Computers and Electrical Engineering 110 (2023) 108913

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Integrating explainability into deep learning-based models for white blood cells classification

Kunal Bhatia [a], Sabrina Dhalla [a,*], Ajay Mittal [a], Savita Gupta [a], Aastha Gupta [b], Alka Jindal [c]

[a] Department of CSE, UIET, Panjab University, Chandigarh, India
[b] School of Technology, Management and Engineering (STME), SVKM's NMIMS University (Deemed to be University), Chandigarh Campus, Chandigarh, India
[c] Department of CSE, Punjab Engineering College, Chandigarh, India

ARTICLE INFO

Keywords:
Explainable AI (XAI)
Deep learning
White blood cells (WBCs)
LIME

ABSTRACT

White blood cells (WBCs) are crucial constituents of the blood that protect the human body against infections and viruses. The classification of WBCs in a blood smear image is used to diagnose a range of haematological disorders. The manual identification of WBCs can result in potential errors, necessitating the need for automated systems that can assist in classification. Recently, various deep learning models, such as DenseNet121, Xception, MobileNetV2, ResNet50, and VGG16, have been used to classify WBCs. However, the available classification models are black boxes because their decisions are difficult for humans to understand without further exploration. The interpretability and explainability of these models are essential, as their decisions can have severe consequences for patients. In this paper, we integrate an explainable AI (XAI) technique called local interpretable model-agnostic explanations (LIME) with the DenseNet121 classification model for WBC classification. Interpretable results would allow the users to understand and verify the model's predictions, enhancing their confidence in the automated diagnosis.

1. Introduction

Cellular entities in the blood perform crucial functions in the human body, such as regulating blood circulation, oxygen transportation, and immune response to infections. The primary components of blood include white blood cells (WBCs), red blood cells (RBCs), platelets, and plasma. WBCs can be classified as *eosinophils*, *basophils*, *monocytes*, *lymphocytes*, and *neutrophils*, each with a specific role in fighting different types of infections caused by bacteria, viruses, parasites, and fungi. Fig. 1 depicts the appearance of different types of WBCs in a blood smear image. *Neutrophils* constitute up 50%–70% of circulating WBCs and are the first line of defence against infections. *Eosinophils* constitute 0%–5% of WBCs and combat parasitic infections. *Basophils* release histamine, which responds to the body's sensitivity to various allergies and is the least present among other types of WBCs. *Monocytes* aid the immune system in killing dangerous organisms, accounting for only 3%–9% of total WBCs. *Lymphocytes* generate antibodies and constitute 25%–30% of WBCs. A deficiency in these components can be dangerous and lead to fatal cancers such as leukemia. Therefore, classifying and counting WBCs are critical for diagnosing various haematological diseases.

* Corresponding author.
E-mail address: dhallasabrina@gmail.com (S. Dhalla).

https://doi.org/10.1016/j.compeleceng.2023.108913
Received 24 December 2022; Received in revised form 8 June 2023; Accepted 6 August 2023
Available online 23 August 2023

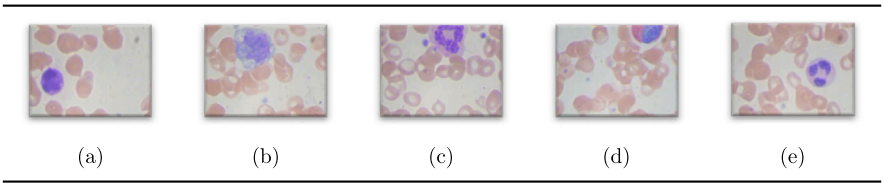

Fig. 1. Types of WBCs: (a) Basophils, (b) Monocytes, (c) Lymphocyte, (d) Eosinophils, and (e) Neutrophils.

WBCs and RBCs in the human blood in varied abundance, 1:600, skews the cardinality of classification classes, making WBC classification extremely challenging. The conventionally employed manual classification process is labor-intensive, time-consuming, error-prone, and subject to large inter- and intra- observer bias. These limitations necessitate the automation of the WBC classification process. Various deep learning models have recently been proposed to classify WBCs automatically [1,2]. However, deep learning-based models are black boxes, and it is difficult to understand how they arrive at their predictions or decisions. Suppose a non-interpretable AI model produces a diagnosis different from what a physician would expect based on their expertise and experience. In that case, the physician may be reluctant to accept the AI's diagnosis, leading to delayed treatment or missed diagnoses. Additionally, non-interpretable AI models can lead to biases and discrimination in medical diagnoses, as the underlying decision-making process of the model needs to be more transparent and can be influenced by various factors that may not be immediately apparent. World Health Organization (WHO) has also emphasized ensuring transparency, explainability and intelligibility in using AI for health [3].

Model-agnostic interpretation methods, known as *eXplainable AI* (XAI) models, gained popularity in 2017 and can help bring transparency to AI-models' results by understanding the intrinsic mechanism and logic of AI-models' inference. XAI models such as instances-based (LIME), feature-based (SHAP), and gradient-based methods like DeepLIFT or Attribution Maps have been developed. Among these, LIME provides locally reliable predictions and minimizes the locality-aware loss using an approximation technique, which samples instances by randomly removing features to yield perturbed instances for which an interpretable model is learned [4]. The XAI has been a game-changer for the healthcare industry because it helps medical professionals and researchers better understand how AI systems work and how they arrive at certain decisions. It has made AI more transparent, accountable, and reliable, making it easier for healthcare providers to trust AI systems and make decisions based on their recommendations. Most of the XAI algorithms are post-hoc model-agnostic, indicating that they can be used with any black box model [5].

In this paper, we present the progress of integrating explainability into the deep learning models for WBC classification. Five deep learning models namely DenseNet121 [6], Xception [7], ResNet50 [8], VGG16 [9], and MobileNetV2 [10] pretrained over ImageNet dataset [11] have been taken. The pre-trained models have learned a set of available features useful for various related tasks. Transfer learning has been applied to fine-tune the generalized pre-trained models for the WBC classification task by training them on smaller specific datasets, namely Raabin-WBC dataset [12] and BCCD dataset [13]. The fine-tunned models classify WBCs into four categories, and to explain the model's prediction, LIME [4] is employed. Thus, the significant contributions of the research paper are as follows:

1. An automated system has been developed to analyze and classify WBCs into various types.
2. Five different pre-trained models have been fine-tuned on two benchmark datasets for WBC classification.
3. Explainable AI is integrated with the help of LIME model to determine why specific model's result is better than others.

The rest of the paper is structured as follows. Section 2 presents a brief review of state-of-the-art methods for WBC classification. Both types of deep learning methods for WBC classification that either employ explainability or do not are included in the review. Section 3 presents the material and methods used in the study. Experimental results are presented in Section 4, and discussions are made. Finally, conclusions are drawn in Section 5.

2. Related work

This section presents a succinct review of recent deep learning methods proposed for WBC classification, with a focus on explainable AI models used in medical images. Numerous machine-learning algorithms have been proposed for WBCs classification, such as random forest [14], Naïve Bayes [15], logistic regression [1], and k-Nearest Neighbor [16]. Various deep learning algorithms that use convolutional neural networks (CNNs), such as DenseNet121, Xception, VGG16, MobileNetV2, and ResNet50 have also been proposed for WBC classification. The CNN models are popular and produce excellent outcomes, as demonstrated by Girdhar et al. [17] who developed a CNN model for the categorization of WBCs, except for basophils due to lack of images in that category. The model obtained an overall accuracy, precision, recall, specificity, and F1-scores of 98.55%, 97.16%, 97.11%, 99.03%, and 97.11%, respectively, on the BCCD dataset [13]. Similarly, Wang et al. [18] classified WBCs using a 3D CNN model and achieved an accuracy of 97.72%. An alternative strategy used by some researchers involves a separate segmentation and classification process. For instance, Dong et al. [19] implemented a strategy to segment and classify WBCs. They created a technique with the objective of segmenting WBCs, where the nucleus and cell are segmented separately. The method used k-means and Otsu-based thresholding for segmentation and is evaluated using metrics such as sensitivity, Dice similarity coefficient (DSC), and positive predictive value (PPV).

Table 1
Machine-learning methods WBC classification.

Author [Ref.]	Dataset	Segmentation methods	Classification methods	A	P	R	S	F
Girdhar et al. [17]	BCCD	CNN model		98.55	97.16	97.11	99.03	97.11
Dong et al. [19]	Custom dataset with 500 images taken from various datasets	*k*-means and Otsu thresholding (Nucleus Segmentation- DSC (95.98), PPV (96.29) and Sensitivity (95.85) Cell Segmentation- DSC (97.58), PPV (97.77), Sensitivity (97.40))	PSO-SVM	99.76	99.40	99.40	99.85	–
Banik et al. [20]	Cellavision, ALLIDB2, JTSC, BCCD	*k*-means (Accuracy for Cellavision – 98.86, ALLIDB2 – 98.61, JTSC – 97.57 and BCCD - 99.42)	5-layer CNN model	97.95	94.75	96.0	98.61	96.0
Bozkurt et al. [22]	BCCD	DenseNet121		94	94.22	93.74	–	93.81
Mohamed et al. [1]	BCCD	Pre-trained model and FCN	Fine-tunned DenseNet169	94.0	94.22	93.74	–	93.81

A- Accuracy, P-Precision, R-Recall, S-Specificity, F-F1-Score. All values in percent.

The nucleus segmentation method achieved an average score of 95.984%, 96.296%, and 95.856% for the respective metrics, while the cell segmentation method achieved 97.58%, 97.774%, and 97.404%, respectively. After segmenting the WBCs, the PSO-SVM classifier is used to classify them segmented cells. It achieved an accuracy, recall, specificity, and precision of 99.76%, 99.40%, 99.85%, and 99.40%, respectively, and an F-score of 99.40%.

Banik et al. [20] proposed an approach that makes use of a range of datasets, including Cellavision,[1] ALLIDB2,[2] JTSC [21], and BCCD [13] for WBC segmentation and classification. To perform segmentation, RGB images are first converted into HSI and CIELab color spaces. The converted color spaces are perceptually uniform and help in the segmentation process. The *k*-means technique is then used for segmentation, giving an overall accuracy of 99.42%, 98.61%, 97.57% and 98.86% for the BCCD, ALLIDB2, JTSC, and Cellavision datasets, respectively. For classification, a CNN model with five convolutional layers, two fully connected layers, three overlapping average-pooling layers, a hidden layer, and an output layer is proposed.

The model achieved an accuracy, precision, recall, specificity, F1-score and Cohen's kappa index of 97.95%, 94.75%, 96%, 98.6175%, 96%, and 94%, respectively. Some researchers have also used transfer learning to adapt pre-trained models for WBC classification task. For instance, Bozkurt et al. [22] used pre-trained DenseNet121 model and adapted it for WBC classification task by fine-tuning it on the BCCD dataset. The experiment is repeated with pre-trained Xception, VGG19, and EfficientNetB1 models as well. The pre-trained DenseNet121 model attained an accuracy of 94% on the BCCD dataset, surpassing the performance of Xception, VGG19, and EfficientNetB1 by a considerable margin. Similarly, Mohamed et al. [1] used pre-trained models and a fully connected network (FCN) to segment WBCs, which are further classified using logistic regression, decision tree, random forest, Naïve Bayes, *k*-nearest neighbors and linear discriminant analysis classifiers. These classifiers outperformed the baseline classifier in terms of performance. The consolidation of recently presented machine learning-based methods used for WBC classification is presented in Table 1.

XAI is becoming increasingly popular for adding interpretability and trustworthiness to deep learning methods. It has been applied to various medical image analysis tasks such as for Alzheimer's analysis [23], glaucoma's treatment planning [24], and for other domain tasks such as bankruptcy prediction [25] and solar photovoltaic (PV) forecasting [26]. In one study [23], support vector classifier (SVC), Xboost and *k*-NN classifiers are used to analyze gene expression data, and SpinalNet and CNN are used to classify Alzheimer's disease from MRI images. LIME is then used to analyze gene expression, identifying the features with a greater and lesser impact on the output, to make the model easier to understand. In another study [24], XAI is applied to the analysis of glaucoma prediction to understand the risk factors for treatment planning. Hazard factors such as blood glucose level, gender, fundus pixel intensity, and age are explained using Sub-modular Pick Local Interpretable Model-agnostic Explanation (SP-LIME) in the output of Adaptive Neuro-Fuzzy Interface System (ANFIS). Girdhar et al. [17] applied XAI for leukemia diagnosis. Pre-trained models such as ResNet101V2, VGG19 and InceptionV3 are used to diagnose leukemia with accuracy rates of 98.61%, 96.14%, 99.14%, and 98.38%, respectively. Later, LIME is used to add explainability to the inferred results. Park et al. [25] used tree-based models like XGB and lightGBM for bankruptcy prediction. LIME and SHAP are used to help interpret and better understand the results. Kuzlu et al. [26] have used XAI in the models for solar PV forecasting. ELI5, LIME, and SHAP are used to explain specific predictions. Some

[1] https://www.cellavision.com
[2] https://homes.di.unimi.it/scotti/all/

Table 2
Studies from different domains using XAI.

Author [Ref.]	Domain	Task	AI model	XAI model
Girdhar et al. [17]	Medical	Leukemia Prediction	Pre-trained models such as ResNet101V2, VGG19, InceptionResNetV2 and InceptionV3	LIME
Kamal et al. [23]	Medical	Alzheimer's Analysis	KNN, SVC and Xboost classifiers	LIME
Kamal et al. [24]	Medical	Glaucoma treatment planning	Adaptive Neuro-Fuzzy Interface System	SP-LIME
Parl et al. [25]	Finance	Bankruptcy prediction	Tree-based models such as XGB and lightGBM	LIME and SHAP
Kuzlu et al. [26]	Energy	Solar PV forecasting	Random forest regression	EL15, LIME and SHAP

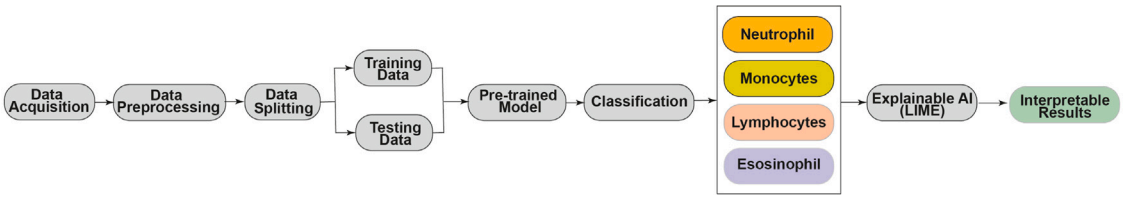

Fig. 2. System pipeline.

studies using XAI are listed in Table 2. In a nutshell, XAI has been beneficial for a wide range of problems in different domains, from healthcare and finance to autonomous systems and social sciences. By providing transparency and interpretability to machine learning models, XAI can help improve their accuracy, reliability, and trustworthiness. Inspired by the benefits of XAI, we next present the integration of XAI with the deep learning models for WBC classification.

3. Materials and methodology

This section presents the datasets used in the study, and the different deep learning and XAI models used.

3.1. Datasets

In this study, the following two datasets have been used for training, testing and validation of different deep learning and XAI models.

1. **BCCD dataset** [13]: The BCCD dataset consists of 410 JPEG and XML format images of monocytes, eosinophils, basophils, lymphocytes, and neutrophils. Each image has dimensions of 320 × 240 pixels. These original images are pre-processed using different techniques, resulting in approximately 12,500 enhanced images, with approximately 3,000 images for each type of cell.
2. **Raabin-WBC dataset** [12]: The Raabin-WBC dataset is extensive, comprising of over 40,000 images. However, for this study, only a subset of double-labeled cropped cell images, consisting of approximately 14,213 images, are used. These images are in JPG format and have a size of 575 × 575 pixels, representing monocytes, eosinophils, lymphocytes, and neutrophils. Basophils were not included in this study as their images are limited in number as compared to other types of cells.

3.2. Methods

In this study, we have used five pre-trained deep learning models fine-tuned over above mentioned datasets for WBC classification. Later, XAI is integrated for a better understanding of the predictions made by the deep learning models. The following is the pipeline of the processes adopted: (i) information gathering from BCCD and Raabin-WBC datasets, (ii) pre-processing dataset to make it apt for the application of deep learning models, (iii) fine-tuning pre-trained models, namely DenseNet121, ResNet50, Xception, MobileNetV2, and VGG16 on the domain-specific datasets, and (iv) integration of XAI method called LIME for the better understanding of the predictions made by the fine-tunned models. The system pipeline is depicted in Fig. 2.

The modules in the system pipeline are explained as follows.

1. **Data pre-processing:** All the images are resized to 224 × 224 and processed in RGB format. The presence of salt-and-pepper noise in the images is removed using a median filter. After pre-processing, dataset is split into train and test sets to cross-check the performance of models. The number of images of the BCCD dataset for the train and test set are 10,028 and 2,472, respectively. Similarly, the number of images in train and test sets for Raabin-WBC dataset is 9,963 and 4,250, respectively. It is to mention that 30% of the total training images are utilized for validating the performance of models after every epoch.

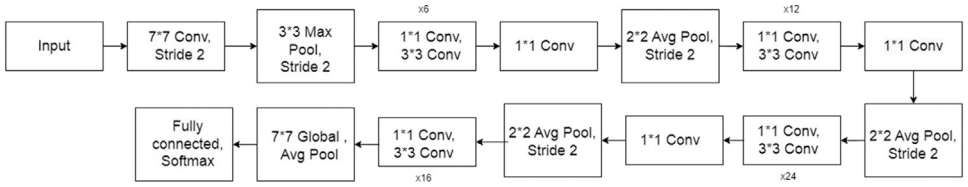

Fig. 3. DenseNet121 Architecture.

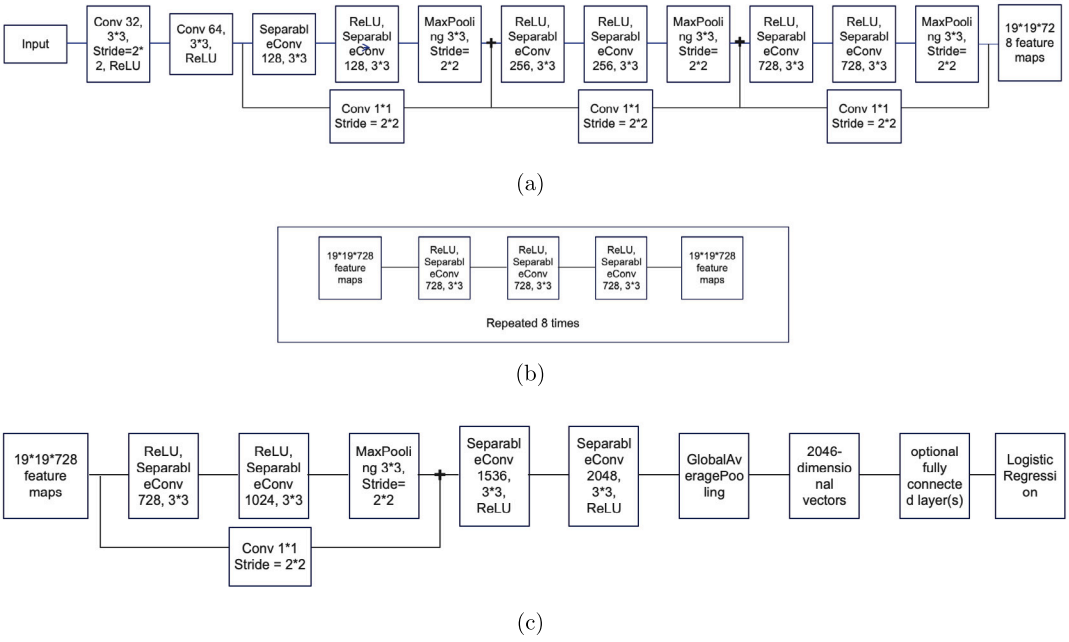

Fig. 4. Xception Architecture: (a) Entry flow, (b) Middle flow, (c) Exit flow.

2. **Pre-trained deep learning models:** A convolutional neural network (CNN) is a type of deep learning model structured in a way that allows it to automatically learn and extract relevant features from images through a process called *convolution*. This process involves sliding a set of small filters (also known as *kernels*) over the input image, performing a dot product between the filter and the underlying pixel values, and then applying a non-linear activation function to the result. The output of this operation is a new set of feature maps that highlight different aspects of the input image, such as edges, textures, and shapes. The output of the final pooling layer is flattened and fed into one or more fully connected layers, which perform classification or regression tasks based on the extracted features. The following are layers used in CNN models.

   (i) Input Layer: This layer takes the input image/ images as a block and forwards it to the next layer after processing.
   (ii) Convolutional Layer: The purpose of a convolutional layer is to perform the operation of convolution on the input image or feature maps. Convolution involves taking a small filter (also known as a kernel) and sliding it over the input image or feature maps to extract local features. The filters in a convolutional layer are typically learned during the training process, allowing the CNN to automatically learn and extract relevant features from the input data.
   (iii) Pooling Layer: The pooling layer reduces the spatial dimensions of the feature maps and helps to improve the computational efficiency of the model.
   (iv) Dropout Layer and BatchNormalization Layer: The purpose of a dropout layer is to randomly drop a fraction of the neurons in the layer during training. This helps to prevent overfitting by reducing the reliance of the model on specific neurons, which can become too specialized to the training data. By randomly dropping out neurons, dropout encourages the model to learn more robust and generalizable features, and helps to prevent the model from memorizing the training data.

   Batch normalization, on the other hand, is a technique used to normalize the activations of the previous layer before feeding them into the next layer. This can help to improve the stability and efficiency of the model by reducing the internal covariate shift, which is the change in the distribution of the activations of a layer due to the changing parameters of the previous layer during training. Batch normalization can also help to speed up the training process by reducing the number of epochs needed to reach convergence.

(v) Flatten Layer: The purpose of a flatten layer in a neural network is to reshape the output of the previous layer into a one-dimensional vector that can be fed into a fully connected layer for classification or regression.

(vi) Fully Connected Layer: The purpose of a fully connected layer in a neural network is to perform a mathematical transformation on the input data to generate an output that is suitable for the task at hand, such as classification or regression.

(vii) Output Layer: The purpose of the output layer is to produce the final output of the model, which could be a prediction or a probability distribution over the possible classes.

In this study, the following CNN models pre-trained over a large natural image dataset, known as *ImageNet*, are used.

(i) **DenseNet121**: DenseNet [6] is a deep learning architecture for image classification that is introduced in 2017 by Huang et al. DenseNet is a variant of the convolutional neural network (CNN) architecture that uses skip connections between layers to improve the flow of information and the gradient signal through the network. Some of the key characteristics of DenseNet are dense connectivity, skip connections, bottleneck layers, growth rate that determines the number of feature maps that are added to each layer, and transition layers. As compared to the traditional CNNs, DenseNet has been developed as a deeper network with a large number of network layers. There are several different versions of DenseNet, including DenseNet121, DenseNet169, DenseNet201, and DenseNet264 [6]. The DenseNet121 model is used in this study to perform WBC classification. As shown in Fig. 3, the architecture of DenseNet121 model consists of 121 convolutional layers (one 7×7 convolution, 58 3×3 convolutions, 61 1×1 convolutions, and one fully connected layer) and 4 AvgPool (average pooling layer) make up this convolutional model.

(ii) **Xception**: Xception (Extreme Inception) [7] is a deep learning architecture for image classification that is introduced in 2016 by François Chollet. Xception is a variant of the Inception architecture that uses depthwise separable convolutions to reduce the number of parameters in the network and improve its efficiency. The key characteristics of Xception includes depthwise separable convolutions, extreme version of Inception architecture, efficient feature extraction and improved performance. As shown in Fig. 4, the 71-layer Xception is divided into three parts: entry flow, middle flow and exit flow. The entry flow starts by taking an input of $299 \times 299 \times 3$ images and then use convolutional layers, ReLu and maxpooling multiple times, giving an output of $19 \times 19 \times 728$ feature maps. The middle flow starts by taking the output of the entry flow as an input. Here, ReLu and SeparableConv are used 3 times, and this process is repeated 8 times to generate $19 \times 19 \times 728$ feature maps. The exit flow also takes an output of the middle flow as an input. ReLu, SeparableConv, and MaxPooling are used multiple times with the global average pooling layer. Lastly, an optional fully connected layer with logistic regression is applied on a 2048-dimensional vector to generate the output.

(iii) **ResNet50**: ResNet (Residual Network) [8] is a deep learning architecture for image classification that is introduced in 2015 by Kaiming He et al. ResNet is based on the idea of residual learning, which involves adding shortcut connections between layers in a feedforward neural network to create residual blocks. The use of residual blocks help to mitigate the vanishing gradient problem and improve the performance of the network. The key characteristics of ResNet includes residual blocks, deep architecture, pre-activation, and improved performance. One of the various variations of the ResNet model is ResNet50. It has 48 convolutional layers, one maxpool and one averagepool layer each, making it a 50-layer deep architecture.

This network starts with an input size of $224 \times 224 \times 3$ images, then performs the convolution with max pooling of 7×7 and a kernel size of 3×3. As shown in Fig. 5, the residual blocks are grouped into 4 stages, with the number of filters in each stage increasing from 64 to 2048. The output of the final residual block is passed through a global average pooling layer, which averages the feature maps across the spatial dimensions of the tensor. The output of the global average pooling layer is passed through a fully connected layer with 1000 units, followed by a softmax activation function. This produces a probability distribution over the 1000 classes in the ImageNet dataset.

(iv) **MobileNetV2**: Google introduced MobileNet and its MobileNetV1 and MobileNetV2 variants. In this study, MobileNetV2 is used, which has 53 convolution layers and one AvgPool layer. The two primary components of this model are the inverted residual block and the residual bottleneck block [10]. Unlike MobileNetV1, which had only one block, MobileNetV2 consists of two blocks, as shown in Fig. 6. The first is a residual block (stride = 1), and the other is for downsizing (stride = 2). Both the blocks consist of 3 layers starting from 1×1 convolution with ReLu6, followed by depthwise convolution. Lastly 1×1 convolution is repeated, but without non-linearity.

(v) **VGG16**: The VGG network is a deep convolutional neural network architecture proposed by Karen Simonyan and Andrew Zisserman in 2014. The key characteristic of VGG network is its use of small 3x3 convolutional filters with a fixed stride of 1 pixel, which allows for a deeper network architecture while maintaining a relatively small receptive field. The VGG network has two main versions: VGG16 and VGG19, which differ in the number of layers. VGG16 contains 16 layers, while VGG19 contains 19 layers. In this study, VGG16 network has been used. The architecture of VGG16 is shown in Fig. 7.

3. **Transfer Learning**: Deep learning requires significant computing power and a large dataset to train network from scratch. In certain cases where availability of data is limited, a process known as transfer learning is utilized. It takes into consideration the weights of pre-trained models, previously trained on huge datasets (ImageNet weights in our case [11]). Idea behind using such weights is that the patterns learnt by pre-trained models can be reused or refined to improve performance of new model, even if the dataset or task differs from the original dataset. Thus, transfer learning can reduce the computational

Algorithm 1 SLIC algorithm for image segmentation

(a) Divide I into K regular grid cells of size $S \times S$. Place a cluster center C_i at the center of each grid cell, where $i = 1, 2, \ldots, K$.
(b) For each pixel (x, y) in I, find the closest cluster center C_i in a window of $2S \times 2S$ pixels around (x, y). Compute the distance $D(x, y, i)$ between the pixel (x, y) and the cluster center C_i, and assign the pixel (x, y) to the cluster with the smallest distance $D(x, y, i)$.
(c) Update each cluster center C_i by computing the mean position (u_i, v_i) and color value C_i^{mean} of all pixels assigned to it, and moving the cluster center C_i to the new position (u_i, v_i) and color value C_i^{mean}.
(d) Refine the segmentation by repeating the following steps until no more changes are made:
    i. For each cluster center C_i, search for the pixel with the smallest distance $D(x, y, i)$ in a window of size 3×3 centered at (u_i, v_i).
    ii. Reassign this pixel to the cluster with the smallest distance $D(x, y, i)$.
(e) Assign each pixel (x, y) to the cluster with the smallest distance $D(x, y, i)$ among all cluster centers C_i to obtain the segmentation map S.

resources required to train a model as pre-trained models have already learnt significant number of features. Such models consists of series of layers that extract low and high-level features to perform a specific task such as image segmentation of classification. Last layer of pre-trained models is usually a classification layer which outputs the number of features to the number of output classes.
In this study, last layer of the pre-trained models is replaced by a new fully connected layer which consists of 4 output channels, each corresponding to different type of WBC. The predictions made by fine-tuned model are made interpretable and explainable by applying XAI described as follows.

4. **Explainable AI (XAI)**: XAI is a field of study that strives to create AI systems easily interpretable by humans. The techniques and methods used in XAI are aimed at enhancing the transparency and interpretability of AI models and decision-making processes for end-users, including policymakers, domain experts, and other stakeholders. There are two primary categories of XAI methods: *model-specific* and *model-agnostic* methods. *Model-specific methods* are focused on explaining how a particular AI model reaches its decisions by analyzing its internal workings and parameters. On the other hand, *model-agnostic methods* aim to explain the decision-making process of any AI model, regardless of its internal structure, by evaluating the model's inputs and outputs. In this study, we have used Local Interpretable Model-Agnostic Explanations (LIME), which is a model-agnostic technique for explaining the predictions of machine learning models, particularly for classification tasks. LIME provides local explanations for individual predictions, by highlighting the features that contribute the most to the predicted outcome. To generate local explanations for image classification, LIME use any of the segmentation algorithm described below to identify the most important regions of the image for the classification decision.

    (i) **Felzenszwalb (FHA):** Felzenszwalb et al. [27] proposed this method for segmenting an image using a graph-based algorithm. It uses tree-based clustering to generate an over-segmentation of an RGB image.
    (ii) **Quickshift:** Vedaldi et al. [28] proposed a method called QuickShift for Mode Seeking. One of the major applications of Quickshift is the segmentation of an image, it segments using cluster pixels. Controlling the ultimate number of superpixels is not possible with any specified setting.
    (iii) **Simple Linear Iterative Clustering (SLIC):** Achanta et al. [29] proposed this superpixel algorithm called SLIC where the segmentation of an image is done using k-means clustering in color space.

    In this study, the SLIC segmentation algorithm is used to clarify the model's functionality. This choice of algorithm is made since it is more comprehensible compared to other available algorithms. Given an image I of size $N \times M$ with pixels (x, y) and color values $C(x, y)$, and a desired number of superpixels K, the SLIC Algorithm 1 proceeds as follows:
    For the classification of WBCs into monocytes, eosinophils, lymphocytes, and neutrophils, pre-trained deep learning models namely DenseNet121, Xception, ResNet50, VGG16, and MobileNetV2 are applied and then the LIME explainer is used to explain the predictions of each deep learning model using Algorithm 2.

The LIME algorithm, which uses the SLIC (Simple Linear Iterative Clustering) superpixel segmentation technique, follows a step-by-step process. Initially, the input image is divided into K superpixels using the SLIC algorithm, which clusters together pixels with similar attributes into compact regions. This step helps to decrease the input image's dimensionality and enhance the algorithm's efficiency. Afterward, N perturbed images are generated by randomly switching on or off the superpixels, and each perturbed image is resized to a fixed width and height. The perturbed images are then processed through the black box model, and the output probabilities are recorded. To determine the most informative perturbed images, a similarity function calculates the distance between each perturbed image and the original input image. The top M perturbed images with the highest similarity scores are selected for further analysis. An interpretable model like linear regression or decision trees is then used to fit the selected perturbed images using

Algorithm 2 LIME algorithm using SLIC for image segmentation with a pre-trained network

(a) Let $f_\theta(x)$ be the pre-trained image classifier to be explained, and let x be an input image.

(b) Generate M random perturbations of the input image x to obtain a set of modified images $\{x'_i\}_{i=1}^{M}$. For each modified image x'_i, apply SLIC algorithm (Algorithm 1) with K clusters to obtain a segmentation map S_i.

(c) For each modified image x'_i, compute the feature vector $\phi(x'_i)$ as follows:

- Compute the color value C_j^{mean} and the location (u_j, v_j) of each cluster j in the segmentation map S_i.
- Let $c_i = f_\theta(x'_i)$ be the predicted class probability for modified image x'_i.
- Compute the weight w_j for each cluster j as follows:

$$w_j = \sum_{i=1}^{M} c_i \cdot d(\phi(x'_i), \phi(x))$$

where $d(\cdot,\cdot)$ is a distance metric between feature vectors.

- Normalize the weights $\{w_j\}_{j=1}^{K}$ to sum to one.
- Compute the weighted coverage map $L(x'_i)$ for each modified image x'_i as follows:

$$L(x'_i) = \sum_{j=1}^{K} w_j \cdot \mathbb{1}(S_i = j)$$

where $\mathbb{1}(\cdot)$ is the indicator function.

(d) Fit an interpretable model to the feature vectors $\{\phi(x'_i)\}_{i=1}^{M}$ and the corresponding class probabilities $\{c_i\}_{i=1}^{M}$ to obtain a local explanation model $g(x')$.

(e) Use the local explanation model $g(x')$ to compute the feature importance scores for the original input image x.

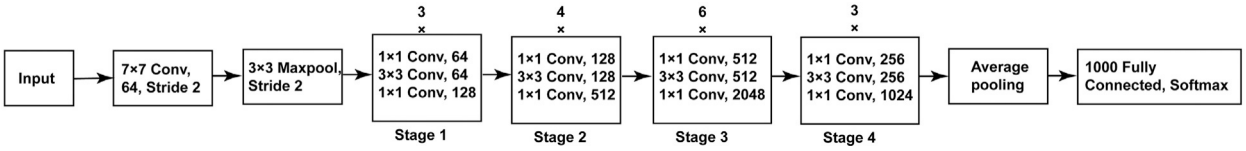

Fig. 5. ResNet50 Architecture.

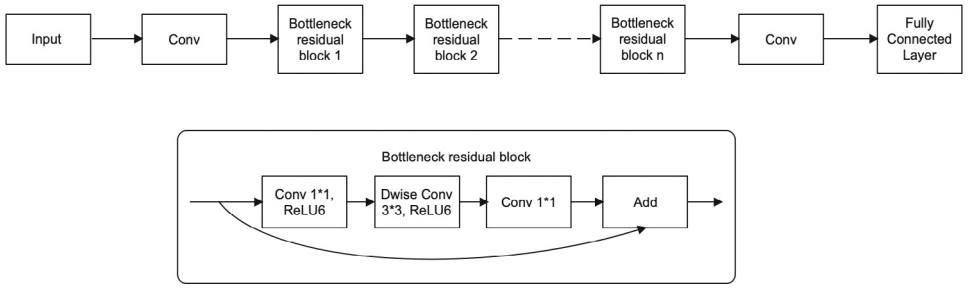

Fig. 6. MobileNetV2 Architecture.

their corresponding output probabilities as labels. The coefficients of the interpretable model are utilized to calculate the feature importance weights, which show the contribution of each superpixel to the model's output for the input image. Lastly, the local feature importance weights are returned as the algorithm's output. These weights enable an understanding of how the model makes predictions for a specific input image and uncover any potential biases or inconsistencies in the model's behavior. The algorithm employs an indicator function to identify the activated superpixels for generating the perturbed images. The indicator function is implemented as a binary mask that is applied to the image, where each element corresponds to a superpixel and takes on a value of 0 or 1 to indicate whether the superpixel should be turned off or on, respectively.

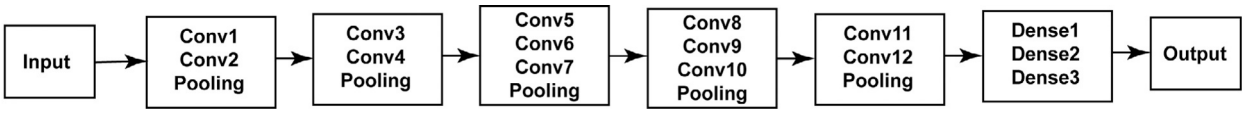

Fig. 7. VGG16 Architecture.

3.3. Evaluation metrics

Accuracy, recall, precision, specificity and F1-score are the evaluation metrics used in this study. In order to compare the models, a confusion matrix is used to get the metric values. A confusion matrix consists of the following:

1. **True Positive (TP):** TP represents that the projected value is positive, and the predicted value is also true. E.g., while finding the type of WBCs, if the projected value is neutrophil and the predicted value is also neutrophil, TP is equal to 1.
2. **False Positive (FP):** FP represents that the projected value is positive, and the predicted value is false. E.g., the projected value is neutrophil, but the predicted value contradicts.
3. **True Negative (TN):** TN represents that the projected value is negative, and the predicted value is true. E.g., the projected value is not neutrophil, but the predicted value is also not neutrophil.
4. **False Negative (FN):** FN represents that the projected value is negative, and the predicted value is also false. E.g., the projected value is not neutrophil, but the predicted value is neutrophil.

Using the above values, the used evaluation metrics are calculated as follows:

1. **Accuracy:** It is the most well-liked and extensively applied metrics for segmentation and classification. As mentioned in Eq. (1), it is determined by dividing the number of right predictions by the sum of all valid predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

2. **Precision:** Precision, or positive predictive value, is ratio of total number of correct positive predictions to the total number of positive predictions. It is calculated as mentioned in Eq. (2), and range between 0 and 1.

$$\text{Precision} = \frac{TP}{FP + TP} \tag{2}$$

3. **Sensitivity:** As calculated in Eq. (3), sensitivity, also known as recall, is the ratio of the number of correct positive predictions to the total number of positives.

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{3}$$

4. **F1-score:** Recall and precision are used for calculating the F1-score. It is the harmonic mean of precision and recall and is expressed as a single value between 0 and 1, with 1 being the best possible score. As mentioned in Eq. (4), F1-score is calculated as:

$$\text{F1-score} = \frac{2 * PrecisionRecall}{Precision + Recall} \tag{4}$$

5. **Specificity:** Specificity represents the number of right negative predictions divided by the total number of negatives. It is calculated as in Eq. (5).

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{5}$$

4. Experimental results and discussion

The experiments are conducted on a workstation with Intel(R) Xeon(R) Gold 5218 CPU @ 2.30 GHz 2.29 GHz (2 processors), NVIDIA GeForce RTX 2080Ti Graphics with 64 GB primary memory. VSCode software with Python version 3.8.0 is used for implementation. The tools used for implementing the present work are VSCode, Python, Scikit-learn, Keras (with TensorFlow backend), Pandas, NumPy, OpenCV, and LIME. The pre-trained models are fine-tunned on BCCD dataset and Raabin-WBC dataset for 100 epochs with batch size of 8.

4.1. Quantitative performance evaluation

Five pre-trained deep learning models, namely DenseNet121, Xception, ResNet50, VGG16 and MobileNetV2 are evaluated to find the best model for WBCs' classification. The last fully-connected layer of the pre-trained models is replaced with a fully-connected layer with 4 output channels. The shallower layers are further fine-tuned to adjust weight of the network as per the

Table 3
Performance evaluation on BCCD dataset.

BCCD dataset	Type	Accuracy	Precision	Recall/ sensitivity	F1-score	Specificity
DenseNet121	E	0.9533	0.9095	0.9036	0.9100	0.9699
	L	1.0000	1.0000	1.0000	1.0000	1.0000
	M	0.9609	1.0000	0.8435	0.9200	1.0000
	N	0.9384	0.8248	0.9583	0.8900	0.9318
Average		**0.9587**	**0.9335**	**0.9284**	**0.9300**	**0.9754**
ResNet50	E	0.8745	0.9018	0.5601	0.6900	0.9796
	L	0.7571	0.7352	0.4032	0.8000	0.9951
	M	0.7957	0.5696	0.7387	0.6400	0.8146
	N	0.6501	0.4025	0.8141	0.5400	0.5952
Average		0.7693	0.6552	0.6290	0.6675	0.8461
Xception	E	0.6449	0.3574	0.5232	0.4200	0.6856
	L	0.8025	0.6608	0.4274	0.5200	0.9271
	M	0.7679	0.6806	0.1306	0.2200	0.9796
	N	0.6763	0.4142	0.7003	0.5200	0.6682
Average		0.7229	0.5283	0.4453	0.4200	0.8151
VGG16	E	0.6980	0.3638	0.2744	0.3100	0.8395
	L	0.7523	0.8333	0.0800	0.0200	0.9994
	M	0.6119	0.3874	0.9580	0.5500	0.4970
	N	0.7868	0.5983	0.4583	0.5200	0.8969
Average		0.7122	0.5457	0.4426	0.3500	0.8082
MobileNetV2	E	0.9296	0.8510	0.8715	0.8600	0.9490
	L	0.9947	1.0000	0.9790	0.9900	1.0000
	M	0.9296	0.9587	0.7500	0.8400	0.9892
	N	0.8669	0.6935	0.8413	0.7600	0.8754
Average		0.9302	0.8758	0.8604	0.8625	0.9534

E: Eosinophil, L: Lymphocyte, M: Monocyte, N: Neutrophil

dataset. For evaluating the performance of various models, five metrics were used: accuracy, precision, recall, specificity and F1-score. These metrics have been calculated for each sub-type of WBC for BCCD and Raabin-WBC dataset in Table 3 and Table 4 respectively. Confusion matrices with respect to all the experiments have been shown in Figs. 8 and 9 for BCCD and Raabin-WBC dataset respectively . In addition to this, convergence graphs for loss and accuracy of all the models as summed in Figs. 10 and 11 show that the models were trained appropriately. Table 3 demonstrates that among the models tested, DenseNet121 achieved the highest performance. Specifically, for the BCCD dataset, the average values for all metrics are superior to those of the other models tested. Average values for accuracy, precision, recall, F1-score and specificity are 0.95, 0.93, 0.92, 0.93, 0.97 respectively. While the MobileNetV2 model provided a relatively close competition to the DenseNet121 model, its accuracy results were approximately 2% lower.

VGG16 model gave the lowest values of all the metrics involved, specially F1-Score as 0.35. For Raabin-WBC dataset, average values of DenseNet121 for accuracy, precision, recall, F1-score and specificity are 0.98, 0.95, 0.91, 0.93 and 0.98 respectively.

On the other hand, numerical values of these metrics change to 0.97, 0.86, 0.94, 0.89 and 0.98 respectively. Values of performance metrics for other models can be referred from Table 4. It can be clearly seen that DenseNet121 has outpowered all the models in terms of accuracy, precision, F1-Score and specificity but MobileNetV2 has higher recall than Densenet121. It clearly mean that MobileNetV2 has better ability to identify true positives than DenseNet. After careful observation and analysis for both the datasets, it can be concluded that DenseNet121 should be the choice of classifier for WBC classification problem.

4.2. Qualitative performance evaluation (XAI)

In this study, LIME is used as reference to XAI to evaluate results of each model qualitatively. It starts by taking an image as its input, and four images from each dataset were chosen at random to show final output results for each model. The model's prediction, which might be any of the four cells, monocytes, eosinophils, lymphocytes, and neutrophils has been examined. SLIC segmentation algorithm is applied on the final resultant images. *Images*, *num_samples*, *top_label* and *classifier_fn* are the inputs for the LIME explainer. The term *Images* is referred to the random images that were chosen from the beginning, *top_label* is the no. of labels that LIME needs to show. In this case, that number is one because we only want to show one with the highest probability. *Classifier_fn* is the model used for the predictions such as DenseNet121, Xception, MobileNetV2, VGG16 and ResNet50. Later, artificial points i.e., the selected pixels are turned on and off to see how the output changes. In this case, LIME repeats this process 1000 times, and the result is an output where the model's dominant pixels from the original image are highlighted. In Fig. 12, 13, 14, and 15, LIME analysis is performed on 20 randomly selected images of each type of WBC from both datasets. Both correct and incorrect predictions were evaluated to determine which pixels played a significant role in determining the model's decisions. Analyzing all of the LIME-predicted images provides a better understanding of the deep learning model's behavior.

Table 4
Performance evaluation on raabin-WBC dataset.

Raabin-WBC dataset	Type	Accuracy	Precision	Recall/ sensitivity	F1-score	Specificity
DenseNet121	E	0.9917	0.9309	0.9627	0.9500	0.9941
	L	0.9851	0.9550	0.9854	0.9700	0.9850
	M	0.9821	0.9540	0.7094	0.8100	0.9980
	N	0.9849	0.9850	0.9909	0.9900	0.9748
Average		**0.9859**	**0.9562**	0.9121	**0.9300**	**0.9879**
ResNet50	E	0.9832	0.9035	0.8726	0.8900	0.9923
	L	0.9762	0.9864	0.9114	0.9500	0.9959
	M	0.9689	0.6554	0.9188	0.7700	0.9718
	N	0.9703	0.9777	0.9748	0.9800	0.9628
Average		0.9746	0.8806	0.9194	0.8975	0.9807
Xception	E	0.9477	0.6096	0.8633	0.7100	0.9546
	L	0.9590	0.9118	0.9206	0.9200	0.9713
	M	0.9769	0.8269	0.7350	0.7800	0.9910
	N	0.9284	0.9634	0.9206	0.9400	0.9415
Average		0.9530	0.8279	0.8598	0.8375	0.9646
VGG16	E	0.8294	0.3030	0.9627	0.4600	0.8184
	L	0.9550	0.9596	0.8510	0.9000	0.9888
	M	0.9611	0.8876	0.3376	0.4900	0.9975
	N	0.8783	0.9824	0.8203	0.8900	0.9754
Average		0.9059	0.7831	0.7429	0.6850	0.9450
MobileNetV2	E	0.9650	0.6331	0.9968	0.7700	0.9526
	L	0.9851	0.9840	0.9545	0.9700	0.9950
	M	0.9856	0.8446	0.9059	0.8700	0.9902
	N	0.9578	0.9983	0.9342	0.9700	0.9974
Average		0.9733	0.8651	**0.9478**	0.8950	0.9838

E: Eosinophil, L: Lymphocyte, M: Monocyte, N: Neutrophil

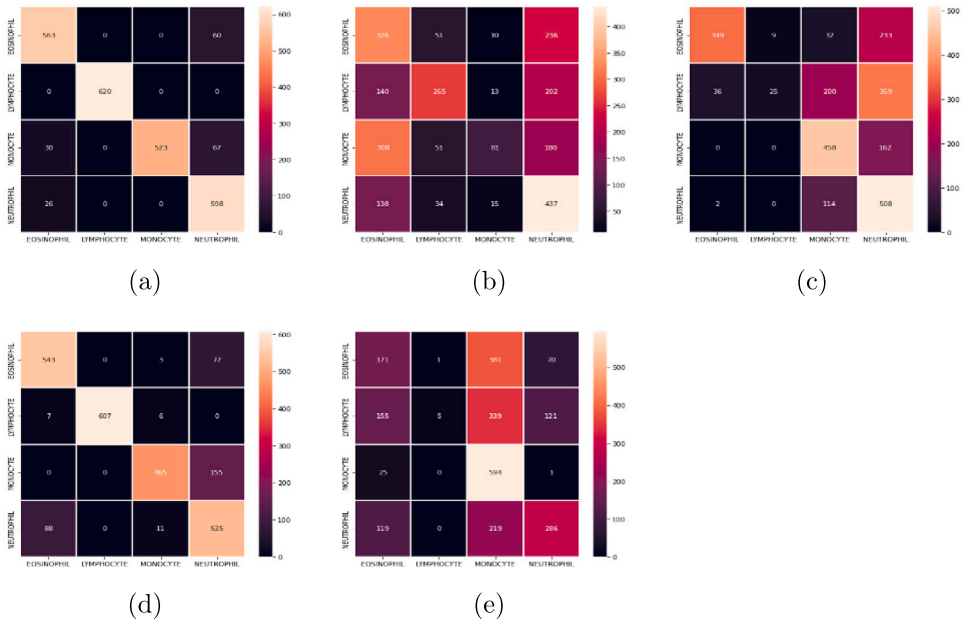

Fig. 8. Confusion Matrix of DenseNet121 (a), Xception (b), ResNet50 (c), MobileNetV2 (d) and VGG16 (e) for BCCD Dataset.

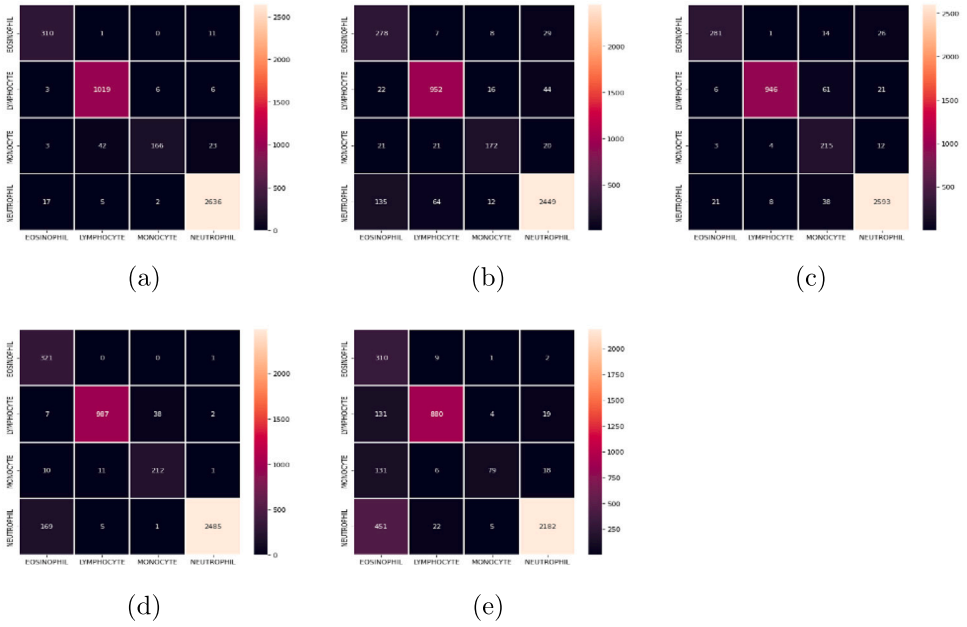

Fig. 9. Confusion Matrix of DenseNet121 (a), Xception (b), ResNet50 (c), MobileNetV2 (d) and VGG16 (e) for Raabin-WBC Dataset.

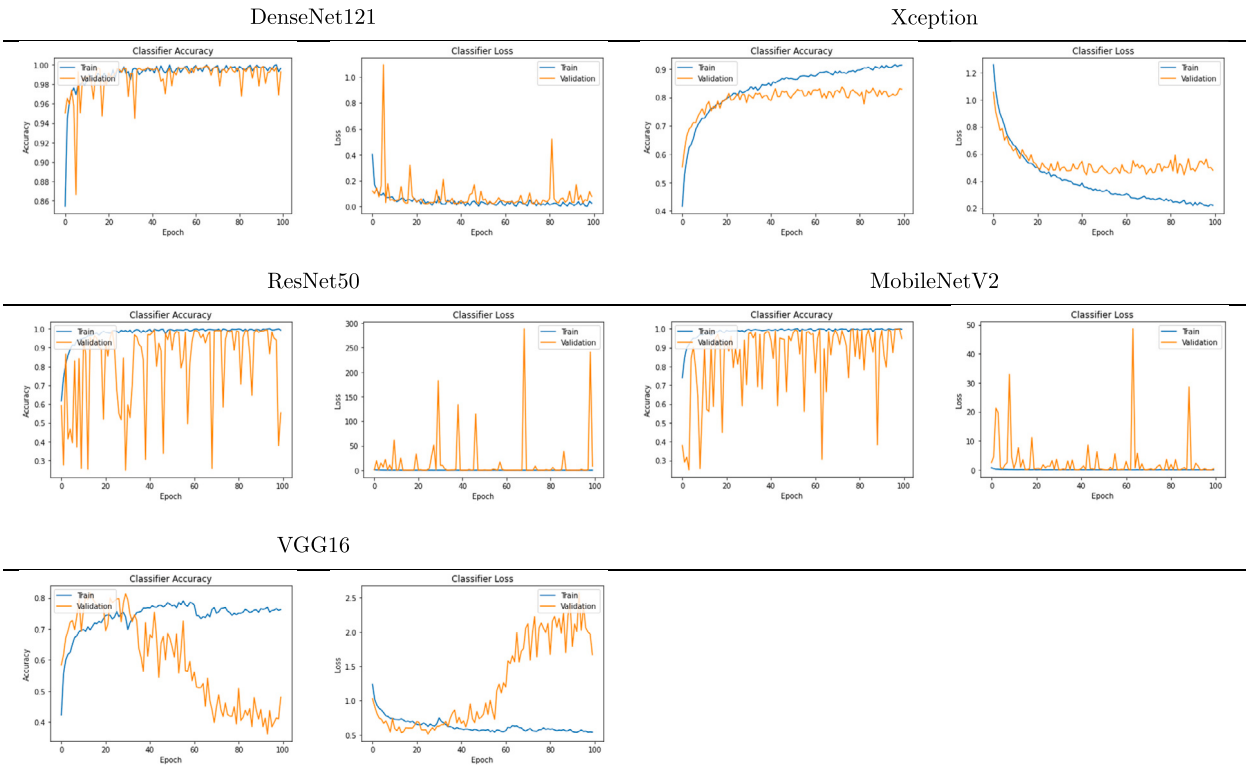

Fig. 10. Classifier accuracy and loss graphs of DenseNet121, Xception, ResNet50, MobileNetV2 and VGG16 for BCCD Dataset respectively.

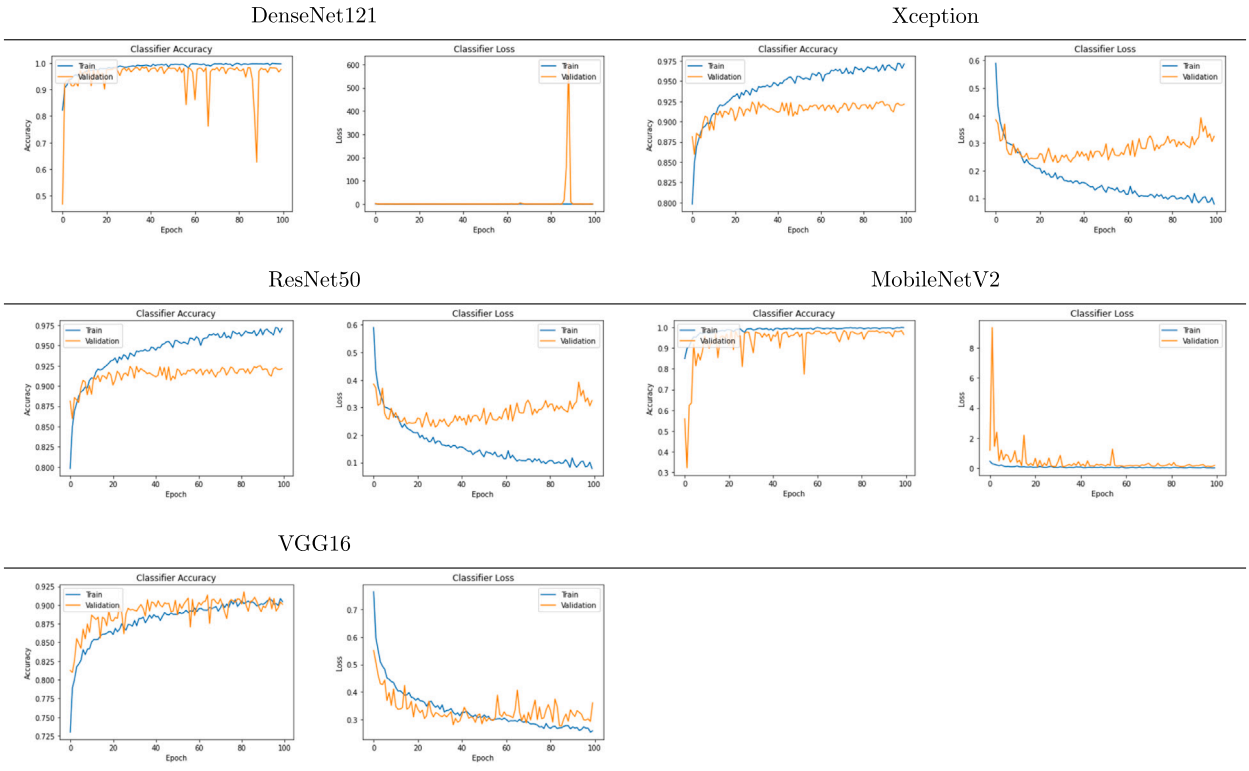

Fig. 11. Classifier accuracy and loss graphs of DenseNet121, Xception, ResNet50, MobileNetV2 and VGG16 for Rabbin-WBC Dataset.

By analyzing the images, we can identify the pixels that assist the model in making both correct and incorrect decisions. LIME can be used to validate the correctness of the model's predictions when it predicts the correct outcome.

In contrast, when the model makes an incorrect prediction, LIME can be used to improve the model. Hence, LIME can be used to predict the real winner for any deep learning problem. For instance, the Raabin-WBC dataset demonstrates that DenseNet121 has the highest average accuracy for classifying WBCs, but not all types are equally accurate. The MobileNetV2 model, on the other hand, provided better accuracy for detecting monocytes compared to DenseNet121. Therefore, by using the LIME explainer, we can identify the most suitable model for a particular type of WBC.

In-depth analysis of results of all models is given as below:

1. **DenseNet121:** The initial model used for classification is Densenet121, which is one of the top-performing models for both datasets. Due to the model's average accuracy of over 95% and 98% in the BCCD and Raabin-WBC datasets, respectively, it is critical to thoroughly analyze the images, as shown in Fig. 16. The right labels predicted for the BCCD dataset mainly focus on cell pixels and disregard other image pixels. However, for the Raabin-WBC dataset, the predictions also consider non-WBC pixels in some cases. Although the model's performance is impressive, there is still room for improvement, as demonstrated by one incorrect case where the correct label is monocyte, but the model predicted lymphocyte. Thus, examining the highlighted pixels may improve the model more quickly and easily before real-time testing.
2. **Xception:** The next WBC classification model employed is Xception. It showed an average accuracy of 72% and 95% for the BCCD and Raabin-WBC datasets, respectively. However, precision, recall, and F1-score for the BCCD dataset decreased considerably. Fig. 17 offers visual evidence of such poor scores. For both datasets, the model appears to concentrate on non-relevant areas. Despite the lymphocyte image in the Raabin-WBC dataset being accurately identified as a lymphocyte, some features were still improperly learned.
3. **VGG16** The VGG16 model, which is one of the five models tested, achieved an average accuracy of only 71% and 90% for the BCCD and the Raabin-WBC datasets respectively, making it the least accurate of all the models. This is a significant drop in performance compared to the other models. Additionally, the VGG16 model has an average precision of only 54% and 78% for the BCCD and the Raabin-WBC datasets respectively. This indicates that the model is not very good at correctly identifying true positives, and may even classify some false positives as true positives. The poor performance of the VGG16 model is

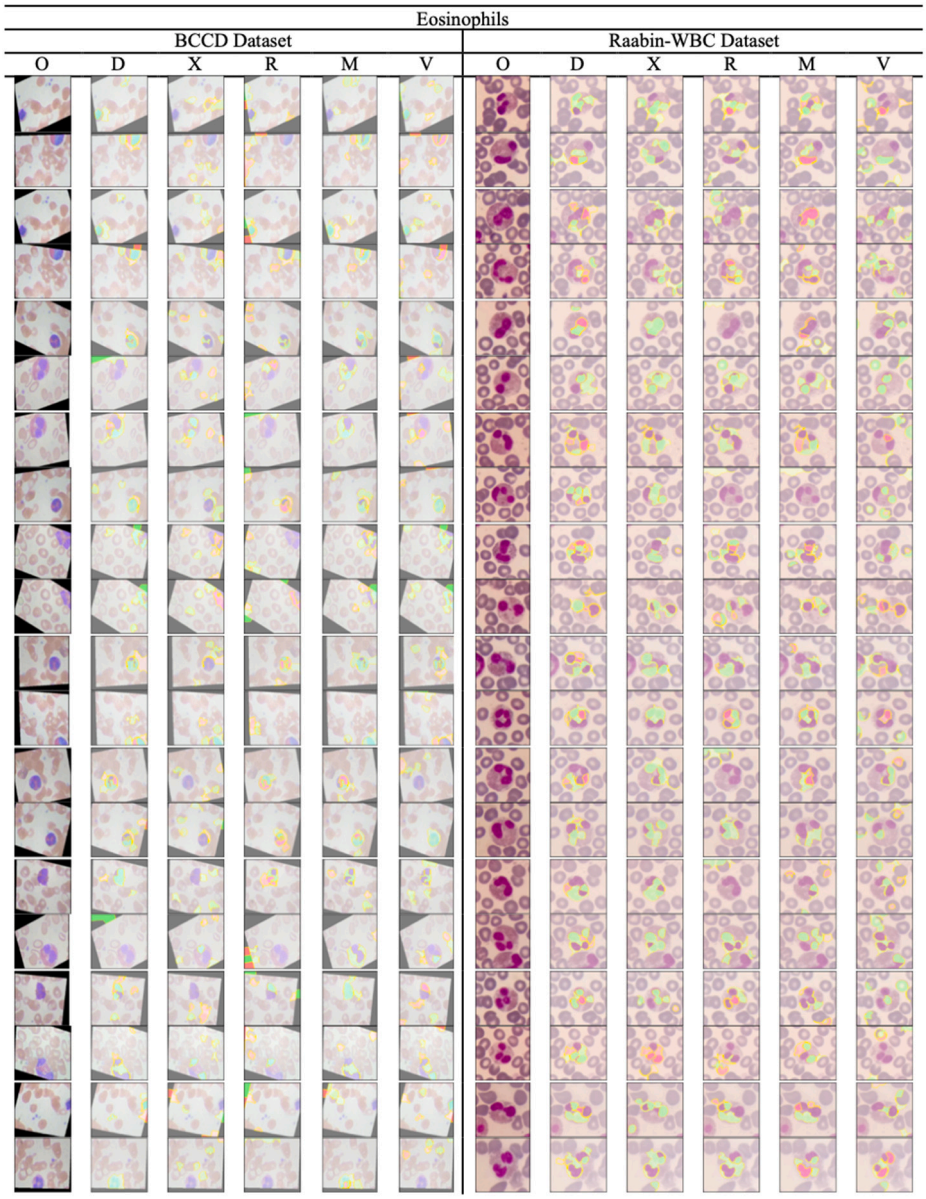

Fig. 12. LIME predicted Images for Eosinophils (Original Image (O), DenseNet121 (D), ResNet50(R), Xception (X), MobileNetV2 (M), VGG16 (V)).

further confirmed by the visual analysis shown in Fig. 18, which illustrates that the model heavily relies on incorrect pixels of the image and does not take into consideration the correct pixels. These results highlight the limitations of the VGG16 model for WBC classification, and suggest that other models may be more suitable for this task.

4. **MobileNetV2:** The MobileNetV2 model performed well in classifying WBCs for both BCCD and Raabin-WBC datasets, achieving an average accuracy of 93% and 97%, respectively. In comparison to other models, such as VGG16 and Xception, MobileNetV2 demonstrated superior performance. However, even with mostly accurate predictions, analysis using the LIME explainer, depicted in Fig. 19, revealed that some unnecessary pixels were still being selected along with the correct ones.
5. **ResNet50:** ResNet50 achieved an average accuracy of 77% and 97% for WBC classification in the BCCD and Raabin-WBC datasets, respectively. Despite accurate predictions, LIME analysis revealed that the model is choosing inappropriate pixels as shown in Fig. 20. Instead of focusing on the pixels that correspond to WBCs, the model is prioritizing other pixels. This information can be used to improve the model's performance more efficiently.

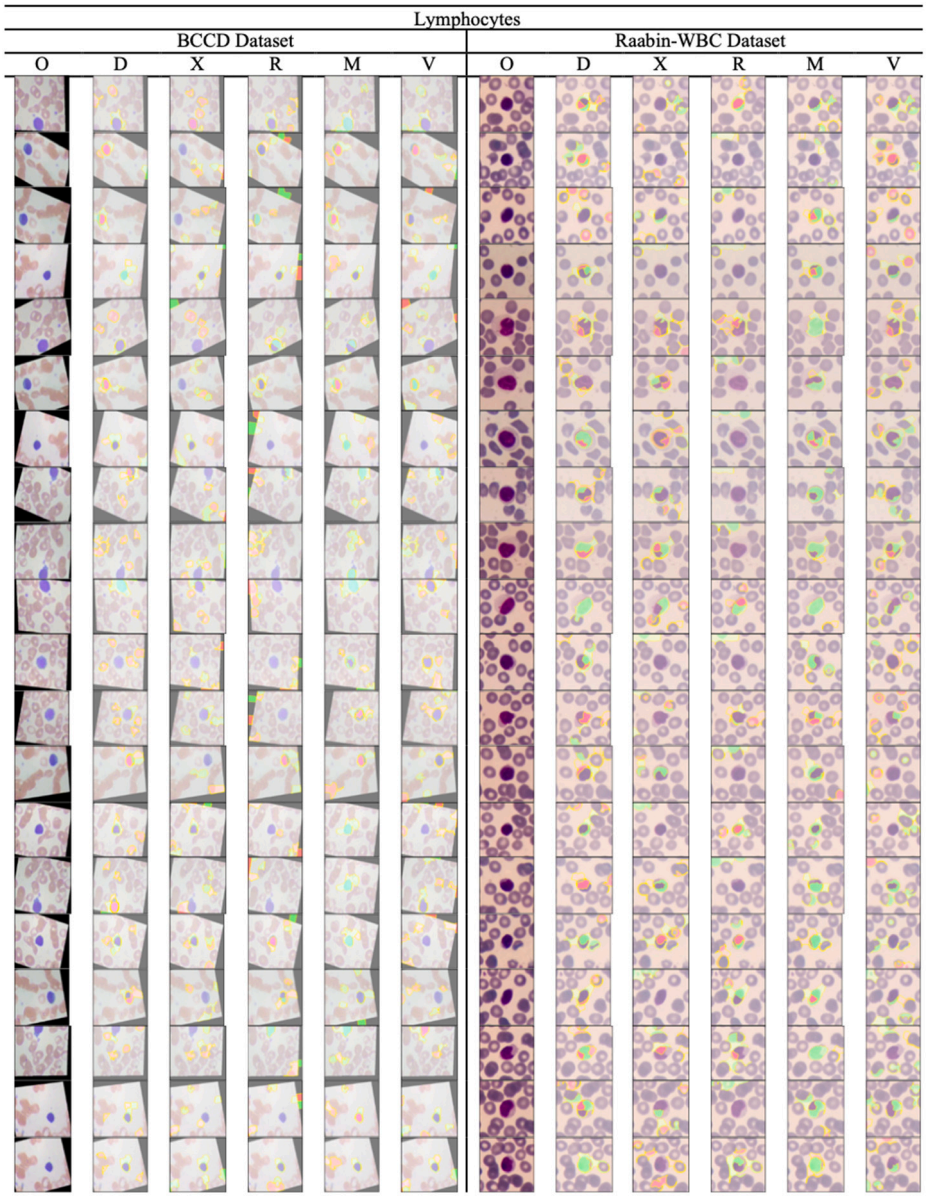

Fig. 13. LIME predicted Images for Lymphocytes (Original Image (O), DenseNet121 (D), ResNet50(R), Xception (X), MobileNetV2 (M), VGG16 (V)).

5. Conclusion

This paper presents a comparative study of five popular deep learning models to classify white blood cells (WBCs) and evaluates their performance using the LIME explainer. LIME is used to check the reliability and validity of the models by explaining them to the best of their ability. The robustness of the models is further validated using two publicly available datasets, namely the BCCD and Raabin-WBC datasets. The LIME explainer is used to identify the dominant image pixels that the model used to make its predictions, indicating that the model is making predictions based on its observation of the highlighted pixels. LIME is able to explain the reasons for both correct and incorrect predictions. Having a high accuracy does not guarantee that the model is correctly identifying the relevant features of the data it is processing. By examining the LIME-predicted images, it becomes possible to determine which pixels the model is relying upon to make its predictions and whether they are genuinely important or not. This analysis can help refine the model and ensure that it is not only accurate but also utilizing the appropriate features to make its predictions.

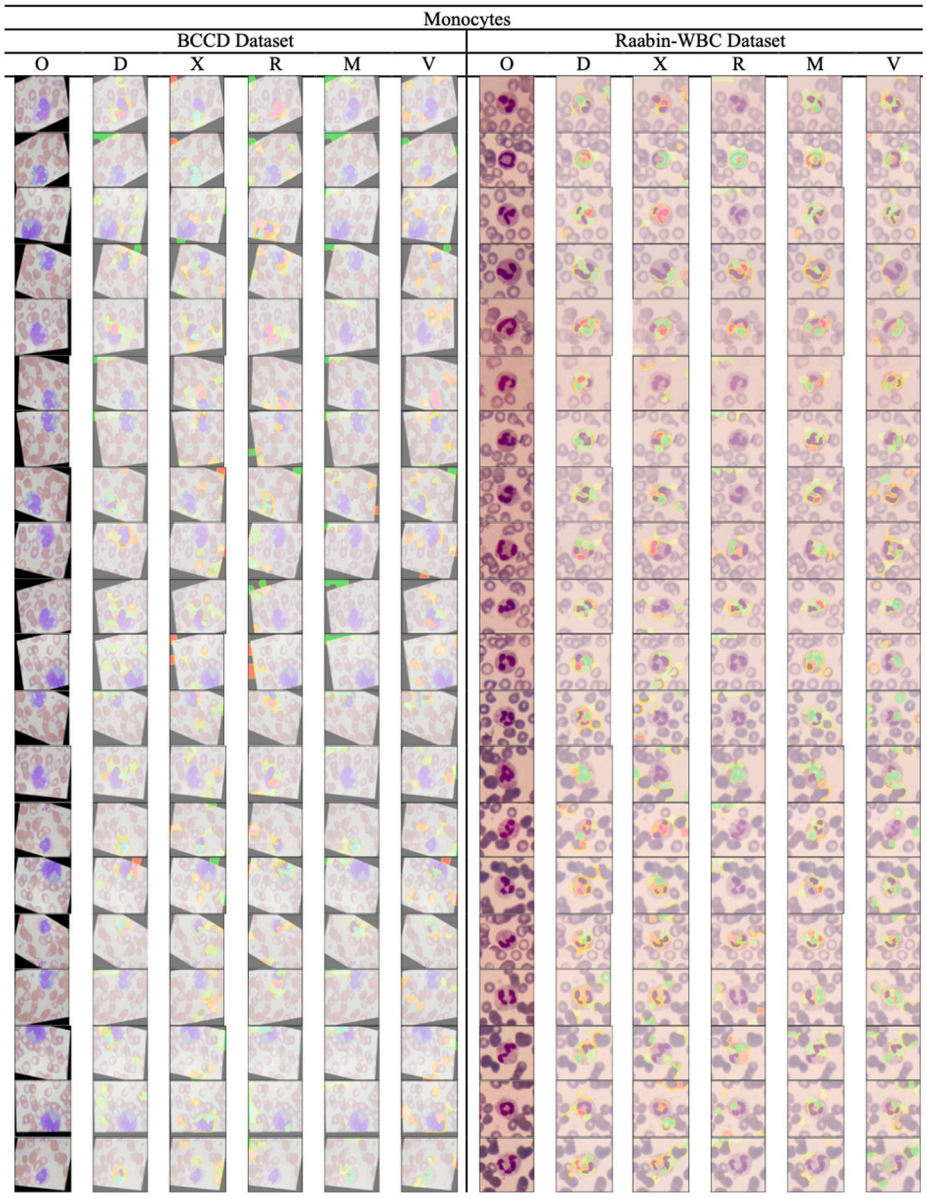

Fig. 14. LIME predicted Images for Monocytes (Original Image (O), DenseNet121 (D), ResNet50(R), Xception (X), MobileNetV2 (M), VGG16 (V)).

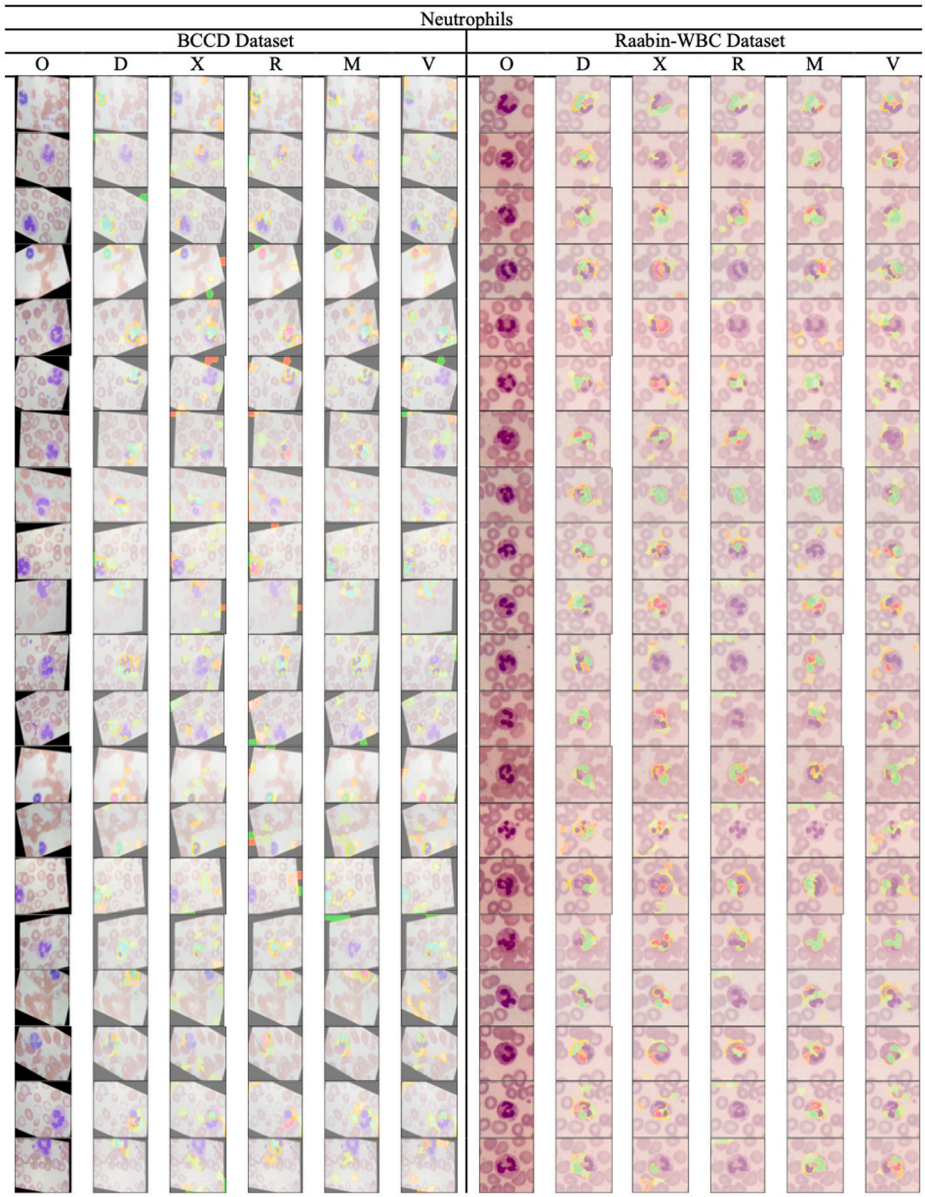

Fig. 15. LIME predicted Images for Neutrophils (Original Image (O), DenseNet121 (D), ResNet50(R), Xception (X), MobileNetV2 (M), VGG16 (V)).

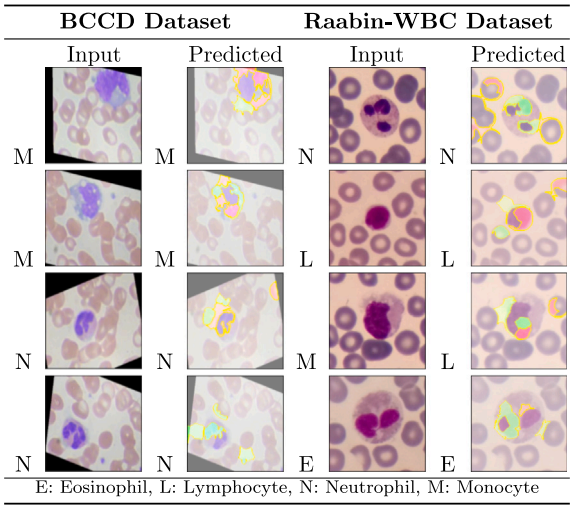

Fig. 16. LIME predictions by DenseNet121.

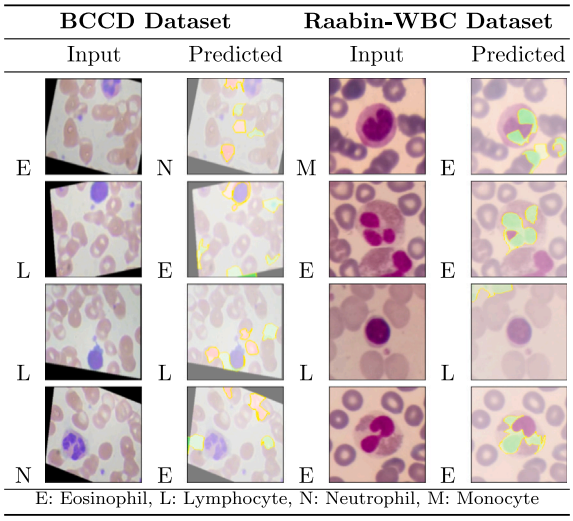

Fig. 17. LIME predictions by Xception.

CRediT authorship contribution statement

Kunal Bhatia: Methodology, Software, Writing – original draft. **Sabrina Dhalla:** Conceptualization, Writing – review & editing, Supervision. **Ajay Mittal:** Visualization, Supervision, Writing – review & editing. **Savita Gupta:** Formal analysis, Supervision, Project administration. **Aastha Gupta:** Validation, Supervision. **Alka Jindal:** Validation, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request

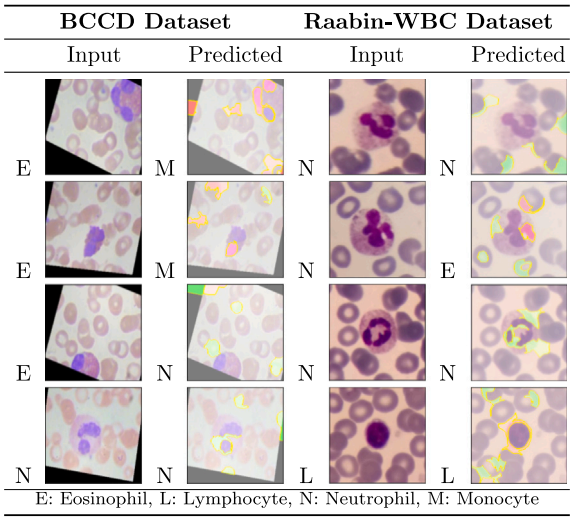

Fig. 18. LIME predictions by VGG16.

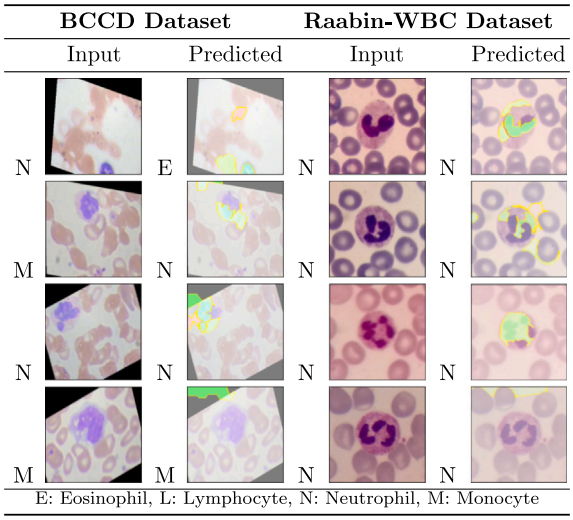

Fig. 19. LIME predictions by MobileNetV2.

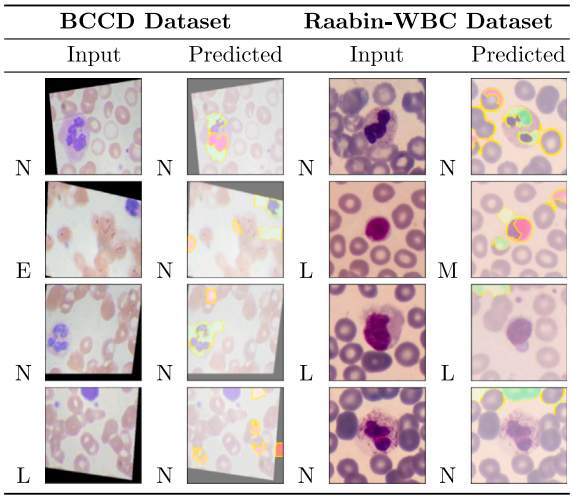

Fig. 20. LIME predictions by ResNet50.

References

[1] H Mohamed E, H El-Behaidy W, Khoriba G, Li J. Improved white blood cells classification based on pre-trained deep learning models. J Commun Softw Syst 2020;16(1):37–45.
[2] Sharma S, Gupta S, Gupta D, Juneja S, Gupta P, Dhiman G, Kautish S, et al. Deep learning model for the automatic classification of white blood cells. Comput Intell Neurosci 2022;2022.
[3] Organization WH, et al. Ethics and governance of artificial intelligence for health: WHO guidance. 2021.
[4] Ribeiro MT, Singh S, Guestrin C. " Why should i trust you?" Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International conference on knowledge discovery and data mining. 2016, p. 1135–44.
[5] Korica P, Gayar NE, Pang W. Explainable artificial intelligence in healthcare: Opportunities, gaps and challenges and a novel way to look at the problem space. In: International conference on intelligent data engineering and automated learning. Springer; 2021, p. 333–42.
[6] Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, p. 4700–8.
[7] Chollet F. Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, p. 1251–8.
[8] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, p. 770–8.
[9] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. 2014, arXiv preprint arXiv:1409.1556.
[10] Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on computer vision and pattern recognition. 2018, p. 4510–20.
[11] Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L. Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on computer vision and pattern recognition. Ieee; 2009, p. 248–55.
[12] Kouzehkanan ZM, Saghari S, Tavakoli E, Rostami P, Abaszadeh M, Mirzadeh F, Satlsar ES, Gheidishahran M, Gorgi F, Mohammadi S, et al. Raabin-WBC: a large free access dataset of white blood cells from normal peripheral blood. 2021, bioRxiv.
[13] Mooney P. Blood cell images. 2017, URL: https://www.kaggle.com/paultimothymooney/blood-cells.
[14] Saraswat M, Arya K. Feature selection and classification of leukocytes using random forest. Med Biol Eng Comput 2014;52:1041–52.
[15] Gautam A, Singh P, Raman B, Bhadauria H. Automatic classification of leukocytes using morphological features and naïve Bayes classifier. In: 2016 IEEE Region 10 Conference. IEEE; 2016, p. 1023–7.
[16] Santhosh Krishna B, Jijin Godwin J, Tharanee Shree S, Sreenidhi B, Abinaya T. Detection of leukemia and its types using combination of support vector machine and K-nearest neighbors algorithm. In: Next Generation of internet of things: proceedings of ICNGIoT 2021. Springer; 2021, p. 435–44.
[17] Girdhar A, Kapur H, Kumar V. Classification of white blood cell using convolution neural network. Biomed Signal Process Control 2022;71:103156.
[18] Wang Q, Wang J, Zhou M, Li Q, Wen Y, Chu J. A 3D attention networks for classification of white blood cells from microscopy hyperspectral images. Opt Laser Technol 2021;139:106931.
[19] Dong N, Zhai M-d, Chang J-f, Wu C-h. A self-adaptive approach for white blood cell classification towards point-of-care testing. Appl Soft Comput 2021;111:107709.
[20] Banik PP, Saha R, Kim K-D. An automatic nucleus segmentation and CNN model based classification method of white blood cell. Expert Syst Appl 2020;149:113211.
[21] Source code and blood image dataset.
[22] Bozkurt F. Classification of blood cells from blood cell images using dense convolutional network. J Sci Technol Eng Res 2021;2(2):81–8.
[23] Kamal MS, Northcote A, Chowdhury L, Dey N, Crespo RG, Herrera-Viedma E. Alzheimer's patient analysis using image and gene expression data and explainable-AI to present associated genes. IEEE Trans Instrum Meas 2021;70:1–7.
[24] Kamal MS, Dey N, Chowdhury L, Hasan SI, Santosh K. Explainable AI for Glaucoma prediction analysis to understand risk factors in treatment planning. IEEE Trans Instrum Meas 2022;71:1–9.
[25] Park MS, Son H, Hyun C, Hwang HJ. Explainability of machine learning models for bankruptcy prediction. IEEE Access 2021;9:124887–99.
[26] Kuzlu M, Cali U, Sharma V, Güler Ö. Gaining insight into solar photovoltaic power generation forecasting utilizing explainable artificial intelligence tools. IEEE Access 2020;8:187814–23.
[27] Felzenszwalb PF, Huttenlocher DP. Efficient graph-based image segmentation. Int J Comput Vis 2004;59(2):167–81.

[28] Vedaldi A, Soatto S. Quick shift and kernel methods for mode seeking. In: European conference on computer vision. Springer; 2008, p. 705–18.
[29] Achanta R, Shaji A, Smith K, Lucchi A, Fua P, Süsstrunk S. SLIC superpixels compared to state-of-the-art superpixel methods. IEEE Trans Pattern Anal Mach Intell 2012;34(11):2274–82.

Kunal Bhatia is pursuing his Master degree (ME) in Computer Science and Engineering (CSE) at UIET, Panjab University (2019-2023). His fields of interest are machine learning, deep learning and computer vision.

Sabrina Dhalla is pursuing Ph.D. from UIET, Panjab University. Broadly conceived, her doctoral work is on deep learning based methods for leukemia detection. Her interest areas are image processing, machine learning, medical image analysis and deep learning.

Ajay Mittal is a professor in the department of CSE at UIET, Panjab University, Chandigarh, India. He has rich experience of nearly two decades in the areas of image processing, machine learning, computer vision, medical image analysis.

Savita Gupta is a professor at UIET, Panjab University, Chandigarh. Her research interest includes medical image analysis, speckle reduction, wavelet applications, cognitive enhancement, and cognitive radios.

Aastha Gupta is an assistant professor in school of technology, Management and Engineering (STME), SVKM's NMIMS University (Deemed to be University), Chandigarh, India.

Alka Jindal is an assistant professor in CSE department of PEC university of technology, Chandigarh, India.