BloodMNIST Image Classification

281 Final Project, Summer 2024

Shaki Pothini, Austen Lowitz, Annie Deforge

GitHub Notebook

Abstract

In this project, we developed an advanced image classifier to accurately distinguish between eight classes of blood cells using the BloodMNIST dataset. Through meticulous data preprocessing involving binary thresholding and a strategic selection of features including color histograms, GLCM, light to dark cell count ratio, and Visual Bag of Words, we achieved significant classification accuracy. We trained multiple machine learning models, including Logistic Regression, Random Forest, XGBoost, and SVMs, with SVM (RBF kernel) emerging as the most effective model, demonstrating 95% test accuracy. Our results underscore the importance of feature engineering, model selection, and hyperparameter tuning in tackling complex multi-class classification problems. This tool has the potential to aid medical professionals in diagnosing and treating endocrine and blood-related diseases, highlighting its practical and societal value.

Introduction

In the human body, there are several different types of white blood cells that serve different functions. The ability to properly identify the type of white blood cells in a population can be an indication of recovery from a disease or signal state of immune system health. The application of computer vision towards developing highly accurate image classification models is especially important in the field of medicine and health care as a model can predict classes for a large dataset much faster than it would take a human to identify class labels, thus it can be used to speed up diagnosis, and potentially even pick up on things that a human might miss, which can greatly improve patient outcomes. We attempted to build image classification models which could distinguish between different types of white blood cells utilizing the images of stained cells under a microscope. We experimented with several different preprocessing methods to highlight the most essential parts of the image and feature extraction to obtain the most useful feature vector from the raw image. We built and tested several different classifiers, including logistic regression, decision trees and SVMs, to identify a model that was both accurate and parsimonious to run.

Data

We used the BloodMNIST dataset, a part of the larger MedMNIST collection, which provides various medical image datasets designed for research and development in biomedical image analysis. BloodMNIST contains 17,092 high-resolution images of different types of blood cells captured by a blood cell microscope. The source images begin with a resolution of 3x360x363 pixels then are center-cropped into 3x200x200, and eventually resized to our desired resolution of 224x224.

There are 8 classes stratified across training (70%), validation (10%), and test (20%) sets at the patient level to prevent data leakage. This stratification is essential to ensure that evaluation metrics are accurate and not artificially inflated by the same patient's data appearing in both training and test sets.

Dataset Details

- MedMNIST website
- MedMNIST GitHub Repo

Figure 1 is an example of images in each class from the BloodMNIST train dataset:

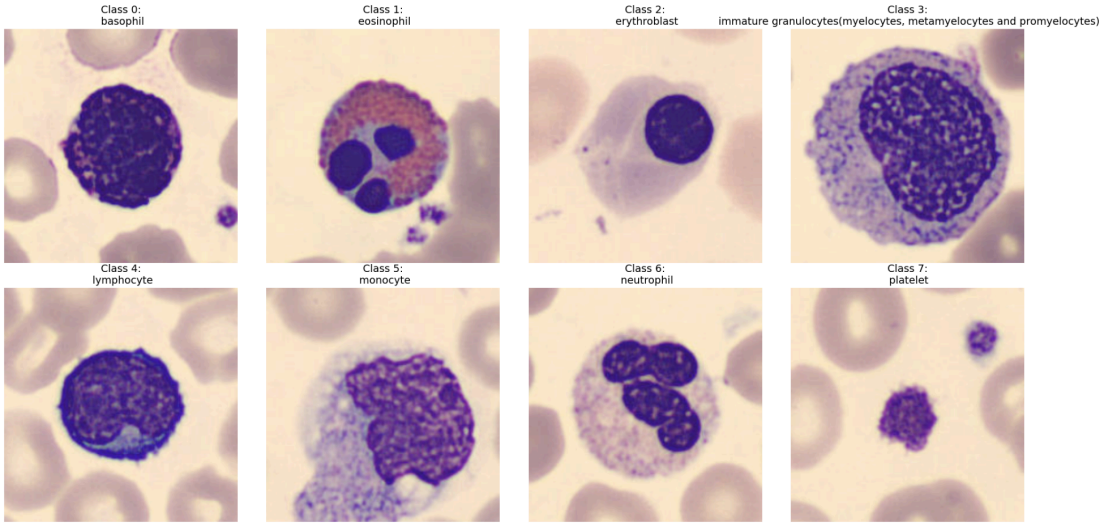

Figure 1: *from left to right: Class 0: Basophil, Class 1: Eosinophil, Class 2: Erythroblast, Class 3: Immature Granulocytes, Class 4: Lymphocyte, Class 5: Monocyte, Class 6: Neutrophil, Class 7: Platelet*

The BloodMNIST dataset contains images of blood cells categorized into eight classes. The images are provided in high resolution (224x224 pixels) and in RGB format (3 channels). The classes are as follows:

- Class 0: Basophil
    - White blood cells involved in inflammatory responses and allergic reactions.
- Class 1: Eosinophil
    - White blood cell key to fighting parasitic infections and involved in allergic reactions.
- Class 2: Erythroblast

    - Immature red blood cells found in bone marrow, precursor to mature red blood cells.
- Class 3: Immature Granulocytes (myelocytes, metamyelocytes, promyelocytes)
    - Early-stage white blood cells (myelocytes, metamyelocytes, promyelocytes) typically found in bone marrow.
- Class 4: Lymphocyte
    - White blood cell crucial for immune response: includes B cells, T cells, and natural killer cells.
- Class 5: Monocyte
    - Large white blood cell that differentiates into macrophages and dendritic cells, important for phagocytosis.
- Class 6: Neutrophil
    - Most abundant white blood cell, first responder to infections, involved in phagocytosis.
- Class 7: Platelet
    - Small blood cell fragments essential for blood clotting and wound healing.

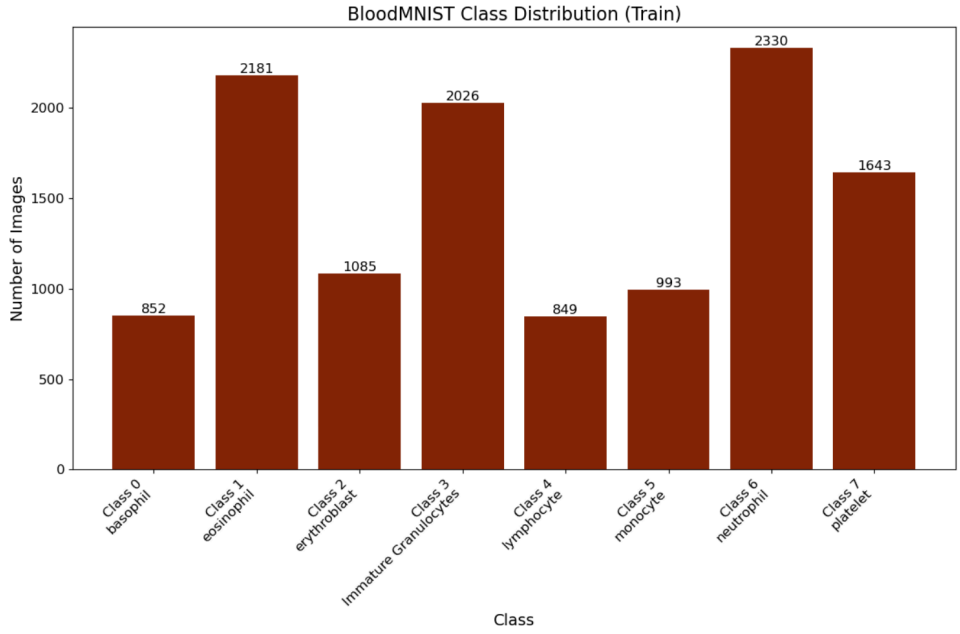

***Figure 2**: count of classes in the train set. Same distribution applied across validation and test sets*

Pre-processing

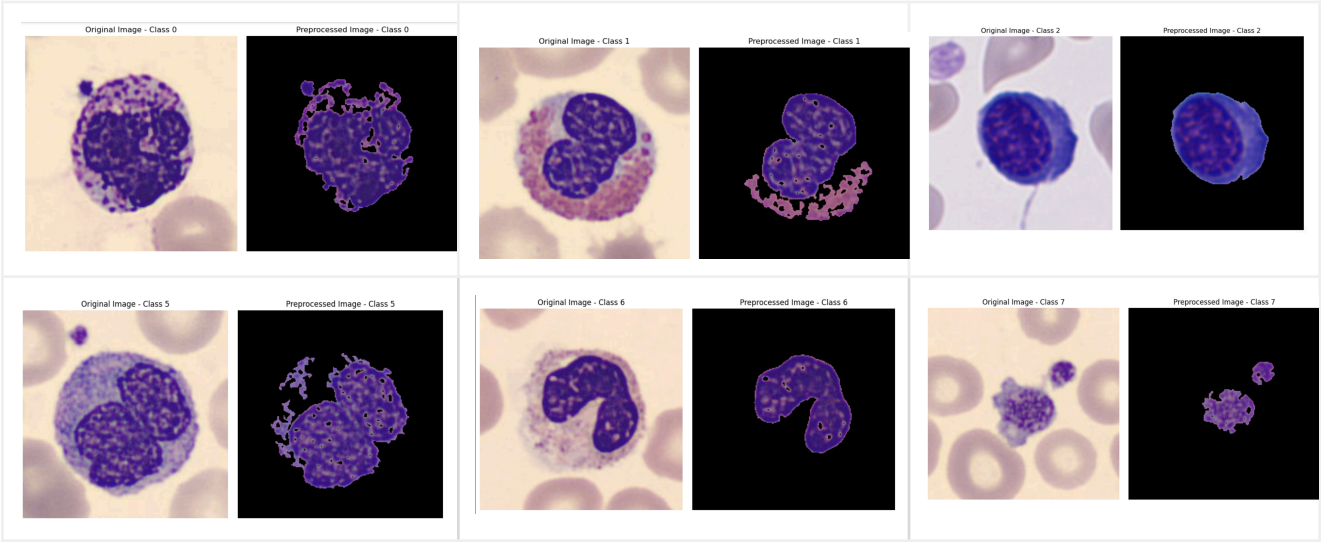

Figure 3: *pre processed images from left to right: Class 0: Basophil, Class 1: Eosinophil, Class 2: Erythroblast, Class 3: Immature Granulocytes, Class 4: Lymphocyte, Class 5: Monocyte, Class 6: Neutrophil, Class 7: Platelet*

1. Convert to Grayscale: Converts the RGB image to grayscale by eliminating the hue and saturation information while retaining the luminance. This step reduces the complexity of the image data, as it changes from three color channels (Red, Green, Blue) to a single channel (intensity).
2. Apply Binary Threshold: converts the image to black and white based on a specified intensity threshold (128 in this case). Pixels with intensity above the threshold are set to 0 (black), and those below are set to 255 (white), due to the cv2.THRESH_BINARY_INV flag which performs inverse binary thresholding. This step highlights the target cells (darker regions) in white and the background in black
3. Remove Small Objects: Eliminates connected components in the binary image that are smaller than the specified size (500 pixels). This step helps in retaining only significant structures, likely corresponding to the target cells, and removing noise.
4. Label Connected Components: assigns a unique label to each connected component in the binary image. The connectivity=2 parameter specifies that both 8-connectivity (diagonal connections) and 4-connectivity (horizontal and vertical connections) are considered. The result is an image where each connected region is labeled with a unique identifier.
5. Keep the Largest Connected Component: It keeps only the connected components that are larger than a specified size (500 pixels), which helps in isolating the largest and presumably most significant component, likely representing the target cell.
6. Mask the Original Image: Highlight the target cells while suppressing the background using the binary mask.

Feature extraction

By incorporating both simple features, like color histograms, and more complex ones, like Visual Bag of Words, we aimed to create a robust feature vector that captures both the basic and nuanced aspects of the blood cell images, ultimately enhancing model performance.

Summary of Features:

1. Simple features

Edge Detection

Our goal with an edge detection feature was to extract the shape information of the cells. We utilized a canny filter in order to find the edges between the dark cell body and the light background.

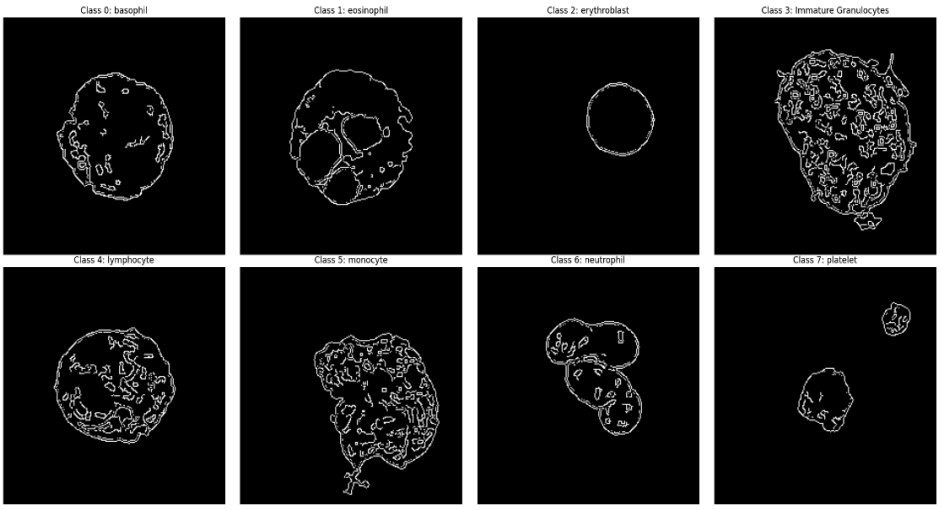

Figure 4: *Edge images from left to right: Class 0: Basophil, Class 1: Eosinophil, Class 2: Erythroblast, Class 3: Immature Granulocytes, Class 4: Lymphocyte, Class 5: Monocyte, Class 6: Neutrophil, Class 7: Platelet*

Color Histograms

Between the classes, there seemed to be variation in hue and pixel intensity of the cell bodies. We used HSV color histograms in order to try and capture the color information from the images.

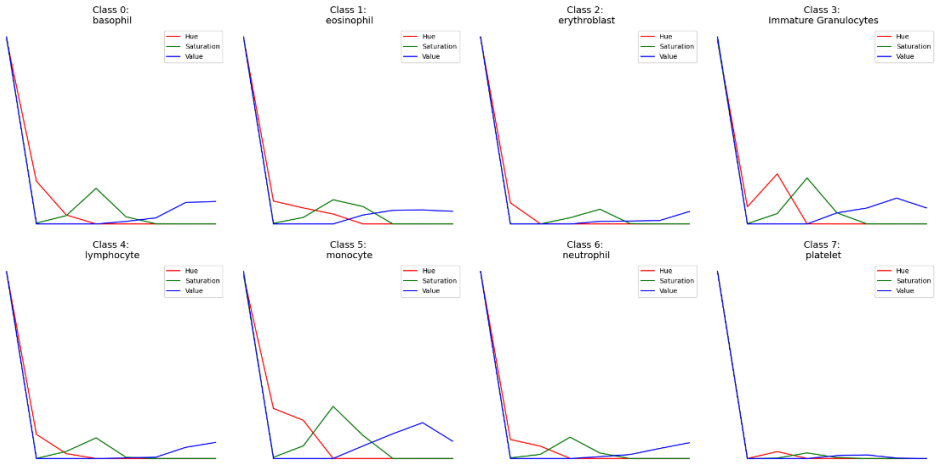

Figure 5: *Color histogram images from left to right: Class 0: Basophil, Class 1: Eosinophil, Class 2: Erythroblast, Class 3: Immature Granulocytes, Class 4: Lymphocyte, Class 5: Monocyte, Class 6: Neutrophil, Class 7: Platelet*

Histogram of Oriented Gradients (HOG)

The hog feature vector captures the distribution of local edges. The cells were centered very uniformly across images, so capturing the orientation of edges extract spatial information that can differentiate between classes. The local gradient information was captured by dividing the image into small cells of 16x16 pixels.

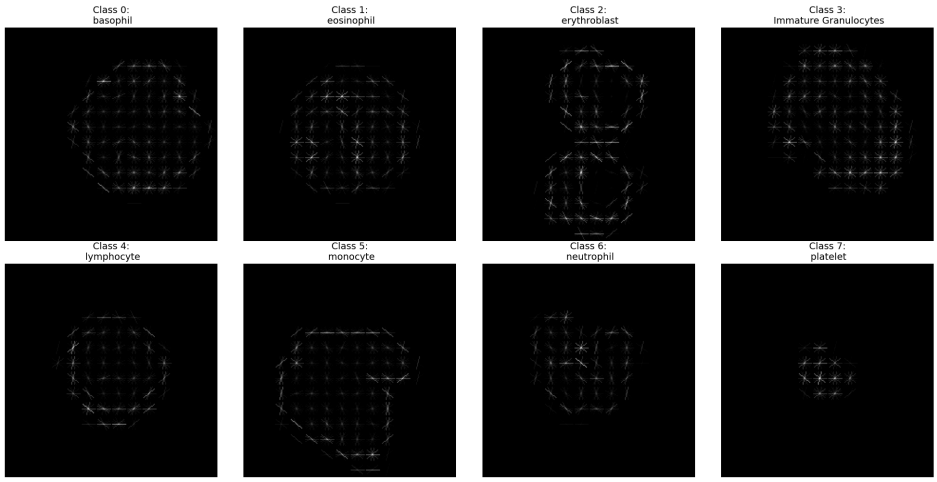

Figure 6: *HOG mages from left to right: Class 0: Basophil, Class 1: Eosinophil, Class 2: Erythroblast, Class 3: Immature Granulocytes, Class 4: Lymphocyte, Class 5: Monocyte, Class 6: Neutrophil, Class 7: Platelet*

Gray Level Co-Occurrence Matrix (GLCM)

We utilized Gray-Level Co-occurrence Matrix (GLCM) to extract texture features from images and analyze the distinctive patterns across different classes

- Dissimilarity: High values indicate complex texture; low values indicate smooth texture.

- Homogeneity: High values indicate smooth texture; low values indicate complex texture.
- Energy: High values indicate very uniform texture.
- Correlation: High values indicate regular and predictable texture; low values indicate random texture.
- ASM (Angular Second Moment): High values indicate regular and uniform texture; low values indicate complex texture.

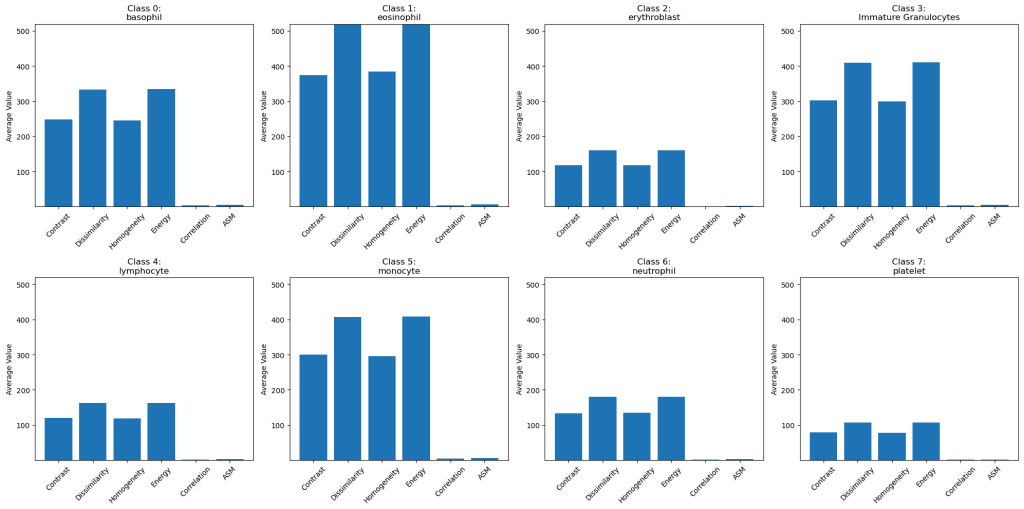

Figure 7: *GLCM images from left to right: Class 0: Basophil, Class 1: Eosinophil, Class 2: Erythroblast, Class 3: Immature Granulocytes, Class 4: Lymphocyte, Class 5: Monocyte, Class 6: Neutrophil, Class 7: Platelet*

Light:Dark Pixel Count Ratio

The image consisted of the stained cell body which was represented by purple pixels in order to create contrast from the background. In general there seemed to be two general ranges of color with which the purple pixels fell into. A dark purple, which represented more dense parts of the cell body, and a light purple, which represented the membrane wall of the cell. We isolated a non-overlapping range in which the RGB pixel values of each of the classes generally fell and extracted a count for each image of the number of dark and light purple pixels (sample patches to extract pixel ranges shown below), and additionally calculated the ratio between the two counts.

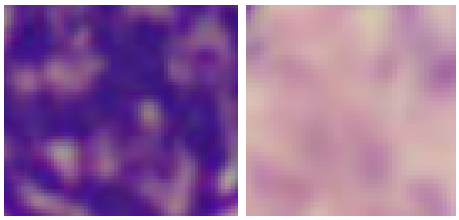

Figure 8: *Pixel color patches. Quantiles of pixel values in patches were analyzed to determine relevant range. Dark cell pixels (left) and light cell pixel (right)*

In this feature, we were trying to extract size information from the image. The images were all taken under similar microscope conditions, so any variance in the amount of space that the cells

take up in the image should represent an intrinsic characteristic about the cell, which we can leverage to differentiate between classes.

2. Complex features

ResNet101

ResNet overview: ResNet101 model was employed for feature extraction to leverage its powerful pre-trained convolutional layers. We wanted to use the learned embeddings of the deep layers of the NN in order to extract complex non-linearities within the images. The final classification layer was removed to retain only the convolutional layers, which are responsible for capturing rich, hierarchical features from the input images. In the model, the total trainable params were 42,500,16.

Each image was preprocessed to fit the input requirements of ResNet, converting to PIL format and performing normalization (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

Visual Bag of Words

To enhance our model's feature extraction, we incorporated a Visual Bag of Words (VBoW) approach using SIFT features, which are effective for identifying key points and extracting local descriptors in images. We applied K-Means clustering to group similar SIFT descriptors into 100 clusters, forming a visual vocabulary where each cluster center represents a specific pattern or texture. For each image, we generated a histogram of visual word occurrences, providing a compact and discriminative feature vector. The 2D visualization of these descriptors and cluster centers illustrates the distinct patterns identified across different cell types, enhancing the model's ability to differentiate between visually similar classes and contributing to its overall accuracy.

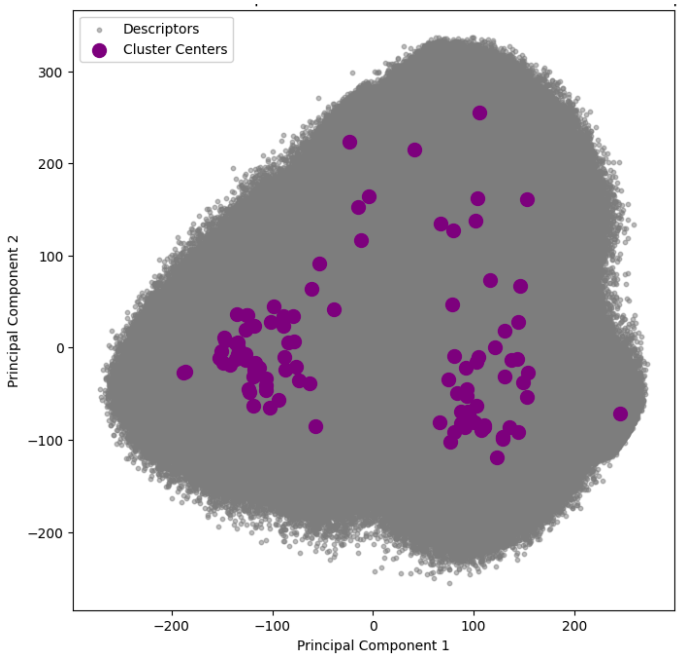

Figure 9: *Visualization of SIFT Descriptors and k-means clusters in reduced 2D space*

3. Analysis of features

We performed PCA analysis for each of the features (excluding the pixel count as that was only 3 dimensions) and analyzed how much of the variance in the data was explained by the feature.

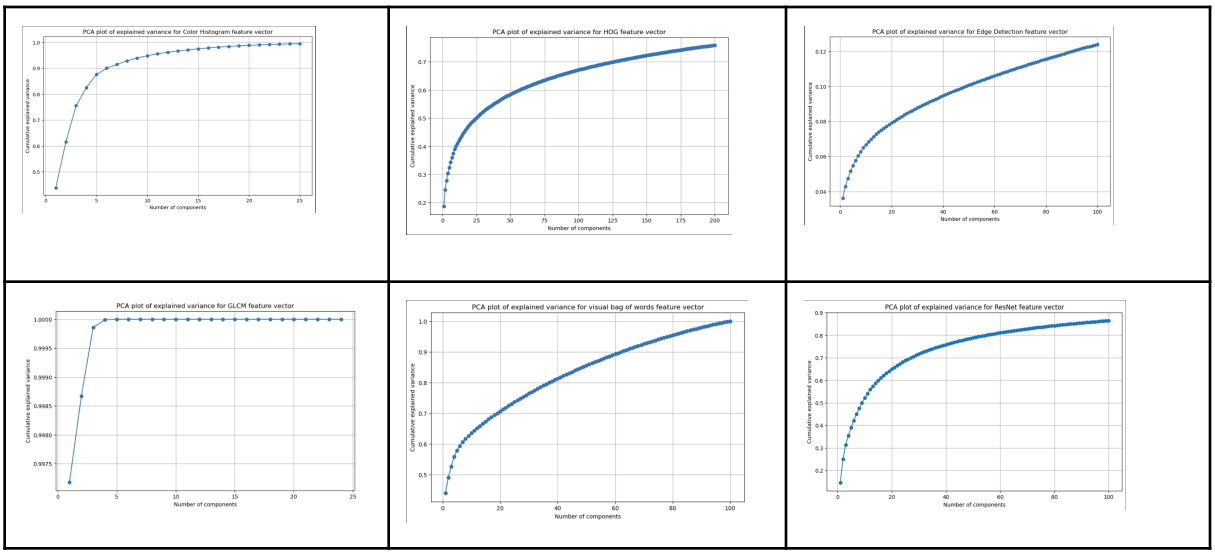

Figure 10: *PCA plots from left to right: Color Histogram, HOG, Edge detection, GLCM, Visual bag of words, ResNet*

As seen in the charts, 3 of the feature vectors reached 100% explained variance in a reduced number of components. These were the color histogram, GLCM and visual bag of words features. Although this general indicates that these are features that will be highly effective for classification, the GLCM plot indicated that all of the variance was condensed into only 3 components. We found that GLCM alone was not particularly effective at class differentiation (see t-sne plot below), but it was helpful when included in the model.

HOG and ResNet both reached at least 75% explained variance, however, because these were lower than some of the other feature vectors, concatenating them with the other features would have lowered the performance of the model, thus they were excluded from the model.

In a more extreme case, the edge detection feature only explained 13% of the variance. The low utility of the edge detection may be explained by the fact that cells borders have natural, somewhat fractal edges to them which may have introduced too much noise into the data, thus making them an ineffective feature, which we ultimately also excluded from the model.

Feature	Purpose	% Explained Variance	# Components
Edge Detection	Boundary Extraction	13%	100
Color Histogram	Color Distribution	100%	25
HOGs	Shape Recognition	75%	200
GLCM	Texture Analysis	100%	4
VBOW	Keypoint Clustering	100%	100
ResNet	Deep Feature Extraction	87%	100

Table 1: *Comparison of different feature vectors. Features in red were excluded.*

In the t-SNE plots shown below, the strengths of the color histogram and visual bag of words are shown especially in their ability to cluster class 5 (represented by dark blue) where many of the other features struggled with this. Class 5 was one of the more frequently mis-identified classes (see results section), so the features that were best at clustering it were our strongest.

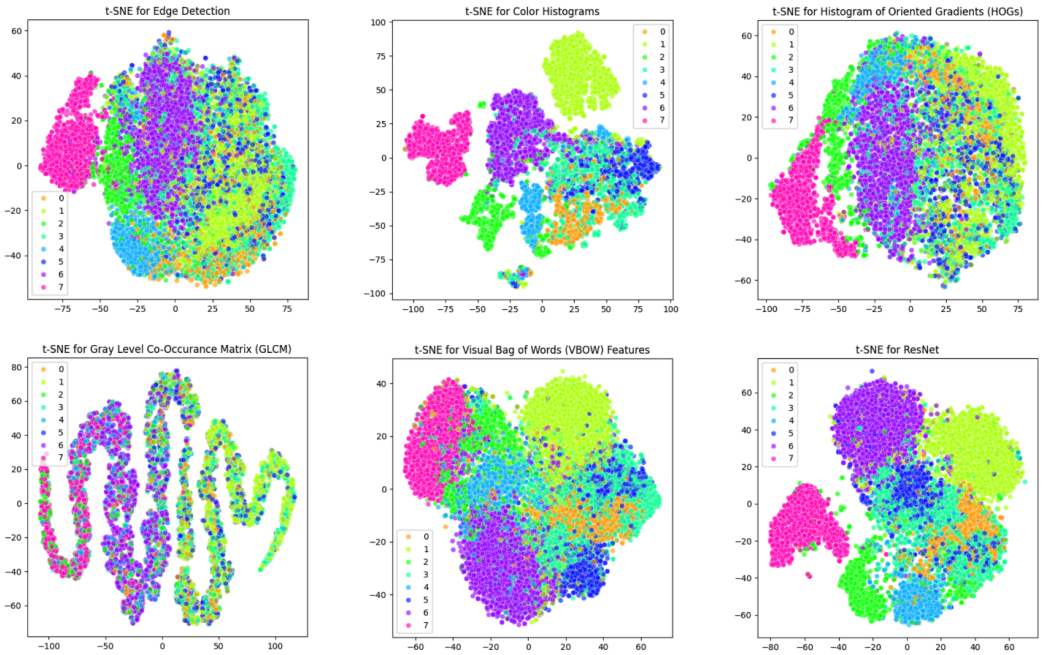

Figure 11: *tSNE plots from left to right: Color Histogram, HOG, Edge detection, GLCM, Visual bag of words, ResNet*

Lastly, we plotted the pixel counts and ratios by the average for each class and observed that we could obtain fairly good differentiation between classes, particularly for classes 2, 6, and 7.

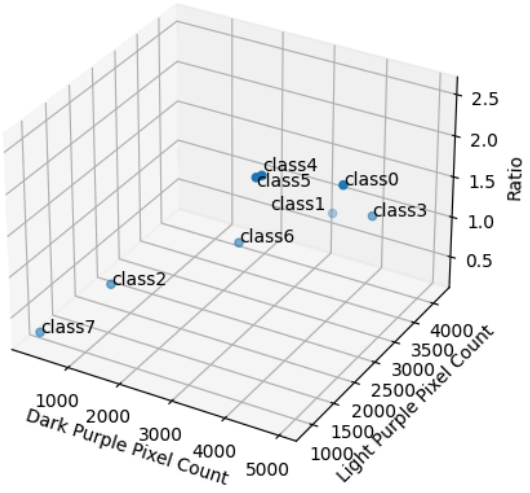

Figure 12: *Averages of light and dark pixel counts and ratio by class*

Model building

After testing various feature permutations, we finalized our feature vector: Color Histograms, GLCM, Pixel Count Ratio, and Visual Bag of Words.

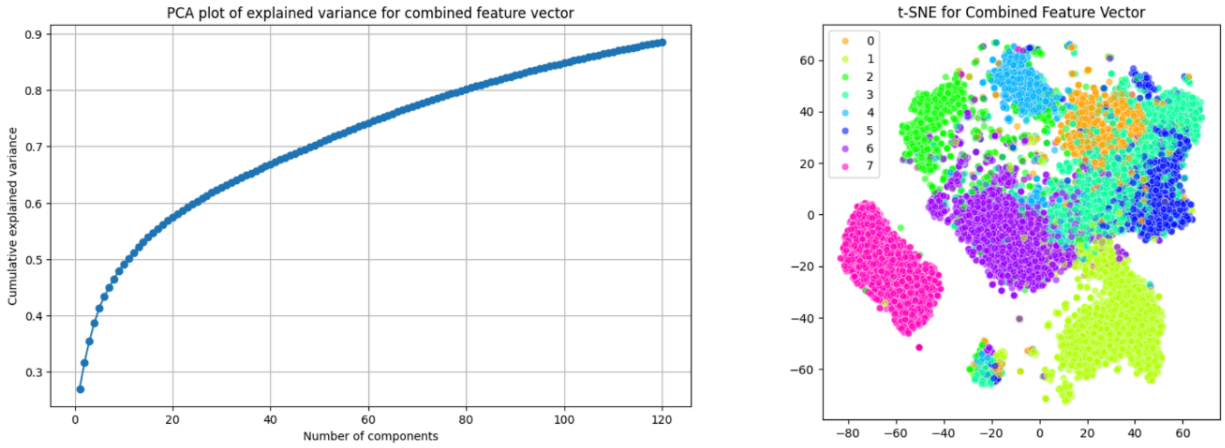

Figure 13: *PCA and tSNE for features Color Histogram, GLCM, Pixel count ratio and Visual Bag of Words*

The figures above show the PCA plot and t-SNE visualization for our combined feature vector. In 120 PCA components, we are explaining close to 90% of the variance, which is notably less than the 100% explained variance we got with a more parsimonious model, using only simple features. It's therefore important to note the difference between explained variance and discriminative power: while PCA aims to maximize the explained variance with the principal components by capturing the most significant patterns in the data, it does not mean those patterns are necessarily the most relevant for distinguishing

between classes. Despite the 90% explained variance from our PCA plot, our t-SNE visualization illustrates the great separation we see between classes, with only a couple classes with boundaries that appear to be a bit more ambiguous (i.e., class 3 and 5).

In the following section, we will discuss the classification results of our combined feature vector.

Model Architectures Utilized

Logistic Regression: Logistic Regression is a simple yet effective classifier that models the probability of different classes by using a multinomial logistic function. It is suitable for linearly separable problems where there is a linear relationship between the independent and dependent variables. It therefore serves as a good baseline model due to its simplicity and ability to help us determine the complexity of our problem.

Random Forest Description: Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes of the individual trees. It tends to handle non-linear relationships well.

XGBoost Description: XGBoost, or Extreme Gradient Boosting, is an optimized distributed gradient boosting model designed to be highly efficient, flexible, and portable. It uses a more regularized model formalization to improve performance over standalone decision trees.

SVM (Linear) Description: Linear SVM is a powerful linear classifier that aims to find the optimal hyperplane that maximizes the margin between different classes in a dataset. It is particularly effective in high-dimensional spaces and is robust against overfitting.

SVM (RBF) Description: The SVM with Radial Basis Function (RBF) kernel is a non-linear classifier that maps input features into higher-dimensional spaces using the RBF kernel. This allows the model to handle non-linear relationships between the features and the target classes effectively.

Dealing with Overfitting

During our initial experiments, we encountered overfitting issues, particularly with our tree-based models, Random Forest and XGBoost, which showed nearly perfect accuracy on the training set but lower accuracy on the validation and test sets. To combat overfitting, we reduced the number of PCA components, which was effective in lowering the dimensionality of the feature space, but it came at the expense of reduced classification accuracy, illustrating the delicate balance between model complexity and generalization.

Hyperparameter Tuning

As illustrated by Tables 2 and 3 below, we employed GridSearchCV to fine-tune hyperparameters across all our models. This process involved iterating over a predefined set of parameter values and using accuracy as the evaluation metric to identify the optimal settings. Through hyperparameter tuning, we sought to strike a balance between model complexity and predictive power.

Although hyperparameter tuning didn't fully eliminate overfitting in tree-based models, it improved model robustness, with SVM (RBF) achieving the best balance between accuracy and generalization.

Results

We trained a diverse set of linear and non-linear classification models to achieve optimal performance and evaluate the complexity of our dataset. The models included: (1) Logistic Regression (Baseline), (2) Random Forest, (3) XGBoost, and (4) Support Vector Machines (SVMs).
Additionally, we utilized GridSearchCV to optimize the hyperparameters of our models by iterating over a predefined set of parameter values and using accuracy as the evaluation metric to select the best value:

Model	Parameter	Search Values	Best Value	Best Accuracy	Search Time (Seconds)
Logistic Regression	max_iter	[100, 500, 1000, 2000]	500	91.6%	13.2
Random Forest	max_depth n_estimators	[2, 4, 6, 8, 10] [10, 50, 100, 200]	10 200	89.4%	385.6
XGBoost	max_depth n_estimators	[2, 4, 6, 8, 10] [10, 50, 100, 200]	4 200	92.8%	274.3
SVM (Linear)	C	[1, 2, 5, 10]	1	85.3%	13.2
SVM (RBF)	C	[1, 2, 5, 10]	10	93.4%	32.0

Table 2: Hyperparameter Tuning Details

Using the "Best Value" hyperparameters from above, we trained each model on our train (70%), validation (10%), and test (20%) sets.

Model	Train Accuracy		Validation Accuracy		Test Accuracy	
	Before Tuning	After Tuning	Before Tuning	After Tuning	Before Tuning	After Tuning
Logistic Regression	94%	94%	93%	93%	92%	92%
Random Forest	99%	97%	93%	92%	91%	91%
XGBoost	99%	99%	94%	94%	94%	95%
SVM (Linear)	88%	89%	87%	87%	87%	87%
SVM (RBF)	96%	98%	94%	95%	94%	95%

Table 3: Model Accuracy Pre/Post Tuning

Confusion Matrices

The figure below shows confusion matrices for each of our models, after hyperparameter tuning. Overall, we see dark blue squares along the diagonal, which represents our high true positive rate. The annotations are displayed as a percentage so we can easily compare the model accuracy across classes. We can see our models often confuse the Immature Granulocytes (class 3) and Monocytes (class 5). However, our nonlinear models, SVM RBF in particular, did much better at distinguishing between these two classes, achieving over 90% accuracy for all eight of our classes.

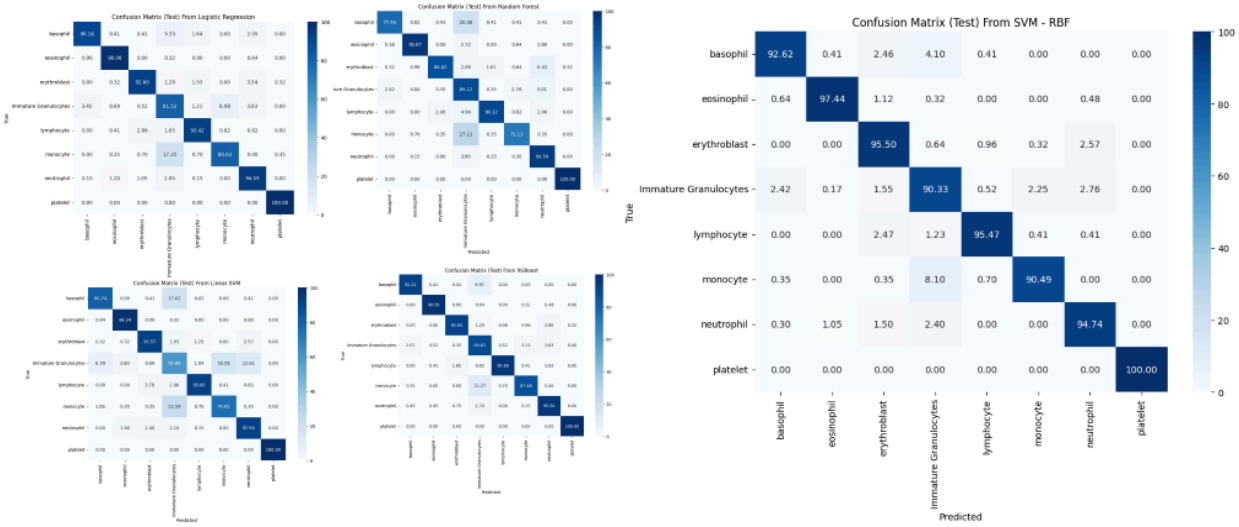

***Figure 14**: Confusion matrices of Logistic Regression, Random Forest, XGBoost, and linear SVM (left) and our best model, non-linear SVM (right)*

Logistic Regression (Baseline)

Results: Logistic Regression performed consistently across the training, validation, and test sets with an accuracy of 94%, 93%, and 92%, respectively, before and after hyperparameter tuning. This stability

indicates that Logistic Regression is robust and generalizes well to unseen data. Despite its simplicity, it provided a solid baseline for comparison with more complex models.

Tree-Based Models: Random Forest and XGBoost

Results: Random Forest achieved high accuracy on the training set (99% before tuning, 97% after tuning), indicating its ability to learn complex patterns. However, it showed a decrease in validation accuracy (93% before tuning, 92% after tuning) and test accuracy (91% both before and after tuning), which suggests the model may be overfitting. Despite this, it performed well, but the additional complexity and training time does not increase the classification accuracy beyond our baseline model.

Results: XGBoost displayed excellent performance across all datasets, achieving 99% accuracy on the training set both before and after tuning. It also showed impressive validation accuracy (94% both before and after tuning) and test accuracy (94% before tuning, 95% after tuning). The slight improvement post-tuning highlights the model's ability to generalize well while effectively handling complex, non-linear relationships in the data. Similar to our random forest classifier, the overfitting observed in the training set suggests the need for careful regularization and parameter tuning.

Support Vector Machines

Results: The Linear SVM achieved 88% training accuracy before tuning and 89% after tuning, with validation and test accuracies remaining at 87% both before and after tuning. These results indicate that the Linear SVM model struggled with the non-linearity in the dataset, leading to relatively lower performance compared to both non-linear models and even our logistic regression baseline.

Results: The RBF SVM outperformed the Linear SVM significantly, achieving 96% training accuracy before tuning and 98% after tuning. It also demonstrated strong validation accuracy (94% before tuning, 95% after tuning) and test accuracy (94% before tuning, 95% after tuning). The RBF SVM's performance underscores its ability to handle the non-linear nature of the dataset effectively, making it one of the top-performing models.

Discussion

While Logistic Regression, our linear baseline, performed well, our non-linear models, particularly SVM with the RBF kernel, outperformed it in terms of accuracy, generalizability, and efficiency. The SVM (RBF) model proved to be the most effective, avoiding the overfitting issues that plagued our tree-based models while simultaneously achieving the highest test accuracy of 95%. This kernel's ability to create non-linear decision boundaries allowed it to differentiate between classes that even trained professionals might struggle to distinguish. Our results suggest that the BloodMNIST classification problem is largely linear, with some non-linearities, as indicated by the superior performance of non-linear models and t-SNE visualizations.

Furthermore, the success of our classifier was heavily reliant on a combination of binary thresholding, strategic feature engineering, and rigorous model selection and hyperparameter tuning. Key pre-processing steps, such as reducing background noise, and the thoughtful selection of feature sets—like color histograms, GLCM, light to dark cell count ratio, and Visual Bag of Words—were instrumental in achieving high performance. Although tree-based models struggled with overfitting, this experience highlighted the crucial balance between model complexity and generalizability. Future efforts could focus on enhancing the robustness of these models, possibly through data augmentation techniques, to further improve their performance on unseen data.

Efficiency

From the table below, we can see that our linear SVM model trained in the least amount of time, under ¾ of a second. However, this model also performed the worst, highlighting the tradeoff between accuracy and efficiency. Our tree based models, which performed better, generally took longer to train, while our SVM model with the RBF kernel showcases a nice balance between training time and accuracy, training in under 1.5 seconds. That said, this model did take the longest to predict at around 3.3 seconds on the training set.

Model	Training Time (Seconds)	Prediction Time (Seconds)		
		Training Set	Validation Set	Test Set
Logistic Regression	1.1313	0.0030	0.0010	0.0010
Random Forest	16.1569	0.1980	0.0440	0.0740
XGBoost	4.1701	0.0620	0.0110	0.0180
SVM (Linear)	0.6892	0.7782	0.1120	0.2230
SVM (RBF)	1.2453	3.3376	0.4791	0.9570

Table 4: Model Efficiency

Generalization

Ensuring the generalizability of our models was a critical focus throughout our experimentation process. We adhered to best practices by splitting our data into training (70%), validation (20%), and test (10%) sets, ensuring that each set was stratified at the patient level. This stratification maintained an even distribution of samples across all 8 classes, which is essential for robust model evaluation. By keeping the

class distribution consistent across all sets, we minimized the risk of bias and ensured that our models could accurately generalize to unseen data. The test set was treated as a true ‘hold out’ set and reserved solely for final evaluation, allowing us to make informed decisions regarding feature selection and model tuning based on the performance observed in the training and validation sets.

Conclusion

In this project, we developed a robust image classifier that accurately distinguishes between 8 classes of blood cells with a high degree of accuracy. By effectively leveraging data pre-processing, feature engineering, and careful model selection, we achieved a 95% accuracy across various classification models. Our best-performing model, the SVM with an RBF kernel, demonstrated superior generalizability and efficiency, making it a strong candidate for assisting medical professionals in diagnosing and treating various endocrine and blood-related diseases. The success of this classifier underscores the potential for machine learning to play a pivotal role in enhancing healthcare outcomes, offering a powerful tool that can contribute to more accurate diagnoses and better treatment plans, ultimately improving patient care and advancing public health.