Leukemia Cancer Classification via Attention-Guided Deep Networks on Microscopic Smear Images

Mageshwari V. [1],*, Jana U. Sagar [1], and Ashok K. M. [2]

[1] Department of Mathematics, Amrita School of Physical Sciences Coimbatore, Amrita Vishwa Vidyapeetham, India
[2] Open Labs, Bluecrest University, Monrovia, Liberia
Email: v_mageshwari@cb.amrita.edu (M.V.); cb.ps.p2asd23009@cb.students.amrita.edu (J.U.S.); coe@bluecrest.edu.lr (A.K.M.)
*Corresponding author

Abstract—**Blood cancer is a type of cancer which affected many people and their lives have flipped totally while battling the cancer. Most common type of blood cancer is Acute Lymphocytic Leukemia (ALL), Acute Myeloid Leukemia (AML) and Multiple Myeloma (MM). One of the primary tests done by pathologist to confirm the diagnosis is microscopic blood smear analysis, which requires human interpretation and consumes a lot of time. To overcome these problems, the proposed study helps the pathologist with a distinctive and a novel automated neural network mechanism, called HemoNet, which is a Convolutional Neural Networks (CNNs) architecture with a Convolutional Block Attention Module (CBAM) mechanism, incorporated to correctly identify the type of cancer within less time. This study poses a helping hand rather than completely taking over the human interpretation. The proposed work has been performed over ALL, AML and MM blood smear image dataset. Data transformation was performed to overcome the overfitting, allowing the model to learn the features with different characteristics. The baseline model 'DeepLeukNet' which when trained on the dataset, obtained an accuracy of 99.56% with a validation loss of 0.4. Whereas the proposed HemoNet model which has CBAM module, obtained an accuracy of 99.56% with a validation loss of 0.34, and about 79% faster.**

Keywords—**deep learning, Convolutional Neural Networks (CNN), attention mechanisms, optimization, blood smear analysis, Convolutional Block Attention Module (CBAM), image analysis**

I. Introduction

Leukemia is a type of blood cancer which would affect the white blood cells [1]. Acute Lymphocytic Leukemia (ALL) is a type of blood cancer that grows rapidly and if not diagnosed and treated quickly, it will become fatal in a few months [2, 3]. "Lymphocytic" means development of immature forms of lymphocytes, which are a type of white blood cells under the lymphoid lineage [4]. Lymphocytes are the main cells of lymph tissue, a major part of the body's immune system.

Acute Myeloid Leukemia (AML) is another type of blood cancer which originates from bone marrow where abnormal blood cells are produced at a high rate [5, 6]. The disorder affects the myeloid lineage of blood cells, which include White Blood Cells (WBCs), Red Blood Cells (RBCs) and platelets [7]. Person diagnosed with AML, has increased number of immature cells which later become abnormal cells, called myeloblasts, which are not healthy. As the number of abnormal cells increase, they block the free space to produce other blood cells in the bone marrow. This may lead to anemia, fatigue and infections. There is a very high possibility that AML can spread and attack our Central Nervous System (CNS) which can become very fatal.

Multiple Myeloma (MM) is a type of blood cancer affecting the plasma cells of the human body, which are responsible for producing antibodies and help fighting infections. MM accounts for about 10% of all hematologic malignancies and primarily affect older adults [8]. MM causes immature and cancerous plasma cells to accumulate in the bone marrow and stop the bone marrow from producing healthy plasma cells. These unhealthy plasma cells accumulate and release a protein called abnormal immunoglobulins (M-protein), which can cause organ damage, sometimes end-organ failure. Common symptoms of MM include elevated calcium levels in the body, renal failure, fatigue and compromised immune system.

Deep learning algorithms are very powerful tool and are very helpful in the field of medicine, when it comes to diagnosing a disease. Convolutional Neural Network (CNN), a sub-category of deep learning, are specially designed for processing image data [9–11]. Till now, various cancer diagnostic systems have been developed using deep learning architectures, which are very powerful in diagnosing using medical image data [12–14]. However, image datasets are often large in size and contain a lot of information, thus needing an efficient

Manuscript received July 22, 2025; revised September 3, 2025; accepted October 24, 2025; published December 19, 2025.

doi: 10.18178/joig.13.6.709-721

feature extraction. Deep learning architecture is designed to extract most important and significant features and provide really good accuracy [15].

The identification of blood cancer through microscopic smear images is a crucial diagnostic test, aiding in the early detection of conditions such as ALL, AML, and MM. Since these cancers share common symptoms, accurate classification is essential. However, traditional diagnostic methods can often be costly for patients, making automated and efficient classification techniques are highly valuable. Not only it benefits the patients, but also medical experts by reducing medical errors, diagnosing the disorder at early stages and also reduce the workload by processing large data efficiently. This study, proposes an Attention-Guided deep networks model using CNN to classify the images into ALL, AML and MM cancer types. The model is trained on microscopic blood smear images of above-named type of cancers.

II. Literature Survey

There has been a constant study happening in medical science on applying Artificial Intelligence (AI), helping in faster diagnosis and prognosis of the disease [16–18]. In the department of oncology, there have been many studies published on using machine learning and deep learning concepts to help the experts in classifying the type of cancers based on the results obtained from performing clinical tests, and these new updates cut down the process time and any human error involved [19].

Saeed *et al.* [20] performed a study on classifying Acute Lymphoblastic Leukemia (ALL). Their study comprised of analyzing a total of 268 blood smear images, which they used to develop a CNN architecture called ‘DeepLeukNet’. Upon training the model, they obtained an accuracy of 99.61% in classifying the image into either ALL or normal blood smear images. Their proposed model comprised of 23 trainable layers, and they have used Rectified Linear Unit (ReLU) activation function as their study focused on binary classification type. The author has successfully demonstrated and presented the performance of DeepLeukNet in terms of accuracy. However, there is a clear need to compare their proposed model with the existing models in terms of time efficiency.

Rahman *et al.* [21] proposed a study on multiclass blood cancer classification using ResNet50 model with Particle Swarm Optimization (PSO) and Cat Swarm Optimization (CSO) which had given them an accuracy of 99.84%. Their study was done on blood smear images collected from 89 different patients. Of the 89 different samples, 64 samples were identified as ALL and the rest 25 samples were identified as healthy and normal. Their study also comprised of comparing the accuracy scores of different classifiers like Decision Tree, Support Vector Classifier (SVC), Random Forest (RF), Logistic Regression with various CNN models like VGG19, ResNet50, Inception V3. Out of which, ResNet50 model had given the highest accuracy.

Zheng *et al.* [22] proposed a study on classifying nodules with an attention—guided deep neural network with a multichannel architecture. His study introduces a multi-channel attention mechanism in a deep learning model to precisely classify malignant lung nodules. He had done this study using 1018 Computed Tomography (CT) scans. Their proposed model took only 29.09% of the total time taken by the ResNet50 model, and also obtained an accuracy of 90.11%. In this study, the researcher has cropped the images into 3 cubes, each of different scale. Then they used multi-channel attention model on the 3 inputs, which allowed this model to process and learn the images in less time compared to standard CNN models.

Dese *et al.* [23] had performed their study on classifying leukemia using machine learning based approach. They obtained an accuracy of 97.69% in their study on 520 blood smear images. Their study mainly focused on classifying types of Leukemia, viz, ALL, AML, and Chronic Lymphoblastic Leukemia (CLL). They have used Support Vector Machine (SVM), a supervised machine learning model, and the output was compared with other models like K-Nearest Neighbors (KNN) and Artificial Neural Network.

Shaheen *et al.* [24] conducted an extensive study involving 4000 microscopic blood smear images to accurately detect ALL. The researchers implemented and trained two deep learning architectures, AlexNet and LeNet-5 to evaluate their effectiveness in leukemia classification. The models were developed and trained using the MATLAB environment. Their findings demonstrated a high classification performance, achieving an impressive accuracy of 98.58% with AlexNet and 96.25% with LeNet-5.

Allegra *et al.* [15] reviewed ways to detect multiple myeloma by leveraging machine learning and deep learning techniques, and also help with treatment selection and track prognosis of the disease. Their study involved reviewing multiple research articles involving the application of deep learning methodologies, at various steps in the process of diagnosis and prognosis. Through their study, they have discovered that deep learning procedures have given better outputs, compared to traditional ML procedures, despite their complex computational and training time.

Xu *et al.* [25] conducted a comprehensive study on the application of ResNet architectures in the medical imaging domain. Their research highlights the widespread adoption of ResNet models in various diagnostic tasks, particularly in areas such as lung tumor classification, skin disease identification, breast cancer detection, and brain disorder diagnosis. The study emphasizes the effectiveness of ResNet due to its skip connection mechanism, which mitigates the vanishing gradient problem—a common limitation in deeper neural networks. This architectural advantage enables ResNet to maintain performance and stability even as the network depth increases, making it highly suitable for complex medical image analysis.

Das *et al.* [26] has done a study on the identifying ALL. They proposed a method using Tangent Sand Cat Swarm Optimization with Long Short-Term Memory (LSTM), which they incorporated with LeNet architecture. This approach allowed them precisely classify malignant leukemia blood smear images. Their study has proven to give an accuracy of 98.7% and precision of 97.9%.

Ramaneswaran *et al*. [27] has done a study on classifying blood cancer using blood smear images. They proposed a hybrid architecture that integrates machine learning and deep learning architectures. They used InceptionV3 architecture to extract salient features from the images. These extracted features were then processed as input to the XGBoost, which used a classification head in the architecture. This hybrid model has reported an impressive F1-Score of 98.6%.

III. Methodology

This study proposes a deep learning model to correctly classify the type of blood cancer. This model is a combination of CNN layers and attention mechanism layer. DeepLeukNet is a fine-tuned deep learning model which is built on top of the pre-trained architectures like ResNet and AlexNet. The model is engineered to produce high accuracy by automatically identifying most significant features. HemoNet is an optimized model with faster processing time and lower computational requirements which makes it more suitable for blood cancer analysis.

Images from the image dataset have been transformed and normalized to tackle overfitting and enabling the model to learn the images represented in various formats. The images, then, have been split into training and testing set in the ratio of 70:30 respectively, using train-test split function from scikit-learn package. The baseline CNN architecture selected for the study was 'DeepLeukNet' [20]. As discussed earlier, this method aims on building a CNN model with attention mechanism block. Its performance was compared with the base model. Fig. 1 is the proposed workflow followed during the study.

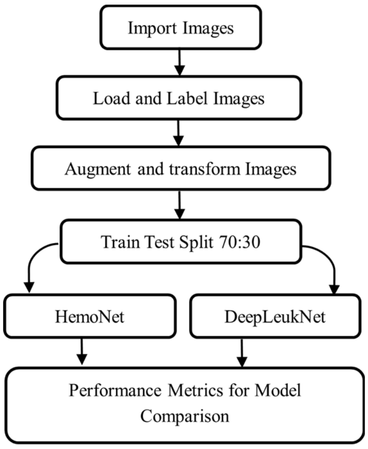


Fig. 1. Workflow diagram.

A. Data Collection

Collection of data is the most crucial part for any research. If the data is ambiguous, or does not align with the objective of the study, it ends up giving wrong results which can affect the practical applications. For this study, data was collected from recognized and approved websites, viz., National Cancer Institute which is a website managed by the United States government, and Kaggle data repository. The images include high quality colored blood smear images. The data was primarily segregated into 3 distinct folders, viz., ALL, AML, MM, corresponding to the type of cancer under investigation. Originally, each folder contained images between the range 240–260 images, mitigating the class imbalance and supporting robust model training.

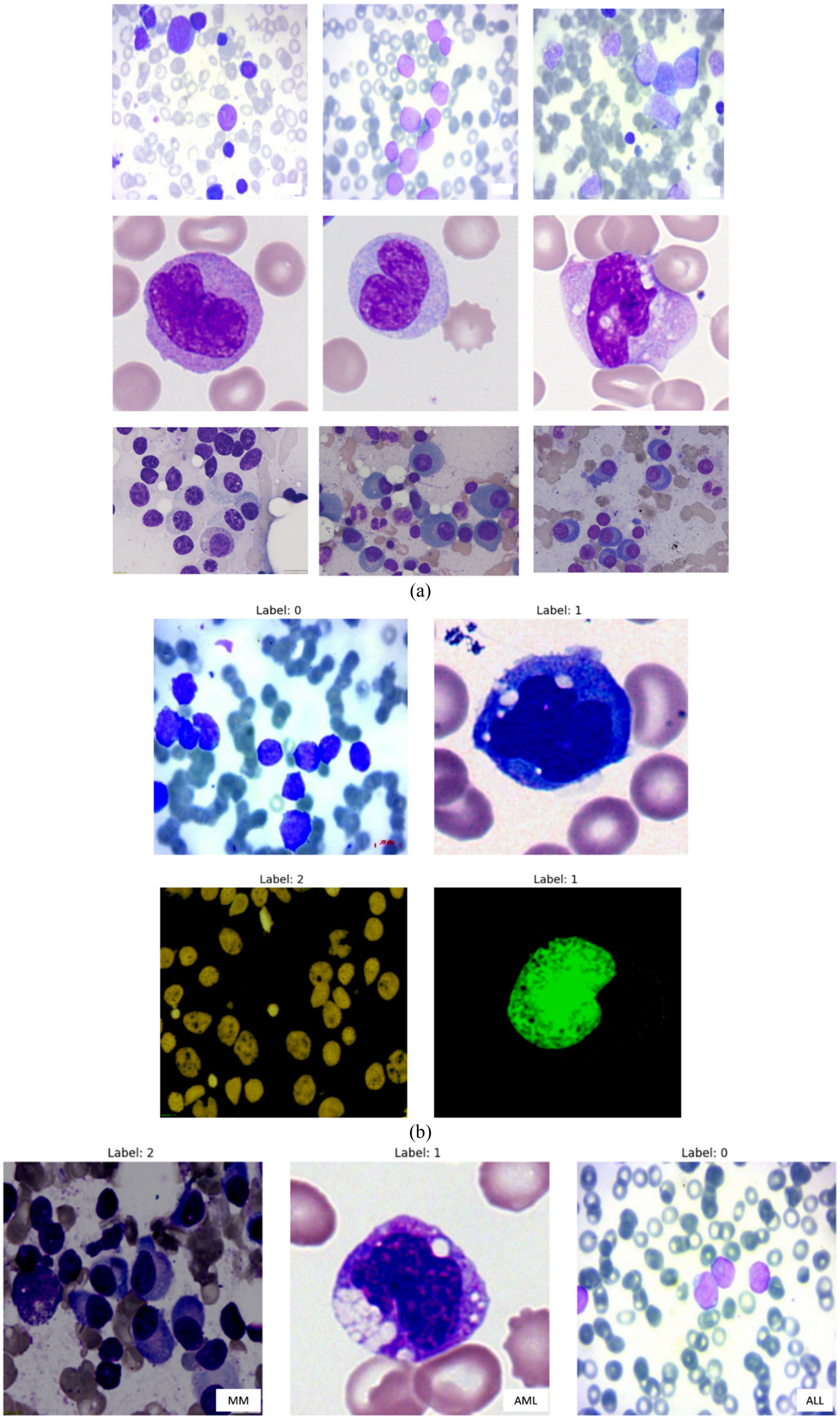
B. Data Preparation

To ensure that the model interprets the data effectively, the images have been processed, and that was done using the Transformers module from PyTorch package. The images were scaled to 224×224 pixels. Since the original dataset was unlabeled, labelling of the images was done using Label Encoder method, based on the folder to which they belonged to. The resulting labels for the images were [0, 1, 2] corresponding to [ALL, AML, MM] respectively. The images were split into training and testing set in the ratio of 70:30.

The total number of images in the training set were 527, and in the testing set were 227. All the images in training set and testing set were of different sizes, and hence, they had to be resized to 224×224 pixels and RGB color channels were normalized to [0.4, 0.4, 0.4]. Additionally, to enhance the robustness of the model, other transformations were also applied exclusively to the training dataset. These transformations include, adjusting the brightness, inverting the color channels, and amplifying brightness. Fig. 2(a) represents the sample images in the dataset. Fig. 2(b) and (c) illustrate the images with labels and transformations applied onto them. Once the images were ready, both training and testing datasets were processed batchwise with each batch size of 40. Batch processing was done to reduce the load on the system while training the model.

Following the data preparation, the next step was to define the process for training the proposed model. A comprehensive functions with all the essential components for the model to learn, was defined. It included early stopping, configured with a patience of 5 epochs, which preventing the model to learn if outputs of 5 consecutive saw no improvement. Additionally, a nested function was defined to mutate the learning rate, to tackle stagnation in the performance. Whenever the mutation triggered, learning rate is mutated by a factor of 1.2. Throughout the training process, the Key Performance Indicators (KPI) including training time, training loss and validation loss were recorded at each epoch, which provided detailed insights about the model's learning performance.

The training of the proposed model was conducted on a local system with the configuration of 20 GB of memory and no GPU acceleration. The loss function used for this study was 'Cross Entropy Loss'. It was used because of its ability to learn from the previous errors and allow the model to accurately classify the image and inherently performed the role as SoftMax layer, simplifying the model's architecture.



(a)

(b)

(c)

Fig. 2. Images sampled from the dataset. (a) All types of sample images (b) Sample of training image set after transformation and labelling the images as Label:0 (ALL), Label:1 (AML) and Label:2 (MM) (c) Sample of labelled images from the testing set, after scaling to 224×224 pixels and normalizing the RGB color channels to [0.4, 0.4, 0.4].

In PyTorch, cross entropy loss function combines LogSoftMax and Negative Log-Likelihood loss function.

LogSoftMax function is defined as Eq. (1).

$$LogSoftMax\left(Z_i\right) = log\left(\frac{e^{Z_i}}{\sum_{j=1}^{k} e^{Z_j}}\right) \tag{1}$$

Negative Log Likelihood loss:

$$\mathcal{L}= -log\left(\hat{y}_{true\ class}\right) \tag{2}$$

Thus, the cross-entropy loss function is defined as:

$$Loss = -log\left(\frac{e^{Z_i}}{\sum_{j=1}^{k} e^{Z_j}}\right) \tag{3}$$

where Z_y is the predicted value of the correct class; Z_j are the values of all the classes (output from the last linear layer).

C. Proposed Model

The proposed model, named HemoNet, is an adaptation of the DeepLeukNet architecture which initially had 29 layers comprised of 22 layers in the convolutional block and 7 layers in the fully connected layer, and it was primarily designed for binary classification i.e., ALL or normal cells [20]. In a similar way, this study proposes a new CNN architecture, HemoNet which has an attention mechanism block i.e., CBAM module, making the architecture more efficient in learning. Architecture of the proposed model is given in Fig. 3.

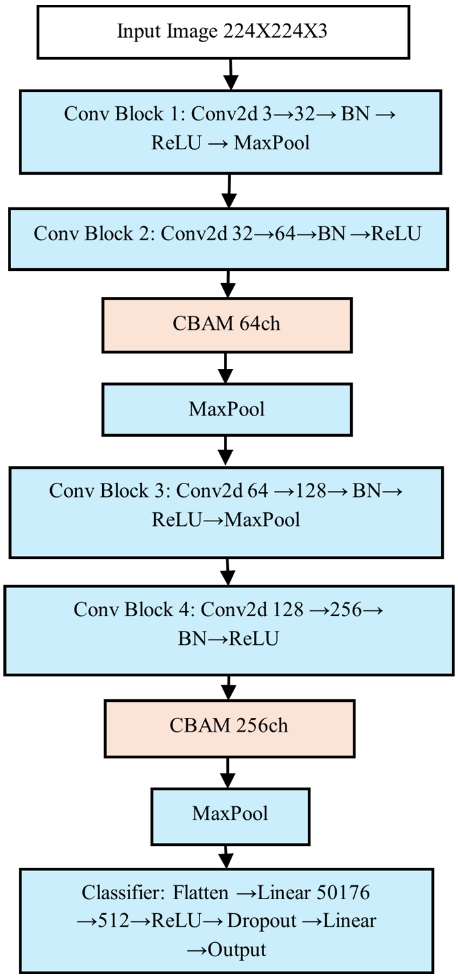

Input Image 224X224X3
Conv Block 1: Conv2d 3→32→ BN → ReLU → MaxPool
Conv Block 2: Conv2d 32→64→BN →ReLU
CBAM 64ch
MaxPool
Conv Block 3: Conv2d 64 →128→ BN→ ReLU→MaxPool
Conv Block 4: Conv2d 128 →256→ BN→ReLU
CBAM 256ch
MaxPool
Classifier: Flatten →Linear 50176 →512→ReLU→ Dropout →Linear →Output

Fig. 3. Architecture of the proposed HemoNet model.

Convolutional Block Attention Module (CBAM) is an attention module proposed by Woo *et al*. [28] at Korea Advanced Institute of Science and Technology. The CBAM helps improve how well a deep neural network can classify images by letting it focus on the most important parts of an image. It does this by looking at both the "what" and "where" aspects of the image. By applying attention mechanism in two steps—first looking at the channels and then the spatial layout—CBAM makes the image features more useful for classification without adding much extra computation.

CBAM helps improve classification in several ways. It allows the model to learn which features are useful and which are not, making the learning process more efficient. By combining both channel and spatial attention, CBAM captures the big picture (like the type of object in an image) and the detailed parts (like the exact location of a feature). This is especially useful in medical imaging, where the model can not only recognize the presence of a disease but also locate specific areas of concern.

CBAM also helps the model perform better on new and different types of data by making it focus on real, important features instead of getting confused by irrelevant details. This makes the model more reliable, especially in complex analysis like medical diagnosis. Another benefit of CBAM is that it makes the model's decision-making clearer. The attention maps created by CBAM show which parts of image the model focuses on, helping us to understand how the model works. This transparency is important for building trust and improving the model.

CBAM is also very efficient, making it easy to add to existing models without changing much of the original structure. It doesn't require a lot of extra parameters or computation, so it's a great way to improve performance without making the model bigger or more complex.

The mechanism of CBAM is to effectively infer the attention maps at two dimensions, one is channel dimension, and the other is spatial dimension. To do this, CBAM has two modules, Chancel Attention Module (CAM) and Spatial Attention Module (SAM). CAM emphasizes on the important features of feature map across all the channels, whereas SAM emphasizes on the location of the important features of the feature map. SAM takes refined features from the CAM and applies a similar pooling process but along the channel dimension. These features are then combined and passed through a small convolution layer to create a 2D attention map. This map, after applying a sigmoid function, highlights the important parts of the image. These highlighted regions are then used to adjust the final feature map, making it more focused on the relevant parts for classification.

Advantage of using CBAM is that, it was specially designed to focus only on the feature maps which are important. All the features that were extracted through the previous layers are passed to CBAM module, which then refines by highlighting only the important feature maps.

The HemoNet's feature learning is improved by the CBAM module, which allows the network to concentrate on the most pertinent data in the channel and spatial dimensions of its feature maps. This procedure increases the accuracy of feature extraction and eliminates superfluous information, which is essential for a task like blood vessel segmentation or analysis, which HemoNet probably completes.

D. Architecture HemoNet with Convolutional Block Attention Module (CBAM) Mechanism

This version of CNN architecture has 41 layers, including the input layer. The layers were primarily clubbed to form Convolutional Blocks. In total, there are 4 convolutional blocks and 1 classifier block. Convolution Blocks 2 and 4 have CBAM layer, which enhances the feature learning. The following is the structure of the CNN architecture with CBAM mechanism

1) Input layer

Each input image fed to the model is an RGB image with dimensions (224×224×3), where 224 denotes the spatial dimensions (heights and width), and 3 denotes the number of color channels. The images are classified into one of the following 3 blood cancer types represented as categorical variables, viz., ALL (0), AML (1), MM (2).

2) First convolutional block (4 layers)

The first convolutional block comprises of the following layers in the sequence, convolutional layer, batch normalization layer, ReLu function, max pooling layer.

This block captures the 32 low-level features from the images such as color pattern, textures, and basic edges. The convolutional operation is applied using a set of learnable filters to compute the features. Output of the convolutional operation is given by Eq. (4):

$$Y(i,j)=\sum_{m=1}^{k-1}\sum_{n-1}^{k-1}X(i+m,j+n)\cdot W(m,n) \tag{4}$$

where:

X = input image/feature map.

W = filter of size (k×k).

$Y(i, j)$ = resulting feature map at the position (i, j).

Following the convolutional operation, batch normalization was applied which accelerated training by normalizing the output of the convolutional layer. Batch normalization is defined by the Eq. (5):

$$y=\frac{x-E[x]}{\sqrt{Var(x)+\varepsilon}}\times\gamma+\beta \tag{5}$$

where: x is input value; γ, β are learnable parameters; ε constant value added for stability.

Next, ReLU is applied which adds a non-linearity to the model, enabling the model to learn complex patterns. ReLU is defined by the Eq. (6).

$$f(x)=\max(0,x) \tag{6}$$

Finally, maximum pooling has been applied to reduce the spatial dimensions while preserving important features, reducing the computational complexity. Max pooling is defined as Eq. (7):

$$Y(i,j)=\max_{m,n}X(i+m,j+m) \tag{7}$$

where: m, n are dimensions of the pooling window. i, j are dimensions of the output feature map.

As a result of this convolutional block, original input image of the size (224×224×3) is transformed into (32×112×112) which is then passed on to subsequent convolutional blocks for deeper feature extraction.

3) Second convolutional block (4 layers + CBAM)

The second convolutional block shares a similar structure with the first convolutional block. Additionally, CBAM is integrated to enhance feature refinement through attention mechanism. This block is responsible for extraction of mid-level features. Output from the First Convolutional Block, (32×112×112) is fed as the input. It consists of the following sequential layers; convolutional layer, batch normalization layer, ReLU function, CBAM, max pooling layer.

Once the mid-level features are extracted through convolutional operations and batch normalization and ReLU activation are added, the resulting feature maps are passed through CBAM layer, which refines the features using attention mechanism. CBAM layer is comprised of two modules:

a) Channel Attention Module (CAM)

CAM aims to focus on what is meaningful in a given feature map. CAM operates through the following layers; Global Average Pooling (GAP), Global, Max Pooling (GMP), shared Multi-Layer Perceptron (MLP), Sigmoid activation function.

Important features from each feature map are selected by Global Average Pooling (GAP) and Global Max Pooling (GMP). GAP takes the average of all the pixels in the feature map using the Eq. (8).

$$F_{avg}^{c}=\frac{1}{H\times W}\sum_{i,j}X^{c}(i,j) \tag{8}$$

GMP takes the average of all the pixels in the feature map using the Eq. (9).

$$F_{max}^{c}=\min_{i,j}X^{c}(i,j) \tag{9}$$

where $X^{c}(i,j)$ is the pixel at height i, width j, in the feature map c. H×W is the spatial dimension.

Now both F_{avg}^{c} and F_{max}^{c} are passed through the same multi-layer perception followed by element wise addition. The result is then passed through a sigmoid function. This tells us how much to emphasize on each feature map, which is called as the channel attention mask. Channel attention mask is calculated using the Eq. (10).

$$M_c = \sigma\left(MLP(F_{avg}^{c}) + MLP(F_{\max}^{c})\right) \tag{10}$$

where σ is the sigmoid function; M_c is the attention mask.

Obtained attention mask is now multiplied to each input feature map.

$$X' = M_c \cdot X \tag{11}$$

b) Spatial Attention Module (SAM)

After channel wise refinement, SAM determines where the important features are located in the feature map. This is achieved by calculating the spatial summary along the channel axis effectively summarizing spatial characteristics. To compute the spatial summary, average pooling and max pooling are applied independently. Average pooling highlights the overall spatial distribution while max pooling highlights the salient spatial activations. Eqs. (12) and (13) are used to calculate the spatial descriptors.

$$F_{avg}^{spatial} = \frac{1}{c}\sum X_c \tag{12}$$

$$F_{max}^{spatial} = maxX_c \tag{13}$$

The resultant $F_{avg}^{spatial}$ and $F_{max}^{spatial}$ are passed through a 7×7 convolutional layer, followed by sigmoid function. The output obtained is called as spatial attention mask and is calculated by Eq. (14).

$$M_s = \sigma\left(Conv_{7\times 7}([F_{avg}^{spatial} \times F_{max}^{spatial}])\right) \tag{14}$$

Obtained spatial attention mask M_s is applied on all the feature maps on which channel attention mask has already been applied. This is represented by the Eq. (15).

$$X'' = M_s \cdot X' \tag{15}$$

The obtained X'' is the enhance feature map with channel wise and spatial wise attention mask.

Following CBAM, max pooling is applied on the obtained enhanced features to down sample spatial resolution while retaining the enhanced features. Input to this block, originally of the size (32×112×112), is transformed into a more abstract representation of size (64×56×56), which is then broadcasted to the next convolutional block for further feature extraction. Fig. 4 represents architecture of CBAM module.

4) Third convolutional block (4 layers)

The third convolutional block replicates the operation and structure of the first convolutional block, focusing on capturing deeper and higher-level features. It consists of the following sequential layers: convolutional layer, batch normalization layer, ReLU function, max pooling layer.

The block gets the input from the second convolutional block, which had the dimensions (64×56×56). The convolutional layer applies a series of filters to extract the features while batch normalization ensures stability and ReLU activation function adds non-linearity allowing the model to learn intricate patterns in the high-level features.

Finally max pooling reduces the spatial dimension while preserving the important and informative features. The output of this block is of the dimension (128×28×28), which is passed through fourth convolutional block.

5) Fourth convolutional block (4 Layers + CBAM)

This layer is same as the second convolutional block with layers in the same sequential order: Convolution layer, batch normalization layer, ReLU function, CBAM, max pooling.

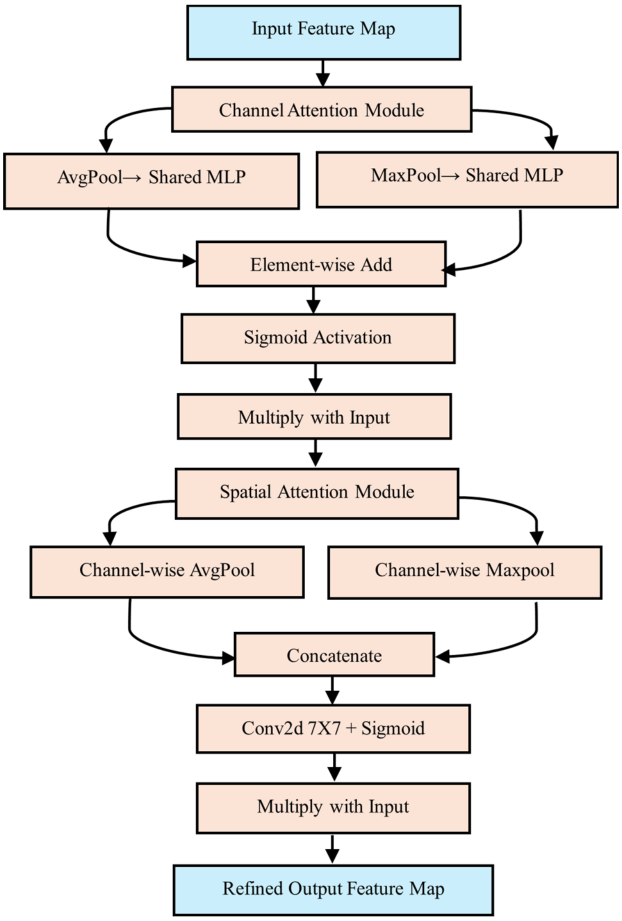


Fig. 4. Architecture of the CBAM module.

Input to this block is the feature maps with high level feature representation that were obtained in the third convolutional block with the dimensions (128×28×28). The convolutional layer further deepens the feature extraction. Batch normalization and ReLU ensure stability and efficient learning respectively. The CBAM module, comprising channel attention and spatial attention, adaptively emphasize on the important features of the input feature map.

After further refinement of the features, max pooling is applied to reduce the spatial dimensions while preserving the informative features. Finally, the resulting output is of the dimension (256×14×14), which is fed to the classifier block.

6) Classifier block (4 layers)

The classifier block, serves as the final stage of the network, responsible for transforming the refined feature maps into categorical predictions corresponding to the target classes: ALL (0), AML (1), MM (2). It consists of the following sequential layers: Linear (fully connected) layer, ReLU function, dropout layer, final linear (output) layer.

Input to this block, is from the fourth convolutional layer, which is of the dimension (256×14×14). This feature map is flattened and a new feature vector of the dimension 256×14×14 = 50,176, is created, consolidating the channel and spatial awareness in a suitable dense classification. The obtained feature vector is passed through fully connected layer which reduces the dimension to 512 features, effectively focusing on the significant features. ReLU function is applied to introduce the non-linearity to capture complex feature interactions and decision boundary. Then the dropout layer with 30% probability is applied, to randomly omit a subset of neurons, preventing the model to overfit and enhance the generalization.

Finally, the second linear layer (output layer) projects the 512-dimension feature vector to the output classes, which in our case is 3. The output of this layer is a set of logits, which is passed through 'Cross Entropy Loss' function which mimics the SoftMax function, producing the class probabilities ultimately yielding predicted class labels for each input image.

When training a CBAM with a large dataset, Stochastic Gradient Descent (SGD) is used to tackle the high computational cost and memory issues of standard gradient descent. SGD updates the model step by step using small batches of data, which is crucial for managing the large number of parameters in deep neural networks where CBAM is usually added. Handling vast parameters is another challenge, as CBAM adds more parameters to existing networks. SGD tackles this by updating the model using small batches, which saves time and makes training more efficient. SGD also speeds up the training process by providing frequent updates. This allows the model to adjust quickly, helping it learn faster, especially at the beginning.

SGD helps in rapid convergence by making the model to achieve good performance quicker. It helps explore the loss landscape better. The random updates from mini-batches or single data points assist in avoiding shallow local minima, allowing the model to find better solutions in complex, non-convex problems common in deep learning. SGD is the basics for advanced optimizers. Technically like momentum or adaptive learning rates are built on top of the SGD for smoother and faster convergence. For attention-based models like CBAM, adaptive optimizers are often more effective. This is because the noise in SGD has a heavy-tailed distribution, and adaptive methods can adjust learning rates per parameter, which is especially helpful for the varying attention weights.

Using SGD with a low starting learning rate and weight decay is a common and effective way to train deep learning models. This setup combines the fast and efficient optimization of SGD with techniques that help keep the model stable and reduce the risk of overfitting. A high learning rate can cause the optimizer to jump past the best parameter values, making the loss go up and down unpredictably. A small learning rate of 0.0001 ensures that each update is a small and careful steep, helping the training process stay stable. This prevents overshooting and divergence.

Even though a fixed learning rate of 0.0001 might slow things down, it is often used as a starting point when combined with a learning rate schedule. This steep allows the learning rate to drop over time as the model gets closer to the best solution, letting it take bigger steps at first and smaller ones later.

Studies show that using SGD with large learning rates can add some kind of built-in regularization. However, a smaller learning rate can also have this effect. A low learning rate with slow decay is less likely to mess up the natural optimization process that helps the model perform well on new data.

Weight decay is a technique that helps prevent overfitting by punishing large weights in the model. The value of $1e^{-4}$ is typical starting point. Weight decay, also known as L2 regularization, adds a cost to the loss function based on how big the weights are. This encourages the model to use smaller weights, leading to a simpler model that is less likely to memorize the training data. This also enhances implicit regularization.

Recent research suggests that weight decay, especially in deep networks, doesn't just act as a regularizer. It can also boost the implicit regularization from the noisy updates of SGD, helping the model find solutions that work well on unseen data. Adding weight decay changes the path the optimizer takes, stopping the weight values from becoming too big. This makes training more stable, can lead to better performance and has a great impact on optimization dynamics.

For models that aren't trained enough, weight decay helps manage the tradeoff between bias and variance, which can result in a lower training error. By doing this the bias-variance tradeoff is balanced.

So, in order to optimize the learning process of the proposed architecture, SGD is used as the optimizer function, with initial learning rate set to 0.0001 and weight decay to $1e^{-4}$, which adds penalty to the loss function. Setting SGD to low initial learning rate and weight decay is an effective training strategy in deep learning. Following is the SGD optimization equations.

$$v_t = \mu \cdot v_{t-1} - \eta \cdot \nabla_\theta \mathcal{L}(\theta_t) + \lambda \theta_t \tag{16}$$

$$\theta_{t+1} = \theta_t + v_t \tag{17}$$

where:

θ_t is the model parameter.

η is the learning rate (0.0001).

$\nabla_\theta \mathcal{L}(\theta_t)$ is gradient loss of the function.
μ is momentum coefficient (0.5).
v_t is velocity.
λ is weight decay $1e^{-4}$.

IV. Results

Following extensive data preprocessing and rigorous training, a comprehensive evaluation of the 'HemoNet' model was carried out. The performance of the model was assessed based on the three key benchmarks: accuracy score, loss value, and training time per epoch. These benchmarks were compared against those of the baseline model 'DeepLeukNet' and two widely recognized CNN architectures 'ResNet50' and 'ResNet101', both of which are commonly employed for medical imaging tasks such as tumor detection, cancer diagnosis, and radiological image analysis like X-Ray and MRI scans.

The proposed 'HemoNet' model demonstrated superior performance achieving an impressive training accuracy of 99.8% upon training on 527 images and validation accuracy of 99.56% upon training on 227 images, within 12 epochs, indicating the model's ability to learn the features quickly and effectively.

The key contributor to the HemoNet model is the CBAM mechanism. By adaptively focusing on the informative channel-wise and spatial-wise features, it allows to significantly enhance the model's ability to identify the salient patterns in the microscopic blood smear images. Integration of CBAM not only accelerated the convergence, but also reduce the overall computation overhead compared to models without attention mechanism.

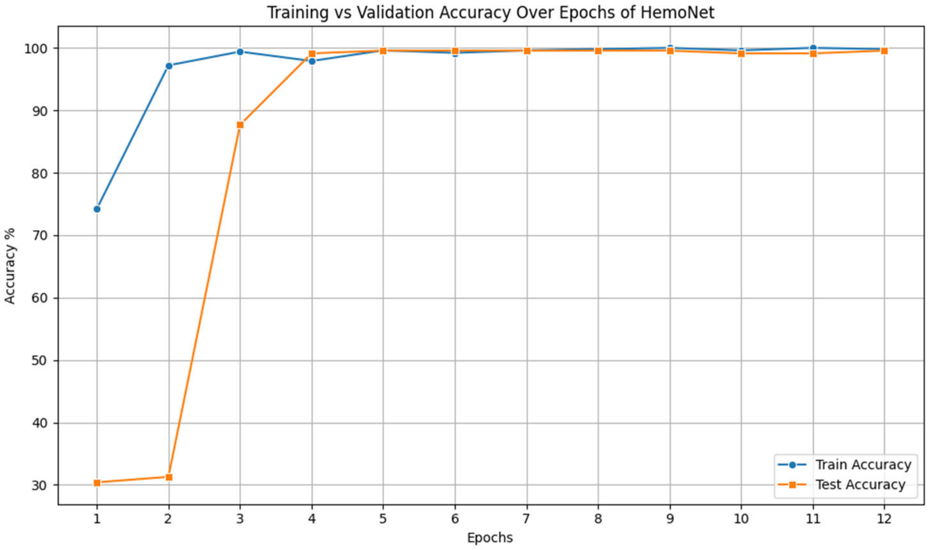


Fig. 5. Illustration of training and validation accuracy of HemoNet at every epoch.

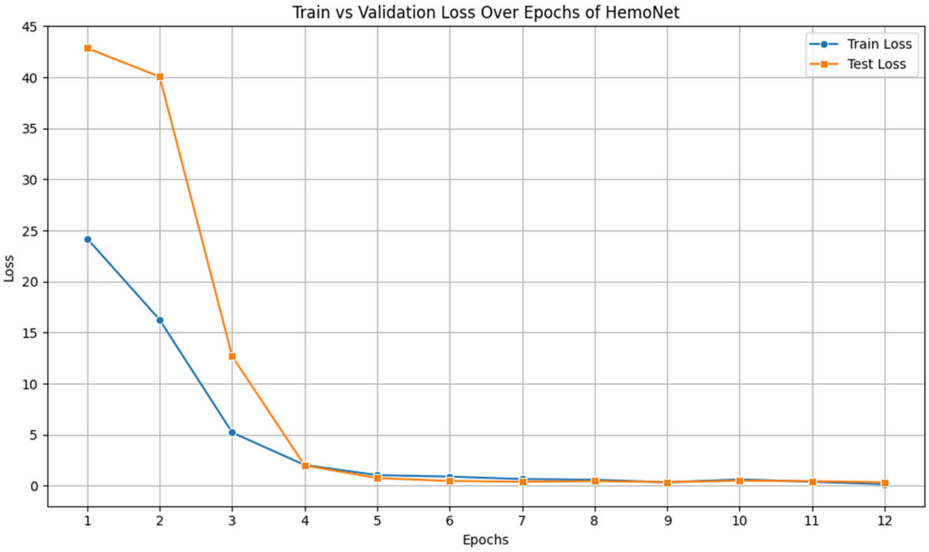


Fig. 6. Illustration of training and validation loss values of HemoNet at every epoch.

Fig. 5 illustrates the training and validation accuracy curve of the HemoNet model over 12 epochs of training. In the initial iterations, the model's accuracy improves sharply from approximately 75% to near 98%, indicating that the model quickly learnt meaningful patterns. Meanwhile the validation accuracy increased rapidly after 3 epochs, from near 30% to 88%. From epoch 4 onwards, training and validation accuracies converge and stabilize near 100% mark. Both training and validation accuracy maintained over 99% over epoch 5 and from epoch 4 there was minima gap between them at a consistent rate, which indicated excellent generalization by HemoNet model, suggesting that is not overfitting the training data, which is a critical aspect in deep learning. The smooth plateau demonstrates that the model retains its learned features with no significant signs of accuracy degradation or model instability.

In machine learning and deep learning, loss value is as significant as other evaluation metrics. Loss value indicates whether the model is performing well or needs an update on how the input is being fed and the model is learning. Essentially, lower loss value means better performing model. Fig. 6 illustrates the training and validation loss obtained on implementing HemoNet model over 12 epochs. It was observed that the loss values in the initial epochs were very high, which is an expected sight. By epoch 4, both training and validation loss values dropped near to 1.5. In the upcoming epochs, the loss values remained constant near to 0 with minimal gap between training and validation loss. This phenomenon is consistent with the earlier accuracy curve, explaining the model's ability to retain the important information throughout the epochs, which explains that model not simply recognizing the training samples but is instead learning robust features, aided by CBAM module.

Fig. 7 illustrates the training time required (in seconds) for four different deep learning models: DeepLeukNet, HemoNet, RestNet50 and ResNet101. Each model was measured over 12 epochs, under similar experimental configurations. This provides valuable insights into computational efficiency and stability of each model during the training. ResNet101 demonstrated high training time, with 360 s in the early epochs and gradually increasing, with reaching a maximum of 800 s. This indicates higher computational cost, highlighting the complex architecture of ResNet101, which requires process during forward and backward passes. On the other hand, ResNet50 presents moderate training time compared to ResNet101.

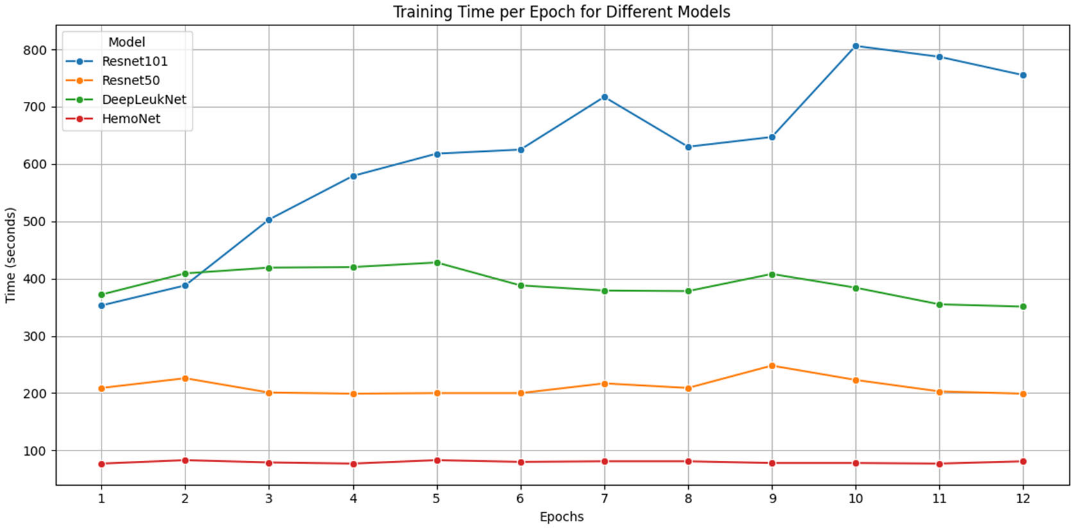


Fig. 7. The training time curve for different deep learning models.

The baseline model 'DeepLeukNet' exhibits higher training time compared to ResNet50, and lower training time ResNet101. Higher training time per epoch indicates that the model is taking more time to extract and learn the features. The training time of the HemoNet model ranges between 70 s to 85 s across multiple epochs, which is two times less than Resnet50 and comparatively four times lesser than DeepLeukNet and Resnet101 models. Since the computational cost is directly proportional to the training time of the model, faster the model training would significantly reduce the computational overhead. In such scenario the proposed model is more suitable for the vital fields like medical where milliseconds can make a huge difference in decision making. Among all the models, the proposed 'HemoNet' model stands out to as the most computationally efficient, with very less training time compared to other deep learning models. This outstanding speed while achieving high accuracy values with very minimal loss, is because of the CBAM mechanism, that was integrated into the custom CNN architecture.

Fig. 8 presents a comparison of accuracy scores achieved on 4 models: ResNet50, ResNet101, DeepLeukNet, HemoNet. Starting with ResNet50, the model achieved 100% training and testing accuracy, indicating excellent generalization, and has learned the underlying data very well when evaluated on unseen samples. While achieving 100% validation accuracy seems impressive, it is not ideal and there are subsequent drawbacks to it. Having such accuracy often raises concerns about its robustness and data leakage. At times, the model might not be feasible to be deployed in the real world as the unseen data is always prone to bias and noise.

Talking about the ResNet101 architecture, the model fits well on the training data, achieving a training accuracy of 100%, indicating that the model learnt the training samples very well, but when it comes to unseen data (validation data), the model generalized poorly. Additionally, ResNet101 being a very complex model, took more time to complete one epoch, which was illustrated in Fig. 7.

The baseline model achieved impressive training and testing accuracy of 99.8% and 99.56%, respectively. The model performed very well on both training and testing set, but incurred higher training and validation loss, compared to the proposed model. Fig. 9 illustrates the comparison of loss values obtained after training the baseline model and proposed HemoNet on blood smear images.

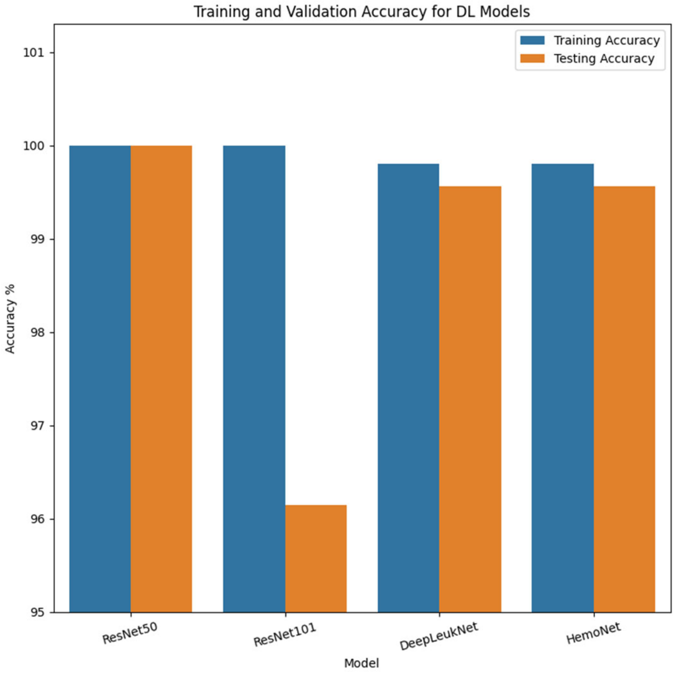


Fig. 8. Comparison of accuracy scores.

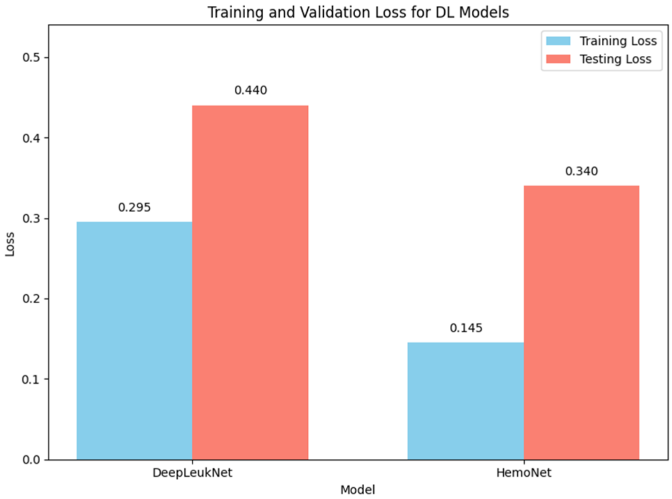


Fig. 9. Comparison of loss values of DeepLeukNet and HemoNet.

TABLE I. DETAILED COMPARISON BETWEEN HEMONET AND OTHER CNN ARCHITECTURES

Aspects	Models			
	ResNet50	ResNet101	DeepLeukNet	HemoNet
Number of layers	50	101	29	40
Attention Module	No	No	No	Yes
Average Training time per epoch	3 min 31 s	10 min 17 s	6 min 31 s	1min 20 s
Training Accuracy	100%	100%	99.85	99.80%
Validation Accuracy	100%	96.15%	99.56%	99.56%
Training Loss	0.195	0.135	0.295	0.145
Validation Loss	0.06	0.335	0.44	0.34

It was observed that the proposed model, demonstrated high validation accuracy and lower loss values—99.56% and 0.145 respectively. This superior performance of the model compared to other models can be attributed to the integration of CBAM, allowing HemoNet to prioritize informative features, effectively reducing computational time, and providing better results in the end. The comparison between HemoNet and other CNN architectures is given in Table I.

V. Conclusion

Blood cancer remains a life-threatening disease and its diagnosis has multiple steps, often leading to longer wait time, especially when symptoms of different types are similar. This study addresses the challenge by proposing a faster and effective deep learning-based approach to help medical professionals in classifying the type of cancer, upon analyzing the microscopic blood smear images. The effectiveness of the proposed HemoNet model which had CBAM module is evaluated and compared with the baseline model 'DeepLeukNet' and with the standard CNN architectures like ResNet50 and ResNet101. Evaluation metrics, including classification accuracy score, loss values, and train time per epoch, consistently demonstrated the impressive performance of the proposed model.

The base line model 'DeepLeukNet', resulted in high accuracy, but it took more train time per epoch and incurred higher loss values proposed HemoNet model. HemoNet achieved faster convergence and less training time. This significant improvement directs us to the CBAM mechanism computations. These computations refine the feature learning by applying channel attention mask and spatial attention mask. This allows the model to filter out irrelevant features, which accelerated the training time and enhancing the model's interpretability.

Overall, integration of attention module like Convolution Block Attention Module (CBAM), not only maintains high classification performance, but also effectively reduces the computational cost, training time, making it highly promising solution for real time automated blood cancer classification using blood smear images.

Despite notable results in research settings, DeepLeukNet and HemoNet face significant limitations in real-time clinical practice as collection of large, diverse and expertly annotated dataset is challenging due to patient privacy concerns. These models have to undergo rigorous and independent validation in order to be employed in clinical settings.

DeepLeukNet serves as a model specifically for identifying various types of leukemia through microscopic images of blood cells, whereas HemoNet denotes a broader category of hematological disease processes and is not a particular deep learning model aimed at cancer detection. However, the other deep learning models such as CNNs are generally used for classifying other types of cancers such as lung and liver cancer. In future, the proposed models along with other deep learning models can be explored in classifying other types of blood cancers such as Lymphoma.

Further going forward, the DeepLeukNet model can be employed for investigating cell morphology because of its outstanding feature extraction capabilities, while the HemoNet model can be utilized in drug discovery owing to its ability to manage complex features. While both models leverage deep learning to automate analysis, the DeepLeukNet can be used exclusively for Computer-Aided Diagnosis (CAD) of leukemia.

Ethical Approval

This article does not contain any studies with human participants or animals performed by any of the authors.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

All authors contributed to the study conception and design. Material preparation, methodology development, data collection and analysis were performed by Mageshwari V and Jana Uday Sagar. Ashok Kumar M has contributed to methodology development and writing review and editing. The first draft of the manuscript was written by Mageshwari V and all authors commented on previous versions of the manuscript. All authors had approved the final version.

References

[1] H. M. Kantarjian, N. J. Short, A. T. Fathi *et al.*, "Acute myeloid leukemia: Historical perspective and progress in research and therapy over 5 decades," *Clin. Lymphoma Myeloma Leuk.*, vol. 21, no. 9, pp. 580–597, 2021. doi: 10.1016/j.clml.2021.05.016

[2] M. A. Lichtman, "Obesity and the risk for a hematological malignancy: Leukemia, lymphoma, or myeloma," *The Oncologist*, vol. 15, no. 10, pp. 1083–1101, 2010. doi: 10.1634/theoncologist.2010-0206

[3] H. M. Kantarjian, C. D. DiNardo, T. M. Kadia *et al.*, "Acute myeloid leukemia management and research in 2025," *CA: A Cancer J. Clin.*, vol. 75, no. 1, pp. 46–67, 2025. doi: 10.3322/caac.21873

[4] K. K. Verma and N. Hussain, "A challenging case of acute lymphoblastic leukemia with bone marrow necrosis," *Med. Rep.*, vol. 12, 100208, 2025. doi: 10.1016/j.hmedic.2025.100208

[5] A. Rafae, F. V. Rhee, and S. A. Hadidi, "Perspectives on the treatment of multiple myeloma," *The Oncologist*, vol. 29, no. 3, pp. 200–212, 2024. doi: 10.1093/oncolo/oyad306

[6] W. Ni, B. Hu, C. Zheng *et al.*, "Automated analysis of acute myeloid leukemia minimal residual disease using a support vector machine," *Oncotarget*, vol. 7, no. 44, pp. 71915–71921, 2016.

[7] Z. King, S. R. Desai, D. A. Frank, and A. Shastri, "STAT signaling in the pathogenesis and therapy of acute myeloid leukemia and myelodysplastic syndromes," *Neoplasia*, vol. 61, 101137, 2025. doi: 10.1016/j.neo.2025.101137

[8] M. S. Abduh, "An overview of multiple myeloma: A monoclonal plasma cell malignancy's diagnosis, management, and treatment modalities," *Saudi J. Biol. Sci.*, vol. 31, no. 2, 103920, 2024. doi: 10.1016/j.sjbs.2023.103920

[9] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 7794–7803. doi: 10.1109/CVPR.2018.00813

[10] C. Yang, C. Zhang, X. Yang, and Y. Li, "Performance study of CBAM attention mechanism in convolutional neural networks at

different depths," in *Proc. IEEE 18th Conf. Ind. Electron. Appl. (ICIEA)*, 2023, pp. 1373–1377. doi: 10.1109/ICIEA58696.2023.10241832

[11] W. Gu and K. Sun, "AYOLOv5: Improved YOLOv5 based on attention mechanism for blood cell detection," *Biomed. Signal Process. Control*, vol. 88, 105034, 2024. doi: 10.1016/j.bspc.2023.105034

[12] D. Sarwinda, R. H. Paradisa, A. Bustamam, and P. Anggia, "Deep learning in image classification using Residual Network (ResNet) variants for detection of colorectal cancer," *Procedia Comput. Sci.*, vol. 179, pp. 423–431, 2021. doi: 10.1016/j.procs.2021.01.025

[13] K. S. Kumar, A. S. Radhamani, S. Sundaresan, and T. A. Kumar, "Medical image classification and manifold disease identification through convolutional neural networks: A research perspective," in *Proc. Handbook of Deep Learning in Biomedical Engineering and Health Informatics*, 2021, pp. 203–225.

[14] S. Tamuly, C. Jyotsna, and J. Amudha, "Deep learning model for image classification," in *Proc. Computational Vision and Bio-Inspired Computing*, 2020, pp. 312–320.

[15] A. Allegra, A. Tonacci, R. Sciaccotta *et al.*, "Machine learning and deep learning applications in multiple myeloma diagnosis, prognosis, and treatment selection," *Cancers*, vol. 14, no. 3, p. 606, 2022. doi: 10.3390/cancers14030606

[16] W. Yan, H. Shi, T. He *et al.*, "Employment of artificial intelligence based on routine laboratory results for the early diagnosis of multiple myeloma," *Front. Oncol.*, vol. 11, p. 933, 2021. doi: 10.3389/fonc.2021.608191

[17] K. H. Yu, A. L. Beam, and I. S. Kohane, "Artificial intelligence in healthcare," *Nat. Biomed. Eng.*, vol. 2, pp. 719–731, 2018. doi: 10.1038/s41551-018-0305-z

[18] M. Yurdakul, K. Uyar, Ş. Taşdemír, and Í. Atabaş, "ROPGCViT: A novel explainable vision transformer for retinopathy of prematurity diagnosis," *IEEE Access*, vol. 13, pp. 77064–77079, 2025. doi: 10.1109/ACCESS.2025.3564213

[19] L. Xu, G. Tetteh, J. Lipkova *et al.*, "Automated whole-body bone lesion detection for multiple myeloma on 68Ga-Pentixafor PET/CT imaging using deep learning methods," *Contrast Media Mol. Imaging*, vol. 2018, 2391925, 2018. doi: 10.1155/2018/2391925

[20] U. Saeed, K. Kumar, M. A. Khuhro *et al.*, "DeepLeukNet—A CNN based microscopy adaptation model for acute lymphoblastic leukemia classification," *Multimed. Tools Appl.*, vol. 83, pp. 21019–21043, 2024. doi: 10.1007/s11042-023-16191-2

[21] W. Rahman, M. G. G. Faruque, K. Roksana *et al.*, "Multiclass blood cancer classification using deep CNN with optimized features," *Array*, vol. 18, 100292, 2023. doi: 10.1016/j.array.2023.100292

[22] R. Zheng, H. Wen, F. Zhu, and W. Lan, "Attention-guided deep neural network with a multichannel architecture for lung nodule classification," *Heliyon*, vol. 10, no. 1, e23508, 2024. doi: 10.1016/j.heliyon.2023.e23508

[23] K. Dese, H. Raj, G. Ayana *et al.*, "Accurate machine-learning-based classification of leukemia from blood smear images," *Clin. Lymphoma Myeloma Leuk.*, vol. 21, no. 11, pp. 903–914, 2021. doi: 10.1016/j.clml.2021.06.025

[24] M. Shaheen, R. Khan, R. R. Biswal *et al.*, "Acute Myeloid Leukemia (AML) detection using AlexNet model," *Complexity*, vol. 2021, 6658192, 2021. doi: 10.1155/2021/6658192

[25] W. Xu, Y. L. Fu, and D. Zhu, "ResNet and its application to medical image processing: Research progress and challenges," *Comput. Methods Programs Biomed.*, vol. 240, 107660, 2023. doi: 10.1016/j.cmpb.2023.107660

[26] S. Das and K. Padmanaban, "Incremental learning for acute lymphoblastic leukemia classification based on hybrid deep learning using blood smear image," *Comput. Biol. Chem.*, vol. 118, 108456, 2025. doi: 10.1016/j.compbiolchem.2025.108456

[27] S. Ramaneswaran, K. Srinivasan, P. M. D. R. Vincent, and C. Y. Chang, "Hybrid inception v3 XGBoost model for acute lymphoblastic leukemia classification," *Comput. Math. Methods Med.*, vol. 2021, 9924565, 2021. doi: 10.1155/2021/9924565

[28] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 3–19. doi: 10.1007/978-3-030-01234-2_1