"Blood Cell Type Prediction"

Abstract—**Accurate classification of blood cell types is crucial for early detection of hematological disorders and improving diagnostic efficiency. This study evaluates the performance of deep learning architectures for automated blood cell classification using the BloodMNIST dataset, which contains 17,092 microscopic images of eight blood cell types. We systematically implement and compare CNNs, ResNet, VGG-16, and Inception-v3, along with attention mechanisms such as Squeeze-and-Excitation (SE) and Convolutional Block Attention Module (CBAM), training all models from scratch to ensure a fair evaluation. The study explores the impact of architectural depth, regularization techniques, and attention mechanisms on classification accuracy. Our results show that VGG-16 achieves the highest accuracy (96.43%), closely followed by ResNet (95.67%), demonstrating that depth, normalization, and attention mechanisms significantly enhance classification performance. These findings contribute to advancing automated blood cell analysis and provide a foundation for future applications in hematological diagnostics.**

Index Terms—**Deep Learning, Convolutional Neural Networks, BloodMNIST, Classification, CNN, VGG-16, RestNet, Inception**

I. Introduction

Advancements in deep learning have significantly improved the ability to analyze medical images, particularly in tasks such as blood cell classification. Identifying and categorizing blood cell types accurately is critical for diagnosing various hematological conditions, including anemia, infections, and leukemia. Manual microscopic examination of blood samples is time-consuming and requires expert interpretation, making automated classification a valuable tool in clinical practice.

The BloodMNIST dataset, a part of the **MedMNIST** collection, provides a well-structured benchmark for evaluating deep learning models in **biomedical image classification**. It consists of **17,092** labeled microscopic images representing **eight distinct blood cell types**, including *basophils, eosinophils, erythroblasts, lymphocytes, monocytes, neutrophils, immature granulocytes, and platelets*. The dataset is carefully curated, containing images exclusively from healthy individuals without infections or blood disorders. Figure 1 illustrates the different blood cell types included in the dataset.

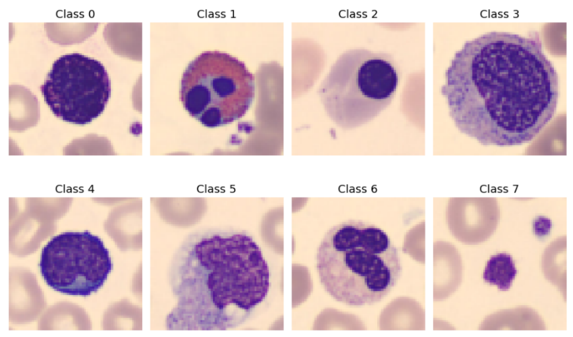


Fig. 1: Blood Cell Types

This study aims to **evaluate and compare multiple deep learning architectures** to determine the most effective model for **blood cell classification**. We systematically implement and analyze **Convolutional Neural Networks (CNNs)**, **ResNet**, **VGG-16**, and **Inception-v3**, alongside attention mechanisms such as Squeeze-and-Excitation (SE) and Convolutional Block Attention Module (CBAM). Unlike previous studies that often rely on transfer learning, all models in this study are **trained from scratch** to ensure a fair comparison.

By assessing these architectures, we seek to **identify the key factors** influencing classification accuracy, including network depth, regularization strategies, and **the role of attention mechanisms**. The results will provide insights into the most effective model configurations, guiding future research in automated blood cell analysis and potential real-world medical applications.

We acknowledge that in resource-constrained environments, it is essential to develop lightweight models that maintain high accuracy while minimizing computational cost. Therefore, this study aims to bridge this gap by addressing the following key research questions:

Q1: Can a lightweight model with a low number of parameters and computational cost still achieve competitive performance in blood cell classification?

Q2: Can incorporating soft attention mechanisms, such as Squeeze-and-Excitation (SE) blocks, enhance the performance of these lightweight models?

Q3: How do more complex architectures, like ResNet and Inception, compare to lightweight models in terms of the trade-offs between model complexity, accuracy, and computational efficiency?

The remainder of this paper is structured as follows: *Section II* reviews related work and existing approaches. *Sections III and IV* describe the methodology, dataset preprocessing, and architectural details of each model. *Section V* presents the experimental results and identifies the best-performing architecture. Finally, *Section VI* summarizes key findings, and *Section VII* discusses implications and future research directions.

II. Related Work

This section reviews **key developments** in the classification of blood cell types. In recent years there have been studies related to the development of the machine learning techniques for **medical image classification**.

In [1] , the authors conducted a study on the classification of five types of normal peripheral blood (PB) cells using **two distinct approaches**. The first approach followed a traditional pipeline consisting of image segmentation, feature extraction, and classification. In this method, they employed a **support vector machine (SVM)** as the classifier, while **principal component analysis (PCA)** was utilized for feature selection to reduce dimensionality and improve efficiency. The second approach leveraged deep learning by utilizing **a convolutional neural network (CNN)**, where the entire image was directly fed into the model without requiring manual feature extraction. The results demonstrated that **the CNN-based approach** outperformed the traditional method, achieving an **accuracy of 85%**, highlighting the effectiveness of deep learning in blood cell classification.

Other studies [2] explored the use of convolutional neural networks (CNNs) for the classification of five types of peripheral blood (PB) cells. Their study involved fine-tuning two well-established deep learning architectures, **AlexNet** and **LeNet-5**, to optimize their performance on this specific classification task. The fine-tuned AlexNet model achieved an accuracy of 91.2%, while the LeNet-5 model obtained a slightly lower accuracy of 84.9%. In addition to these experiments, the authors also developed and trained a custom CNN model using a dataset consisting of 2,551 images. Their proposed model demonstrated superior performance, achieving a maximum **testing accuracy of 96%**, showcasing the potential of deep learning techniques for highly accurate PB cell classification.

The usage of **more complex architectures** such as **VGG-16** and **Inception-v3** have shown a higher performance when it comes to the classification task for images. In [3] the authors propose a classification framework utilizing VGG16 and Inception-v3 to distinguish between eight types of peripheral blood cells. By fine-tuning a state-of-the-art architecture, they trained **an end-to-end classifier** using BloodMnist dataset. Their model (VGG-16) achieved an **overall classification accuracy of 96.2%**, with excellent precision, sensitivity, and specificity.

RestNet-50 is another architecture proposed for image classification, in [4] proposes an automated approach for malaria diagnosis using **transfer learning with the ResNet-50 architecture**. The authors fine-tune a pre-trained ResNet-50 model on a dataset of malaria cell images, classifying cells as either parasitized (infected) or uninfected. This method addresses the limitations of manual diagnosis, which is time-consuming and error-prone. The fine-tuned model achieves high accuracy, precision, recall, and F1-score, outperforming traditional machine learning methods and other CNN architectures. The study demonstrates that transfer learning with ResNet-50 is a robust, efficient, and scalable solution for malaria cell-image classification, with potential applications in **resource-limited settings** and other medical image analysis tasks. Also in [5] ResNet-50 **outperformed** other competitor algorithms in terms of **average accuracy (96.83%)** and **average F1-score (96.82%)** for microscopic image classification.

The work presented in this paper aims to compare architectures designed **to reduce complexity** while **optimizing both computational efficiency and performance accuracy** in blood cell type prediction.

III. Processing Pipeline

This and the following sections outline **the end-to-end processing pipeline** for **blood cell classification** using deep learning models. The pipeline consists of **multiple stages**: data acquisition, preprocessing, model architecture design, and attention mechanism integration. A structured and efficient processing pipeline ensures that the models receive high-quality input data, enabling optimal learning and classification performance.

A. Model Architectures

To evaluate the effectiveness of different deep learning models in blood cell classification, we implemented multiple architectures, each designed **to extract hierarchical features** and **improve classification performance**.

1) Convolutional Neural Networks (CNNs): CNNs form **the foundation** of modern image classification models. They employ **convolutional layers** to extract spatial features from images, followed by **pooling layers** that reduce dimensionality while preserving essential information. **Fully connected layers** at the end of the network transform extracted features into classification outputs. For this study, we implemented a standard CNN architecture with:

- Multiple convolutional layers with **ReLU activation**.
- Max-pooling layers to downsample feature maps.
- **Dropout layers** to prevent overfitting.
- Fully connected layers for final classification.

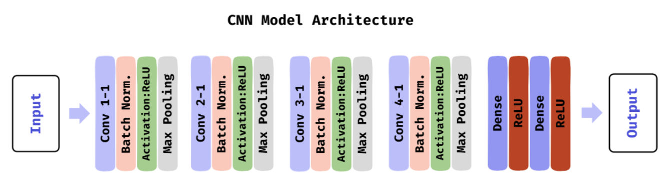


Fig. 2: CNN Architecture

2) Residual Networks (ResNet): ResNet introduces **skip connections** to enable deeper architectures while mitigating the **vanishing gradient problem**. These identity mappings allow gradients to propagate effectively, facilitating stable training of deep networks. In this study, we implemented a **custom ResNet-18-inspired model**, consisting of:

- 18 layers with **residual blocks** for feature learning.
- **Batch normalization** to improve gradient flow.
- Global average pooling instead of fully connected layers to reduce model complexity.

This lightweight ResNet variant enhances performance while maintaining computational efficiency, making it well-suited for blood cell classification.

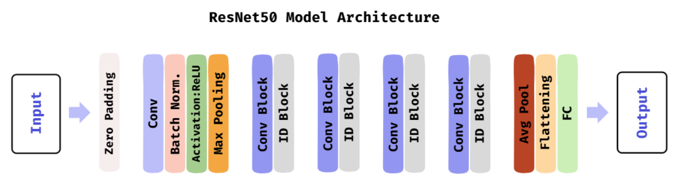


Fig. 3: ResNet Architecture

3) Inception-v3: Inception-v3 is optimized for efficient feature extraction by using **parallel convolutional filters** of **varying sizes**. This architecture enables the model to capture fine-grained and high-level representations simultaneously. Key components include:

- Factorized convolutions that reduce computational cost.
- **Asymmetric kernels** that enhance spatial feature extraction.
- **Auxiliary classifiers** that improve gradient propagation.

By leveraging **multi-scale feature learning**, Inception-v3 provides strong performance while being computationally efficient.

4) VGG-16: VGG-16 is a deep CNN architecture composed of **13 convolutional layers** followed by **three fully connected layers**. It is known for its structured, **uniform kernel size** and ability to capture hierarchical features effectively. The architecture consists of:

- Stacked 3×3 convolutional layers for fine-detail extraction.
- Max-pooling layers to progressively reduce spatial dimensions.
- Fully connected layers for classification.

VGG-16 provides a balance between simplicity and high accuracy, making it a strong candidate for blood cell classification.

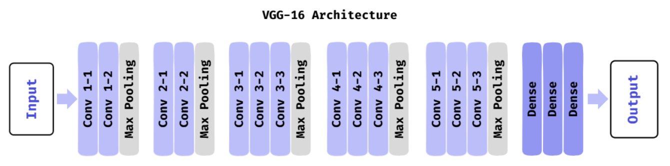


Fig. 4: VGG-16 Architecture

B. Attention Mechanisms

To further enhance classification performance, we integrated **attention mechanisms** into CNN and VGG-16 architectures. These mechanisms allow models **to focus on the most relevant features**, improving decision-making.

1) Squeeze-and-Excitation (SE) Networks: SE Networks apply **channel-wise attention** by dynamically recalibrating feature maps. The mechanism consists of:

- **Squeeze Operation:** Global average pooling compresses spatial information.
- **Excitation Operation:** A multi-layer perceptron (MLP) learns channel-wise feature importance.
- **Reweighting:** The recalibrated weights enhance the most important feature maps.

By applying SE blocks, the network emphasizes critical feature representations, improving classification accuracy.

2) Convolutional Block Attention Module (CBAM): CBAM extends SE by incorporating **both channel and spatial attention**, allowing the model to refine feature importance holistically. It follows a two-step process:

1) **Channel Attention:** Similar to SE, it weights feature maps using global average and max pooling.
2) **Spatial Attention:** A convolutional layer processes pooled feature maps to highlight important spatial regions.

While SE focuses **solely on channel-wise importance**, CBAM refines feature selection at **both channel and spatial levels**, making it more effective for complex images.

C. Summary of Processing Pipeline

The **complete processing pipeline** consists of:

1) **Data Loading and Preprocessing:** Normalization, resizing, augmentation, and train-test splitting.
2) **Model Selection:** Implementation of CNN, ResNet-50, Inception v3, and VGG-16.
3) **Regularization Techniques:** Dropout, batch normalization, and L2 weight regularization.
4) **Attention Mechanism Integration:** SE in CNN, and both SE and CBAM in VGG-16.
5) **Optimization:** Models trained with Adam and SGD optimizers, using categorical cross-entropy loss.

IV. Signals and Features

In this paper, we work with the BloodMNIST dataset, which consists of **17,092 digital images** of **8 different blood cell types**. Each image is **in RGB color space** with a resolution of **64x64 pixels**. During the preprocessing phase, several steps were undertaken to standardize and augment the dataset:

- **Normalization:** Pixel values were scaled from their original range (typically 0 to 255) to a range of 0 to 1 by dividing by 255.0.
- **Data Augmentation:** To enhance the model's robustness and generalization capabilities, various augmentation techniques were applied using Keras's **ImageDataGenerator**. Since the dataset has class imbalances in the training, validation, and test sets, applying **data augmentation** to the training set is a well-known method in image classification tasks. The augmentations included:
  - *Rotation:* Images were randomly rotated up to 90 degrees.
  - *Horizontal Flip:* Images were flipped horizontally.
  - *Vertical Flip:* Images were flipped vertically.

  However, after conducting initial experiments, we observed that training with augmented data led to **a decline in classification accuracy**. Since blood cell images have specific morphological structures, excessive transformations such as flipping and rotation introduced distortions that **did not align with biological reality**. As a result, we decided not to train our models with augmented data to ensure that they learn from authentic, unaltered representations of blood cells.
- **Filling missing pixels:** The "nearest" fill mode replaces missing pixels, created during transformations like rotation or flipping, with the nearest neighboring pixel value to maintain image consistency.

Train, Validation, and Test Split: The process of splitting a dataset into training, validation, and test sets is a crucial step in data preprocessing. In the case of the BloodMNIST dataset, it is already pre-split, eliminating the need for manual partitioning. The dataset split is presented below.

Set	Total Images
Train	11959
Validation	1712
Test	3421

TABLE 1: Train/Validation/Test Set

In Figure 5 is shown the distribution plots the training set for each of the classes of blood cell type before and after data augmentation.

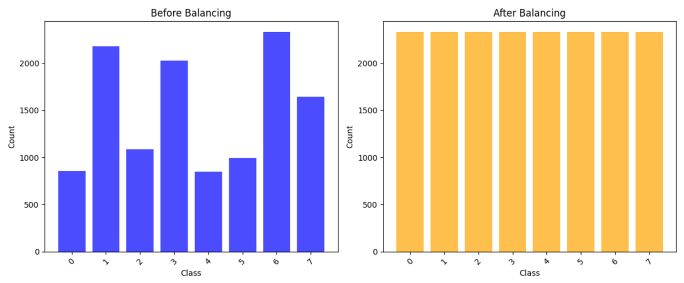


Fig. 5: Data Augmentation

V. Learning Framework

In this section, we describe **the learning strategy** and **methodology** we employed to develop an optimized deep learning model for BloodMNIST cell type classification. Our framework follows an **iterative refinement process**, where we systematically enhance model architecture, apply regularization techniques, and optimize training strategies to achieve higher classification performance while maintaining computational efficiency.

A. Model Development Strategy

We designed our learning framework through a structured, **step-by-step approach**, starting with a simple **Convolutional Neural Network (CNN) baseline model** and progressively increasing its complexity. This iterative approach allowed us to analyze how architectural changes impact performance. In addition to the custom CNN, we **implemented ResNet, Inception-v3, and VGG-16 from scratch** to compare their effectiveness in feature extraction and classification.

1) Baseline Model Implementation: Our **baseline model** was a straightforward CNN consisting of **two convolutional layers**, each followed by **ReLU activation** and **max pooling**. These layers were responsible for extracting hierarchical spatial features from the microscopic blood cell images. After the feature extraction phase, we included **fully connected layers** to process the learned representations and map them to the eight blood cell classes. This initial model served as **a foundational reference**, helping us evaluate how additional architectural modifications influence classification accuracy.

2) Architectural Refinements and Optimization: To systematically improve feature extraction, we experimented with **increasing the depth** of our CNN by adding more convolutional layers, progressing from **two to four layers**. The goal was to determine an **optimal network depth** where additional layers enhance learning **without excessive computational cost or overfitting**.

Following the structural refinements, we applied **regularization techniques** to prevent overfitting and improve generalization:

- **Dropout:** We implemented dropout at different rates (ranging from **0.3 to 0.5** in the fully connected layers. This technique **randomly deactivates neurons** during training, forcing the network to learn more robust features rather than memorizing specific patterns.
- **Batch Normalization:** To stabilize training, we integrated **batch normalization** across convolutional layers. This method normalizes activations across mini-batches, reducing **internal covariate shift** and allowing for smoother and faster convergence.
- **L2 Regularization:** We applied **L2 weight regularization** to convolutional layers to penalize large weight values, encouraging the model to maintain smaller, more stable weights. We experimented with different values for the regularization parameter (**$\lambda = 0.01$ to 0.0001**) to find the optimal balance between reducing overfitting and preserving essential feature representations.

Through this systematic optimization process, we developed **multiple variations** of our CNN architecture, allowing us to compare their effectiveness against **ResNet, Inception-v3, and VGG-16**. Each modification was evaluated based on its impact on accuracy, computational efficiency, and ability to generalize well across the BloodMNIST dataset.

3) Attention Mechanisms: To enhance feature selection and improve classification performance, we integrated **attention mechanisms** into our CNN-based architectures. Specifically, we implemented **Squeeze-and-Excitation (SE) blocks**, which adaptively **recalibrate feature maps** to emphasize the most relevant channels. Our approach involved applying SE blocks at multiple convolutional layers, allowing the network to dynamically prioritize essential features. **CBAM** was used with VGG-16 architecture to enhance feature representation **by sequentially applying channel attention** (to focus on important feature maps) and spatial attention (to highlight significant spatial regions).

- **Squeeze-and-Excitation (SE) Networks:** SE blocks consist of three main steps:
  1) **Squeeze:** Global Average Pooling (GAP) is applied to each channel to aggregate spatial information.
  2) **Excitation:** Fully connected layers process the pooled features using a bottleneck layer with ReLU activation, followed by a sigmoid activation to generate attention weights.
  3) **Scale:** The learned attention weights are multiplied with the original feature map to enhance relevant channels.
- **Improved CNN with SE Attention:** We integrated SE blocks after every convolutional stage in our CNN model. The final architecture consists of:
  - Three convolutional blocks with **Batch Normalization** and **ReLU activation**.
  - **Max-pooling layers** for downsampling.
  - SE blocks at each stage to apply channel-wise attention.
  - Fully connected layers for classification.
- **L2 Regularization with SE Blocks:** In a further enhancement, we incorporated **L2 weight regularization** ($\lambda = 0.0001$) into the fully connected layers of the SE module. This approach prevents overfitting by penalizing large weight values, stabilizing feature learning while maintaining the benefits of attention mechanisms.
  By applying **SE attention blocks** with **L2 regularization**, we improved the network's ability to focus on informative regions while reducing overfitting. These modifications significantly enhanced classification accuracy, as later analyzed in the *Results* section.
- **Convolutional Block Attention Module (CBAM):** consists on:
  - **Channel Attention:** Global Average Pooling (GAP) and Global Max Pooling (GMP) are applied to the input feature map to capture global context, followed by a shared fully connected network to learn channel-wise attention weights.
  - **Spatial Attention:** A convolutional layer is applied to the concatenated results of GAP and GMP to generate a spatial attention map, which is then used to focus on important spatial regions in the feature map by element-wise multiplication.

4) Deep Architectures: Beyond CNN optimizations, we implemented and trained **deep architectures from scratch** to evaluate their effectiveness in feature extraction and classification. Our focus was on **ResNet, Inception-v3 and VGG-16**, both widely recognized for their ability to capture hierarchical representations in image data.

- **Residual Networks (ResNet):** ResNet introduces **residual connections** that allow gradient information to flow more effectively, mitigating the **vanishing gradient problem** in deep networks. Our implementation of ResNet includes:
  - **ResNet blocks** consisting of two convolutional layers, each followed by **Batch Normalization** and **ReLU activation**.
  - **Shortcut (skip) connections** that enable identity mappings when possible, ensuring stable gradient propagation.
  - **Downsampling blocks** that adjust feature map dimensions using a **stride of 2** in select layers.

  - A 7×7 **initial convolutional layer**, followed by **max pooling** to reduce spatial dimensions early in the network.
  - **Global Average Pooling (GAP)** before the final dense classification layer, reducing parameter overhead while preserving important features.

  Additionally, we optimized training using the **SGD optimizer with Cosine Decay Learning Rate Scheduling**, enabling gradual adaptation of learning rates. We also incorporated **early stopping** to prevent overfitting while ensuring model convergence.
- **Inception-v3:** Inception-v3 is an advanced deep learning architecture designed to efficiently extract features while maintaining computational efficiency. Our implementation of Inception-v3 was built from scratch, incorporating several optimizations to enhance blood cell classification performance:
  - **Stem Block:** The model begins with a sequence of convolutional layers with increasing filter sizes $(32, 64, 96)$, each followed by **Batch Normalization** and **ReLU activation**. This initial stage reduces spatial dimensions while capturing fundamental low-level features.
  - **Inception Modules:** We employed stacked **Inception modules** consisting of:
    - **Factorized convolutions** that break down large kernels (5×5) into smaller ones (3×3) to enhance computational efficiency.
    - **Asymmetric convolutions**, further optimizing the processing of spatial features.
    - **Parallel convolutional branches** with different kernel sizes $(1 \times 1, 3 \times 3, 5 \times 5)$, allowing multi-scale feature extraction.
    - **Depthwise separable convolutions**, reducing parameter count while preserving representational power.
  - **Reduction Blocks:** To downsample spatial dimensions, we incorporated **reduction modules**, which use:
    - **Max-pooling layers** to aggressively reduce spatial size while retaining important features.
    - 3×3 **stride-2 convolutions**, allowing efficient spatial reduction with minimal information loss.
    - **Concatenation layers**, combining different branches to improve feature diversity.
  - **Global Feature Aggregation:** The final feature map undergoes **Global Average Pooling (GAP)**, which reduces the number of parameters while maintaining high discriminative power.
  - **Classification Layer:** The final dense layer consists of an **8-class softmax activation**, predicting the probability distribution for each blood cell type.
- **VGG-16:** We implemented a 16-layer deep learning architecture, adjusting the number of filters in each block to optimize performance. Given our 64x64 input size, we started with **32 filters** in the first block and progressively increased the number, reaching **256 filters** in the fifth block. This approach ensures that the network captures both low-level and high-level features effectively.
  However, due to the complexity introduced by multiple layers, we focused on understanding how each block contributes to overall performance. To enhance model efficiency and prevent overfitting, we applied several optimization techniques. **Dropout layers** were added after the dense layers to reduce overfitting by randomly deactivating neurons during training, improving generalization. Additionally, we experimented with removing some layers from the last two blocks, simplifying the architecture while maintaining its predictive power. Furthermore, we explored Attention Mechanisms such as **SE and CBAM**, which dynamically prioritize important features within an image, allowing the model to focus on relevant regions.
  This technique has proven effective in various image classification tasks, enhancing interpretability and overall accuracy. By combining these strategies, we aimed to refine the architecture, ensuring an optimal balance between complexity, performance, and computational efficiency.

B. *Optimization and Training Strategy*

To ensure robust model training, we employed the **Adam optimizer** with an initial learning rate of 0.001, along with **learning rate scheduling** for improved convergence. In some cases, we switched to **Stochastic Gradient Descent (SGD)** to enhance generalization and prevent overfitting in later training stages.

All models were trained using **categorical cross-entropy loss**, and performance was evaluated based on:

- Accuracy
- Weighted F1-score
- Weighted Precision and Recall
- Confusion Matrix

VI. Results

This section presents **the performance evaluation** of different deep learning architectures for blood cell type classification. The primary objective is to identify the most effective model **based on key performance metrics**. We systematically optimized model configurations, starting with a baseline CNN and progressively incorporating additional layers, regularization techniques, attention mechanisms, and deeper architectures.

A. *Effect of Convolutional Layers*

To determine **the optimal network depth**, we experimented with 2-layer, 3-layer, and 4-layer CNN architectures.

TABLE 2: Comparison of Models Based on Convolutional Layer Configurations

Model Configuration	Test Accuracy
Baseline CNN (Two Layers)	88.95%
Basic CNN (Three Layers)	88.07%
Basic CNN (Four Layers)	88.78%

Discussion: As shown in Table 2, the baseline 2-layer CNN achieved the highest test accuracy (88.95%). Adding a third layer resulted in a slight drop (88.07%), but the learning curve improved, suggesting better feature extraction. The 4-layer CNN showed a marginal improvement (88.78%), demonstrating the benefit of deeper networks. However, the accuracies remained very similar across configurations, indicating that excessive depth alone did not yield significant gains, highlighting the need for additional techniques such as regularization.

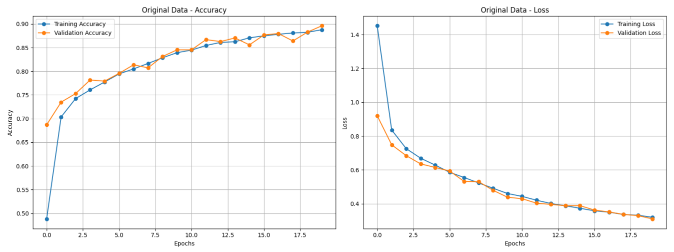


Fig. 6: Training and Validation Performance of 4-Layer CNN on BloodMNIST

B. *Impact of Regularization*

After identifying the 4-layer CNN as the best base architecture, we applied various regularization strategies, including dropout, batch normalization, and L2 weight regularization.

TABLE 3: Comparison of Regularization Techniques

Regularization Strategy	Test Accuracy
CNN with Dropout Rate (0.4)	91.9%
CNN with Dropout Rate (0.5)	90.0%
CNN with L2 Regularization (0.01)	90.47%
CNN with L2 Regularization (0.001)	86.64%
CNN with L2 Regularization (0.0001)	91.11%
CNN with Batch Normalization	93.92%
CNN with Batch Norm. + Dropout (0.4)	95.59%

Discussion: Regularization significantly improved performance. As shown in Table 3, dropout at 0.4 boosted accuracy (91.9%) compared to 0.5 (90%), indicating that excessive dropout removed too much information. L2 regularization was most effective at $\lambda = 0.0001$ (91.11%), while stronger penalties ($\lambda = 0.01$) reduced accuracy (90.47%). The best performance (95.59%) was achieved by combining batch normalization and dropout (0.4), demonstrating the stabilizing effect of normalization on training.

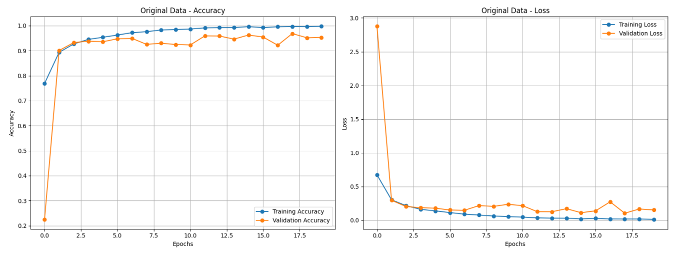


Fig. 7: Training and Validation Performance of 4-Layer CNN with Batch Normalization (dropout: 0.4)

C. *Effect of Attention Mechanisms*

To further refine feature extraction, we integrated attention mechanisms, Squeeze-and-Excitation (SE) and Convolutional Block Attention Module (CBAM).

TABLE 4: Comparison of Attention Mechanisms

Model Configuration	Test Accuracy
CNN + Attention Mechanism (SE)	95.03%
VGG-16 + Attention Mechanism (SE)	91.70%
VGG-16 + Attention Mechanism (CBAM)	93.31%

Discussion: Attention mechanisms significantly enhanced accuracy. As shown in Table 4, the CNN with SE achieved 95.03%, demonstrating the effectiveness of channel-wise attention. VGG-16 with SE and CBAM both achieved (92% - 93%), suggesting that attention-based feature weighting is particularly effective in deep architectures.

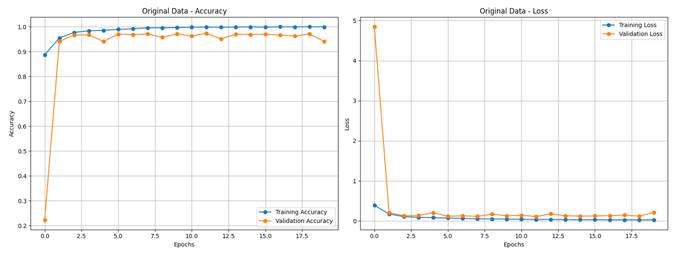


Fig. 8: Training and Validation Performance of 4-Layer CNN with Attention (SE) on BloodMNIST

D. *Comparison of Deep Architectures*

Finally, we implemented and evaluated deeper architectures to assess their performance against our optimized CNN.

TABLE 5: Comparison of Deep Architectures

Model Configuration	Test Accuracy
Custom CNN with Residual Connections	92.43%
ResNet	95.67%
Inception-v3	88.02%
VGG-16	96.43%

Discussion: Among deep architectures, as shown in Table 5, VGG-16 achieved the highest accuracy (96.43%), closely followed by ResNet (95.67%), likely due to their deep structures and effective feature extraction. Inception-v3 struggled (88.02%), possibly due to its complex factorized convolutions being less suited for BloodMNIST. The custom CNN with residual connections performed competitively (92.43%), validating the effectiveness of residual learning in improving model performance.

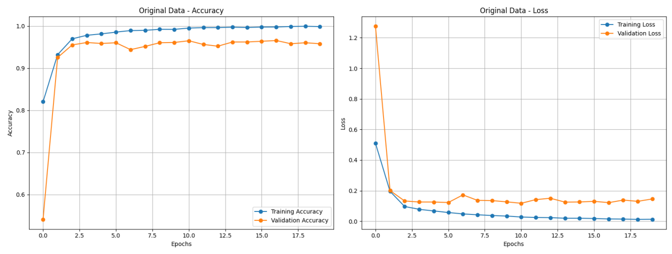


Fig. 9: Training and Validation Performance of ResNet on BloodMNIST

E. Identifying the Best Model

Identifying the best model depends not only in the accuracy of the model, but also some other metrics such as: memory occupation and execution time. Based on these three metrics in Figure 10 is shown the performance of each architectures.

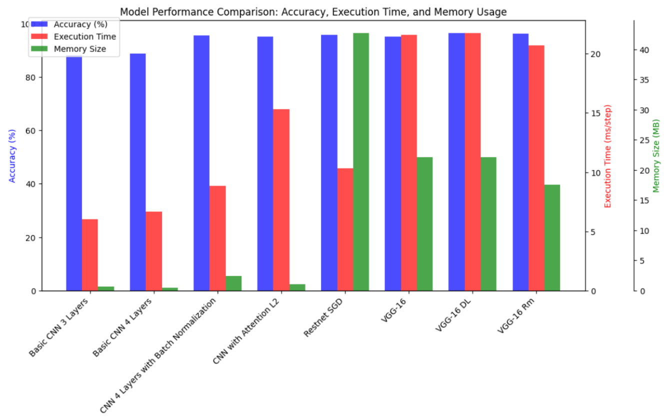
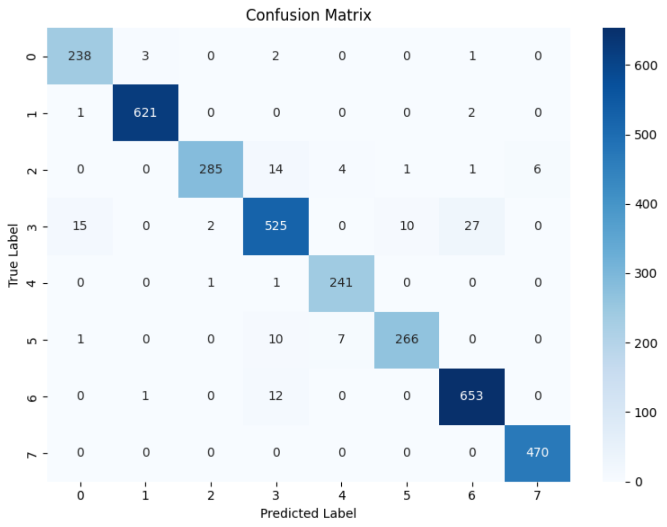


Fig. 10: Comparison of Architecture

- **Accuracy:** The highest accuracy (96.43%) is achieved by VGG-16 with dropout layers, followed closely by VGG-16 removing layers from last blocks (96.20%). Basic CNN models show significantly lower accuracy (88.07% - 88.78%), making them less effective for complex tasks.
- **Execution Time:** Basic CNN models perform the fastest (6.05 - 6.65 ms/step), making them efficient but less accurate. VGG-16 and its variants (dropout later and removing layers) take the longest (20.69 - 21.75 ms/step), showing higher computational demands. While CNN with Attention L2 has a particularly high execution time (15.30 ms/step) despite a moderate accuracy improvement.
- **Memory Usage:** ResNet has the highest memory consumption (42.71 MB), indicating a significant resource demand. VGG-16 and its variants also consume substantial memory (22 MB), making them suitable for powerful hardware. Basic CNN models are extremely lightweight (0.51 - 0.67 MB), making them ideal for low-memory environments.

To further analyze the classification performance of VGG-16 with dropout layers (DL), we present the confusion matrix in Figure 11. This visualization provides insight into how well the model differentiates between blood cell types, highlighting any misclassification trends.



True \ Predicted	0	1	2	3	4	5	6	7
0	238	3	0	2	0	0	1	0
1	1	621	0	0	0	0	2	0
2	0	0	285	14	4	1	1	6
3	15	0	2	525	0	10	27	0
4	0	0	1	1	241	0	0	0
5	1	0	0	10	7	266	0	0
6	0	1	0	12	0	0	653	0
7	0	0	0	0	0	0	0	470

Fig. 11: Confusion matrix for the VGG-16 with dropout layers, showing classification performance across the eight blood cell types.

The results highlight **a clear trade-off** between accuracy, execution time, and memory consumption across different models. VGG-16 with dropout layers delivers the highest accuracy (96.43%) but demands significant computational resources, making it suitable for **high-performance environments**. CNN with four layers and batch normalization strikes a good balance between accuracy (95.59%) and efficiency, making it a viable option for resource-constrained scenarios.

For applications requiring speed and low memory usage, Basic CNN models are preferable, although their accuracy (around 88%) may not be sufficient for complex tasks. ResNet achieves high accuracy as you can also see in Figure 12, but is highly memory-intensive, which could be a limiting factor for deployment on low-resource hardware.

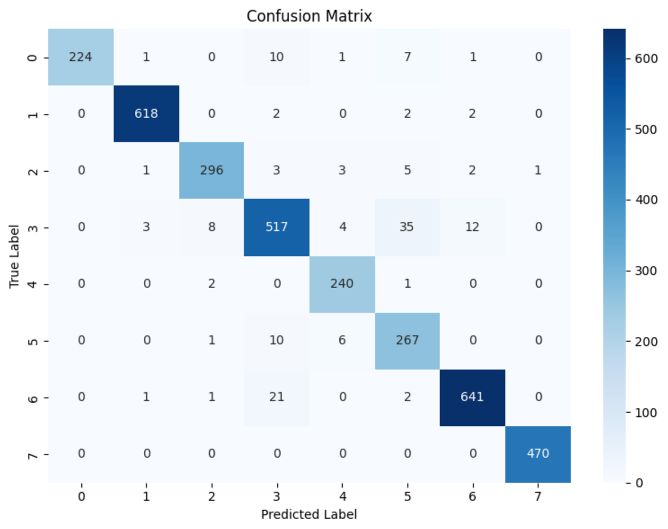


Fig. 12: Confusion matrix for the ResNet Model, showing classification performance across the eight blood cell types.

VII. Concluding Remarks

In summary, this paper presents a framework for comparing different machine learning algorithms for image classification. The results highlight that the choice of the best algorithm depends **not only on prediction accuracy** but also on **memory usage** and **execution time**. The selection of an appropriate model should align with the specific application requirements - whether prioritizing accuracy, speed, or computational efficiency. Balancing complexity with simplicity while fine-tuning hyperparameters is a challenging task, requiring a deep understanding of how each model makes predictions and the ability to interpret its decision-making process effectively.

Using **a small dataset**, such as BloodMNIST, poses challenges for more complex models, as they may struggle to generalize effectively **due to limited training data**. Deep architectures with numerous layers often require large datasets to fully capture patterns and avoid overfitting. As a result, **choosing the right model architecture** becomes crucial **for achieving optimal performance**.

References

[1] M. Habibzadeh, A. Krzyzak, and T. Fevens, "White Blood Cell Differential Counts Using Convolutional Neural Networks for Low Resolution Images," in *Artificial Intelligence and Soft Computing*, (Springer).

[2] A. Shahin, Y. Guo, K. Amin, and A. A. Sharawi, "White blood cells identification system based on convolutional deep neural learning networks," *Computer Methods and Programs in Biomedicine*, vol. 168, pp. 69–80, Jan. 2019.

[3] A. Acevedo, S. Alférez, A. Merino, L. Puigví, and J. Rodellar, "Recognition of peripheral blood cell images using convolutional neural networks," *Data Brief*, vol. 30, Aug. 2019.

[4] A. S. B. Reddy and D. S. Juliet, "Transfer Learning with ResNet-50 for Malaria Cell-Image Classification," *International Conference on Communication and Signal Processing*, Apr. 2019.

[5] R. Liu, W. Dai, T. Wu, M. Wang, S. Wan, and J. Liu, "AIMIC: Deep Learning for Microscopic Image Classification," *Data Brief*, vol. 226, Nov. 2022.