Title: Developing Deep Learning Methods for Peripheral Blood Cell Classification

Team Members: Harshi Saha, Diana Hou, Cece Jen, Tina Tu, Clementine Chen

Code Files: https://github.com/harshi-saha/blood-medmnist

Abstract: Peripheral blood cell classification is relevant to diagnosing infections and tracking disease progression, but manual labeling is time and labor intensive. Deep learning can be used to automate classification, and here we evaluate ResNet-50 and ResNet-101 on the BloodMNIST dataset at 64×64, 128×128, and 224×224 pixel image resolutions. Hyperparameter tuning was also conducted for dropout rate, batch size, learning rate, and optimizer selection, and indicated that simpler configurations, such as smaller batch sizes and the exclusion of attention layers or class-weighted loss functions, achieved superior performance, particularly with deeper models (ResNet-101) and higher resolution (224 pixels). The top performing model used ResNet-101 (224 resolution input), batch size 16, dropout 0.3, and Adam optimizer (learning rate = 1e-4). However, classification of morphologically ambiguous classes, that is, monocytes and immature granulocytes, remained an ongoing challenge across all model configurations. Grad-CAM visualizations confirmed that model decisions were based on biologically relevant features, and external validation demonstrated generalizability, though low performance of ambiguous classes remained. These findings support using ResNet architectures for accurate, interpretable peripheral blood cell image classification, though there is room for further improvement.

Introduction:

Background: Identifying peripheral blood cell types is relevant to diagnosing infections, monitoring immune responses, and tracking disease progression. Traditional methods of manual assessment using unique morphological characteristics of the different cells are labor-intensive and prone to error[1,2]. To automate and improve the accuracy of classification, deep learning has been applied to these efforts and has shown promise. Custom CNNs from Anand et al. have achieved accuracy rates up to ~97.98% by using sequential layers[4], however, such models require extensive hyperparameter tuning and architecture optimization, offering limited advantages over fine-tuned pre-trained models. In addition, pre-trained architectures like VGG-16 and InceptionV3 have achieved accuracies~96% in a previous study by Acevedo et al.[1] VGG-16 utilizes multiple ReLU-activated convolutional layers but suffers from the vanishing gradient problem and relatively shallow depth, while InceptionV3 addresses multi-scale feature extraction but is computationally demanding due to its complex architecture and extensive training requirements. According to Habibzadeh et al., Inception architectures consistently perform worse than ResNet models.[8] ResNet offer significant advantages by enabling deeper architectures without encountering vanishing gradient issues through the use of residual connections that allow for effective gradient flow during backpropagation.[5, 7] ResNet models are also easy to train using high-level APIs like Keras and are compatible with interpretability tools such as Grad-CAM, making them suitable for medical imaging tasks where interpretability and high performance are desirable.

Objectives: Our primary objective is to develop and evaluate ResNet-based CNN models for accurate classification of different types of normal peripheral blood cells. Using ResNet architectures will maintain model depth without degradation through the use of residual connections, while enhancing feature localization and interpretability using tools like Grad-CAM to highlight the most important visual features used for classification[9]. We will ensure model robustness and generalizability through regularization, data augmentation, and evaluation across various image resolutions. Additionally, we will assess performance across different architectures and hyperparameter settings, and evaluate applicability to other datasets.

Methods:

How the dataset was collected/obtained:

The main dataset in this project consists of 17,092 RGB images of normal peripheral blood cells: neutrophils, eosinophils, basophils, lymphocytes, monocytes, immature granulocytes, erythroblasts, and platelets[1,2,3]. The cells are represented by 1,214-3,329 images each, and manually labeled by clinical pathologists at the original data source, Hospital Clinic of Barcelona[1,2,3]. The images are center cropped from the original to 360×363 to 200 pixels and resized to 28, 64, 128, and 224 pixels, contained in the MedMNIST python package through which this dataset will be accessed[1,2,3]. The MedMNIST package provides the data through the BloodMNIST module, including train:validation:test in a 7:1:2 ratio, and in each split, the cell types are represented in proportions of: basophil 0.07, eosinophil 0.18, erythroblast 0.09, immature granulocyte 0.17, lymphocyte 0.07, monocyte 0.08, neutrophil 0.19, and platelets 0.14[1,2,3]. The website describing the dataset includes model benchmarks with automated hyperparameter tuning, with those relevant to this project being ResNet-50 28 resolution (accuracy 0.956, AUC 0.997) and ResNet-50 224 resolution (accuracy 0.950, AUC 0.997)[3]. The external dataset to evaluate model robustness and generalizability was from Raabin health database. It includes 4339 images of 5 cell types: basophils, eosinophils, lymphocytes, monocytes, and neutrophils[20] labeled by hematologists, with 89 basophils, 322 eosinophils, 1034 lymphocytes, 234 monocytes, and 2660 neutrophils.

The deep learning approaches used:

To classify the 8 cell types, we fitted ResNet models with various layers and input image resolutions, using Keras throughout this project. ResNet-18 was excluded since it is not available in Keras, and 28 pixel resolution was omitted since Keras minimum requirement is 32. Our four initial models are ResNet-50 and ResNet-101 with input resolutions 64 and 224 pixels each, followed by ResNet-50 and ResNet-101 fitted with image resolution 128. For all the models, the same general architecture was used, with changes in the model type by layer and input image resolution.

The general architecture is such: Since our training set is large, we did not fix the layers of the base models, and created a convolutional backbone by removing the original dense classifier from imageNet, followed by adding GlobalAveragePooling2D layer to collapse each feature map into scalar to turn pretrained feature extractor into an accurate 8-way cell-type classification model. To further prevent overfitting, we randomly drop 30% of units each step, followed by an 8-unit softmax output layer for the 8 classes. We used Adam optimizer (learning rate 1e-4), [6] to balance training speed with stability, used categorical cross-entropy loss to learn probability distributions over 8 cell types, and tracked training and validation accuracy and AUC for each epoch. Input images of resolution 224 or 64, depending on the model, were put through keras ImageDataGenerator and rescaled to [0,1]. Training had data augmentation with random shear [-0.2, 0.2] radians, zoom adjustments [80%, 120%], and horizontal flips, with only rescaling for validation and test. All initial models used a batch size of 16, preferred to try in practice[10].

Rapid convergence was observed on 224 resolution images, with ResNet-101 and ResNet-50 models reaching >97% validation accuracy by epoch 2. As per Figure 1, training loss drops drastically across all models in the first 3 epochs. Comparing models by resolution, there are a few validation loss spikes for 224 resolution, suggesting higher resolution with larger and higher variance gradients might overshoot the minimum loss with the current learning rate or dropout rate. In contrast, the 64 models converged more slowly due to smaller and less variable gradients, so the validation curve remains smoother. On the other hand, under the same resolution, ResNet-50 has smoother changes in validation loss than ResNet-101, as in Figure 1, implying models with more layers and larger capacity would be

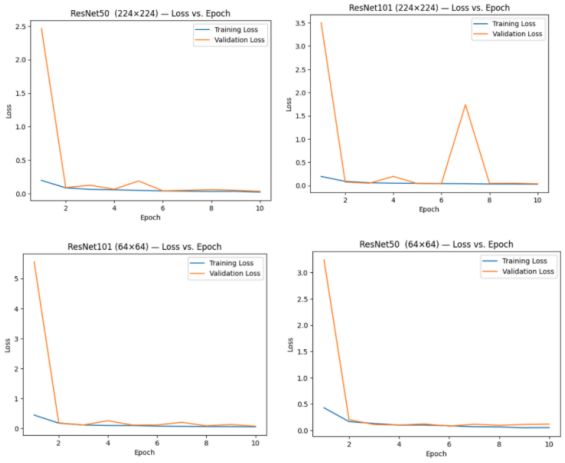


Figure 1: The Loss vs. Epoch curve for all four initial models

more sensitive to training hyperparameters such as learning rate and regularization and need to be specifically tuned when scaling up model depth.

Hyperparameter Tuning:

Hyperparameter tuning was conducted on ResNet50 with 64 resolution, for lower computation costs.

Batch Size Comparison: We first evaluated the impact of batch size by comparing models trained with 16 and 32 samples per iteration. While larger batch sizes improve efficiency by reducing weight updates, smaller batches can introduce stochastic noise that benefits generalization[14]. This comparison allowed us to assess how this trade-off influenced ResNet50 performance.

	Batch Size 16	Batch Size 32
Training Accuracy	0.9831	0.9856
Training AUC	0.9989	0.9993
Training Loss	0.0525	0.0412
Validation Accuracy	0.9690	0.9743
Validation AUC	0.9950	0.9969
Validation Loss	0.1170	0.0858

Table 1. ResNet50 (64x64 pixels) batch size comparison: batch size 16 vs 32.

Batch size of 32 provided better validation performance and was selected for all subsequent experiments.

Attention Layer and Class Weighted Loss Function: To address class-specific lower accuracy, we incorporated a class-weighted loss function to prevent bias toward majority classes and enhance learning of underrepresented classes.[15] Building on this, we added an attention layer to test whether focusing on the most relevant input features could further improve performance.[16]

	Original	Class Weighted Loss	Class Weighted Loss + Attention Layer
Training Accuracy	0.9856	0.9774	0.9542
Training AUC	0.9993	0.9984	0.9973
Training Loss	0.0412	0.0670	0.1310
Validation Accuracy	0.9743	0.9702	0.9620
Validation AUC	0.9969	0.9983	0.9966
Validation Loss	0.0858	0.0899	0.1222

Table 2. ResNet50 (64x64 pixel) attention layer and class weighted loss function loss comparison: batch size 32

Though adding class-weighted loss function and attention layer did not improve performance, we retained both in subsequent analyses to explore if further hyperparameter tuning could enhance model outcomes.

Optimizer Comparison: We compared widely used optimizers[17] Adam, RMSprop, and Stochastic Gradient Descent (SGD), to evaluate which most effectively adjusted weights and learning rates during training to minimize loss and improve accuracy. First performing overall comparison between optimizers:

	Batch Size 32		
Optimizer	Adam (lr = 1e-4)	RMSprop (lr = 1e-4)	SGD (lr = 1e-3, mom = 0.9)
Training Accuracy	0.9542	0.9743	0.8362
Training AUC	0.9973	0.9978	0.9779
Training Loss	0.1310	0.0947	0.4787
Validation Accuracy	0.9620	0.9702	0.9001
Validation AUC	0.9966	0.9931	0.9907
Validation Loss	0.1222	2.4937	0.2894

Table 3. ResNet50 (64x64 pixel) optimizer comparison with batch size 32 with attention layer and class weighted loss function: Adam (learning rate = 1e-4), RMSprop (learning rate = 1e-4), and SGD (learning rate = 1e-3, momentum = 0.9). Note: lr = learning rate; mom = momentum.

SGD Momentum and Learning Rate Comparison: Focusing on SGD, fine-tuning momentum and learning rate, momentum 0.8 performed best, and was tested with varying learning rates.

SGD (learning rate = 1e-3)				
Momentum	**0.9**	**0.8**	**0.7**	**0.5**
Training Accuracy	0.8362	0.9808	0.9639	0.973
Training AUC	0.9779	0.9994	0.9982	0.999
Training Loss	0.4787	0.0545	0.1035	0.0773
Validation Accuracy	0.9001	0.972	0.9626	0.9638
Validation AUC	0.9907	0.9978	0.9981	0.9973
Validation Loss	0.2894	0.0808	0.1053	0.1077

SGD (momentum = 0.8)			
Learning Rate	**1e-2**	**1e-3**	**1e-4**
Training Accuracy	0.9823	0.9808	0.9414
Training AUC	0.9995	0.9994	0.9947
Training Loss	0.0520	0.0545	0.1929
Validation Accuracy	0.9720	0.9720	0.9579
Validation AUC	0.9973	0.9978	0.9975
Validation Loss	0.0883	0.0808	0.1297

Table 4 (left). ResNet50 (64x64 pixel) SGD (learning rate = 1e-3) optimizer comparison with different momentums: batch size 32, with attention layer and class weighted loss function.
Table 5 (right). ResNet50 (64x64 pixel) SGD (momentum = 0.8) optimizer comparison with different learning rates: batch size 32, with attention layer and class weighted loss function.

Adam and RMSprop Learning Rate Comparison: We similarly compared learning rates[18] for Adam and RMSprop (Tables 6 and 7). After identifying that a learning rate of 1e-4 achieved the best performance, we further investigated model performance on slightly higher rates (3e-4, 5e-4).

Adam						
Learning Rate	**1e-2**	**1e-3**	**3e-4**	**5e-4**	**1e-4**	**1e-5**
Training Accuracy	0.8799	0.8405	0.9119	0.8822	0.9542	0.8972
Training AUC	0.9844	0.9764	0.9926	0.9873	0.9973	0.9868
Training Loss	0.3755	0.6370	0.2498	0.3684	0.1310	0.3435
Validation Accuracy	0.8230	0.8347	0.9352	0.9246	0.9620	0.9445
Validation AUC	0.9793	0.9810	0.9960	0.9934	0.9966	0.9937
Validation Loss	0.5631	0.5196	0.1693	0.2138	0.1222	0.3134

RMSprop Optimizer						
Learning Rate	**1e-2**	**1e-3**	**3e-4**	**5e-4**	**1e-4**	**1e-5**
Training Accuracy	0.8631	0.9576	0.9663	0.9027	0.9743	0.9682
Training AUC	0.9802	0.9962	0.9972	0.9872	0.9978	0.9971
Training Loss	0.4470	0.1493	0.1241	0.4902	0.0947	0.0993
Validation Accuracy	0.7699	0.9060	0.9667	0.9246	0.9702	0.9632
Validation AUC	0.9488	0.9875	0.9940	0.9929	0.9931	0.9935
Validation Loss	1.0332	0.3475	0.1411	0.2383	2.4937	0.1700

Table 6 (left). ResNet50 (64x64 pixel) Adam optimizer comparison with different learning rates: batch size 32, with attention layer and class weighted loss function Table 7 (right). ResNet50 (64x64 pixel) RMSprop optimizer comparison with different learning rates: batch size 32, with attention layer and class weighted loss function.

Final Optimizer Summary: We then compared the top-performing optimizers:

	No Class Weighted Loss + Attention Layer			**Class Weighted Loss + Attention Layer**		
Optimizer	**Adam (learning_rate =1e-4)**	**RMSprop (learning_rate =1e-4)**	**SGD (learning_rate =1e-3, momentum = 0.8)**	**Adam (learning_rate =1e-4)**	**RMSprop (learning_rate =1e-4)**	**SGD (learning_rate =1e-3, momentum = 0.8)**
Training Accuracy	0.9856	0.9839	0.9714	0.9542	0.9743	0.9808
Training AUC	0.9993	0.9987	0.9984	0.9973	0.9978	0.9994
Training Loss	0.0412	0.0497	0.0886	0.1310	0.0947	0.0545
Validation Accuracy	0.9743	0.9731	0.9737	0.9620	0.9702	0.9720
Validation AUC	0.9969	0.9941	0.9983	0.9966	0.9931	0.9978
Validation Loss	0.0858	0.1752	0.0824	0.1222	2.4937	0.0808

Table 8. ResNet50 (64x64 pixel) top-performing optimizer comparison without and with attention layer and class weighted loss function: batch size 32.

As observed, the model without an attention layer or class-weighted loss function still performed the best. Thus, we proceeded with no attention layer or class-weighted loss function appended.

Dropout Rate Comparison: Finally, after identifying Adam (learning rate = 1e-4) as the most robust optimizer, we evaluated different dropout rates to assess whether further improvements in generalization could be achieved by introducing regularization through randomly deactivating neurons during training.[19]

Dropout Rate	0.3	0.4	0.5
Training Accuracy	0.9856	0.9870	0.9842
Training AUC	0.9993	0.9995	0.9991
Training Loss	0.0412	0.0976	0.1154
Validation Accuracy	0.9743	0.9597	0.9725
Validation AUC	0.9969	0.9948	0.9971
Validation Loss	0.0858	0.1402	0.0977

Table 9. ResNet50 (64x64 pixels) dropout rate comparisons: batch size 32, without attention layer or class-weighted loss function, Adam (learning rate = 1e-4)

External data validation: To assess robustness and generalizability, we validated the ResNet-50 model on the Raabin white blood cell dataset.[20] Images were resized to 64 pixels, and data augmentation matched the BloodMNIST setup. The dataset was split with 20% (865 images) for validation. The ResNet-50 architecture included a 0.3 dropout layer and a dense output layer with 5 softmax units. The model was compiled with the Adam optimizer (learning rate 1e-4) and categorical cross-entropy loss, tracking accuracy, and AUC. Training was performed for 10 epochs with a batch size of 32.

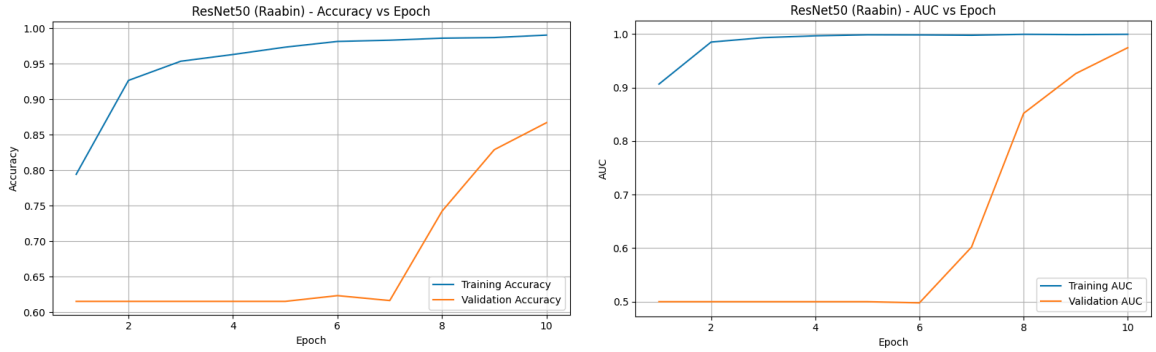


Figure 2: The Accuracy and AUC vs. Epoch curve for the ResNet50 model with Raabin dataset

The evaluation metrics for the deep learning models: Model performance was assessed using overall accuracy and AUC to capture global predictive power. For evaluation by cell type, we reported per-class precision, recall, and f-1 score to gauge model success in avoiding false positives, capturing true positives, and the balance between the two, respectively.

Results:

Initial model results: All four initial models fit have consistent patterns in performance at the class level. Across all models, the classes with the highest prediction accuracy in the test set are eosinophils and platelets (Table 10), which achieve 100% precision and recall in almost every model, and the monocytes and immature granulocytes classes have comparatively lower recall rates <0.9 (Table 10). The consistent lower performance in these classes indicates that there are likely some persistent patterns, either in the images or our modeling strategies, that may be reflective of this lowered class performance, and this is an area for further improvement that we will be investigating as we proceed to tuning models. In addition, the relatively similar overall performances by model layers and input resolution may also indicate persistent characteristics in the model architectures, especially since we used the same strategy across all models, which we will investigate. In addition to better balancing model complexity and ideally increasing accuracy from what we have achieved for the initial model fit, we will further explore both more complex and simpler model variants to see which best predict the most difficult categories while still retaining predictive performance on the other classes for the following hyperparameter tuning.

	ResNet101 (224×224)		ResNet50 (224×224)		ResNet101 (64x64)		ResNet50 (64x64)	
Cell Type	precision rate	recall rate	precision rate	recall rate	precision rate	recall rate	precision rate	recall rate
Basophil	1.00	0.98	1.00	1.00	0.98	0.98	0.98	0.98
Eosinophil	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00
Erythroblast	1.00	0.99	1.00	0.98	0.98	0.99	0.98	0.97
Immature granulocytes	0.97	0.98	0.96	0.97	0.89	0.98	0.90	0.96
Lymphocyte	0.98	1.00	0.97	1.00	1.00	0.93	0.94	0.98
Monocyte	1.00	0.98	1.00	0.95	0.96	0.85	1.00	0.81
Neutrophil	0.97	0.98	0.97	0.98	0.99	0.96	0.97	0.98
Platelet	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Model	Test Accuracy	Test AUC
ResNet101 (224×224)	0.9880	0.9989
ResNet50 (224×224)	0.9857	0.9985
ResNet101 (64×64)	0.9693	0.9978
ResNet50 (64×64)	0.9676	0.9956

Table 10 (left): The test precision and recall for all four models. Table 11 (right): The test accuracy and AUC for all four models

Hyperparameter tuning results:

Batch Size Comparison: Training and validation metrics for ResNet50 (64) as in Table 1 show batch size 32 achieved higher validation accuracy, AUC, and lower validation loss. These results indicate improved generalization and suggest that batch size 32 strikes a better balance between stability and performance. Therefore, batch size 32 was selected for all subsequent hyperparameter tuning experiments.

Attention Layer and Class Weighted Loss Function: From the original model configuration, there are consistently lower precision and recall in some cell types. To improve model performance in them, we integrated a class-weighted loss function into the ResNet50 model. Additionally, we tested whether attention layers, designed to emphasize the most relevant features contributing to classification, could enhance predictive performance. While both modifications preserved strong overall performance, they slightly reduced accuracy and increased validation loss compared to the original configuration. The original configuration achieved the highest validation accuracy 0.9743 and lowest validation loss 0.0858. In contrast, the model with class-weighted loss and attention reached validation accuracy 0.9620 and a higher loss 0.1222. The result suggests a trade-off between convergence efficiency and potential overfitting using additional layers. Although these adjustments did not improve model performance, we retained them in subsequent experiments involving optimizer and learning rate tuning to evaluate whether they would interact beneficially with other hyperparameter settings.

Optimizer Comparison: We evaluated model performance using widely used optimizers, Adam, RMSprop, and SGD across various hyperparameter settings. We began by comparing the optimizers using ResNet50 with a 64 input size, attention layer, and class-weighted loss function. We fixed batch size to 32 due to previously demonstrated superior performance. Results revealed that Adam (learning rate 1e-4) and SGD (learning rate 1e-3, momentum 0.9) outperformed RMSprop (learning rate 1e-4) in both training and validation metrics. While Adam achieved a strong validation accuracy of 0.9620 and loss of 0.1222, SGD has slightly better performance with validation accuracy of 0.9720 and the lowest loss of 0.0808.

SGD, Adam, and RMSprop's and learning rate comparison: We investigated the effect of different momentum values in SGD while keeping the learning rate fixed at 1e-3. Momentum of 0.8 yielded the best overall performance. Further tuning of the learning rate showed that both 1e-3 and 1e-2 performed similarly well, but 1e-3 with momentum of 0.8 consistently achieved the best balance between accuracy, AUC, and loss. For optimizer Adam, we observed best validation performance at a learning rate of 1e-4. Increasing the learning rate to 3e-4 or 5e-4 resulted in degraded accuracy and higher validation loss, indicating potential overfitting or instability. RMSprop similarly showed its best performance at a learning rate of 1e-4, though with a concerningly high validation loss of 2.4937.

Final optimizer comparison: The best-performing configuration is SGD with learning rate 1e-3 and momentum 0.8, showing the best trade-off between training and validation metrics, achieving validation accuracy 0.9720 and AUC 0.9978. However, the original model using Adam optimizer (learning rate 1e-4) without attention layer and class weights still slightly outperformed all modified versions, achieving highest validation accuracy of 0.9743 and the lowest loss of 0.0858, suggesting that simpler models may generalize better in this case. When tested on the original model without attention and class-weighted loss, Adam (learning rate 1e-4) and SGD still showed the best performance, but none performed better than the baseline configuration using Adam (learning rate 1e-4) without an attention layer and class-weighted loss.

Dropout Rate Comparison: We tested dropout rates of 0.3, 0.4, and 0.5 on ResNet-50 with 64×64 inputs (batch size 32, Adam at 1e-4). All configurations yielded high validation and test performance, but 0.3 was optimal: it achieved 97.4 % validation accuracy, 99.69 % AUC, 0.0858 loss, and 97.8 % test accuracy with 99.66 % AUC. Increasing dropout to 0.4 dropped test accuracy to 96.5 % and raised validation loss to 0.1402; at 0.5, accuracy goes to 97.3 % on both sets, but still does not achieve what it has with 0.3. Thus, 0.3 provides the best regularization–generalization balance.

Final model comparison: Batch size 32 improved performance during hyperparameter tuning on the ResNet-50 64 resolution model, giving higher validation AUC and lower validation loss, but this did not occur across all models. Validation accuracy was consistently higher with batch size 16 than 32, except for the ResNet-50 64 resolution model used for hyperparameter training. This suggests that while larger batch size may stabilize training and improve runtime, it can prevent generalization.

	batch size = 16						batch size = 32					
	224×224		128×128		64×64		224×224		128×128		64×64	
	ResNet101	ResNet50	ResNet101	ResNet50	ResNet101	ResNet50	ResNet101	ResNet50	ResNet101	ResNet50	ResNet101	ResNet50
Test Accuracy	0.9880	0.9857	0.9807	0.9687	0.9693	0.9676	0.9848	0.9734	0.9728	0.9860	0.9667	0.9775
Test AUC	0.9989	0.9985	0.999	0.9984	0.9978	0.9956	0.9983	0.9984	0.9975	0.9989	0.9965	0.9966

Table 12: Test set performance of all models (dropout rate = 0.3, Adam 1e-4, batch size 16 vs. 32, without attention layer or class-weighted loss)

Examining class-level performance, F1-scores across most cell types were relatively stable across batch sizes. However, monocytes and immature granulocytes consistently showed the lowest F1-scores, particularly for ResNet-50 (64 pixels) at batch size 32, where monocyte F1 dropped to 0.89 and immature granulocyte to 0.92. This reflects persistent classification challenges for these morphologically ambiguous classes. In contrast, eosinophils and platelets consistently achieved perfect F1-score across all models and batch sizes, showing that morphologically distinctive classes are well identified.

	ResNet101 (224×224)		ResNet101 (128x128)		ResNet101 (64x64)		ResNet50 (224×224)		ResNet50 (128x128)		ResNet50 (64x64)	
Cell Type	Batch size = 16	Batch size = 32	Batch size = 16	Batch size = 32	Batch size = 16	Batch size = 32	Batch size = 16	Batch size = 32	Batch size = 16	Batch size = 32	Batch size = 16	Batch size = 32
Neutrophil	0.98	1.00	0.97	0.92	0.98	0.96	0.98	0.99	0.97	0.99	0.98	0.98
Eosinophil	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Basophil	0.99	1.00	0.99	0.99	0.98	0.98	1.00	0.99	0.94	0.98	0.98	0.98
Lymphocyte	0.99	0.96	0.97	0.93	0.96	0.92	0.98	0.94	0.97	0.97	0.96	0.95
Monocyte	0.99	0.99	0.97	0.99	0.90	0.96	0.97	0.98	0.94	0.99	0.89	0.98
Immature granulocyte	0.97	0.97	0.96	0.98	0.93	0.92	0.97	0.97	0.93	0.99	0.92	0.97
Erythroblast	0.99	0.98	0.99	0.97	0.99	0.97	0.99	0.95	0.96	0.98	0.98	0.97
Platelet	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 13: The class-level performance in terms of F1 score, for dropout rate = 0.3 , optimizer is Adam 1e-4

Validation Result of the ResNet50 Model on the Raabin WBC Dataset:

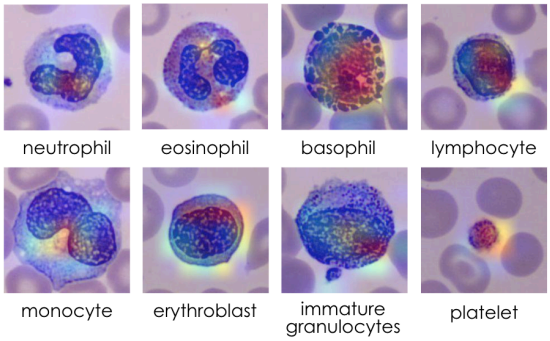
Overall, the model achieved 82% accuracy on the Raabin data. It demonstrated strong performance on lymphocytes and neutrophils, consistent with findings on BloodMNIST data. However, classification performance for monocytes was notably lower, reflecting morphological overlaps and class imbalance.

	Precision	Recall	F1-score	Support
Basophil	0.76	0.94	0.84	17
Eosinophil	0.78	0.83	0.80	64
Lymphocyte	0.96	0.72	0.83	206
Monocyte	0.26	0.89	0.40	46
Neutrophil	0.96	0.85	0.91	532

Class	Accuracy
Basophil	0.9412
Eosinophil	0.8281
Lymphocyte	0.7233
Monocyte	0.8913
Neutrophil	0.8477
Overall	**0.8200**

Table 14 (left). Per-class performance of ResNet-50 of resolution 64×64 on the Raabin WBC Dataset. Table 15 (right). Per-class accuracy of ResNet-50 of resolution 64×64 on the Raabin WBC Dataset

Model visualization: To enhance interpretability and verify that model predictions were based on biologically meaningful features, we applied Gradient-weighted Class Activation Mapping (Grad-CAM). For BloodMNIST (Figure 2), we visualized the highest performing model, ResNet-101 resolution 224. The model captured unique structures of the different cell types, with high attention to intracellular regions, confirming focus on known morphology and not peripheral artifacts. The heatmaps highlighted distinctive features such as the u-shaped nucleus of monocytes, basophilic granules of basophils, dense chromatin of erythroblasts and lymphocytes, and multi-lobed nuclear structures in eosinophils and neutrophils. We also applied Grad-CAM to the ResNet50 model with 64 resolution fitted on the external Raabin data (Figure 3). Resulting heatmaps similarly demonstrated that the model focused on core cellular features such as nuclear morphology and cytoplasmic texture as primary drivers of classification decisions. This consistency across two independent datasets underscores the robustness of the model, provides interpretability, and supports its alignment with established biological knowledge.



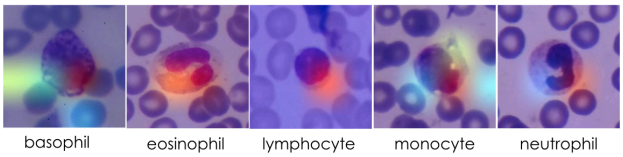


Figure 2 (left): Grad-CAM visualization of ResNet-101 model attention region.
Figure 3 (right): Grad-CAM visualization of ResNet-50 model attention region on the Raabin dataset

Discussion:

Interpretations/implications of results:

Our work shows the effectiveness of ResNet-50 and ResNet-101 architectures in classifying peripheral blood cell images across resolutions. After unfreezing base layers and removing the original ImageNet classifiers, we customized the models with a GlobalAveragePooling2D layer, 8-unit softmax output, and regularization via a dropout rate 0.3. We employed Adam optimizer with learning rate 1e-4, and used categorical cross-entropy loss to optimize performance across 8 cell classes. Across all experiments, ResNet-50 and ResNet-101 showed consistently high performance by accuracy and AUC, with the best model being ResNet-101 (224 resolution) with batch size 16 and dropout rate 0.3, achieving a test accuracy of 0.988 and AUC of 0.999. Despite strong overall performance, class-level performance revealed persistent weaknesses in monocyte and immature granulocyte classification, reflecting morphological ambiguity in these cell types and suggesting future targeted improvements are necessary.

Hyperparameter tuning tested batch sizes (16 and 32), dropout rates (0.3, 0.4, 0.5), learning rates (1e-2 to 1e-5), and optimizers (Adam, RMSprop, and SGD), and results reaffirmed that the initial parameters used generalize best. Specifically, Adam with learning rate 1e-4 consistently had smooth convergence, low validation loss, and strong generalization. Increasing dropout beyond 0.3, using attention layers, and class-weighted loss functions reduced performance, likely due to added model complexity introducing overfitting or noisy gradients. Interestingly, larger batch size 32 marginally improved F1-scores in some classes, suggesting that more stable gradient estimation may aid in addressing minority-class variability. However, batch size 32 does not improve overall test and validation performance for most models, except for ResNet-50 at 64 resolution, and performance for some classes even degrades. Overall, ResNet-101 at 224 resolution with dropout rate 0.3, batch size 16, Adam optimizer with learning rate 1e-4, and no attention layers or class-weighted loss function, proved optimal in balancing test accuracy, AUC, and training efficiency.

Additionally, the ResNet50 model generalized well, maintaining strong performance for major classes, though lower monocyte accuracy as in BloodMNIST was also observed on Raabin data. Finally, Grad-CAM visualizations on models for both BloodMNIST and Raabin revealed consistent model focus on biologically relevant intracellular regions like nuclear morphology and cytoplasmic granularity, providing interpretability and highlighting model reliance on valid known cellular structures.

Limitations of the study:

Model-related challenges: In our models, most classes achieved strong classification performance, but monocytes and immature granulocytes consistently had lower precision and recall across models and datasets. This likely reflects intrinsic morphological challenges in differentiating these cells from other peripheral blood cell types. Similarly, when models were evaluated on the external dataset, monocytes had greatly reduced accuracy compared to the other cell classes. To tackle this class-specific issue, we experimented with class-weighted loss functions, but it did not improve overall prediction performance and, in some cases, degraded overall performance. We also tried to add attention layers to prioritize important regions for classification, but this also negatively impacted performance. These implementations are theoretically beneficial, but may have led to overfitting in our context, especially given that some cell types have subtle morphological differences and are visually similar. Moreover, in terms of computational efficiency, training deeper models, especially with high-resolution images, involved great computational costs. For example, ResNet101 at 224 resolution required over 5 hours to train locally, compared to 30 minutes for ResNet50 with 64 resolution inputs. Though high-resolution models performed better, the marginal improvements did not justify the local training time and load.

Data challenges: BloodMNIST combines metamyelocytes, myelocytes, and promyelocytes into immature granulocytes, which may contribute to model misclassification, as there are visually observable morphological differences between these three subtypes, and the model may misclassify these cells into other mature cells based on morphological similarities. Also, the Raabin data lacks erythroblasts, platelets, and immature granulocytes, limiting full generalization analysis, and future work should explore broader, balanced, multi-center datasets for validation.

Future direction: Future work will first focus on improving the classification accuracy for monocytes and immature granulocytes. The immature granulocyte class encompasses morphologically heterogeneous subtypes that may benefit from subclassification, and future studies should explore splitting this cell type into the subtypes promyelocytes, myelocytes, and metamyelocytes, to better capture cell variation and improve performance. Further investigation should also be done to determine which cell types each of monocytes and these three immature granulocytes subtypes are commonly confused for by the models. This issues should be accounted for through targeted changes to the model architectures using hyperparameter tuning, additional data, and other relevant methods to improve performance on

monocytes and immature granulocytes without compromising performance on the other cell types and overall. In addition, further assessments of model robustness and generalizability should be made by extending validation to other publicly available blood cell imaging datasets beyond Raabin, including different staining protocols, imaging equipment, and class distributions, to help with real-world diagnostic variability. Finally, to deal with issues with computational costs of running some of these more demanding models locally, using an HPC cluster, or investigating model architectures with similar benefits over VGG16 and InceptionV3 but that are more lightweight and require less computational resources, could feasibly be used to deal with this issue.

Group Member Contribution: Harshi Saha: Dataset discovery and preprocessing/EDA, project proposal and check-in writeup, literature review and strategy for model architecture and hyperparameter tuning, project poster, and final report. Diana Hou: initial model architecture and training, project proposal and check-in writeup, literature review and strategy for model architecture and hyperparameter tuning, project poster, and final report. Cece Jen: Hyperparameter tuning, project check-in writeup, literature review for hyperparameter tuning, project poster and final report. Tina Tu: Raabin dataset identification and model adaptation to the dataset, project check-in, poster, and final report. Clementine Chen: Help with initial model, proposal, project check-in writeup, literature review, organizing results, figures, and final report.

References:

[1] Acevedo A. et al. Recognition of peripheral blood cell images using convolutional neural networks. Computer methods and programs in biomedicine.
[2] Acevedo A. et al. A dataset of microscopic peripheral blood cell images for development of Automatic Recognition Systems. Data in brief.
[3] Yang, J.et al (2023, January 19). MedMNIST v2 - a large-scale lightweight benchmark for 2D and 3D Biomedical Image Classification. Nature News.
[4] Anand, V. et al. (2024). Deep learning-based image annotation for leukocyte segmentation and classification of blood cell morphology. BMC
[5] Yuan Y. et al A comparative analysis of eleven neural networks architectures for small datasets of lung images of COVID-19 patients toward improved clinical decisions, Computers in Biology and Medicine,
[6] Jung, C. et al. (2022). WBC image classification and generative models based on convolutional neural network. BMC Medical Imaging, 22(1).
[7] Saidani O, et al. White blood cells classification using multi-fold pre-processing and optimized CNN model Sci Rep. 2024 Feb 12
[8] Habibzadeh M, et al. Automatic white blood cell classification using pre-trained deep learning models: resnet and inception. In: International conference on machine vision (ICMV), 2018. p. 274–81.
[9] Erten, M. et al. (2024, August 2). ConcatNeXt: An automated blood cell classification with a new deep convolutional neural network - multimedia tools and applications.
[10] Chen, H. et al. (2022). Accurate classification of white blood cells by coupling pre-trained ResNet and DenseNet with SCAM mechanism. BMC
[11] Wang, Z.et al. (2022). WBC-AMNet: Automatic classification of WBC images using deep feature fusion network based on focalized attention mechanism. PLOS ONE
[12] Mekruksavanich, S. et al. Hybrid convolution neural network with channel attention mechanism for sensor-based human activity recognition. Sci Rep 13, 12067 (2023).
[13] Zhang Y et al (2023) Deep-Learning Model of ResNet Combined with CBAM for Malignant–Benign Pulmonary Nodules Classification on Computed Tomography Images. Medicina, 59(6), 1088–1088.
[14] Goyal, P. et al. (2018). Accurate, large minibatch SGD: Training ImageNet in 1 hour (arXiv preprint)
[15] Huang, Z., & Sui, Y. (2024). Contour-weighted loss for class-imbalance (arXiv preprint)
[16] Wang, F et al. (2017). Residual attention network for image classification (arXiv preprint)
[17] Choi, D., et al. (2020). On empirical comparisons of optimizers for deep learning (arXiv preprint)
[18] L. N. Smith et al, "Cyclical Learning Rates for Training Neural Networks," 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), Santa Rosa, CA, USA, 2017, pp. 464-472
[19] Cai, S, et al. (2020). Effective dropout for deep convolutional neural networks (arXiv preprint).
[20] Kouzehkanan, et al. (2022). A large dataset of white blood cells containing cell locations and types, along with segmented nuclei and cytoplasm. Scientific reports, 12(1), 1123.