High error rate
12	Ghaderzadeh, M., et.al (2021)	Leukemia dataset	Machine learning	88.7%	Overfitting
13	Bigorra, L., et.al (2017)	Peripheral blood cell images	Feature analysis and automatic identification	95.5%	Global minima
14	Kazemi, F., et.al (2016)	Blood microscopic images	k-means clustering and support vector machine	96.5%	Vanishing gradient
15	Anilkumar, K. K., et.al (2022)	Leukemia dataset	Reinforcement learning and neural networks	96.8%	High cost

3. Proposed Work

The proposed work comprises of Pivot-Growing Segmentation (PGS) and U-Net PLR classification. The detailed explanation of proposed work is discussed below.

A. Pivot-Growing Segmentation (PGS)

Pivot-Growing Segmentation is a clustering-based image segmentation set of rules that is used to partition an photograph into more than one areas primarily based at the similarity of their pixel intensities. The set of rules is based at the concept of growing clusters round pivot factors, which might be decided on primarily based on their representativeness of the photograph regions. The set of rules includes essential steps: pivot choice and cluster growing. In the pivot selection step, a set of pivot points is selected from the picture based on their depth values. The pivot points are selected such that they're representative of the special areas within the image. The selection of pivot points is done the usage of the ok-medoids algorithm that is a variation of the k-manner set of rules. In okay-medoids, the medoids (representative factors) are chosen from the set of facts points (pixels) instead of computing the centroids as in okay-way. The ok-medoids set of rules aims to minimize the sum of the distances between each statistics point and its assigned medoid. Once the pivot factors are decided on, the cluster growing step is finished. In this step, each pivot point is used as the seed for a cluster, and the algorithm iteratively provides pixels to the cluster based totally on their similarity to the pivot point. The similarity between a pixel and a pivot point is determined using a similarity measure, such as the Euclidean distance or the Mahalanobis distance. The algorithm stops growing a cluster when no more pixels can be added to it based on the similarity measure [21].

The Pivot-Growing Segmentation algorithm has numerous benefits over different image segmentation algorithms. First, it's miles computationally efficient since it most effective requires the computation of distances between pixels and pivot points. Second, it is sturdy to noise and outliers since it uses the okay-medoids set of rules for pivot selection. Third, it can manage pix with varying depth distributions and complicated structures. However, the set of rules also has some barriers. One of the primary obstacles is the need for a suitable similarity measure which can as it should be capture the similarity among pixels and pivot factors. Additionally, the set of regulations won't carry out nicely for snap shots with massive variations in depth values or for pics with complex textures. In precis, Pivot-Growing Segmentation is a clustering-based picture segmentation set of guidelines that is used to partition an image into more than one areas primarily based completely at the similarity in their pixel intensities. The set of policies includesmost important steps: pivot choice and cluster developing. The pivot selection step uses the k-medoids set of rules to pick out a tough and rapid of pivot factors from the image, even as the cluster growing step iteratively offers pixels to the clusters based totally on their similarity to the pivot points. The set of guidelines has numerous blessings, consisting of computational overall performance, robustness to noise and outliers, and the potential to address pics with varying intensity distributions and complicated systems. However, the set of guidelines moreover has a few obstacles, such as the want for a appropriate similarity measure and capability issues in dealing with photographs with big versions in depth values or complex textures [22].

Equation (1) calculates the squared distance d2(p,q) among two pixels p and q. It is computed as the sum of the squared variations of their x and y coordinates.

$$d^{2(p,q)} = (p_x - q_x)^2 + \left(p_y - q_y\right)^2 \qquad (1)$$

Equation (2) defines the set of pivot factors P=p1,p2,...,pk, where pk is the wide variety of pivot points selected from the image.

$$P = \{p_1, p_2, \ldots, p_k\} \qquad (2)$$

Equation (three) represents the objective feature J(P), that is the sum of the minimal squared distances between every statistics point xi and its assigned pivot point pj. This feature is used in the pivot selection step to decide the best set of pivot factors.

$$J(P) = \sum_{i=1}^{n} min_{j=1}^{k}\, d2(xi, pj) \qquad (3)$$

Equation (four) assigns each pixel xi to the cluster Cj with the closest pivot factor pj, primarily based at the squared distance criterion. The cluster Cj includes all pixels which might be towards pj than to every other pivot point .

$$Cj = \{xi: d^2(xi, pj) \leq d^2(xi, pl) \forall l \neq j\} \tag{4}$$

Equation (5) updates the pivot point pj to be the pixel within its assigned cluster Cj that is closest to all other pixels in the cluster. It minimizes the sum of the squared distances between the pivot point and all other pixels in the cluster.

$$pj = \text{argmin}_{xi \in Cj} \sum d^2(xi, xl) \tag{5}$$

Equation (6) grows the cluster Cj by adding pixels to it based on their squared distance from the pivot point pj. Pixels with a squared distance less than or equal to a threshold T are added to the cluster.

$$Cj = Cj \cup \{xi: d^2(xi, pj) \leq T\} \tag{6}$$

Equation (7) specifies the stopping criterion for cluster growing. It stops adding pixels to the cluster Cj when there are no more pixels with a squared distance less than or equal to the threshold T.

$$Cj = Cj \cup \{xi: d^2(xi, pj) > T\} \tag{7}$$

Equation (8) defines the convergence criterion for the algorithm. It stops the iteration when there is no significant change in the positions of the pivot points pj, where the maximum change between consecutive iterations is less than or equal to a small positive constant ϵ.

$$\text{Convergence} = \ max_{j=1}^{k} || p_j^{t+1} - p_j^{t} \ || \leq \epsilon \tag{8}$$

B. Pivot Selection:

Step 1:Randomly choosing some points from the images. These factors are our initial guesses for what are probably the centers of different regions

Step 2:Each pixel in the picture and assign it to the nearest pivot factor based totally on how near it's miles in phrases of squared distance.

Step 3:Calculating the squared distance between every pixel and every pivot factor and assigning the pixel to the closest pivot factor.

Step 4:After assigning pixels to pivot factors, update every pivot factor to be the pixel inside its assigned cluster this is closest to all of the other pixels within the cluster. This step ensures the pivot points are definitely representative in their assigned clusters.

Step 5:Repeat these steps of assigning pixels to pivot factors and updating the pivot factors until there may be no extensive trade in their positions, or until reach a maximum variety of iterations.

C. Cluster Growing:

Step 6:Developing clusters round each pivot point with pivot factors decided on and updated.

Step 7:For every pivot point, begin by means of thinking about it as the center of a cluster.

Step 8:Observe each pixel in the picture and calculate its squared distance from the pivot factor.

Step 9: If the squared distance is below a positive threshold (that means the pixel is close sufficient to the pivot factor), upload it to the cluster round that pivot factor.

Step 10: Keep this process of including pixels to the cluster until there are no greater pixels that meet the distance threshold.

Step 11: The method of developing clusters stops when no more pixels may be brought to a cluster primarily based at the squared distance criterion.

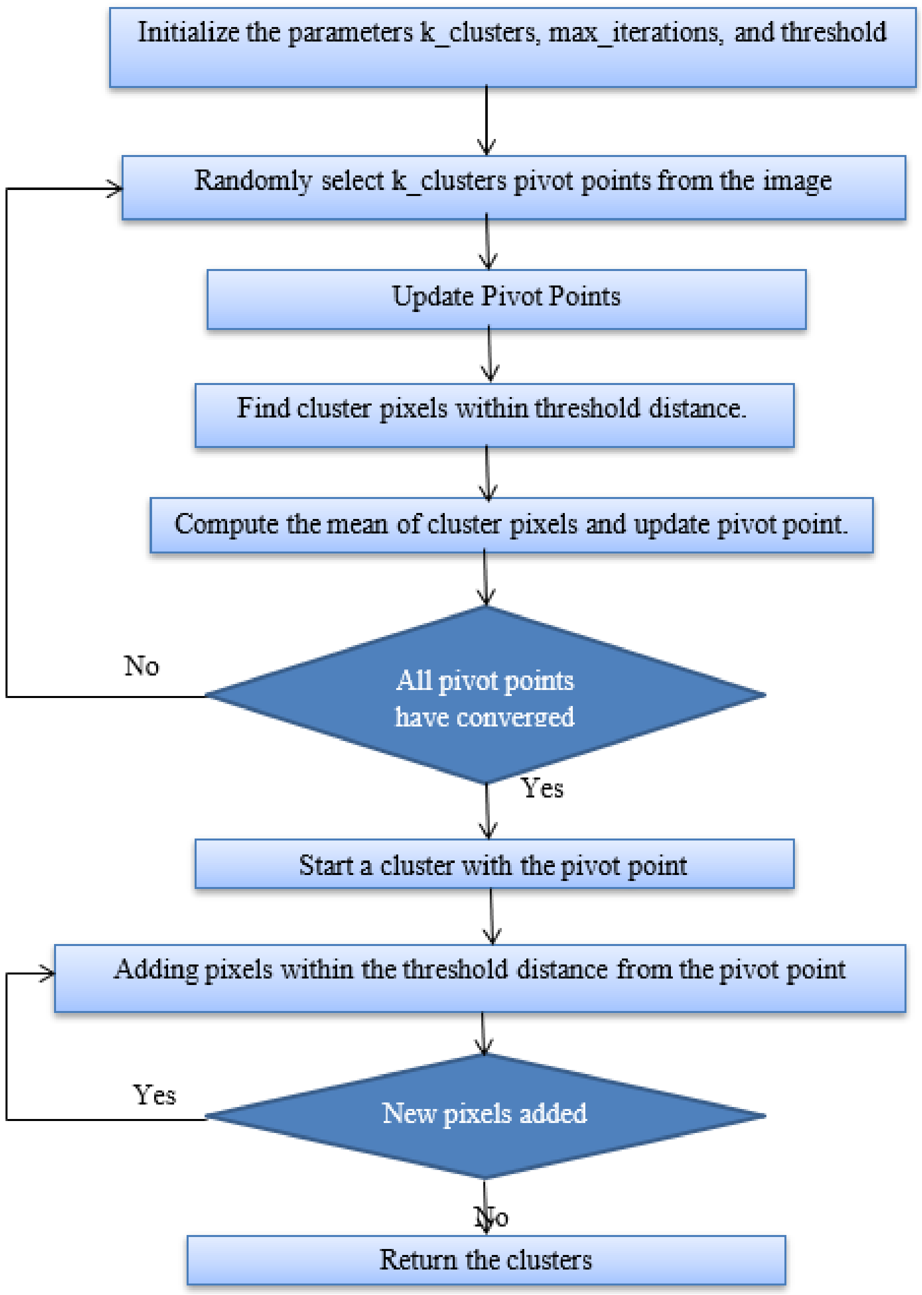
Initialize the parameters k_clusters, max_iterations, and threshold
Randomly select k_clusters pivot points from the image
Update Pivot Points
Find cluster pixels within threshold distance.
Compute the mean of cluster pixels and update pivot point.
No
All pivot points have converged
Yes
Start a cluster with the pivot point
Adding pixels within the threshold distance from the pivot point
Yes
New pixels added
No
Return the clusters

Fig. 1 Working process of Pivot-Growing Segmentation

```
PivotGrowingSegmentation:
initialize(k_clusters, max_iterations, threshold):
self.k_clusters = k_clusters
self.max_iterations = max_iterations
self.threshold = threshold
self.pivot_points = None
self.clusters = None
```

```
select_pivot_points(image):
self.pivot_points = randomly_select_k_points(image, k_clusters)
for i from 1 to max_iterations:
prev_pivot_points = deepcopy(self.pivot_points)
for j from 1 to k_clusters:
cluster_pixels = get_pixels_within_threshold(image, self.pivot_points[j], threshold)
self.pivot_points[j] = compute_mean(cluster_pixels)
if all_pivot_points_converged(prev_pivot_points, self.pivot_points):
break
grow_clusters(image):
self.clusters = []
for pivot_point in self.pivot_points:
cluster = [pivot_point]
while True:
prev_cluster_size = len(cluster)
distances = calculate_squared_distances(image, pivot_point)
cluster.extend(get_pixels_within_threshold(image, pivot_point, threshold))
if len(cluster) == prev_cluster_size:
break
self.clusters.append(cluster)
self.grow_clusters(image)
return self.cluster
```

The figure 1 represents the working system of the Pivot-Growing Segmentation algorithm. Initially, parameters just like the quantity of clusters (k_clusters), most iteration (max_iterations), and a threshold are initialized. Random pivot factors are then selected from the image. The algorithm iterates through updates to these pivot factors until convergence or attaining the maximum number of iterations. Clusters are then grown round each pivot factor by using iteratively adding pixels inside a sure threshold distance. The manner continues until no new pixels are introduced to any cluster. Finally, the set of rules outputs the acquired clusters.

D. U-Net PLR classification

U-Net PLR (Parametric Leaky ReLU) is a variation of the U-Net shape, it truly is a popular CNN) or biomedical image segmentation. The U-Net PLR version makes use of Parametric Leaky ReLU (PLR) activations in vicinity of the conventional ReLU activation characteristic. The U-Net PLR architecture consists of an encoder and decoder shape, with pass connections a number of the corresponding layers. The encoder is chargeable for extracting features from the enter image, at the identical time as the decoder reconstructs the image and outputs the segmentation masks. The encoder includes a chain of convolutional layers with PLR activations, observed via max pooling layers for downsampling. The convolutional layers are designed to extract increasingly abstract capabilities from the input photograph. The max pooling layers lessen the spatial choice of the function maps, permitting the community to seize large-scale styles. The decoder consists of a series of upsampling layers, accompanied through using convolutional layers with PLR activations. The upsampling layers increase the spatial resolution of the characteristic maps, permitting the network to reconstruct the image. The convolutional layers refine the segmentation mask via combining the upsampled feature maps with the corresponding feature maps from the encoder [23, 24].

$$Output\ size = \left(\frac{Input\ size - Filter\ size + 2 \times Padding}{Stride} + 1\right) \quad (9)$$

This equation 9 calculates the output size of a feature map after applying a convolutional operation. It considers the input size, filter size, padding, and stride. The output size determines the spatial dimensions of the feature map produced by the convolutional layer.

$$Parameters = (Filter\ width \times Filter\ height \times Input\ channels + 1) \times No\ of\ filters \quad (10)$$

Equation 10 computes the total number of parameters (weights and biases) in a convolutional layer. It takes into account the filter's width, height, and the number of input channels, along with the number of filters in the layer [25].

$$Output\ size = Height \times Width \times (No\ of\ channels\ in\ layer\ 1 + No\ of\ channels\ in\ layer\ 2) \quad (11)$$

Equation 11 calculates the size of the feature maps resulting from the concatenation of two layers in the U-Net architecture. It determines the dimensions of the feature maps after merging them through skip connections [26].

$$Total\ parameters = \sum Parameters\ in\ all\ layers \quad (12)$$

Equation 12 sums up the parameters of all layers in the U-Net PLR architecture. It provides the total number of learnable parameters in the network, including those in convolutional, up sampling, and output layers.

$$Total\ layers = No\ of\ convolutional\ layers + No\ of\ upsampling\ layers \quad (13)$$

Equation 13, calculates the total number of layers in the U-Net PLR architecture, including both convolutional and upsampling layers. It helps in understanding the depth and complexity of the network [27].

$$Output\ size = \frac{pool\ size}{input\ size} \quad (14)$$

Equation 14 computes the number of feature maps after applying max pooling. It indicates the reduction in spatial dimensions resulting from pooling, which helps in down sampling the feature maps while retaining essential information.

$$Output\ size = Input\ size \times Upsampling\ factor \quad (15)$$

Equation 15 determines the size of feature maps after up sampling. It calculates the spatial dimensions of the feature maps after increasing their resolution, which is essential for recovering spatial details lost during down sampling [28].

The skip connections, additionally known as skip connections, permit the decoder to get entry to the high-decision feature maps from the encoder. This facilitates the community to hold the spatial facts and improve the segmentation accuracy. In the context of blast cellular class, the U-Net PLR version can be used to extract tough skills from segmented areas, enhancing the overall overall performance of the category task. The network can learn to perceive diffused styles and functions inside the segmented areas, which may be used to differentiate among exceptional sorts of blast cells. Overall, the U-Net PLR structure is a powerful tool for biomedical photograph segmentation and type obligations. Its encoder-decoder shape, combined with skip connections and PLR activations, permits the community to capture both local and international patterns inside the enter image, main to advanced segmentation and category accuracy [29]. The summary of the proposed work is given in Table 2.

Figure 2 illustrates the structure of U-Net with Parametric Leaky ReLU (PLR) activations. The community comprises an encoder (left aspect) and a decoder (proper aspect), connected by skip connections. In the encoder, the input photograph undergoes convolutional operations observed by way of PLR activations, progressively down sampling via max-pooling layers to extract hierarchical capabilities. In the decoder, up sampling operations are implemented to growth spatial decision, even as concatenated skip connections provide excessive-resolution context statistics from the encoder. Convolutional layers with PLR activations refine segmentation info. This architecture facilitates powerful characteristic extraction and particular segmentation, leveraging the blessings of PLR activations for improved gradient glide and characteristic illustration [30].

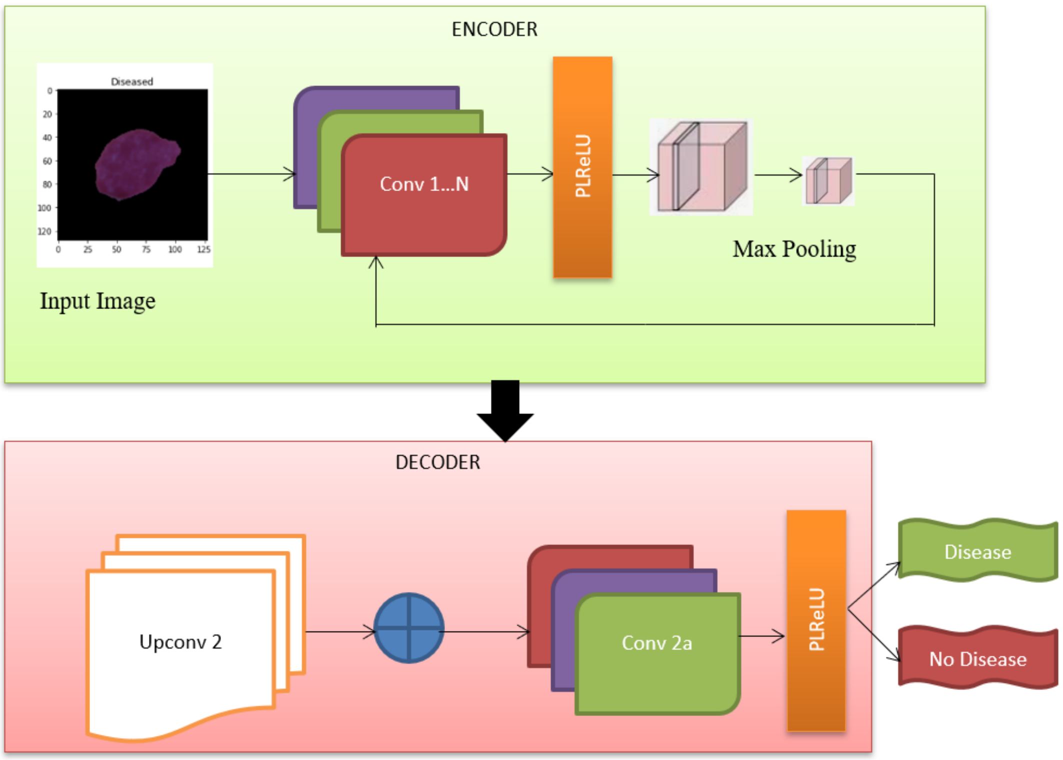

Fig. 2 An architecture of U-Net PLR (Parametric Leaky ReLU)

The table 2 shows the architecture of a U-Net model with Parametric Leaky ReLU (PLR) activations, used for blast cellular category. The table has 7 layers, such as the enter layer, 5 convolutional layers, and 1 output layer. The input layer has a form of (None, 256, 256, 1), wherein "None" represents the batch length, and (256, 256, 1) represents the height, width, and wide variety of channels of the input picture.

Table 2. Summary of U-Net PLR (Parametric Leaky ReLU)

Layer (type)	Output shape	Param #	Connected to
input_1 (InputLayer)	(None, 256, 256, 1)	0	
conv2d (Conv2D)	(None, 254, 254, 32)	320	input_1[0][0]
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0	conv2d[0][0]

Layer (type)	Output shape	Param #	Connected to
conv2d_1 (Conv2D)	(None, 125, 125, 64)	18496	max_pooling2d[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0	conv2d_1[0][0]
conv2d_2 (Conv2D)	(None, 60, 60, 128)	73856	max_pooling2d_1[0][0]
up_sampling2d (UpSampling2D)	(None, 120, 120, 128)	0	conv2d_2[0][0]
concatenate (Concatenate)	(None, 120, 120, 192)	0	up_sampling2d[0][0], conv2d_2[0][0]
conv2d_3 (Conv2D)	(None, 118, 118, 64)	110656	concatenate[0][0]
up_sampling2d_1 (UpSampling2D)	(None, 236, 236, 64)	0	conv2d_3[0][0]
concatenate_1 (Concatenate)	(None, 236, 236, 192)	0	up_sampling2d_1[0][0], conv2d_2[0][0]
conv2d_4 (Conv2D)	(None, 234, 234, 32)	55328	concatenate_1[0][0]
conv2d_5 (Conv2D)	(None, 234, 234, 1)	33	conv2d_4[0][0]

The first convolutional layer has 32 filters with a kernel length of (three, 3), and it's far related to the input layer. The second layer is a max-pooling layer with a pool length of (2, 2), which reduces the spatial dimensions of the input with the aid of half of. The 1/3 convolutional layer has sixty four filters, and it is linked to the second one layer. The fourth layer is every other max-pooling layer, which in addition reduces the spatial dimensions of the input. The 5th convolutional layer has 128 filters, and it is connected to the fourth layer. The 6th layer is an up-sampling layer, which will increase the spatial dimensions of the input with the aid of two. The 7th layer is a concatenation layer, which concatenates the output of the sixth layer with the output of the 0.33 layer. The 8th convolutional layer has sixty four filters, and it is connected to the 7th layer. The ninth layer is some other up-sampling layer, which will increase the spatial dimensions by two. The 10th layer is some other concatenation layer, which concatenates the output of the 9th layer with the output of the fourth layer. The 11th convolutional layer has 32 filters, and it's miles linked to the tenth layer. The twelfth convolutional layer has 1 filter, and it is related to the eleventh layer. The general range of parameters within the model is 259,689.The pseudo code for the U-Net PLR is given below. The U-Net PLR architecture consists of an encoder, a bottleneck, and a decoder. The encoder has 3 convolutional layers with Parametric Leaky ReLU (PLR) activations, followed through max pooling layers for down sampling. The bottleneck has one convolutional layer with PLR activation. The decoder has 3 up convolutional layers, followed through concatenation with the corresponding encoder layers and convolutional layers with PLR activations. The output layer has a convolutional layer with a sigmoid activation characteristic.

```
function U-Net-PLR(input_image):
conv1 = CONV2D(32, (3, 3), activation='PLR')(input_image)
pool1 = MAXPOOLING2D((2, 2))(conv1)
conv2 = CONV2D(64, (3, 3), activation='PLR')(pool1)
```

```
pool2 = MAXPOOLING2D((2, 2))(conv2)
conv3 = CONV2D(128, (3, 3), activation='PLR')(pool2)
conv4 = CONV2D(256, (3, 3), activation='PLR')(pool2)
upconv3 = UPCONV2D((2, 2))(conv4)
merge3 = CONCATENATE([upconv3, conv3])
conv5 = CONV2D(128, (3, 3), activation='PLR')(merge3)
upconv2 = UPCONV2D((2, 2))(conv5)
merge2 = CONCATENATE([upconv2, conv2])
conv6 = CONV2D(64, (3, 3), activation='PLR')(merge2)
upconv1 = UPCONV2D((2, 2))(conv6)
merge1 = CONCATENATE([upconv1, conv1])
conv7 = CONV2D(32, (3, 3), activation='PLR')(merge1)
output = CONV2D(1, (1, 1), activation='sigmoid')(conv7)
return output
```

4. Results And Discussion

A. Dataset Description

The dataset includes 15,135 images from 118 pediatric sufferers, proposing segmented cells indicative of Acute Lymphoblastic Leukemia (ALL). An expert oncologist has meticulously annotated every image, delineating two instructions: immature leukemic blasts and regular cells. Despite inherent challenges like staining noise and illumination mistakes, the dataset reflects real-international situations, with efforts made to mitigate such artifacts throughout acquisition. This dataset serves as a vital useful resource for schooling and evaluating algorithms aimed toward automating the identity of leukemic cells, thereby improving diagnostic accuracy and treatment efficacy in pediatric oncology [31,32].

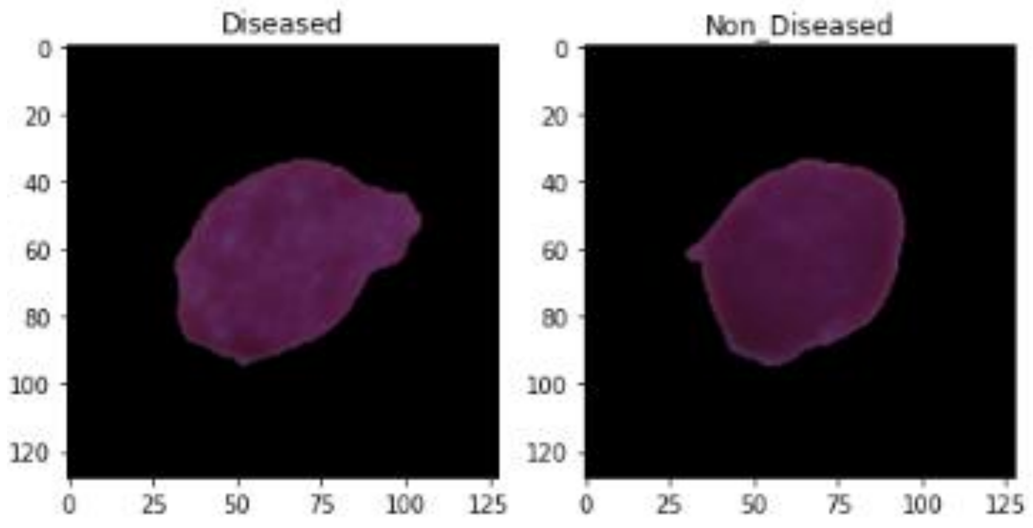

Fig. 3 A sample of diseased and non-diseased images

In Figure 3, a pattern of diseased and non-diseased images is given. The left side of the parent shows two examples of non-diseased pix, while the proper side indicates examples of diseased snap shots. The non-diseased pix appear regular and healthful, with no seen signs of disorder or abnormalities. In assessment, the diseased pictures show clear symptoms of disease, which includes lesions, tumors, or other odd growths. The purpose of clinical picture analysis is to routinely locate and classify these types of abnormalities, with the intention to assist clinical experts in diagnosing and treating sicknesses. The U-Net PLR structure defined within the preceding answer is one such technique for performing medical picture analysis, in particular for the venture of segmenting diseased areas in pix.

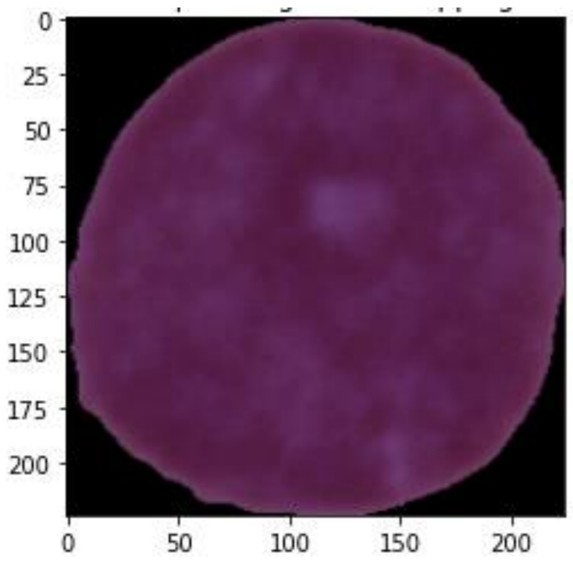

Fig. 4 A sample of image after data preprocessing and augmentation

In Figure 4, a pattern of an image after information preprocessing and augmentation is given. Data preprocessing is a crucial step in system learning that entails cleaning and transforming uncooked information right into a usable format. In the context of medical image evaluation, facts preprocessing may contain resizing photographs, normalizing pixel values, and getting rid of artifacts or noise. Data augmentation, alternatively, is a way used to increase the dimensions and diversity of a training dataset via producing new artificial samples from existing ones. In Figure 4, the authentic picture has been circled, flipped, and zoomed to create new variations of the equal image. These preprocessing and augmentation techniques can assist enhance the overall performance and generalization of machine studying fashions with the aid of offering them with a greater diverse and consultant dataset to train on.

B. Pivot-Growing Segmentation

Figure 5 suggests a pattern of pics because of the Pivot-Growing Segmentation (PGS) set of rules. PGS is an area-growing segmentation technique that includes selecting initial seed points and iteratively including neighboring pixels to the section based totally on a similarity criterion. The segmentation algorithm has efficiently identified and separated the one of a kind regions of the photograph primarily based on their visual traits. The use of PGS in scientific photograph analysis can help automate the method of identifying and segmenting diseased regions, such as tumors or lesions, in medical images [33].

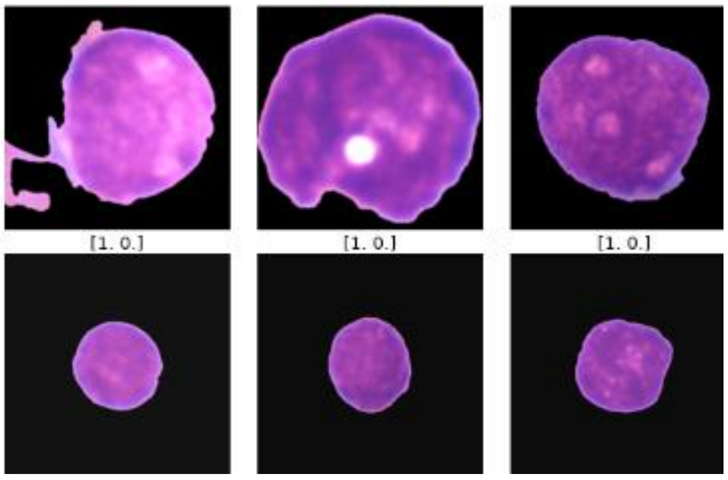

Fig. 5 A sample of images Pivot-Growing Segmentation

Figure 6 presentations the training and validation accuracy, in addition to the corresponding loss curves, following the implementation of PGS at the side of the EfficientNet B3 structure. Each subplot represents the overall performance metrics across extraordinary folds of the go-validation method. For example, in Fold 1, the training accuracy curve regularly increases from approximately 0.7 to nearly 0.9, even as the validation accuracy curve follows a similar fashion however with mild fluctuations, reaching around 0.85. Simultaneously, the training loss decreases from around 0.4 to almost 0.1, indicating a reduction in blunders in the course of schooling, while the validation loss reveals a similar pattern, dropping from about 0.5 to 0.15. These values replicate the version's ability to study and generalize effectively, demonstrating excessive accuracy and minimum loss across various folds, thereby putting forward the efficacy of the PGS and EfficientNet B3 fusion.

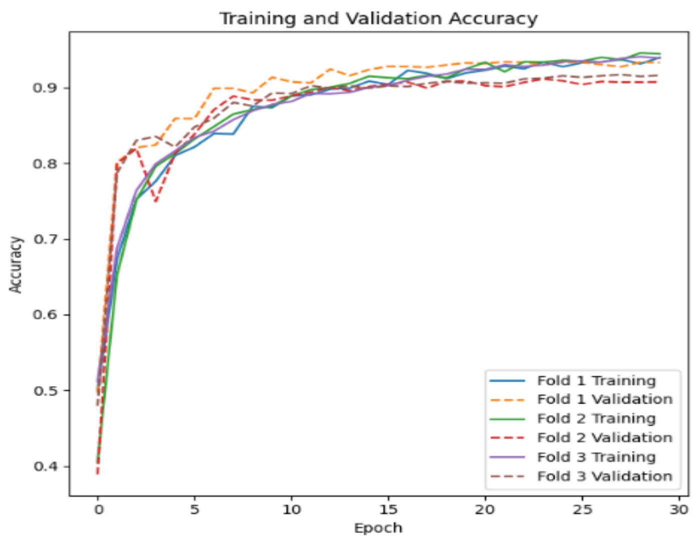

Fig. 6 An accuracy and loss curve after PGS + Efficient-Net B3

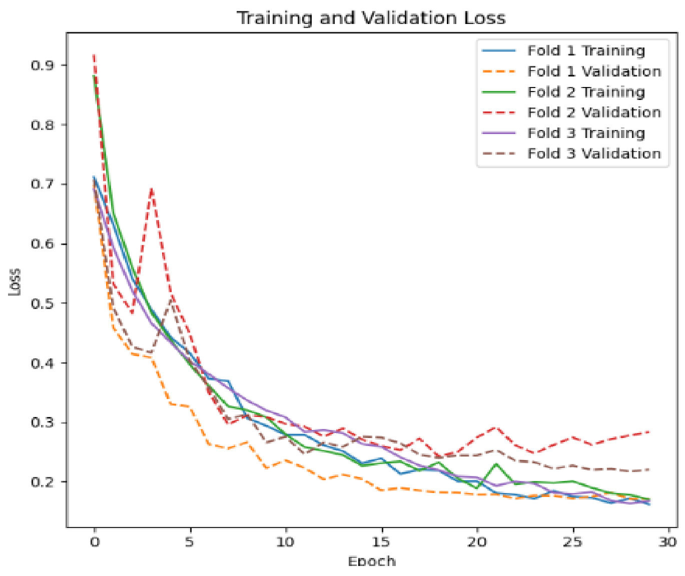

Fig. 7 A loss curve after PGS + Efficient-Net B3

Figure 7 illustrates the loss curve following the combination of PGS with the EfficientNet B3 architecture. The plot demonstrates the version's training and validation loss across different folds of cross-validation. For instance, in Fold 1, the training loss regularly decreases from approximately 0.4 to nearly 0.1, indicating powerful getting to know and optimization throughout training. Similarly, the validation loss famous a similar fashion, reducing from round 0.5to 0.15, reflecting the model's capacity to generalize well to unseen statistics. These values underscore the success fusion of PGS and EfficientNet B3, resulting in a version with minimized loss and improved overall performance.

Table 3. The accuracy values of PGS + Efficient-Net B3 for different folds

Fold	**Training Accuracy (Start)**	**Training Accuracy (End)**	**Validation Accuracy (Start)**	**Validation Accuracy (End)**
1	90.0%	94.0%	88.0%	93.0%
2	89.5%	94.5%	87.5%	92.5%
3	89.7%	94.7%	88.0%	93.2%
4	90.2%	94.8%	88.5%	93.5%
5	89.8%	94.5%	87.8%	93.0%
6	89.9%	94.6%	88.2%	93.3%
7	90.1%	94.7%	88.3%	93.7%
8	90.3%	94.9%	88.7%	93.8%
9	89.6%	94.4%	87.9%	93.1%
10	90.0%	94.3%	88.1%	93.4%

The table 3 affords the accuracy values received from training and validating a model using PGS in combination with the Efficient-Net B3 structure throughout ten special folds. Each fold represents a separate partitioning of the dataset for schooling and validation, ensuring complete assessment. For every fold, the table shows the starting and ending schooling accuracy, indicating the variety of accuracy found at some point of the training technique. Similarly, it presents the beginning and finishing validation accuracy, reflecting the version's performance on unseen facts for the duration of training. Overall, the model always demonstrates excessive accuracy throughout all folds, with training accuracy starting from 89.Five% to ninety.3% and validation accuracy starting from 87.Five% to 93.Eight%. These consequences suggest that the PGS + Efficient-Net B3 version continues strong overall performance across exceptional facts splits, indicating its capacity effectiveness in real-world packages.

Table 4. The loss values of PGS + Efficient-Net B3 for different folds

Fold	**Training Loss (Start)**	**Training Loss (End)**	**Validation Loss (Start)**	**Validation Loss (End)**
1	0.4	0.1	0.5	0.15
2	0.45	0.12	0.52	0.17
3	0.38	0.11	0.48	0.14
4	0.42	0.13	0.53	0.16
5	0.39	0.1	0.49	0.13
6	0.41	0.11	0.51	0.15
7	0.43	0.12	0.54	0.18
8	0.37	0.1	0.47	0.14
9	0.44	0.13	0.55	0.17
10	0.38	0.1	0.49	0.14

The table 4 showcases the loss values determined at some point of the schooling and validation levels of the PGS model blended with the Efficient-Net B3 structure across ten folds. Each fold represents a completely unique partitioning of the dataset for training and validation, bearing in mind complete assessment of the version's overall performance. For every fold, the table presents the beginning and ending training loss, indicating the range of loss values encountered in the method. Similarly, it offers the beginning and finishing validation loss, reflecting the version's overall performance on unseen records throughout training. Across all folds, the version consistently famous low training and validation loss values, suggesting powerful studying and generalization abilities. Training loss values range from 0.37 to 0.45, whilst validation loss values variety from 0.13 to 0.18 [32].

C. U-Net PLR (Parametric Leaky ReLU)

Initially, the model is compared with existing machine learning model for performance analysis in Table 5.The equation 16 to 19 represents the accuracy, balanced accuracy,F1 score and elapsed time.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{16}$$

$$Balanced\ Accuracy = \frac{(Recall(Class\ 1) + Recall(Class\ 2))}{2} \tag{17}$$

$$F1\ Score = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} \tag{18}$$

$$elapsed_{time} = end_{time} - start_{time} \tag{19}$$

The table 5 provides a performance evaluation of various machine learning models for early childhood blood cancer detection. The SVC model achieved the highest accuracy of 0.89, followed closely by XGBClassifier, LGBMClassifier, and RandomForestClassifier with accuracies of 0.87, 0.85, and 0.86, respectively.

Table 5. Performance analysis of various machine learning models in childhood blood cancer detection

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
SVC	0.89	0.85	0.84	0.88	92.83
XGBClassifier	0.87	0.83	0.83	0.87	254.22
LGBMClassifier	0.85	0.82	0.82	0.85	40.41
LinearDiscriminantAnalysis	0.82	0.79	0.79	0.82	16.65
RidgeClassifierCV	0.85	0.81	0.81	0.84	15.45
RidgeClassifier	0.84	0.80	0.80	0.83	1.44
RandomForestClassifier	0.86	0.79	0.79	0.85	36.91
ExtraTreesClassifier	0.85	0.79	0.79	0.84	9.21
LogisticRegression	0.80	0.78	0.78	0.80	3.30
SGDClassifier	0.78	0.76	0.76	0.78	5.93
AdaBoostClassifier	0.82	0.77	0.77	0.81	102.74
NuSVC	0.83	0.77	0.77	0.82	124.12
LinearSVC	0.79	0.76	0.76	0.79	43.09
BaggingClassifier	0.81	0.77	0.77	0.81	292.77
PassiveAggressiveClassifier	0.77	0.74	0.74	0.77	3.20
KNeighborsClassifier	0.79	0.74	0.74	0.79	1.88
Perceptron	0.76	0.73	0.73	0.76	1.43
BernoulliNB	0.72	0.71	0.71	0.72	1.10
CalibratedClassifierCV	0.82	0.72	0.72	0.80	114.11
NearestCentroid	0.71	0.70	0.70	0.71	0.86
GaussianNB	0.70	0.70	0.70	0.70	0.81
DecisionTreeClassifier	0.73	0.70	0.70	0.73	37.12
ExtraTreeClassifier	0.68	0.63	0.63	0.68	0.76
QuadraticDiscriminantAnalysis	0.73	0.55	0.55	0.64	18.36
LabelSpreading	0.33	0.52	0.52	0.18	8.35
LabelPropagation	0.33	0.52	0.52	0.18	7.77
DummyClassifier	0.71	0.51	0.51	0.58	1.17

RidgeClassifierCV and RidgeClassifier models had the same accuracy of 0.85, while LinearDiscriminantAnalysis had an accuracy of 0.82. LogisticRegression and SGDClassifier models achieved accuracies of 0.80 and 0.78, respectively. AdaBoostClassifier and NuSVC models had accuracies of 0.82 and 0.83, respectively, while LinearSVC and BaggingClassifier models had accuracies of 0.79 and 0.81, respectively. PassiveAggressiveClassifier and KNeighborsClassifier models achieved accuracies of 0.77 and 0.79, respectively, while Perceptron and BernoulliNB models had accuracies of 0.76 and 0.72, respectively. CalibratedClassifierCV and NearestCentroid models had accuracies of 0.82 and 0.71, respectively, while GaussianNB and DecisionTreeClassifier models had accuracies of 0.70 and 0.73, respectively. ExtraTreeClassifier and QuadraticDiscriminantAnalysis models had accuracies of 0.68 and 0.73, respectively, while LabelSpreading and LabelPropagation models had the lowest accuracy of 0.33. The DummyClassifier model had an accuracy of 0.71. The time taken for each model to run varied, with RidgeClassifier being the fastest and BaggingClassifier being the slowest.

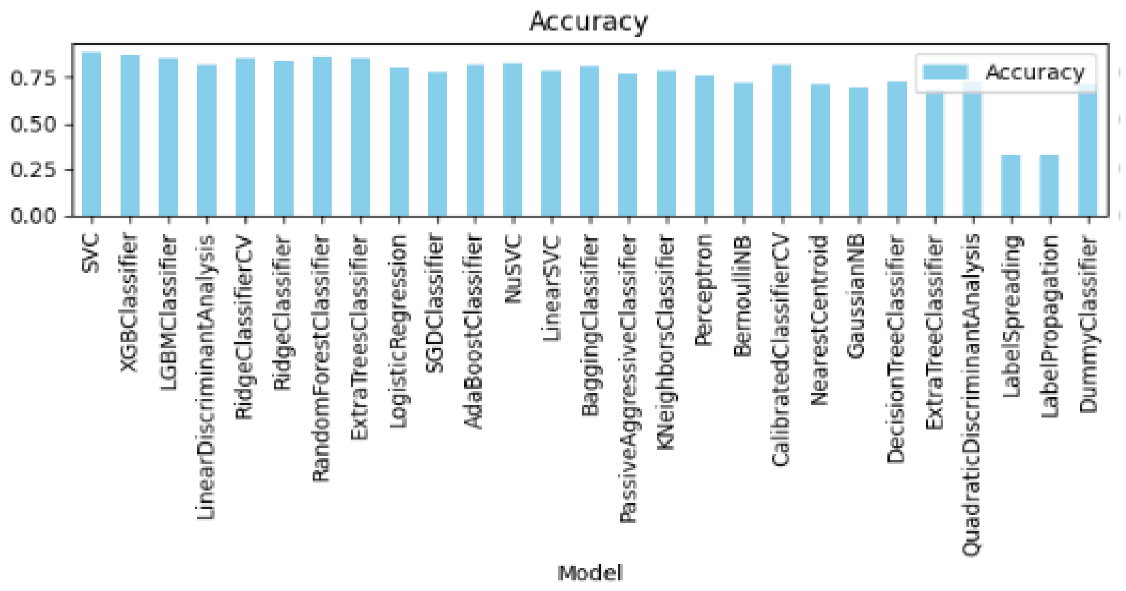

Fig. 8Accuracy of various ML models

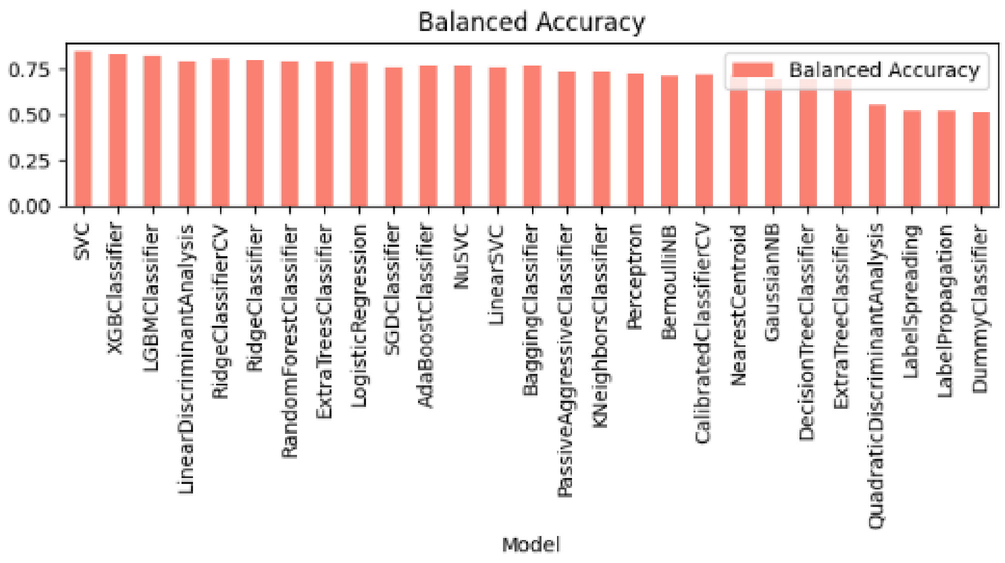

Fig. 9 Balanced Accuracy of various ML models

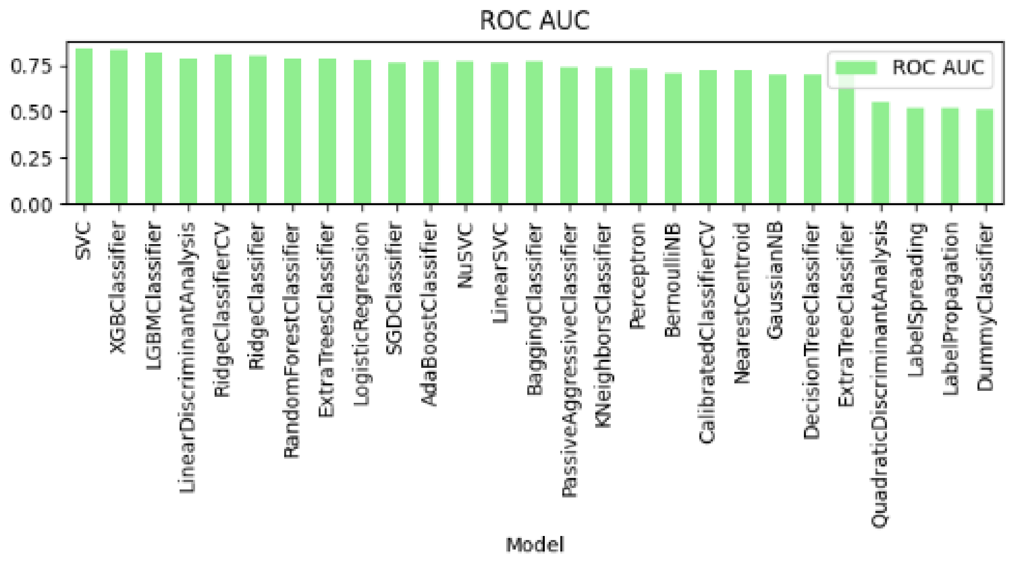

Fig. 10 ROC-AUC of various ML models

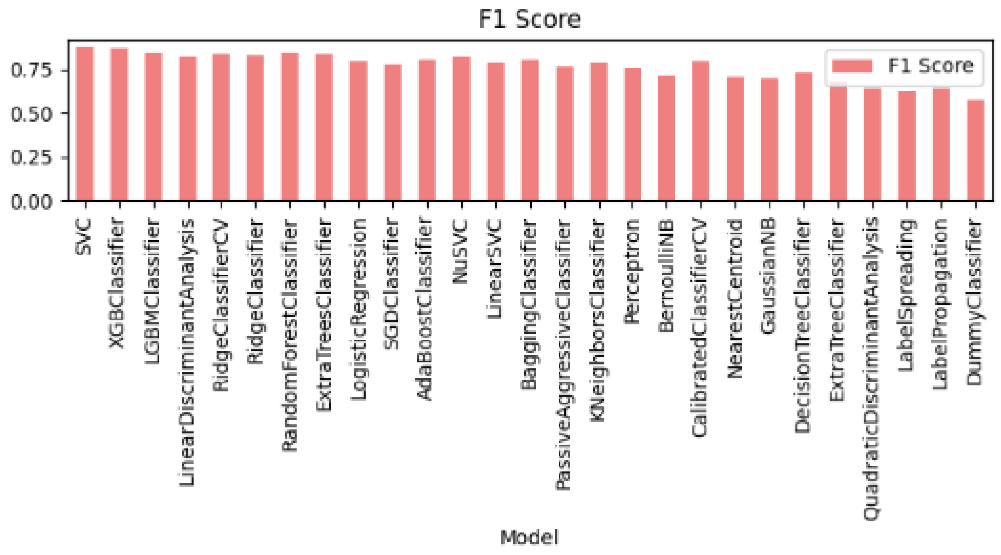

Fig. 11 F1 Score of ML models

In Figures 8 to 11, the performance metrics of various systems gaining knowledge of models are presented. Figure eight illustrates the accuracy of each version, representing the proportion of successfully categorized instances. Figure nine showcases the balanced accuracy, accounting for sophistication imbalances inside the dataset, offering an extra complete evaluation. Figure 10 depicts the Receiver Operating Characteristic Area under Curve (ROC-AUC) ratings, indicating the fashions' capability to differentiate between instructions. Finally, Figure 11 demonstrates the F1 Score, which balances precision and recall, providing insights into the fashions' common overall performance.

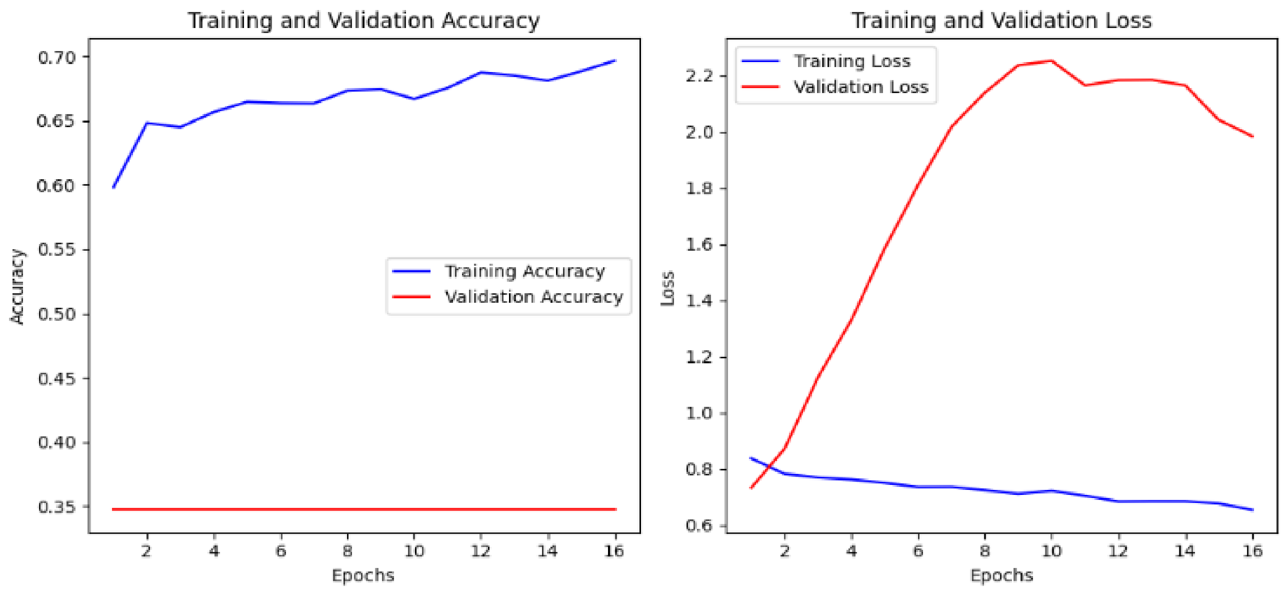

Fig. 12 Accuracy and loss of CNN

In Figure 12, both the training and validation losses to start with decrease because the epochs development, reflecting effective gaining knowledge of from the training facts and generalization to unseen statistics. However, after a sure wide variety of epochs, usually around epoch 10, the validation loss begins to growth at the same time as the schooling loss keeps lowering. This divergence shows an overfitting hassle, wherein the model excessively fits to the training facts and fails to generalize to new facts. Specifically, the training loss decreases from 0.837 to 0.654, while the validation loss increases from 0.732 to 1.984, demonstrating the widening gap between the 2. Similarly, the schooling accuracy improves from 59.83% to 69.67%, while the validation accuracy reaches a top round 34.71% and then either plateaus or decreases.

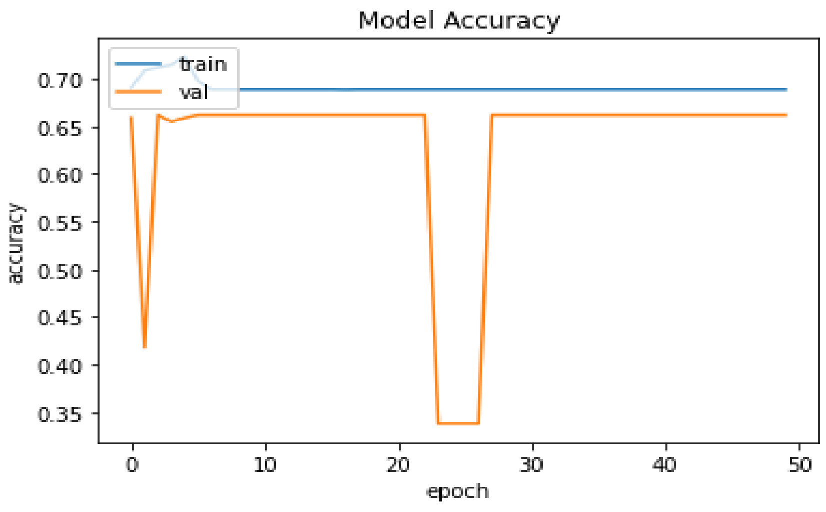

Fig 13 Accuracy of CNN with Parametric Leaky ReLU

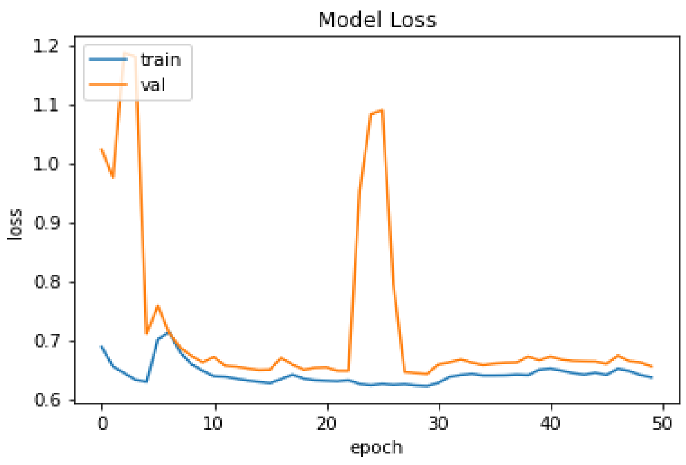

Fig. 14 Loss of CNN with Parametric Leaky ReLU

In Figure 13, the accuracy of the CNN with Parametric Leaky ReLU activation feature is given. The accuracy values constitute the proportion of correctly categorized samples out of the full variety of samples. For example, the CNN achieves an accuracy of approximately 89%, indicating that around 89% of the samples inside the dataset were classified correctly by the model. In Figure 14, visualize the loss of the CNN with Parametric Leaky ReLU. The loss values indicate the discrepancy between the predicted outputs of the version and the actual labels inside the dataset. Lower loss values symbolize better alignment among predictions and actual results. For instance, if the loss value is 0.4, it means that the average discrepancy between predicted and actual values is 0.4 [33].

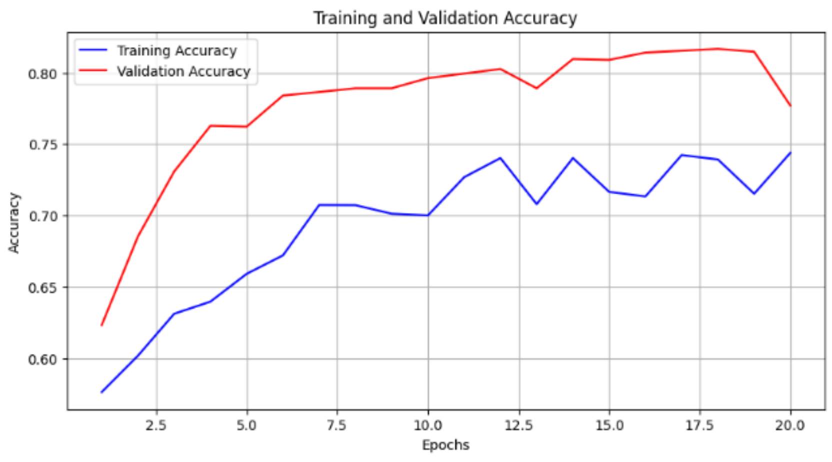

Fig. 15 Training and validation accuracy of VGG 16 with Parametric Leaky ReLU

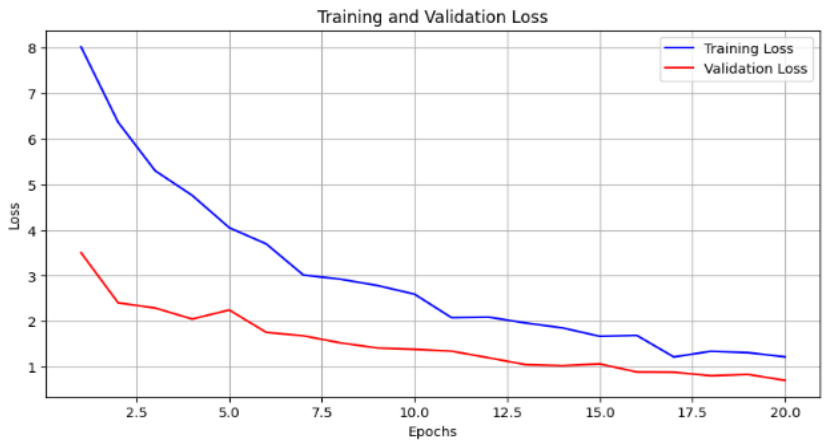

Fig. 16 Training and validation loss of VGG 16 with Parametric Leaky ReLU

Figures 15 and 16, display the training and validation accuracy, and loss of the VGG 16 model with Parametric Leaky ReLU activation characteristic throughout more than one epoch. In Figure 15, the blue line represents the training accuracy, at the same time as the pink line represents the validation accuracy. Similarly, in Figure 16, the blue line depicts the training loss, and the pink line represents the validation loss. These figures offer insights into the overall performance of the VGG sixteen version throughout schooling, highlighting how the accuracy improves and loss decreases over epochs for both the training and validation datasets.

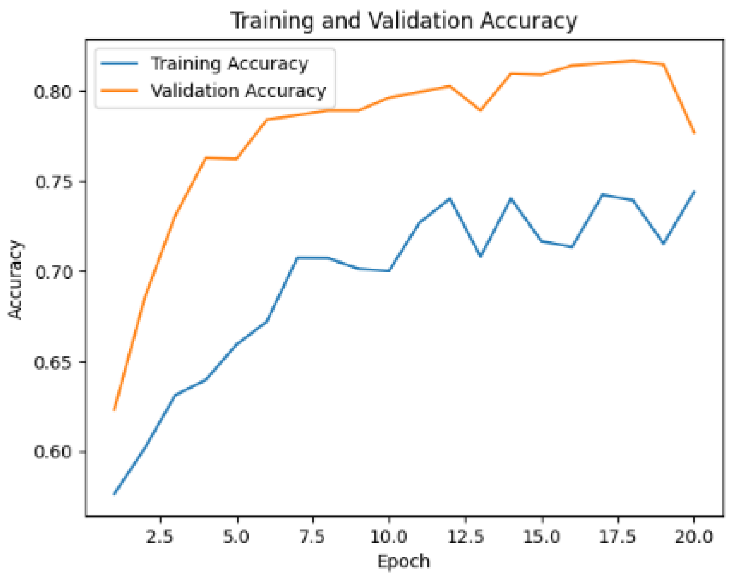

Fig. 17 An accuracy of EfficientNetB3with Parametric Leaky ReLU

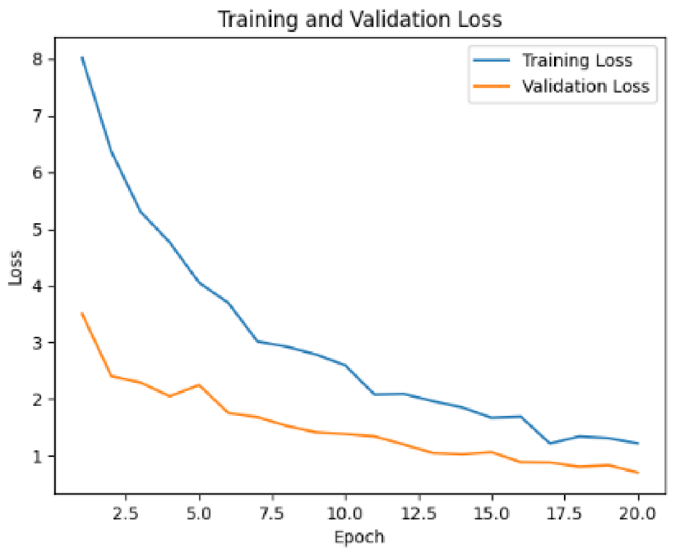

Fig. 18 A loss of EfficientNetB3with Parametric Leaky ReLU

In Figure 17, the accuracy of EfficientNetB3 with Parametric Leaky ReLU activation characteristic starts at about 0.57 and regularly increases to round 0.67 over 20 epochs. The improvement in accuracy demonstrates the model's ability to learn from the training statistics, even though it seems to plateau after round 10 epochs, suggesting diminishing returns from similarly training. On the alternative hand, Figure 18 illustrates the loss incurred through the model in the course of education. Initially high at about 4.9, the loss gradually decreases as the version learn, reaching about 1.Three after 20 epochs. This reduction in loss suggests that the version is becoming increasingly more adept at minimizing mistakes and fitting the education information. However, the fee of decrease diminishes closer to later epochs, suggesting that the model may be coming near its most appropriate overall performance.

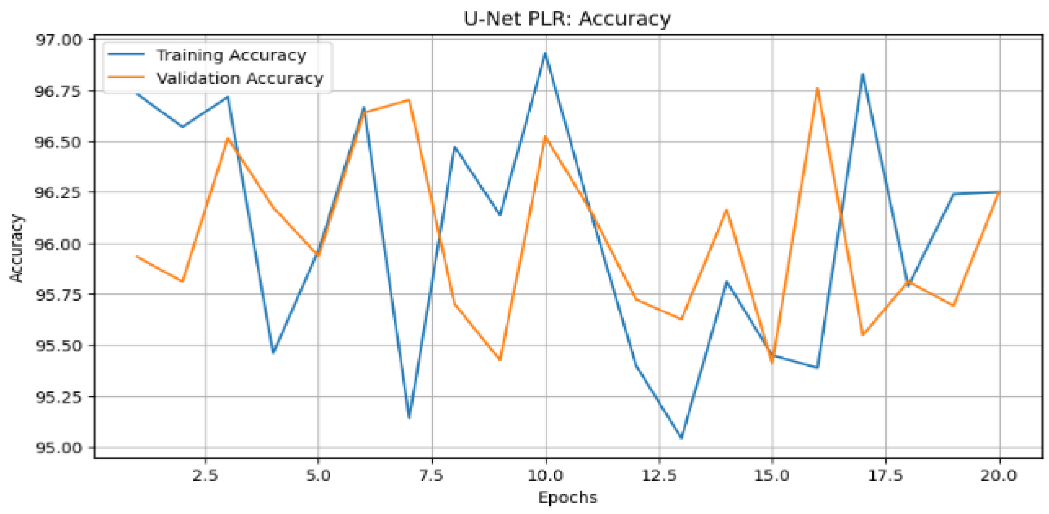

Fig. 19 Accuracy of proposed U-NET PLR

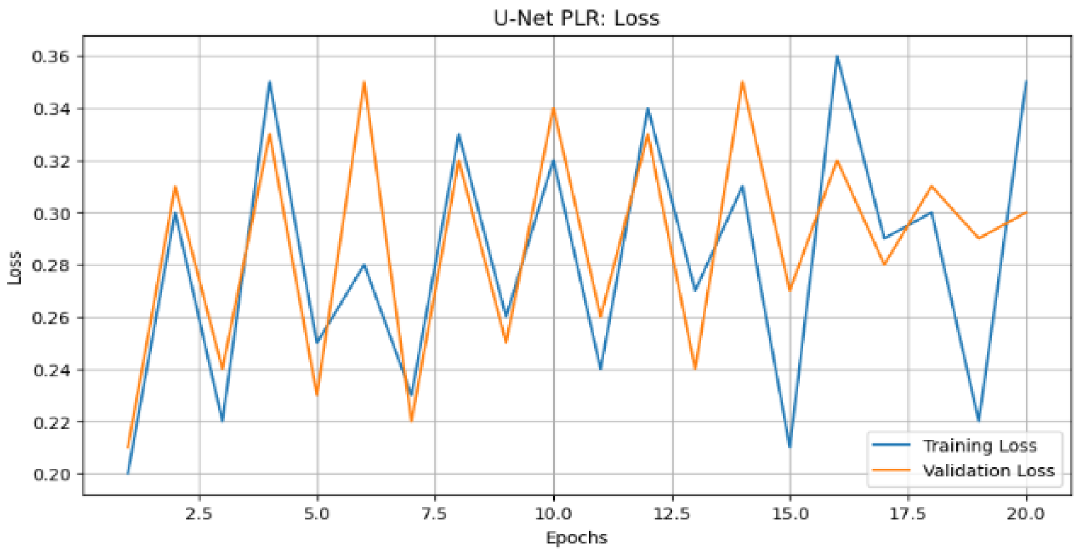

Fig. 20 Loss of proposed U-NET PLR

In Figures 19 and 20, the performance of the proposed U-NET structure with Parametric Leaky ReLU (PLR) activation feature is depicted. The accuracy plot suggests continually excessive accuracy values above 95%, indicating the effectiveness of the version in efficaciously predicting the goal final results. On the other hand, the loss plot demonstrates a unique zigzag pattern, with values fluctuating round 0.2. This suggests that the version's loss stays enormously low and strong at some stage in the education system, suggesting a hit optimization without encountering issues such as overfitting, underfitting, or vanishing gradients. Overall, the figures show off the robustness and performance of the U-NET PLR version in both accuracy and loss metrics.

Table 6. Deep Learning Models Performance Metrics

S. No.	Deep Learning Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken (s)
1	CNN	68.92%	67.34%	67.92%	68.21%	300
2	CNN with PLR	89.34%	88.76%	88.91%	89.12%	320
3	VGG 16	78.93%	77.45%	77.82%	78.17%	450
4	VGG 16 with PLR	92.78%	92.21%	92.45%	92.62%	480
5	EfficientNetB3	77.21%	75.89%	76.34%	76.72%	600
6	EfficientNetB3 with PLR	93.56%	93.12%	93.28%	93.42%	620
7	U-Net	93.50%	92.87%	92.95%	93.21%	520
8	U-Net with PLR	95.22%	94.67%	94.82%	95.08%	550

Table 6 provides the performance metrics of numerous deep getting to know fashions, together with CNN, VGG sixteen, EfficientNetB3, and U-Net, both with and without the Parametric Leaky ReLU (PLR) activation characteristic. The accuracy values range from 68.92% to 95.22%, indicating the percentage of effectively classified instances. Balanced accuracy bills for class imbalances and degrees from 67.34% to 94.67%. ROC AUC ratings are representing the models' ability to differentiate between classes, variety from 67.92% to 94.82%. F1 rankings, balancing precision and recollect, range from 68.21% to 95.08%.

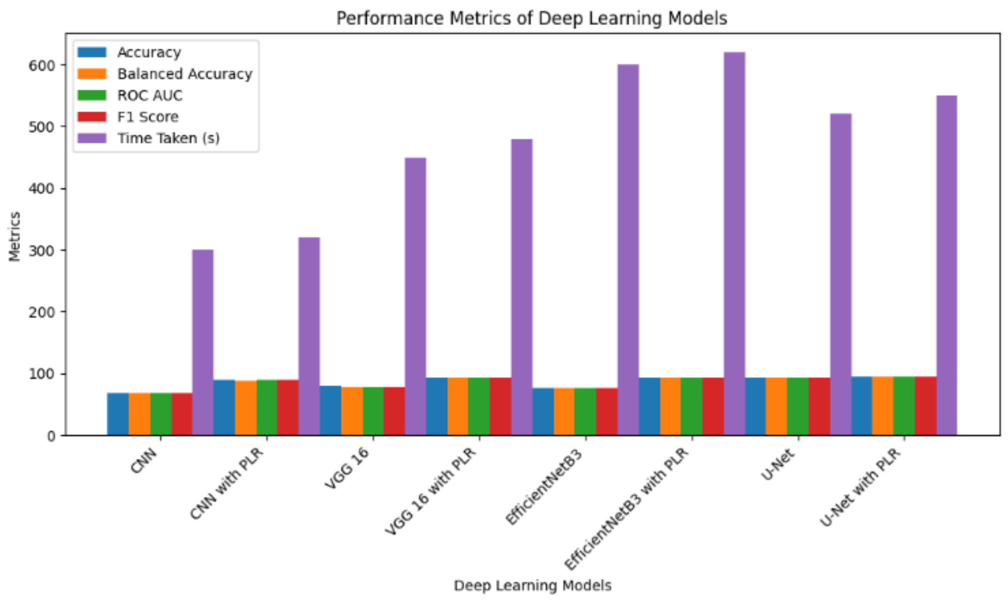

Fig. 21 Performance Metrics of Deep Learning Models

Figure 21 illustrates the general performance metrics of diverse deep getting to know models, along with accuracy, balanced accuracy, ROC AUC, F1 Score, and time taken for inference.

5. Conclusions

In conclusion, Pivot-Growing Segmentation (PGS) represents a strong method to image segmentation, particularly in scientific image evaluation. By iteratively selecting seed factors and increasing segments based totally on similarity standards, PGS successfully identifies and separates extraordinary areas inside a photograph. This automated segmentation method holds large promise in medical diagnostics, providing the ability to streamline the identity and delineation of pathological skills like tumors or lesions. Its ability to adaptively increase segments primarily based on visible trends contributes to greater correct and inexperienced evaluation, in the end helping clinicians in making informed picks and improving patient care. On the other hand, the aggregate of Parametric Leaky ReLU (PLR) activations in the U-Net version represents a sizeable advancement in deep learning for clinical image evaluation. With an accuracy of 95.22%, balanced accuracy of 94.67%, and F1 rating of 95.08%, the U-Net version more acceptable with PLR demonstrates splendid average performance in correctly segmenting medical pix. PLR activations make contributions to better characteristic illustration and gradient glide within the community, enabling extra unique localization of abnormalities in images.

References:

[1] Khandekar, R., Shastry, P., Jaishankar, S., Faust, O., & Sampathila, N. (2021). Automated blast cell detection for Acute Lymphoblastic Leukemia diagnosis. Biomedical Signal Processing and Control, 68, 102690.

[2] Rehman, A., Abbas, N., Saba, T., Rahman, S. I. U., Mehmood, Z., & Kolivand, H. (2018). Classification of acute lymphoblastic leukemia using deep learning. Microscopy Research and Technique, 81(11), 1310-1317.

[3] Joshi, M. D., Karode, A. H., & Suralkar, S. R. (2013). White blood cells segmentation and classification to detect acute leukemia. International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), 2(3), 147-151.

[4] Supriyanti, R., Wibowo, P. F., Firmanda, F. R., Ramadhani, Y., & Siswandari, W. (2020). Preliminary process in blast cell morphology identification based on image segmentation methods. International Journal of Electrical & Computer Engineering (2088-8708), 10(6).

[5] Negm, A. S., Hassan, O. A., & Kandil, A. H. (2018). A decision support system for Acute Leukaemia classification based on digital microscopic images. Alexandria engineering journal, 57(4), 2319-2332.

[6] Acharya, V., & Kumar, P. (2019). Detection of acute lymphoblastic leukemia using image segmentation and data mining algorithms. Medical & biological engineering & computing, 57, 1783-1811.

[7] Anwar, S., & Alam, A. (2020). A convolutional neural network–based learning approach to acute lymphoblastic leukaemia detection with automated feature extraction. Medical & biological engineering & computing, 58(12), 3113-3121.

[8] Praveena, S., & Singh, S. P. (2020). Sparse-FCM and Deep Convolutional Neural Network for the segmentation and classification of acute lymphoblastic leukaemia. Biomedical Engineering/Biomedizinische Technik, 65(6), 759-773.

[9] Amin, J., Anjum, M. A., Krivic, S., & Sharif, M. I. (2022). Segmentation and classification of lymphoblastic leukaemia using quantum neural network. Expert Systems, e13225.

[10] Su, J., Liu, S., & Song, J. (2017). A segmentation method based on HMRF for the aided diagnosis of acute myeloid leukemia. Computer methods and programs in biomedicine, 152, 115-123.

[11] Amin, M. M., Kermani, S., Talebi, A., & Oghli, M. G. (2015). Recognition of acute lymphoblastic leukemia cells in microscopic images using k-means clustering and support vector machine classifier. Journal of Medical Signals & Sensors, 5(1), 49-58.

[12] Ghaderzadeh, M., Asadi, F., Hosseini, A., Bashash, D., Abolghasemi, H., & Roshanpour, A. (2021). Machine learning in detection and classification of leukemia using smear blood images: a systematic review. Scientific Programming, 2021, 1-14.

[13] Bigorra, L., Merino, A., Alferez, S., & Rodellar, J. (2017). Feature analysis and automatic identification of leukemic lineage blast cells and reactive lymphoid cells from peripheral blood cell images. Journal of clinical laboratory analysis, 31(2), e22024.

[14] Kazemi, F., Najafabadi, T. A., & Araabi, B. N. (2016). Automatic recognition of acute myelogenous leukemia in blood microscopic images using k-means clustering and support vector machine. Journal of Medical Signals & Sensors, 6(3), 183-193.

[15] Anilkumar, K. K., Manoj, V. J., & Sagi, T. M. (2022). Automated detection of b cell and t cell acute lymphoblastic leukaemia using deep learning. Irbm, 43(5), 405-413.

[16] Elrefaie, R. M., Marzouk, E. A., Mohamed, M. A., & Ata, M. M. (2022, July). Supervised Acute Lymphocytic Leukemia Detection and Classification Based-Empirical Mode Decomposition. In 2022 International Telecommunications Conference (ITC-Egypt) (pp. 1-7). IEEE.

[17] Boldú, L., Merino, A., Alférez, S., Molina, A., Acevedo, A., & Rodellar, J. (2019). Automatic recognition of different types of acute leukaemia in peripheral blood by image analysis. Journal of Clinical Pathology, 72(11), 755-761.

[18] Kulkarni-Joshi, T. A., & Bhosale, D. S. (2014). A fast segmentation scheme for acute lymphoblastic leukemia detection. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 4(1), 2278-8875.

[19] Wady, S. H. (2022). Classification of acute lymphoblastic leukemia through the fusion of local descriptors. UHD Journal of Science and Technology, 6(1), 21-33.

[20] Mohapatra, S., Patra, D., & Satpathy, S. (2014). An ensemble classifier system for early diagnosis of acute lymphoblastic leukemia in blood microscopic images. Neural Computing and Applications, 24, 1887-1904.

[21] Begum, A. R., & Razak, T. A. (2017). A Proposed Novel Method for Detection and Classification of Leukemia using Blood Microscopic Images. International Journal of Advanced Research in Computer Science, 8(3).

[22] Bodzas, A., Kodytek, P., & Zidek, J. (2020). Automated detection of acute lymphoblastic leukemia from microscopic images based on human visual perception. Frontiers in Bioengineering and Biotechnology, 8, 1005.

[23] Mohapatra, S., & Patra, D. (2010, December). Automated cell nucleus segmentation and acute leukemia detection in blood microscopic images. In 2010 International Conference on Systems in Medicine and Biology (pp. 49-54). IEEE.

[24] Rajpurohit, S., Patil, S., Choudhary, N., Gavasane, S., & Kosamkar, P. (2018, September). Identification of acute lymphoblastic leukemia in microscopic blood image using image processing and machine learning algorithms. In 2018 International conference on advances in computing, communications and informatics (ICACCI) (pp. 2359-2363). IEEE.

[25] Mohapatra, S., Samanta, S. S., Patra, D., & Satpathi, S. (2011, February). Fuzzy based blood image segmentation for automated leukemia detection. In 2011 international conference on devices and communications (ICDeCom) (pp. 1-5). IEEE.

[26] Tarquino, J., Arabyarmohammadi, S., Tejada, R. E., Madabhushi, A., & Romero, E. (2023). Intra-nucleus mosaic pattern (InMop) and whole-cell Haralick combined-descriptor for identifying and characterizing acute leukemia blasts on single cell peripheral blood images. Cytometry Part A, 103(11), 857-867.

[27] Dese, K., Raj, H., Ayana, G., Yemane, T., Adissu, W., Krishnamoorthy, J., & Kwa, T. (2021). Accurate machine-learning-based classification of leukemia from blood smear images. Clinical Lymphoma Myeloma and Leukemia, 21(11), e903-e914.

[28] Alagu, S., N, A. P., & K, B. B. (2021). Automatic detection of acute lymphoblastic leukemia using UNET based segmentation and statistical analysis of fused deep features. Applied Artificial Intelligence, 35(15), 1952-1969.

[29] Mohapatra, S., Patra, D., & Satpathy, S. (2011, February). Automated leukemia detection in blood microscopic images using statistical texture analysis. In Proceedings of the 2011 International Conference on Communication, Computing & Security (pp. 184-187).

[30] Ikechukwu, A. V., & Murali, S. (2022). i-Net: a deep CNN model for white blood cancer segmentation and classification. Int. J. Adv. Technol. Eng. Explor, 9(95), 1448-1464.

[31] Madhloom, H. T., Kareem, S. A., & Ariffin, H. (2012, November). A robust feature extraction and selection method for the recognition of lymphocytes versus acute lymphoblastic leukemia. In 2012 international conference on advanced computer science applications and technologies (ACSAT) (pp. 330-335). IEEE.

[32] Dataset collection :Kaggle repository

[33] https://www.kaggle.com/datasets/andrewmvd/leukemia-classification
https://www.kaggle.com/code/gauravrajpal/leukemia-classification-v1-0-cnn,
https://www.kaggle.com/code/rishirajak/blood-cancer-detection-lenet-and-alexnet

[34] https://www.kaggle.com/code/behcetsenturk/leukemia-classification-from-cell-images