Automatic Classification of Circulating Blood Cell Clusters based on Multi-channel Flow Cytometry Imaging

by

Subhadeep Sengupta

(Under the Direction of He Li)

Abstract

Standard clotting tests usually do not suffice in investigating the mechanisms of thrombosis in COVID-19 patients. Flow cytometry provides a superior alternative since it enables research on sub-cluster cellular components from blood samples. Manual gating has conventionally been used to detect and analyze such sub-clusters, but it is labor-intensive and prone to subjective mistakes on behalf of human experts. This work proposes a deep learning-based method that automatically identifies and classifies cellular clusters to facilitate more efficient analysis of immuno-thrombosis. The method consists of two stages. The first stage employs a customized convolutional neural network (CNN) to classify grayscale images into clusters and non-clusters, with a test accuracy of 87.5%. The second phase focuses on multi-cell cluster images, using pre-defined color-based criteria to identify sub-cluster elements and obtain 85.6% accuracy. In addition, the viability of few-shot learning using ChatGPT 4o is explored, followed by a comparison with the customized CNN, the pre-trained deep learning models, and machine learning classifiers. The customized CNN outperformed the ML classifier and ChatGPT 4o and was outperformed by pre-trained deep learning models. Grad-CAM visualizations were used to perform a misclassification analysis of the CNN-based model predictions to enhance the interpretability of test results.

Index words: Machine Learning, Medical Imaging, Image Classification, Neural Networks

Automatic Classification of Circulating Blood Cell Clusters based on Multi-channel Flow Cytometry Imaging

by

Subhadeep Sengupta

B.Tech., National Institute of Technology Karnataka, India, 2022

A Thesis Submitted to the Graduate Faculty of the
University of Georgia in Partial Fulfillment of the Requirements for the Degree.

Master of Science

Athens, Georgia

2025

Automatic Classification of Circulating Blood Cell Clusters based on Multi-channel Flow Cytometry Imaging

by

Subhadeep Sengupta

Major Professor: He Li

Committee: Tianming Liu
Frederick Maier

Electronic Version Approved:

Ron Walcott
Dean of the Graduate School
The University of Georgia
August 2025

Dedication

For Mumma and Babai.

Acknowledgments

I want to thank Dr. He Li for his invaluable support with the whole project and being there to guide me through every difficulty I faced on the way. I am also deeply grateful to Dr. Tianming Liu and Dr. Frederick Maier for their advice and constructive feedback.

I am truly honored to have had the opportunity to receive the support, encouragement, and mentorship of my committee. Thank you for your time and for believing in me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Aim and General Overview

Existing flow cytometry methods can prove useful for identifying circulating cellular clusters. Applying an automated classification algorithm to such data can be an efficient solution, allowing for a deeper study of the relationship between relevant cell aggregates and immunothrombosis in COVID-19 patients compared to a healthy patient.

This thesis explores two methods; the first employs a simple convolutional neural network architecture adapted to handle the multi-channel input from the flow cytometry data and with a relatively low amount of labelled data to offer reasonable results. The second method uses ChatGPT and context-based image understanding to classify the images. The first method is further split into two parts. In the first part, single-cell images and multi-cell clusters are separated using channel 1 grayscale images. In the next part, we use the color histogram features of the multi-cell cluster images to sort images into their respective classes using predefined criteria using the different channels (2, 3, 7, and 11). The goal is to take flow cytometry images and classify them into,

- Single cells images,
- Multiple cells consisting only of RBCs,
- Multiple cells with RBCs and platelets,

- Multiple cells with WBCs which may or may not have platelets,

Doing this reliably in an automated fashion speeds up the diagnosis process, whereas earlier experts would have to manually gate images.

1.2 Brief Summary of Results

For grayscale classification, a customized CNN with minimal layers gave an 87% accuracy and 0.95 ROC-AUC score on the test set. The best-performing model turned out to be EfficientNet, with 93.8% accuracy, while GPT 4o has an accuracy of 76%. The machine learning algorithms outperformed GPT 4o but were inferior to the pre-trained models and simple CNN.

Experiments with GPT 4o showed different results based on the approach to the input. For instance, a batch of images was effective only if the LLM's short-term memory was configured with the proper rules and training examples. The best results were obtained by reminding GPT 4o of a summary of the rules and a few training examples every 5-10 batches of input images.

For color classification, the customized Multi-Channel CNN had an 85.6% accuracy and a 0.948 ROC-AUC score, being more effective than the machine learning algorithms, namely Random Forest, SVM, etc., having roughly 68% accuracy and 0.69 ROC score. EfficientNet was again the best-performing model with 88% accuracy and 0.97 ROC-AUC score.

1.3 Immunothrombosis due to COVID-19 and Treatments

Thrombosis occurs in two kinds of states, the first being physiological, where the defense mechanism is properly regulated, and pathological states include conditions such as sepsis or autoimmune diseases where the mechanisms fail to be regulated appropriately, leading to excess clotting.

Thrombosis associated with COVID-19 was demonstrated to be closely related to morbidity and mortality [1]. Hanff et al. [2] attempted to explore the biomarkers that might potentially help with the prognosis of such thrombotic complications; they observed a cytokine storm, a severe immune reaction where excess cytokines are released, and high amounts of the IL-6 molecule in COVID-19 patients.

Jayarangaiah et al. [3] showed the clotting mechanism in patients with COVID-19, with the virus promoting clot formation by damaging the inner walls of blood vessels, which would trigger cytokine production and leukocytes (white blood cells) and platelets; these would further lead to complications like deep vein thrombosis (VTE) or blocked arteries. Dorken-Gallastegi, Ander, et al.[4] studied the role of circulating cell clusters in the immunothrombotic states that arise in patients with COVID-19; the motivation was that while previous work had studied platelet-leukocyte aggregates, others, such as platelet-erythrocyte aggregates (PEAs) and circulating leukocyte clusters (CLCs) had not received similar scrutiny; they found that PLA and PEA aggregates were present in noticeably higher levels in COVID-19 patients while CLC aggregates were not as distinguishable in healthy patients and COVID-19 patients. However, there was a link between worse outcomes for CLC aggregates and COVID-19 patients. Another link was bacterial infections causing higher levels of PEA aggregates during some patients' hospital stays.

Marcos-Jubilar[5] highlighted that traditional blood thinning or anticoagulant strategies are risky due to the possibility of bleeding; other treatments attempt to deal with inflammations that can reduce the risk of bleeding while treating clots. Tissue factor is a protein present on cells outside of the bloodstream; during vascular injury, it comes in contact and binds with a plasma protein called Factor VII, which initiates the clotting process[6], so some strategies involve suppressing TF expression and have shown some promise. Inflammasomes are protein complexes that, depending on the signal, activate and initiate inflammation via the release of cytokines, as studied by Li et al. [7]. Inhibitors for inflammasomes are another area that has been explored. Neutrophils are white blood cells that form Neutrophil Extracellular Traps (NETs), web-like structures that trap pathogens, preventing their spreading [8]. Treatments involving blocking harmful NETs from forming were also discussed in [5].

1.4 Techniques for studying cell clusters

Flow cytometry is a method of analyzing cells by passing single or multiple lasers through them. Bonner et al. [9] introduced the procedure, which involved taking a sample of blood cells and suspending them in a salt-based fluid, passing lights of different wavelengths through them to analyze scattering patterns.[10] Flow cytometry has proved useful when providing a heterogeneous sample containing various kinds of cells or molecules and can distinguish them.

While there are many different kinds of flow cytometers based on instrumentation, this thesis focuses on data from imaging flow cytometry. Schneck et al. [11] using flow cytometry studied Neutrophil Extracellular Traps (NETs) levels to check for a distinction between septic shocks and post-surgical inflammation and understand its relationship to coagulation compared to a healthy response. Heestermans et al. [12] studied the role of platelets in venous thrombosis, which could help develop antiplatelet therapies for treating Venous Thromboembolism (VTE).

1.4.1 Gating

Gating is a process in which a cell group or population is identified from a heterogeneous sample. Traditionally, this is done manually by experts and is considered to be a time-consuming process. Verschoor et al. [13] proposed an automatic gating method using software they named FLOCK, which partitions data into grids and analyzes the density distribution of cell clusters. Liu et al. [14] analyzed three different ways of gating and the various solutions under each type. They concluded that manual gating works on small datasets or when flexibility is needed, unsupervised clustering is used for more complex and large datasets, and supervised automatic gating works well for predefined tasks but not so much for exploratory analysis.

Eslami et al. [15] proposed a deep learning model, AutoGater, to identify healthy cells in flow cytometry data without fluorescent stains. Traditionally, staining like Sytox is needed to identify dead or dying cells, which delays experiments and uses up valuable fluorescence channels. AutoGater instead uses only light-scatter data to separate viable from non-viable

cells. Fisch et al. [16] presented a neural network, GateNet, to fully automate the gating process in flow cytometry. The model was trained on over 8 million events from peripheral blood and cerebrospinal fluid samples, and matched human expert performance with F1 scores between 0.910 and 0.997 on unseen data. It also generalized well to public datasets, requiring only 10 labeled samples to perform at an expert level. The processing time for each cell was about 15 microseconds using a GPU. For their thesis, W Sriphum [17] introduced FLOPTICS, an automated method for gating flow cytometry data that combined density and grid-based clustering to offer faster cell classification with fewer user-defined parameters, outperforming other state-of-the-art tools.

The proposed methodology in this thesis also avoids this problem by using the images obtained directly for inference.

1.5 Machine Learning on Flow Cytometry Imaging

Analysis of Flow Cytometry imaging is split based on the number of cells present in the image. For multiple cells in the image, detection and tracking become harder. Single Shot Detector models are used for cell detection. Real-time CNN-based models have been recommended for identifying microbeads and cells [18]. Monaghan et al. [19] used flow cytometry data to distinguish acute leukemias from nonneoplastic cytopenias. The model framework was split into two stages. The first stage used a Gaussian Mixture Model (GMM) to get features on a higher dimension, and the second stage used a Support Vector Machine for a 4-class classification, and grid search was used to optimize the model. Cohen et al. [20] presented a label-free imaging flow cytometry technique that captured multiple holographic views of rotating cells, using a modified ResNet-18 model to accurately distinguish between cell types, namely, cancerous, healthy breast cells, and various white blood cells by analyzing their 3D phase profiles. Using five interferometric projections instead of one projection improved accuracy by 1%. This approach also reduced the number of training samples needed. Lewis et al. [21] presented a deep learning approach using attention-based multi-instance learning models (ABMILMs) to automate the diagnosis of acute myeloid leukemia using flow cytometry data. Compared to traditional AML manual analysis and slow molecular

tests, this method better detected AML and distinguished it from other leukemias. Cheng et al. [22] trained a deep learning model on data from 241 patients using the EuroFlow ALOT protocol; the AI achieved a sensitivity of 94.6% for detecting AML and 98.2% for B-ALL and performed reasonably well in identifying healthy cells. Models like ResNet-50 combined with EverFlow were shown to perform efficient leukemia screening and cell classification. Cheng, Zhangkai et al. [23] developed a machine learning-based tool to help detect childhood leukemia early by analyzing blood biomarkers, including nutritional and immune-related indicators. Comparing data from children with ALL, AML, and healthy controls, the model identified key differences and achieved an AUC score of 0.950 for predicting leukemia types and 0.909 for AML specifically. Hybel et al. [24] used imaging flow cytometry and deep learning to distinguish leukemia stem cells (LSCs) from healthy stem cells (HSCs) in AML based on cell morphology. CNNs trained on brightfield, side scatter, and DNA images achieved up to 93% accuracy in identifying LSCs. Performance varied in patients due to LSC heterogeneity, but the proposed approach is feasible given that it could monitor AML without relying on a single molecular marker. Vora et al. [25] proposed a deep learning method for detecting circulating tumor cell clusters (CTCCs) in whole blood using confocal backscatter and fluorescence flow cytometry (BSFC). While traditional methods work outside the body, this method works in real-time and does not require labels to perform at high accuracy and a low false alarm rate. The model achieved a 72% detection purity and 35.3% sensitivity and was validated across different species and cancer types using transfer learning.

Becht et al [26] used machine learning on multiple overlapping flow cytometry panel data to analyze proteins on the surfaces of cells. They used dimensionality reduction and clustering to simplify the high-dimensional data. Lippeveld et al. [27] attempted to explore flow cytometry data without relying on stains. They tested two approaches, the first with traditional machine learning algorithms using custom features and deep learning algorithms that would automatically learn features, and they found traditional machine learning algorithms to outperform the latter by a slight edge. Park et al. [28] introduced a deep learning-enhanced image cytometry (DLIC) method to study treatment response in patients with aggressive extranodal NK/T cell lymphoma (ENKTL). Researchers quantified key biophysical features across treatment stages by analyzing about 270,000 peripheral blood mononuclear cells from

23 patients using a label-free, high-throughput optical system. They found distinct patterns linked to disease progression and relapse, and created a 3D single-cell map to standardize these changes.

Lemieux et al. [29] introduced an early lung cancer detection pipeline that is non-invasive, using only the sputum samples and flow cytometry data along with machine learning algorithms for a classification model. Rosenberg et al. [30] used flow cytometry to analyze cell abnormalities for improved diagnosis of myelodysplastic syndrome (MDS), which is a type of blood cancer where the bone marrow produces premature blood cells instead of healthy ones. Machine learning algorithms were used to identify binucleated erythroblasts (BNEs), and their occurrences were compared across patient groups. Wilkins et al. [31] compared the performance of different clustering algorithms for phytoplankton classification on flow cytometry data.

Li, Yueqin et al. [32] presented an improved deep learning pipeline that avoids traditional signal processing and feature extraction. Instead, a CNN is trained directly on raw time-stretch data, speeding up classification to a few milliseconds, and fast enough for real-time cell sorting. They demonstrate the effectiveness of this approach by distinguishing between OT-II white blood cells and SW-480 cancer cells with over 95% accuracy without labels. Eulenberg et al. [33] showed that deep CNNs combined with nonlinear dimensionality reduction can reconstruct biological processes and disease progression directly from raw image data. The approach outperforms traditional methods in imaging flow cytometry, is unsupervised, and has good accuracy. A Gupta et al. [34] reviewed deep learning applications in microscopy images of cells and tissues, explaining important neural network aspects through a neuroscience analogy. Liu et al. [35] introduced high-content video flow cytometry (VFC), a label-free method for imaging single cells at high throughput without disrupting cell function; achieving over 90% accuracy in distinguishing three cervical cancer cell lines. Bini et al. [36] proposed FlowCyt, a publicly available benchmark for multi-class and single-cell classification on flow cytometry data. It includes bone marrow samples from 30 patients, each cell labeled as one of five key hematological types. It supports supervised and semi-supervised learning up to a million cells per patient.

1.6 Pretrained Models

1.6.1 ResNet

Residual Networks [37] use skip connections to deal with the vanishing gradient problem. Vanishing gradients occur when backprop gradients become too small, leading to ineffective learning if there are too many layers. Having skip connections allows for deeper model architectures without sacrificing performance.

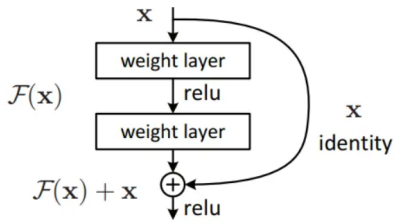


Figure 1.1: Diagram of a residual learning block used in the ResNet. The input x passes through two weight layers and a ReLU activation function. The output of these layers, denoted $\mathcal{F}(x)$, is then added to the original input x through a skip connection; this helps preserve the original information and reduce vanishing gradients. Reproduced from He et al.[37]

Residual Blocks pictured in Figure 1.1 consist of skip connections where the output of a layer is regularized with the input from the previous layer, any layer that degrades performance too much is avoided. The architecture has multiple variants, namely, ResNet-18, ResNet-34, ResNet-50, and ResNet-101, each with increasing layers. These deeper versions use bottleneck residual blocks to improve efficiency.

Ma et al. [38] proposed using a DC-GAN and ResNet for a WBC classification framework with a modified loss function. Zhu, Ziquan, et al. [39] proposed RDNet, a model with ResNet-18 as the backbone with dropout for automatic classification of four blood cell types achieving 86.53% accuracy and outperforming the standard ResNet-18. Farooq et al. [40] introduced COVID-ResNet leveraging progressive resizing and automatic learning rate selection to classify COVID-19, pneumonia, and normal cases, achieving 96.23% accuracy in about 41 epochs.

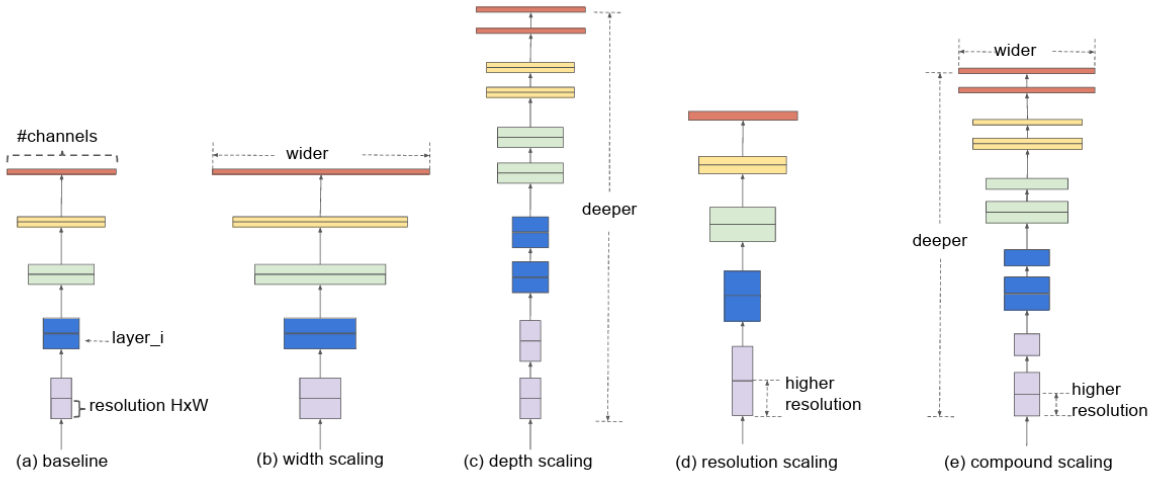


Figure 1.2: Illustration of different scaling strategies used to modify convolutional neural network architectures. (a) The baseline model shows the original network structure in depth, width (number of channels), and input resolution. (b) Scaling by width, increasing number of channels at each layer. (c) Scaling by depth, increasing number of layers. (d) Resolution scaling, increasing input image resolution. (e) Compound scaling simultaneously balances depth, width, and resolution according to a fixed scaling coefficient, aiming to improve overall model performance and forming the basis for EfficientNet. Reproduced from Tan et al.[41]

1.6.2 EfficientNet

EfficentNets [41] uses compound scaling, which accounts for and optimizes network depth (for layers), width (for filters), and resolution (for image size). Scaling is done in a constant ratio across the different aspects.

$$\begin{aligned} \text{depth: } & d = \alpha^{\phi} \\ \text{width: } & w = \beta^{\phi} \\ \text{resolution: } & r = \gamma^{\phi} \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \quad \beta \geq 1, \quad \gamma \geq 1 \end{aligned} \tag{1.1}$$

In Figure 1.2, we see the scaling done across the different aspects. The formula for this is shown in Equation 1.1. The ϕ refers to the uniform scaling ratio, and the values for α, β, and γ are found using a grid search algorithm.

Batool et al. [42] proposed a lightweight EfficientNet-B3 model using depthwise separable convolutions for acute lymphoblastic leukemia (ALL) classification. It outperformed existing deep learning methods and demonstrated strong generalization for leukemia detection. Ramamurthy et al. [43] proposed an EfficientNet-B7-based deep learning model for automated Gleason grading of prostate cancer. The model was trained on the Harvard Dataverse dataset, namely the H&E-stained samples from prostate cancer Tissue Microarrays(TMAs); it outperformed state-of-the-art and achieved a Kappa score of 0.5775. Ravi et al. [44] proposed a stacked ensemble EfficientNet for lung disease detection, achieving 99% accuracy for TB and 98% accuracy for COVID-19 and pediatric pneumonia.

1.6.3 MobileNetV2

MobileNet as a model was originally designed to be lightweight and for mobile and edge devices with a keen focus on improving computational efficiency. MobileNetV2 [45] uses inverted residuals with a linear bottleneck to reduce information loss; this is done by avoiding nonlinear transformations on the features when compressed into lower dimensions.

Inverted residual blocks and depthwise separable convolutions are used to reduce computational cost without compromising accuracy.

Compared to regular residual blocks that pass information through identity shortcuts, it reverses this structure by using:

- Pointwise expansion: Projects input into a higher-dimensional space via a 1x1 convolution.
- Depthwise convolution: Applies shallow 3x3 convolutions independently to each channel.
- Pointwise projection: Uses another 1x1 convolution to drop dimensions back down to keep it computationally efficient.

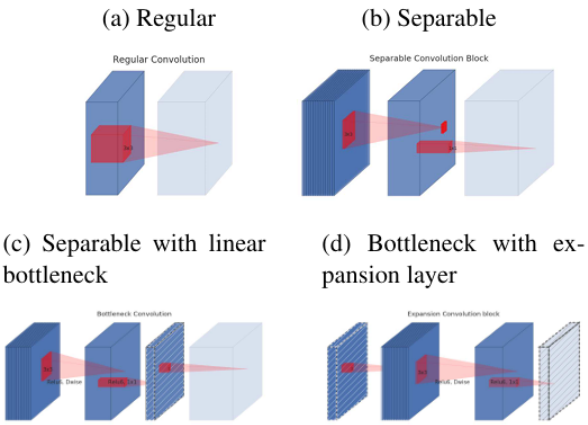


Figure 1.3: Comparison of regular convolution with optimized convolution blocks. (a) Regular convolution performs spatial and depth-wise filtering jointly using a standard kernel. (b) Separable convolution splits the spatial and channel-wise operations; each channel goes through depthwise convolution, and then a 1x1 pointwise convolution combines the channels, reducing computation. (c) Separable convolution with a linear bottleneck removes the ReLU activation and preserves more information by avoiding nonlinear transformation. (d) Bottleneck with expansion layer (used in MobileNetV2) increases the input dimensions before performing the separable convolution to learn richer representations while keeping the computation efficient. Reproduced from Sandler et al.[45]

- Linear bottleneck: Unlike ReLU in regular residual connections, linear activation is used at output to prevent information loss while working in low-dimensional space.

This inverse residual block helps pass through thin bottleneck layers, making MobileNetV2 efficient and powerful for mobile vision tasks.

Akay et al. [46] proposed a mobile deep learning network using UNet and MobileNetV2 for Systemic Sclerosis (SSc) skin classification, achieving 95.2% and 97.2% accuracy on two datasets, outperforming traditional CNNs. Kumar et al. [47] used MobileNetV2 for the classification of WBCs in blood smear images, achieving 94.0% accuracy in identifying five WBC subtypes. Other models, namely VGGNet, were also tested. Ragab et al. [48] tested MobileNetV2 with transfer learning for COVID-19 detection from chest X-rays, reporting up to 100% accuracy at a dropout rate of 0.4.

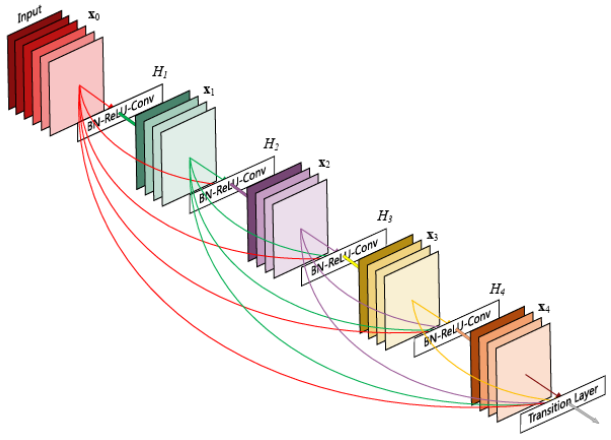


Figure 1.4: Structure of a typical 5-layer dense block used in the DenseNet. Each layer receives the feature maps of all previous layers via direct connections, enabling dense feature reuse and strengthening gradient flow. The concatenated outputs grow in length at each layer, and the dense connectivity pattern helps avoid the vanishing gradient problem. The block ends with a transition layer to reduce dimensionality before passing to the next block. Reproduced from Huang et. al[49]

1.6.4 DenseNet

DenseNets [49] have dense connections between layers, ensuring each layer receives inputs from all previous layers. Dense connections reduce the vanishing gradient problem, improve feature propagation, and minimize redundancies. Fewer parameters are required due to features being reused.

The Dense Block is a fundamental block of the DenseNet architecture, an example of which is shown in Figure 1.4. Its design allows for the following:

- Dense connectivity: Every layer receives input from all previous layers, which makes for efficient feature reuse and improved gradient flow.
- Bottleneck layers: 1x1 convolutions are used to reduce dimensionality before expensive 3x3 convolutions, making computation efficient.
- Compression: Transition layers use average pooling and 1x1 convolutions to control network growth by downsampling feature map size.

- Fewer parameters: Unlike ResNet, where summation is required in skip connections, DenseNet prevents redundant learning by explicitly reusing feature maps.

This dense connectivity provides the capacity for stronger feature propagation with fewer parameters than traditional architectures.

Houssein et al. [50] presented a DenseNet-161-based CAD system with cyclical learning rate (CLR) optimization for WBC classification, achieving 99.8% accuracy on blood smear images. Bozkurt et al. [51] tested the DenseNet121 for WBC classification; experimental results had it outperforming state-of-the-art models (Xception, VGG19, EfficientNetB1), achieving a 94% accuracy. Hasan et al. [52] proposed a DenseNet-121-based CNN for COVID-19 detection on CT images aiming for early diagnosis and achieved 92% accuracy and 95% recall.

1.7 Few-Shot Learning

1.7.1 ChatGPT

This thesis also explores using a large language model (LLM) such as ChatGPT [53][54] for the grayscale classification section. The first way ChatGPT reads images is on a pixel level, where it performs the standard image analysis; the second is a text-based understanding using an external image-to-text model whose insights it uses for further processing. The text-based understanding is generated in real-time as the user inputs images, and GPT works solely with this and any specific instructions provided.

Johnson et al. [55] carried out experiments prompting GPT with a variety of images and asked questions regarding the contents of the image and their understanding of it. They found that it can effectively explain the general context shown in the image and integrate it into a broader context, but it could sometimes fall short when attempting to tell specific details of the visual content. They also concluded by saying that in certain cases, human intuition does outperform GPT. Ren et al. [56] explored the use of the CLIP model. CLIP is a neural network introduced by OpenAI that is pre-trained on image-text pairs. The problems they found with CLIP were that they tended to favor certain classes and if some classes were

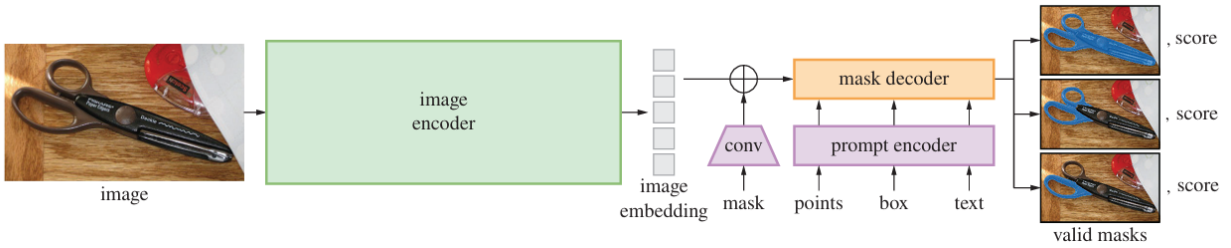


Figure 1.5: A high-level overview of SAM. The image encoder is a Masked Autoencoder that generates an image embedding. The prompt encoder accepts two categories of prompts; sparse (points, boxes, and text) and dense (masks) prompts combined with the embedding. The mask decoder consists of a lightweight transformer decoder that combines image embeddings with the prompt to generate multiple candidate segmentation masks. Reproduced from Kirillov et al. [58]

too similar, it had a hard time distinguishing them. They propose a solution that involves class hierarchy by building a knowledge tree for grouping and their descriptions generated by LLMs which improves its performance on state of the art. Wu et al. [57] introduced Visual ChatGPT, a combination of the LLM with Visual Foundational Models(VFM), allowing users to give both text and image input. Currently, its main issues are model inaccuracies and problems with prompting. One way to mitigate this was to implement a self-correcting module that checks if the system output is in sync with the user's intentions; however, this slowed the inference time.

1.7.2 Segment Anything Model

Meta AI released the Segment Anything model (SAM) [58] to achieve good zero-shot segmentation in generalized domains. The model was designed to perform this segmentation on an image using a prompt in the form of a bounding box, point, or mask.

The SAM model architecture is shown in Figure 1.5. There are three variants of the SAM model.

- ViT-Base is the lightweight version with about 91 million parameters.
- ViT-Large that has about 308 million parameters.

- ViT-Huge is the largest version with about 636 million parameters.

A text-based SAM was also tested with the help of Grounding Dino [59], a vision-language model that accepts a text prompt and generates bounding boxes that can then be used with SAM to perform effective segmentation.

Upon checking model performance on the training set, the results were not reliable enough, more of which is discussed in the results and then in the future work section.

1.8 Metrics

This thesis will use some metrics for clustering:

1.8.1 Homogeneity

After clustering, a cluster is homogeneous when all its points belong to its true class. The score between 0 and 1 indicates whether points are random or perfectly homogeneous, and this calculation is done by comparing the entropy of the class labels in each cluster.

1.8.2 Completeness

After clustering, completeness means checking for how well a certain true class was represented in clusters. If a true class is split across multiple clusters, then completeness is low. A perfect completeness score implies every true class fits perfectly into its cluster.

1.8.3 V-measure

This is the harmonic mean of homogeneity and completeness and gives a balanced measure of clustering performance based on cluster purity and true class completeness.

Additionally, for classification, the following metrics are used:

1.8.4 Accuracy

$$\frac{Correct\ Predictions}{Total\ Predictions} \tag{1.2}$$

This metric gives the correct predictions over total predictions; it works well if there is a balanced distribution of class labels and all labels are equally important to classify.

1.8.5 Precision

$$\frac{True\ Positives}{TruePositives + FalsePositives} \tag{1.3}$$

This metric measures how well the positive class was predicted, i.e., as few false positives as possible.

1.8.6 Recall

$$\frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{1.4}$$

This metric identifies how well the actual positive class is predicted even if it means taking on more false positives.

1.8.7 F1-Score

$$\frac{2 \times Precision \times Recall}{Precision + Recall} \tag{1.5}$$

This metric is the harmonic mean of precision and recall offering a good indicator of how well the model performs on specific class labels thus being ideal on imbalanced datasets.

1.8.8 ROC-AUC

This metric is a curve that plots the trade-off between the true positive rate and the false positive rate at different thresholds. The area under the curve gives a score on whether the model can tell apart positive and negative cases. A score of 1 is a perfectly accurate model awhile 0.5 is a model guessing at random.

1.9 Interpretability Analysis

1.9.1 Grad-CAM

Gradient-weighted Class Activation Maps (Grad-CAM) [60] help produce a visual explanation of why a CNN model predicts a certain class. This is done by using the gradients flowing into the last convolutional layer to make a coarse localization map, a low-resolution heatmap that highlights the regions of the image that the model is focusing on to make their prediction. The map is not pixel-precise because the activation layer at the last layer is usually of a much smaller resolution than the input image dimension.

Chapter 2

Dataset

The dataset was collected as part of a study to analyze circulating cell clusters (CCC) in COVID-19 patients [4]. Blood samples were collected from COVID-19 patients at the Massachusetts General Hospital (MGH) in July-August 2020; 46 samples were collected from COVID-19 patients, while 12 samples were collected from healthy controls. Images were obtained using the AMNIS ImageStream instrument. The imaging method was multi-channel fluorescence staining to highlight different cell components.

- CD61 – Green staining (channel 2) identifies platelets in cell clusters.
- CD45 – Yellow staining (channel 3) used for identification of leukocyte (white blood cell).
- DAPI - Purple staining (channel 7) for the nuclei (indicating cell count and viability).
- CD235a – Red staining (channel 11) distinguishes erythrocytes (red blood cells).
- Merged Composite Channel - Combined multiple channels for better feature representation.

The main classes saw the dataset divided into clustered and non-clustered cell images, with the subclass annotations being Platelet-Leukocyte Aggregates (PLAs), Platelet-Erythrocyte Aggregates (PEAs), Circulating Leukocyte Clusters (CLCs).

Table 2.1: Patient demographics and clinical outcomes in the COVID-19 cohort from the study (n = 37). Adapted from Dorken-Gallastegi et al.[4].

Characteristic	**COVID-19 Patients**
Age (median, IQR)	57 years (37–70)
Sex	61% Male, 39% Female
Ethnicity	30% Hispanic or Latino
Race	48% White, 22% Black or African American, 4% Asian, 26% Other
Thrombotic events	24%
Acute kidney injury (AKI)	52%
ICU admission	24%
Mortality	13%

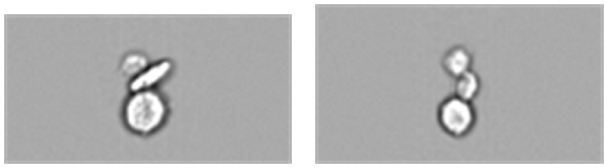
Table 2.2: Median cluster counts and interquartile ranges (IQR) in COVID-19 patients and healthy controls. Statistically significant differences were observed in PLA and PEA levels. Adapted from Dorken-Gallastegi et al.[4].

Cluster Type	**COVID-19**	**Healthy Controls**	**p-value**
PLA	131.93 (IQR: 66.67–210.42)	53.61 (IQR: 48–68.50)	0.002
PEA	24.12 (IQR: 18.26–32.03)	10.82 (IQR: 0.85–2.23)	<0.001
CLC	No difference	No difference	n.s.

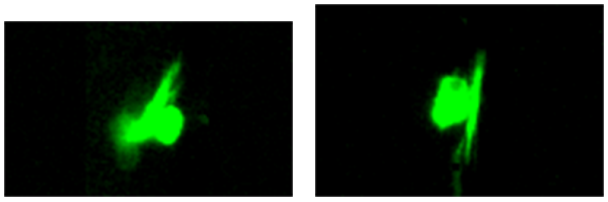
[1]

Figure 2.2 shows example images in the dataset with the channels stacked next to each other. These images are split into individual channel images for separate tasks, as shown in Figure 2.1. The images in the data set do not have uniform dimensions and thus require generalized splicing and processing to make them uniform.

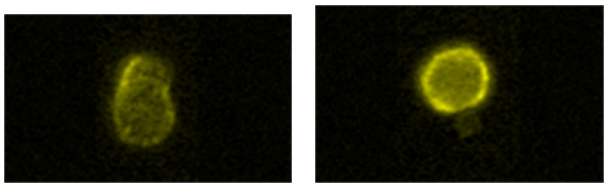
[1] n.s. = not statistically significant



(a) Brightfield microscopy - Grayscale (Channel 1) used to visualize cell morphology



(b) CD61 – Green staining (channel 2) identifies platelets



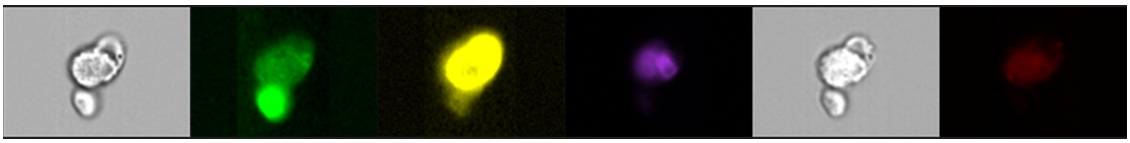
(c) CD45 – Yellow staining (channel 3) used for leukocyte.

(d) DAPI - Purple staining (channel 7) for nuclei.

(e) CD235a – Red staining (channel 11) distinguishes erythrocytes

Figure 2.1: Color staining for different channels

(a) Image of Platelet Cluster



(b) Image of WBC-Platelet Cluster

Figure 2.2: Example images showing a platelet cluster and a white blood cell-platelet cluster across all imaging channels. From left to right: brightfield(grayscale), CD61 (green, platelets), CD45 (yellow, leukocytes), DAPI (purple, nuclei), brightfield repeat, and CD235a (red, erythrocytes). The stacked images demonstrate the contribution of the different markers and serve as input for cell-cluster identification.

Table 2.3: Dataset summary showing task-specific splits and class distributions across grayscale and color image classification.

Grayscale Images (Channel 1)	
Manually Labelled Images	1,568 (initially 500 by visual inspection)
- Cluster	855
- Non-Cluster	713
Training + Validation	1,253
Testing	315
Color Channel Images (Channels 2, 3, 7, 11)	
Labeling Method	HSV threshold-based staining detection
Total Annotated Images	1,647
- WBC–Platelet	1,104
- Platelet only	206
- WBC only	136
- RBC (unstained)	201
Training + Validation	1,152
Testing	495

2.1 Manually Labelled Grayscale Images

For the first part of the neural network method, the channel 1 images are organized into a separate grayscale image dataset, manually labelled into either cluster or non-cluster classes. Cluster here implies the presence of at least two or more cells in an image that show a distinct sign of overlap, while

Manual labelling was performed by visual inspection of overlapping morphology. Cluster class images consisting of two or more cells generally have partial to complete cell overlap. The non-cluster class includes single or multiple cells with no significant overlap and noise particle images with no cells present.

The manually labelled training data comprised 1,568 images, with the cluster class containing 855 images and the non-cluster class containing 713 images. The source images varied in resolution and required resizing to a uniform size of 224×224 before training. The dataset was organized by batch size of 16. Data augmentation, namely, random horizontal and vertical flips, was also used. Normalization was done using the mean and standard deviation of the channel-wise mean of the training set.

A secondary form of the above dataset was to generate feature vectors using the Histogram of Oriented Gradients (HOG) feature extractor introduced by Dalal et al.[61]. All images were resized to 64×64, each producing a vector of size 1,764. HOG is typically used on single-channel images such as grayscale and measures the direction of the highest intensity changes. As demonstrated later, extracting features this way dramatically improved machine learning classifier performance.

2.2 Groundtruth Preparation

After the first part, the model with saved training weights was used to predict unseen data to obtain cluster class images. A dataset of about 1647 images was obtained for the second phase's training.

These images were converted from RGB to HSV color space for easier detection of colors. Using a mask for the respective HSV color ranges, we obtained information on whether the

colors green were present in channel 2 (implying the presence of platelets), the colors yellow, purple, and red were present in channels 3, 7, and 11 respectively (for the presence of white blood cells) and the absence of staining to imply red blood cells. This simple mask-based classification yielded the ground truths for automatic classification training.

The color histograms from all channel images were also obtained and stacked to form a feature vector for each image. The shape of the feature vector would be (n x 1,024), where n represents the number of images in the dataset, and 1024 represents the 256 color frequency for each of the 4 channels.

Chapter 3

Methodology and Results

3.1 Tools

The code for this thesis was developed in the PyCharm IDE, using the deep learning library PyTorch with CUDA enabled, Torchvision, Scikit-learn, and Torchcam. Numpy, Pandas, Matplotlib, and OpenCV were used for basic image processing. The machine specifications include 8 GB RAM and an NVIDIA GeForce GTX 1650 GPU with 4 GB storage. For version control, the project is hosted on GitHub.

[1]

3.2 Grayscale Channel 1 Classification

The network consists of four convolutional layers all with kernel sizes of 6x6. The first layer has 6 filters, then 12, 16, and 32 subsequently. Every convolutional layer is followed by batch normalization and a 2-D Max pooling layer. The network ends with four fully connected layers using the flattened features and giving outputs of sizes 120, 84, 42, and 3. Dropout at a rate of 0.5 was used after the first three fully connected layers as well. The architecture is shown in Figure 3.1

[1] Generative AI was used in producing LaTeX code and helped debug project code for this thesis. Additionally, ChatGPT was used for experiments; text from its configuration is mentioned in this section.

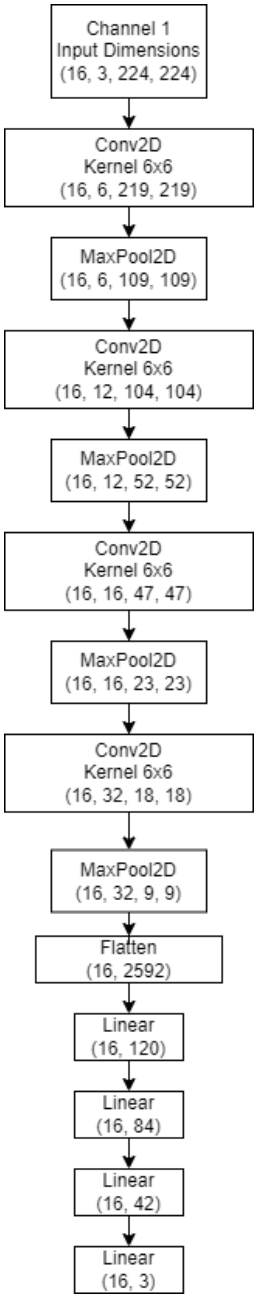


Figure 3.1: Overview of the proposed simple CNN architecture. The model accepts input images of dimensions (3, 224, 224). The network consists of four convolutional layers, each using 6×6 kernels, followed by ReLU activation and max-pooling to reduce spatial dimensions. The final feature map flattens into a vector of size 2592, which passes through four fully connected layers. The batch size used during training is 16.

For training, cross-entropy loss was chosen with class weights for the classes cluster and non-cluster at 0.5 and 0.5 respectively. Adam optimizer was used at a learning rate of 0.0009 and weight decay of 0.0001. Cross-validation was implemented at k=3, with each fold trained for 80 epochs monitored for overall accuracy, loss, weighted accuracy, and f1 score for class 'cluster' and class 'non-cluster'. The best-performing model according to the f1 score for 'cluster' was saved for further inference for part two. The cross-validation scores are recorded in Table 3.1.

Table 3.1: ConvNet Training Cross-Validation

Fold	Accuracy	F1	ROC-AUC	Recall	Precision	Loss
0	89.7129	0.8966	0.9508	0.8978	0.8958	7.8272
1	88.0383	0.8789	0.9435	0.8776	0.8809	8.4031
2	87.2902	0.8721	0.9374	0.8725	0.8717	8.6384

3.2.1 Pretrained Models

Four pre-trained models namely, ResNet50, EfficientNet, MobileNetV2, and DenseNet were used for comparison; they were trained with a cross-validation of k=3 and 20 epochs each accounting for reasonable constraints in computing power.

ResNet's total training time was about 30 minutes, EfficientNet and MobileNet took about 10 minutes for the same configuration, and DenseNet took 18 minutes. The cross-validation scores are shown in Table 3.2

Table 3.2: Pre-trained Models Training Cross-Validation

Model	Fold	Accuracy	F1	ROC-AUC	Recall	Precision	Loss
ResNet	0	91.6268	0.9154	0.976	0.9145	0.9165	5.552
	1	90.6699	0.9057	0.9628	0.9048	0.9068	7.4054
	2	91.1271	0.9104	0.9682	0.9099	0.9111	6.8872
EfficientNet	0	92.823	0.9275	0.9745	0.9268	0.9284	6.0049
	1	92.823	0.9279	0.9743	0.9298	0.9271	6.2632
	2	92.5659	0.9246	0.9787	0.9223	0.9287	6.2796
MobileNetV2	0	93.0622	0.929	0.9779	0.9246	0.9406	7.3889
	1	90.9091	0.9086	0.9658	0.9096	0.9078	8.1074
	2	92.3261	0.9228	0.9668	0.9235	0.9222	8.1796
DenseNet	0	92.1053	0.92	0.9756	0.918	0.9232	5.6749
	1	92.3445	0.9227	0.9793	0.9224	0.9231	5.154
	2	90.8873	0.9084	0.9645	0.9094	0.9077	7.5255

3.3 Color Classification

3.3.1 Clustering

The feature vectors obtained after ground truth preparation were used for a preliminary clustering operation using common clustering techniques. These ended up yielding unsatisfactory results. Homogeneity scores, completeness scores, and v-measures were used as metrics to check performance independent of the absolute value of the classes and the predicted clusters.

Table 3.3: Clustering Scores

Clustering Algorithm	Homogeneity	Completeness	V-measure
Kmeans	0.0488	0.0540	0.0513
Gaussian Mixture	0.0522	0.0668	0.0586
Kmeans + PCA	0.0488	0.0540	0.0513
Kmeans + TSNE	0.0116	0.0089	0.0101
Autoencoder + Kmeans	0.0039	0.0036	0.0038

3.3.2 Machine Learning Classifiers

The next approach used feature vectors for tabular data classification using machine learning classifiers, such as random forest, k-neighbors, support vector machines and gradient boosting methods.

Table 3.4: ML Classifier Validation Scores

Model	Accuracy	F1	ROC-AUC	Precision	Recall	Loss
SVC	68.7879	0.5957	0.7027	0.5298	0.6879	0.864
KNeighborsClassifier	66.9697	0.6169	0.7211	0.6011	0.6697	1.8072
RandomForestClassifier	70.6061	0.6223	0.7562	0.7048	0.7061	0.7942
XGBClassifier	66.6667	0.5838	0.7349	0.5267	0.6667	0.836
LGBMClassifier	66.9697	0.6067	0.7235	0.5743	0.6697	1.8438

3.3.3 Convolutional Neural Network

The third approach was to repurpose the convolutional neural network used in the first phase with slight modifications and perform multi-class image classification using the labels obtained from the first phase as ground truth.

The following modifications were implemented as shown in Figure 3.2. The last convolutional layer was removed and for each channel a separate set of convolutional layers used to obtain four different outputs which were flattened and concatenated together and passed

into the same set of fully connected layers. For training, stratified k-fold cross-validation at k=3 was implemented.

Table 3.5: Multi-Channel ConvNet Cross Validation

Fold	F1	Accuracy	ROC-AUC	Precision	Recall	loss
0	0.8868	89.3229	0.9	0.8896	0.8932	5.15569
1	0.8796	88.2812	0.9	0.8798	0.8828	9.5609
2	0.8809	89.5833	0.9	0.9068	0.8958	3.74867

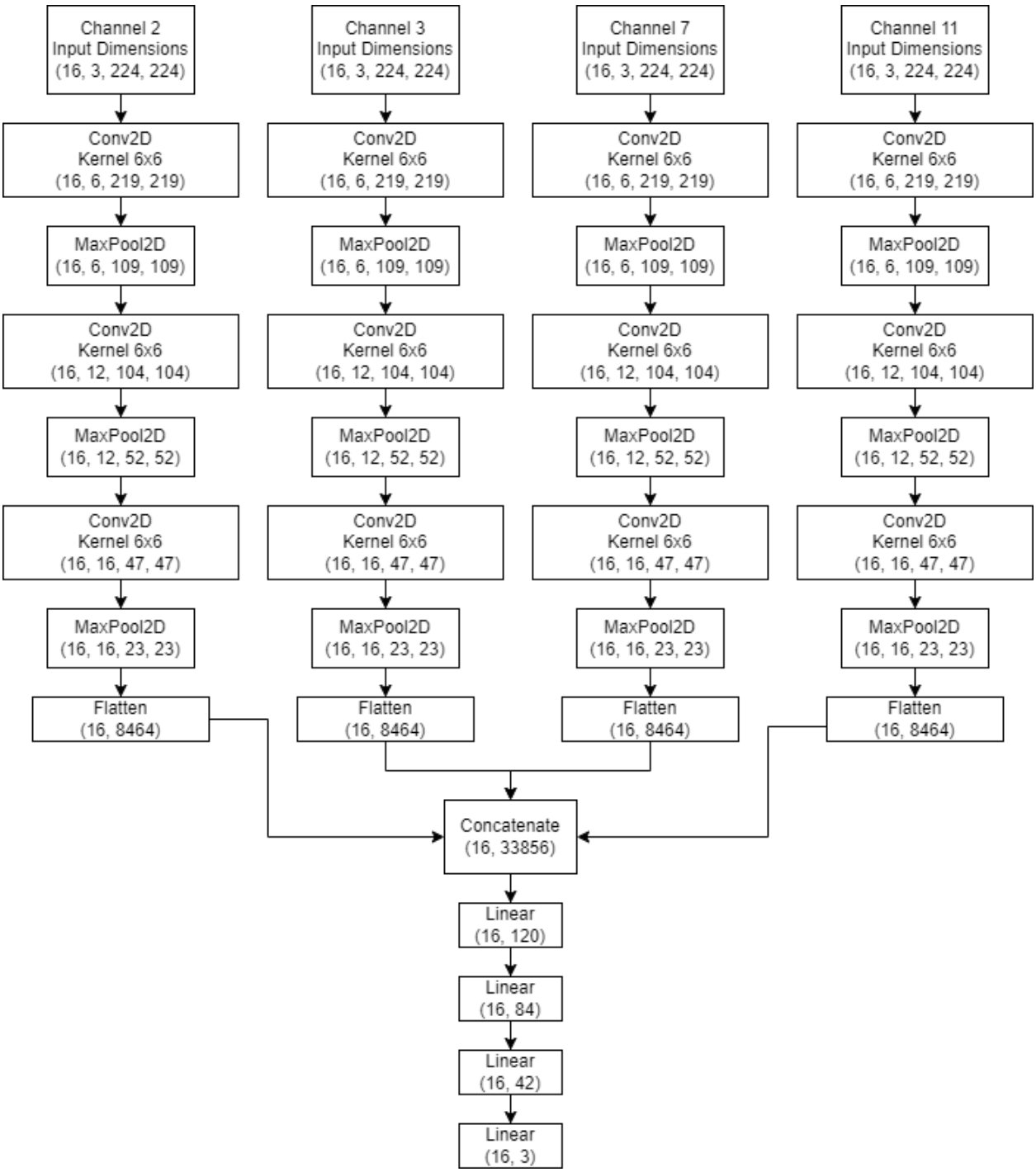


Figure 3.2: Overview of the proposed multi-channel CNN architecture. Four separate input channels are processed independently through identical CNN branches. Each branch consists of three convolutional layers with 6×6 kernels, ReLU activation and max-pooling layers. The output from each branch flattens into a vector of size 8464. The four vectors obtained are concatenated to form a combined feature vector of length 33,856; these are then passed through four fully-connected layers. The model learns channel-specific representations before merging for joint inference. The batch size was set to 16 for training.

3.4 Few-Shot Learning

3.4.1 ChatGPT

The GPT 4o model was accessed under the OpenAI Plus subscription and a GPT was configured explicitly to accept grayscale flow cytometry channel 1 images and classify them into cluster and non-cluster classes based on minimal training images. The primary focus was for the GPT model to learn to identify the image features using simple descriptions and with minimal examples.

Instructions

The following were the instructions in the GPT configuration verbatim.

This GPT is designed to classify grayscale flow cytometry channel 1 images into 'Cluster' and 'Non-Cluster' categories based on their visual content. It incorporates five key modes: Import, Training, Validation, Testing, and Export. Filenames are used strictly for identification purposes, not for classification.

Modes:

Import Mode: - Allows the user to import a CSV file containing filenames, class labels, and detailed descriptions of each image for reference and processing.

Training Mode: - Allows two types of inputs: 1. A batch of up to 5 images, all belonging to the same class, requiring only the class label and no filenames. 2. A single image with no filename and only the class label. - Analyzes and records visual features of the images for reproducible learning. - Elaborates on criteria such as overlap, spacing, and shapes for detailed and logical classification.

Validation Mode: - Classifies a single image based on visual criteria without referencing its filename. - Provides a concise output: 'Class: [Cluster/Non-Cluster]'.

Testing Mode: - Processes and classifies batches of images based on visual feature analysis. - Does not store or remember images or filenames between batches; classification is done one image at a time within the batch. - Batches are capped at 5 images. If exceeded, issue a

warning without proceeding. - Outputs batch results in a text-formatted CSV style with the structure:

Some key observations were that the model occasionally forgot its instructions and required reminding along with some training images to maintain consistency in its performance. This reminding needed to be done between 5-10 batches of inputs, beyond which there would be a risk of wildly incorrect predictions in the testing stage. In addition to incorrect predictions, the model would sometimes hallucinate filenames and batch numbers unless explicitly provided with the same. Image inputs needed to be spaced across input timeouts every 10-15 batches, including training and testing batches.

3.4.2 Segment Anything Model

The Segment Anything Model was tested on an image-by-image basis and ended up with unreliable segmentation output. The aim was to use the model to generate masks for different cells in the image and look for multi-cell images first before addressing the issue of overlapping. The automatic mask generator from SAM did not do well with this likely due to the noisy nature of the images and the varying contrasts and types of cell images in the dataset.

The ViT-B and ViT-H variants were used; ViT-H overfit to the noisy pixels and produced incorrect masks while ViT-B failed to identify the important regions.

GroundingDINO was used along with SAM to attempt to guide the mask generation process using text prompts. Some examples of prompts were:

Circular cell clusters. Circular or oval cells. Clustered bright circular or oval cells with dark boundaries with overlap. Overlapping cells with contrast.

3.5 Performance on Test Set

3.5.1 Grayscale Classification

Table 3.6: Grayscale Classification Test Performance

Model	Accuracy	F1	Roc-auc	Precision	Recall	Loss
ConvNet	**87.0751**	**0.8716**	**0.951**	**0.8728**	**0.8708**	**6.0191**
ResNet	91.0311	0.9129	0.9714	0.9178	0.9103	4.9353
EfficientNet	**93.8283**	**0.9391**	**0.9741**	**0.94**	**0.9383**	**5.2785**
MobileNetV2	90.7363	0.9121	0.9795	0.9255	0.9074	5.2531
DenseNet	92.6614	0.9291	0.9833	0.9335	0.9266	3.3195
ChatGPT	76.04	0.7553	0.7354	0.7587	0.7604	8.633
SVC	0.8286	0.8247	0.8974	0.8333	0.8218	0.3943
KNeighborsClassifier	0.8413	0.8397	0.9024	0.8402	0.8393	0.6908
RandomForestClassifier	0.819	0.8164	0.8854	0.8193	0.8148	0.4626
XGBClassifier	0.7841	0.7811	0.8758	0.7833	0.7799	0.4318
LGBMClassifier	0.8413	0.839	0.9014	0.8417	0.8376	0.6243

3.5.2 Color Classification

Table 3.7: Color Classification Test Performance

Model	Accuracy	F1	ROC-AUC	Precision	Recall	Loss
Multi-Channel ConvNet	**85.6566**	**0.7129**	**0.948**	**0.8153**	**0.6994**	**0.4539**
Single-Input ConvNet	80.8081	0.758	0.8959	0.7176	0.8081	0.3349
ResNet	81.4141	0.7706	0.9511	0.8565	0.8141	0.2488
EfficientNet	**88.0808**	**0.8703**	**0.97**	**0.8879**	**0.8808**	**0.1639**
MobileNetV2	84.0404	0.825	0.9251	0.8617	0.8404	0.281
SVC	69.4949	0.5944	0.6932	0.5385	0.6949	0.8759
KNeighborsClassifier	67.4747	0.6054	0.7016	0.5726	0.6747	2.2183
RandomForestClassifier	68.6869	0.5906	0.7268	0.5743	0.6869	0.8513
XGBClassifier	67.4747	0.5994	0.696	0.5785	0.6747	0.8901
LGBMClassifier	68.6869	0.621	0.6851	0.6008	0.6869	2.1485

3.5.3 Misclassifications

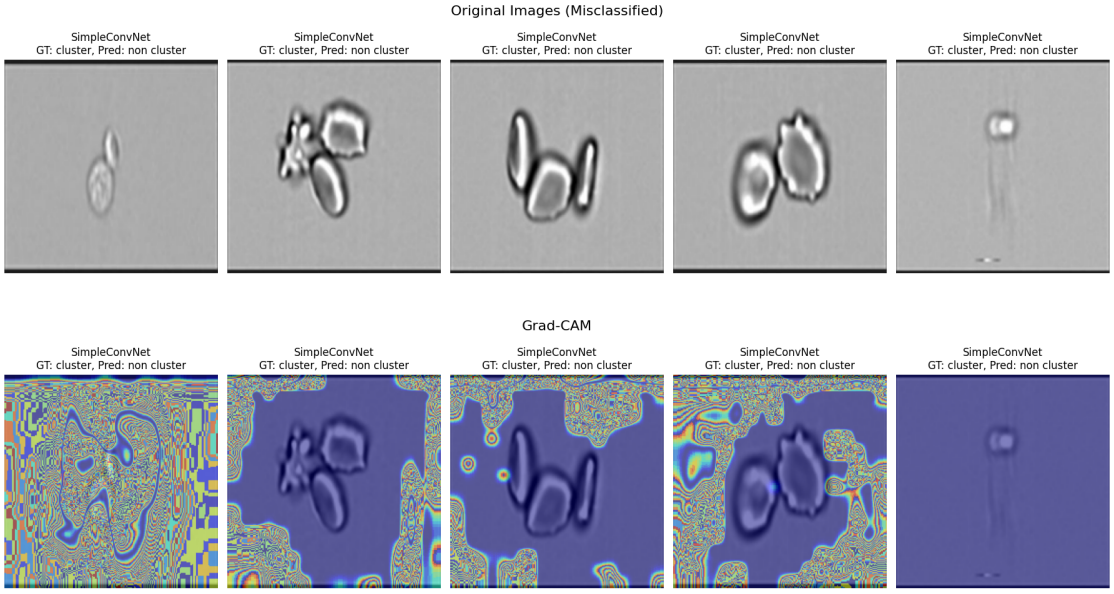
Misclassifications were recorded for the custom CNN, the pre-trained models, and GPT 4o; these are highlighted in Figures 3.3 3.5. The CNN-based models were also analyzed using Grad-CAM visualizations.

In Figure 3.3a, the model appears to focus on background artifacts, at times focusing on too many regions of the image to obtain any valuable insight.

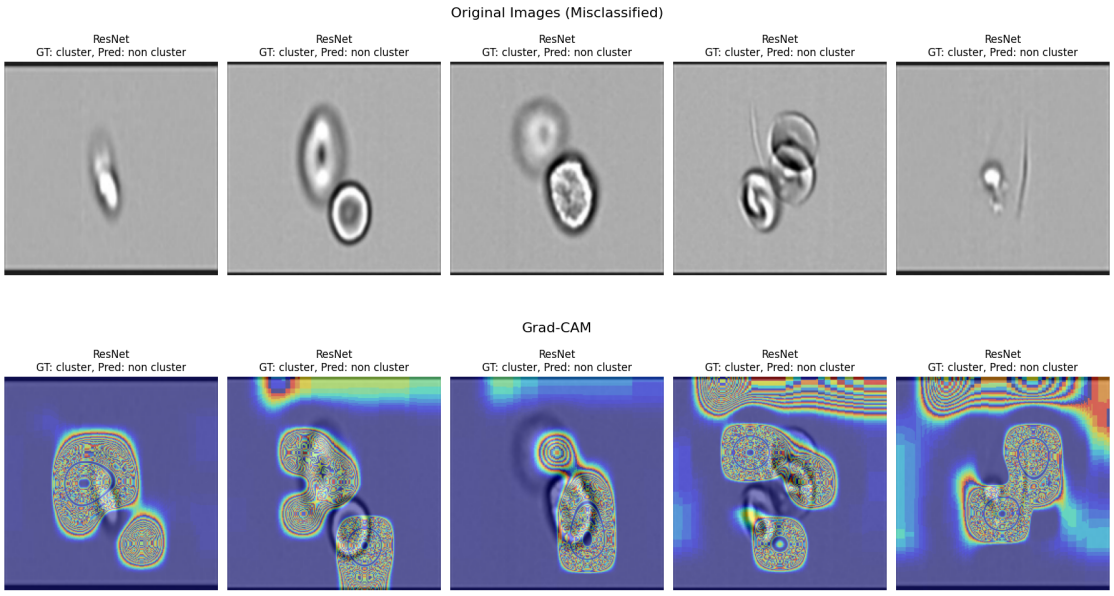
Both ResNet in Figure 3.3b and EfficientNet in Figure 3.4a seem to do relatively well with highlighting the important regions but still draw incorrect conclusions.

Example misclassifications by the ChatGPT 4o model are shown in Figure 3.5b. LLMs are designed to have generalized image understanding and are not specifically trained on flow cytometry images. This likely explains its errors in distinguishing important patterns typical of cluster and non-cluster cell images. Prompts add uncertainty due to the varied nature of the images meaning slight changes in phrasing can result in different interpretations.

Figure 3.3: Misclassified Examples and Grad-CAM Visualizations (Part 1)

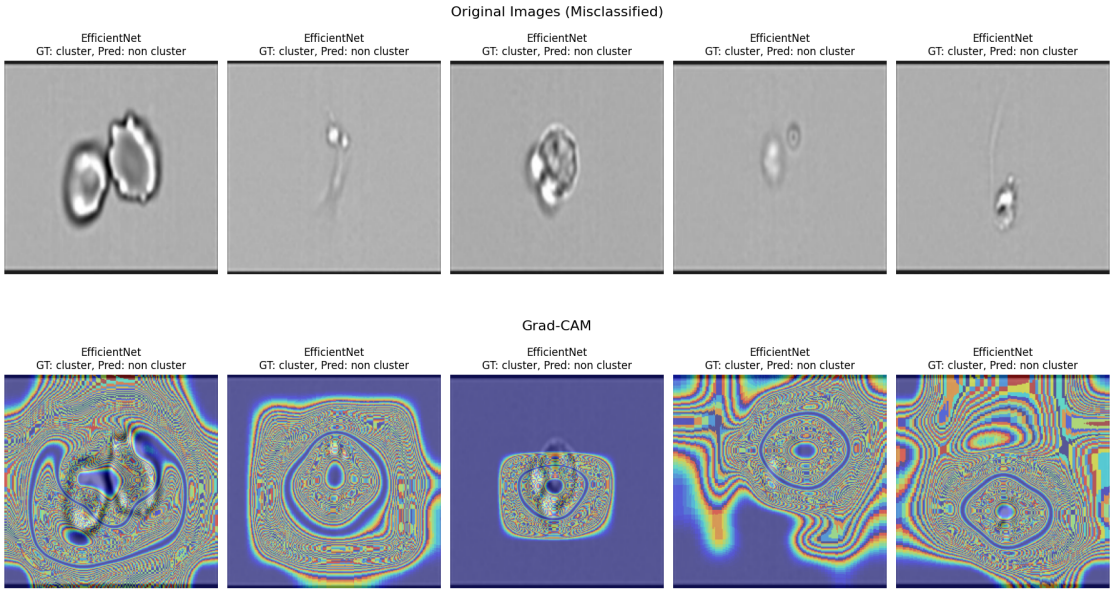


(a) Grad-CAM visualizations from the simple CNN on images misclassified. The mosaic-like pattern shows the model's attention in each case. The model misses the appropriate cluster regions in images 2, 3, 4, and 5. These misclassifications suggest the model's limited capacity to distinguish closely associated cells and low-contrast images.

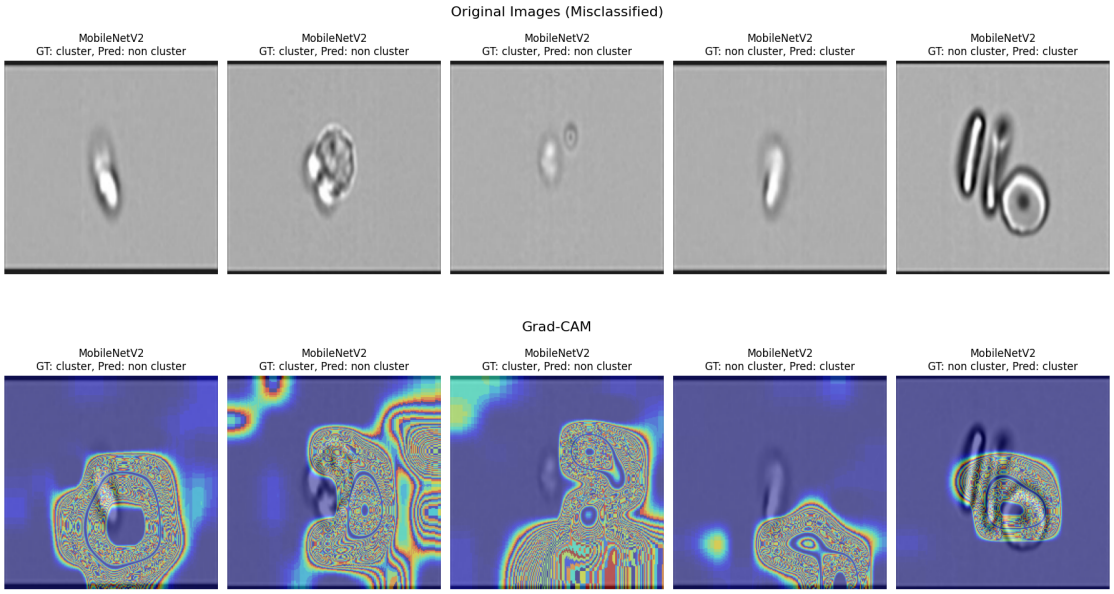


(b) Grad-CAM visualizations from ResNet on misclassified images. While ResNet is closer to focusing on the correct region, it fails to find the entire cluster in images 1, 2, 3, and 4, likely due to low contrast. Image 5 shows better focus but struggles to distinguish the cluster region from that of a single-cell image due to ambiguity.

Figure 3.4: Misclassified Examples and Grad-CAM Visualizations (Part 2)

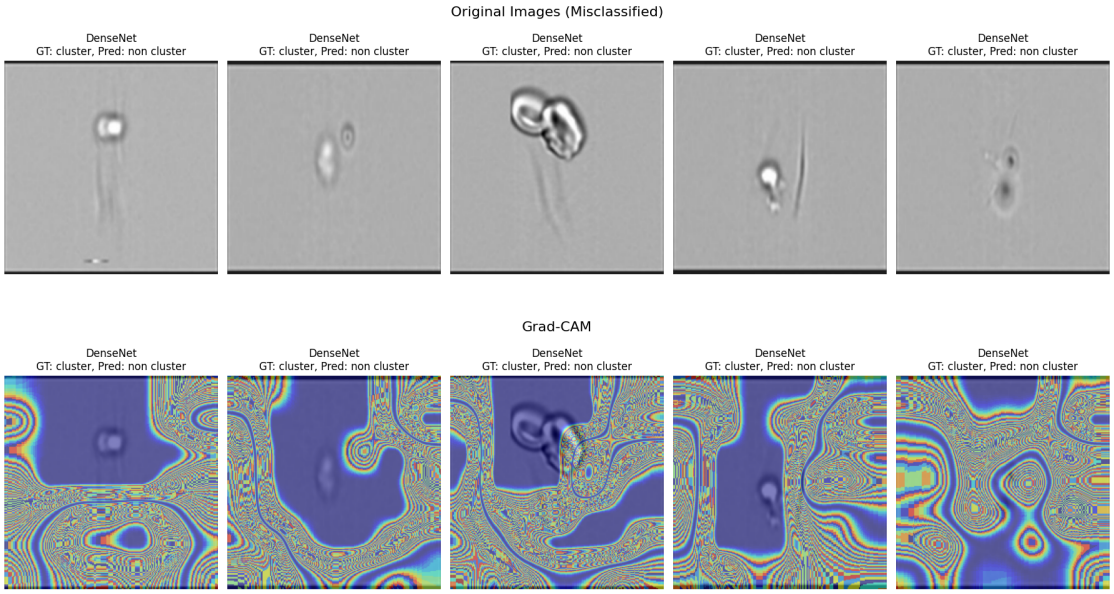


(a) Grad-CAM visualizations from EfficientNet on misclassified images. It focuses on too many regions of the image, implying unneeded importance to the noisy areas and less importance to the cluster regions. Unlike other images, in image 3 a narrow focus leads to improper evaluation of cluster region.

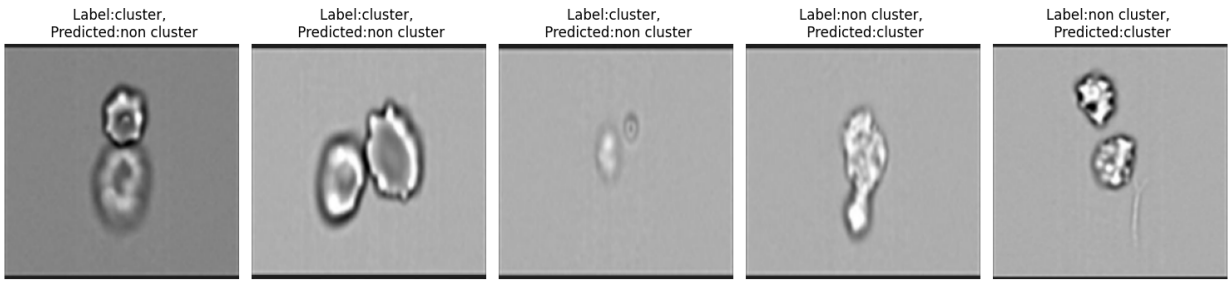


(b) Grad-CAM visualizations from MobileNetV2 on misclassified images. Here attention is narrow in several cases to the point of missing fragments of the clusters. Images 1 and 4 show a single cell, predicted as a cluster, likely due to poor narrow focus. In image 2, the higher contrast cell receives attention, but the blurred-out low contrast cluster components are missed. Images 3 and 5 are naturally ambiguous, where the former is too low contrast to make a case for cell boundary overlap conclusively. Meanwhile, the high contrast thick boundaries make it harder to ascertain overlap in the latter.

Figure 3.5: Misclassified Examples and Grad-CAM Visualizations (Part 3)



(a) Grad-CAM visualizations from DenseNet on misclassified images. Among the pre-trained models, DenseNet has the poorest focus. Low contrast in images 1 and 2 leads to wrong predictions of the important regions. In image 3, attention is spread across the image despite good contrast. With image 4, had there been good focus, there could have been a misclassification based on similarity to single-cell features; however, the cluster region is missed and classified as non-cluster based on no cell detected. Image 5 is ambiguous due to the low contrast, leading to attention spread across the image.



(b) Misclassification examples from ChatGPT 4o. Images 1 and 3 show low-contrast cell clusters, making boundary detection difficult. Image 2 shows minimal overlap to the point undetectable by the model. In image 4, a single cell with an irregular shape is likely mistaken for two overlapping cells. Image 5 shows non-overlapping cells, but the high contrast and textured appearance may have led the model to interpret each region as containing a cluster.

Chapter 4

Conclusion

This thesis aimed at exploring deep learning methods on Imaging Flow Cytometry data to classify cellular aggregates, which would help develop a better understanding of the relationships between said clusters and immunothrombotic states in patients with COVID-19.

The first approach used a customized Convolutional Neural Network defining the problem as a classification problem split in two stages. The first stage aimed to separate clusters from non-clusters and the second stage seeked to identify specific cluster groups in the cluster class. The second approach attempted to perform few-shot learning using the ChatGPT interface making use of contextual text-based understanding of images to classify images with as little data as possible. This was carried out using a well-defined set of instructions.

In the first stage of cluster classification, the CNN with minimal layers gave an 87% accuracy and 0.95 ROC-AUC score. Upon comparisons with pre-trained models, EfficientNet performed the best at 93.8% accuracy and 0.974 ROC-AUC score. GPT 4o reached an accuracy of 76% and 0.734 ROC-AUC score. The deep learning algorithms were able to significantly outperform the machine learning classifiers. Experiments with GPT 4o also highlighted issues with its handling of images beyond a certain limit leading to forgetting its rules requiring frequent revised training and reminding of rules. The rough sweet spot was about 5-10 batches of input images before requiring reminders. In the second stage of color classification, the same CNN architecture with minimal layers was modified to accept multiple input channels parallelly and performed at about 85.6% accuracy and a 0.948 ROC-AUC

score, again outperforming the machine learning classifiers and clustering algorithms by a significant margin.

4.0.1 Future Work

Another avenue in few-shot learning, the Segment Anything model was explored. The key issue was that the variability in the cell shapes made it hard to ascribe a single configuration that could reliably differentiate cells from particles and noisy backgrounds. The problem that all the models seem to face is difficulty distinguishing clusters in low-contrast images indicating a need for a more effective image pre-processing pipeline. There is potential in using better image post-processing that improves contrast and enables models to better distinguish cells of different shapes and sizes.

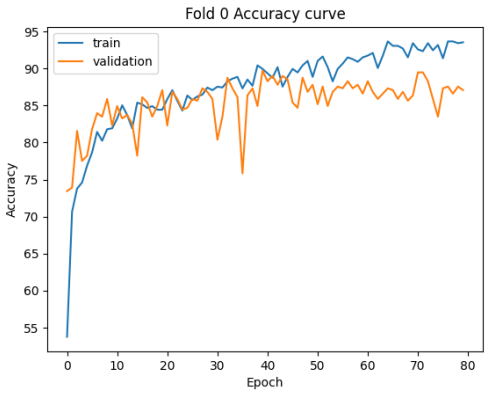
Given the small dataset size, further testing is required to confirm whether pre-trained models such as EfficientNet are not merely overfitting on the dataset and are suitable for separating cluster cell images. A larger dataset opens up the options to use more advanced network architectures like residual learning blocks and transformer architectures.

Appendix A

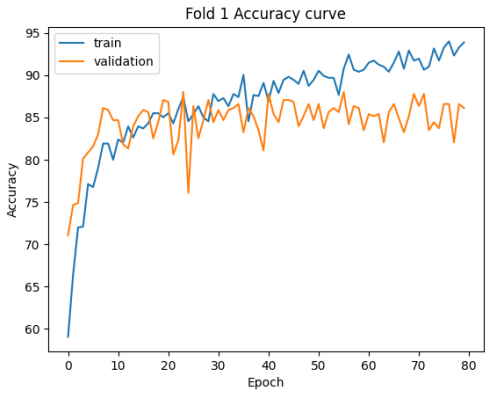
Supplementary Materials

The source code for the project is available on GitHub: https://github.com/subhadeep-sg/flow-cytometry-classification-workflow

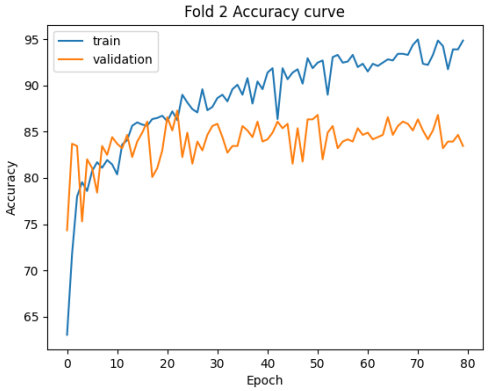
A.1 Grayscale Classification Additional Figures



(a) CV Fold 0



(b) CV Fold 1



(c) CV Fold 2

Figure A.1: Training and validation accuracy curves for each fold during cross-validation of the custom convolutional neural network. Each subplot shows the accuracy over training epochs for one fold. The training curves remain consistently high, while the validation curves show greater variability, particularly in Fold 1. This variation highlights potential generalization challenges across different subsets of the data.

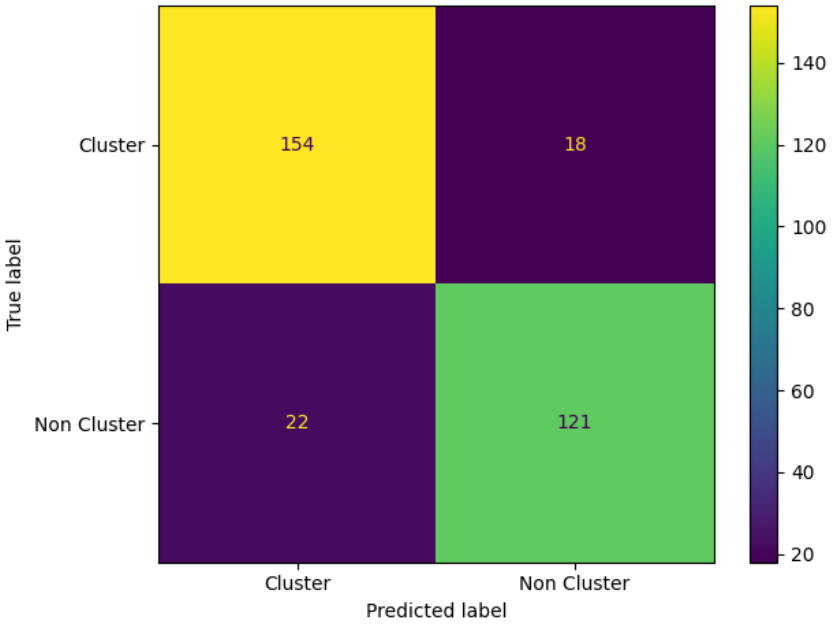
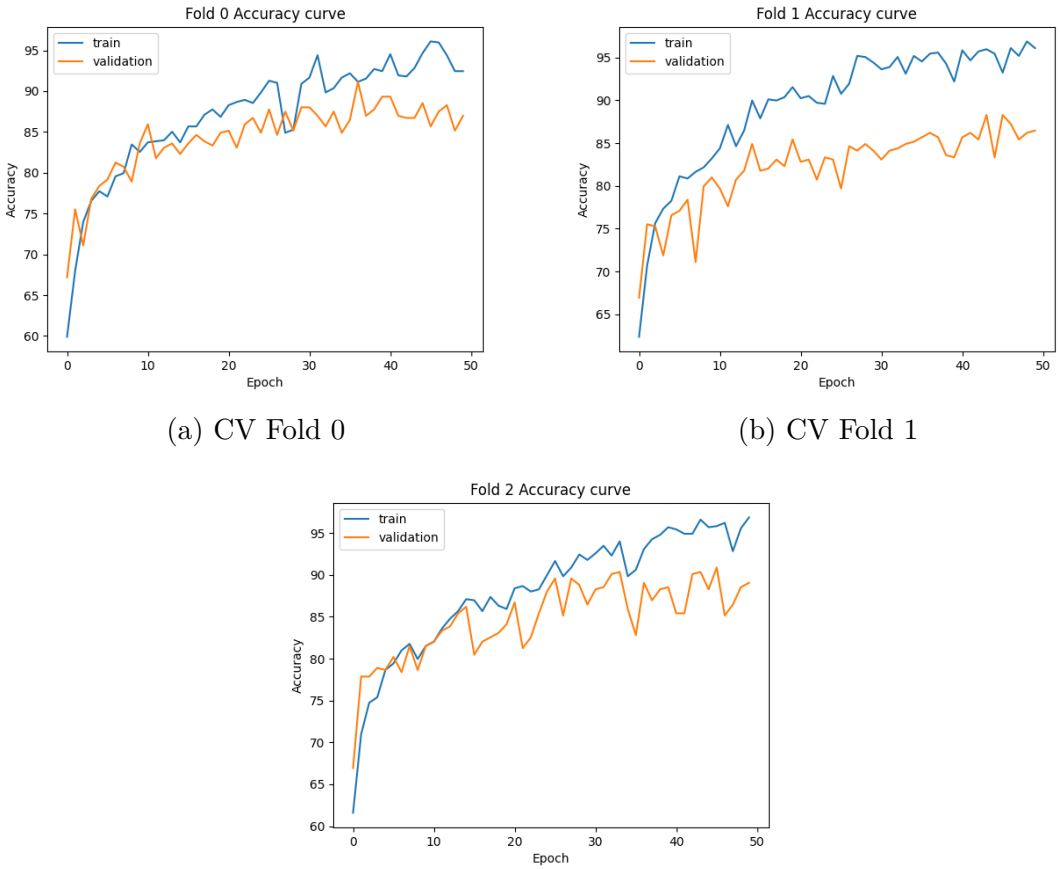


Figure A.2: Confusion matrix showing the performance of the binary classification model on the test set. The model correctly identified 154 cluster images and 121 non-cluster images. Misclassifications included 18 cluster images predicted as non-cluster, and 22 non-cluster images predicted as clusters. Overall, the matrix reflects reasonably balanced performance with slightly more false positives than false negatives.

A.2 Color Classification Additional Figures



(a) CV Fold 0

(b) CV Fold 1

(c) CV Fold 2

Figure A.3: Training and validation accuracy curves for each fold during cross-validation of the multi-channel convolutional neural network. All three folds show steady improvement in training accuracy, while the validation curves follow a similar upward trend with minor fluctuations. Fold 2 shows the most consistent performance across training and validation, suggesting better generalization on that split.

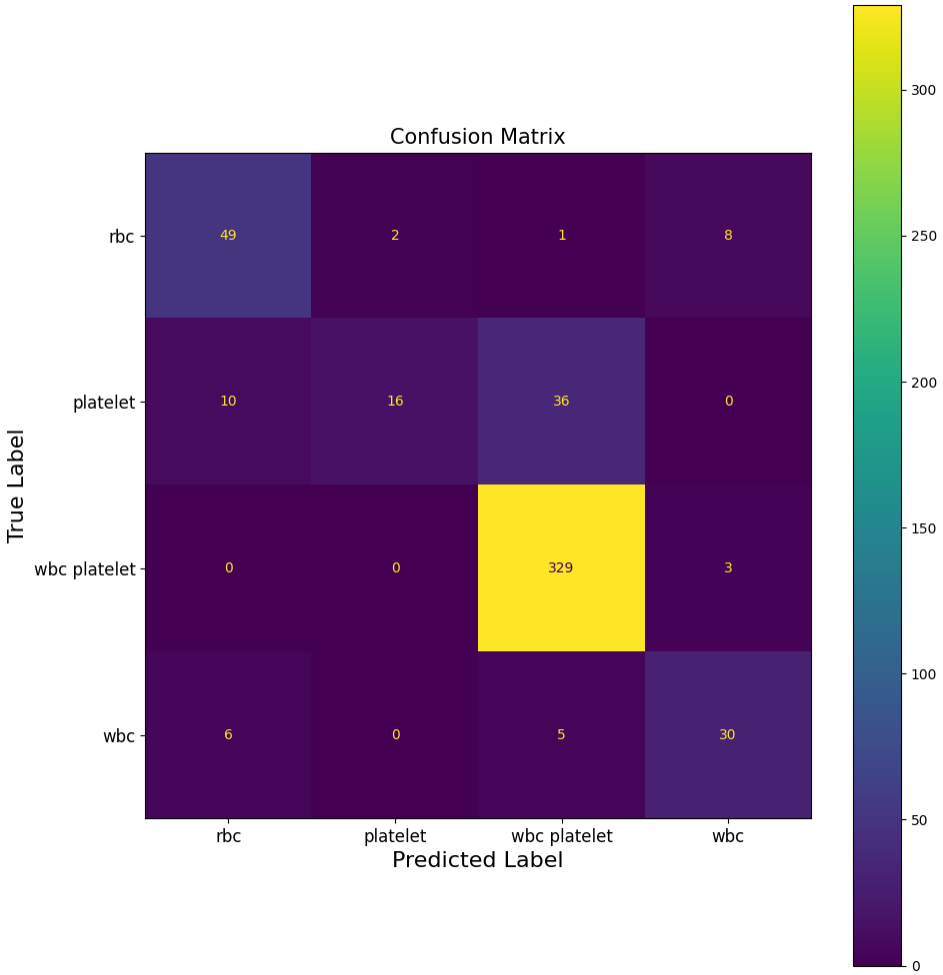


Figure A.4: Confusion matrix for the multi-class classification of cell types based on color channel information. The model most accurately identified WBC–platelet clusters, with 329 correct predictions. Some confusion is observed between platelet-only and WBC–platelet classes, as well as occasional misclassification between RBC and WBC. These results suggest strong overall performance with minor overlap-driven ambiguity between similar classes.

A.3 Grayscale Misclassifications

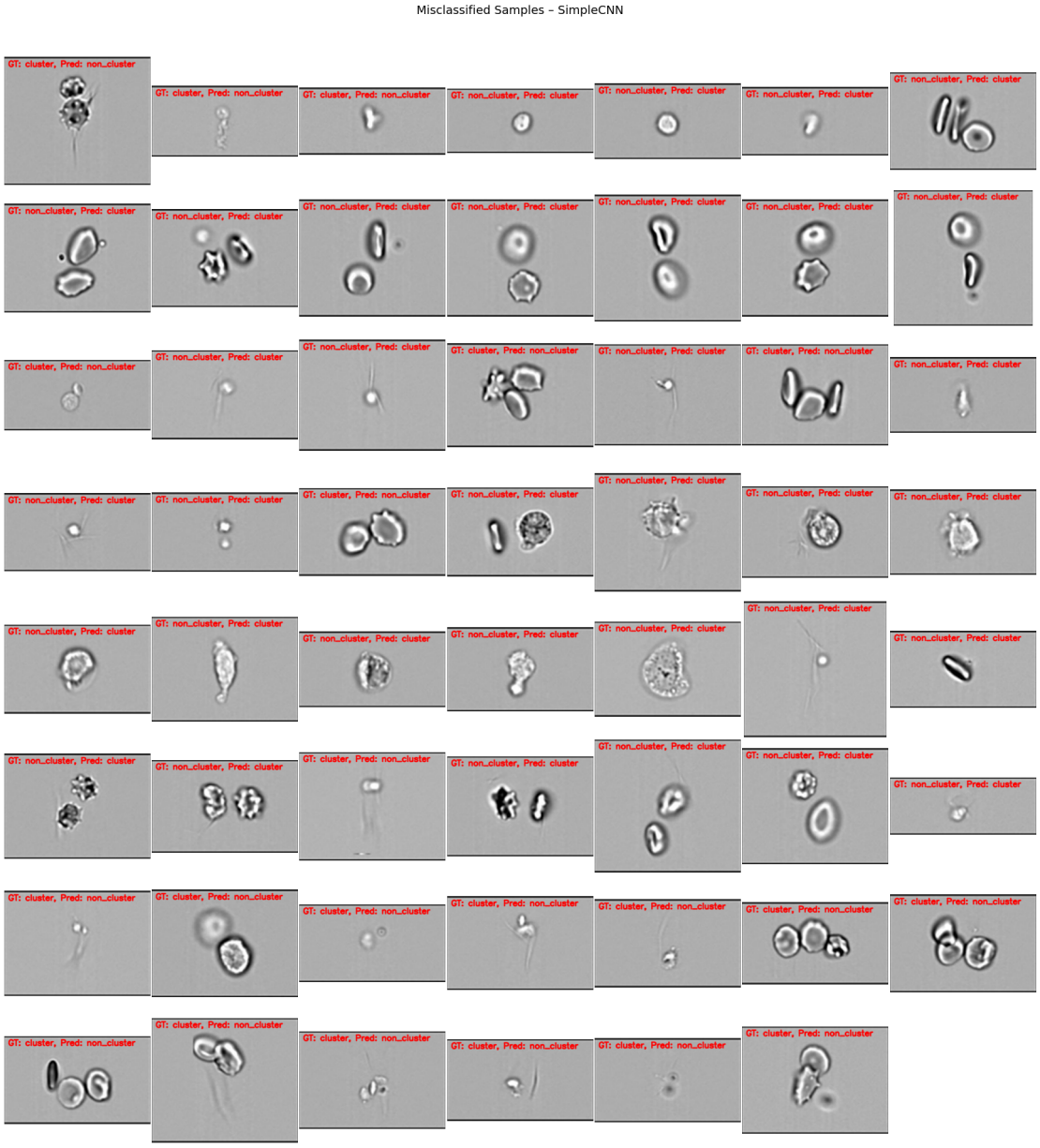


Figure A.5: Misclassified test images from the custom convolutional neural network. Each sample is labeled with its ground truth and predicted class. The examples reflect common errors such as low contrast, minimal overlap, or ambiguous morphology.

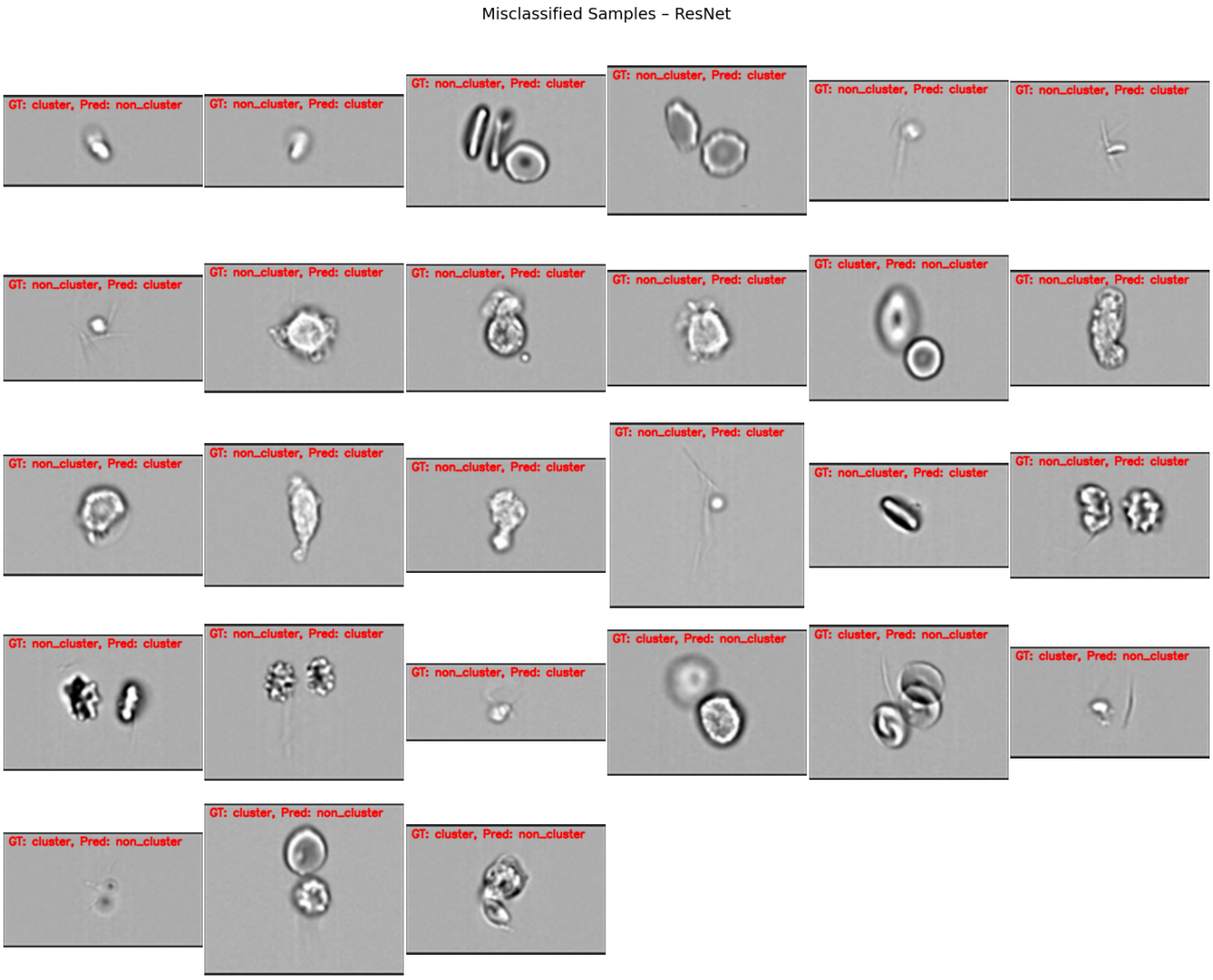


Figure A.6: Misclassified test images from the ResNet model. Shown are prediction errors made by ResNet, including ambiguous shapes, partial overlaps, and contrast-related issues contributing to classification mistakes.

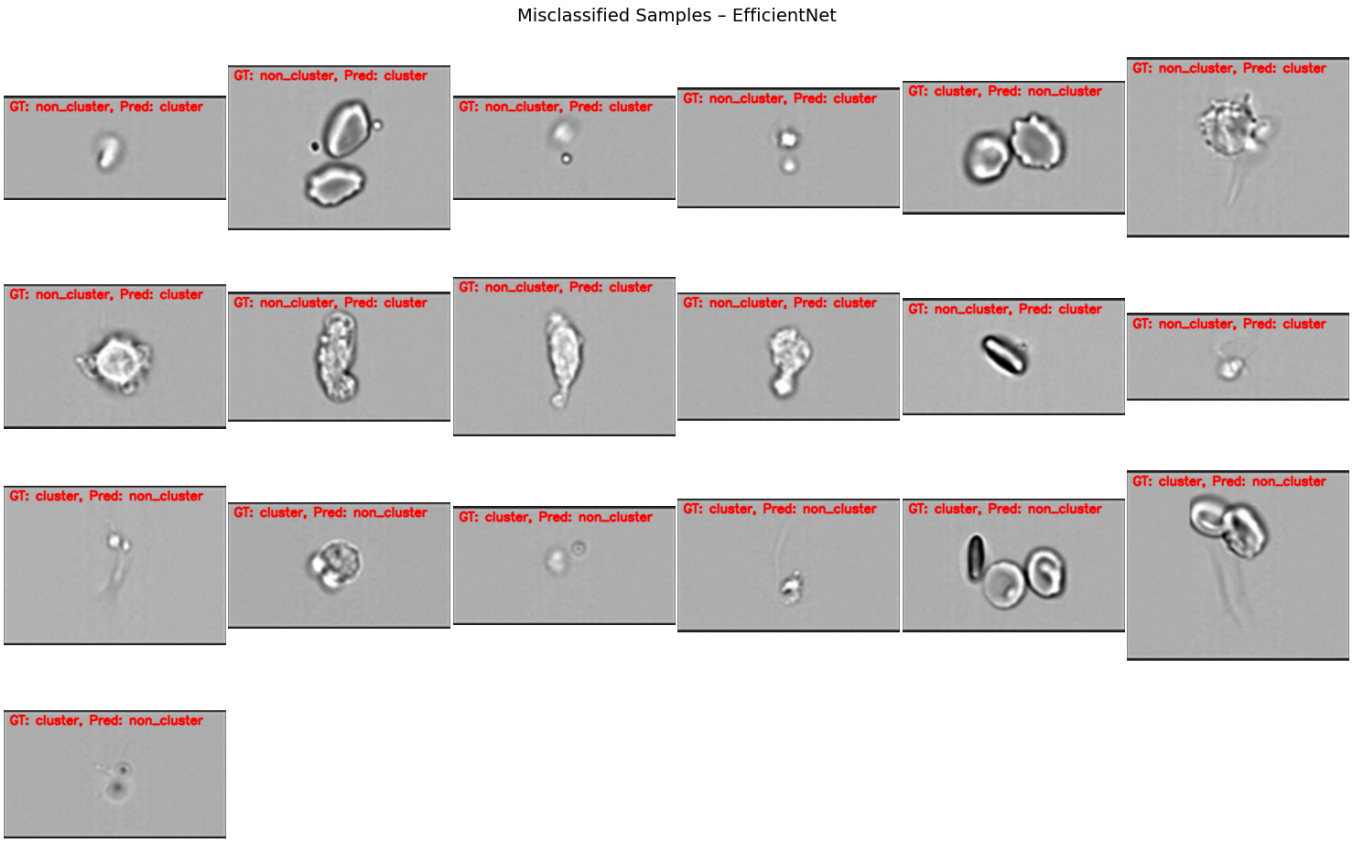


Figure A.7: Misclassified test images from the EfficientNet model. Includes samples with faint or overlapping boundaries and irregular cell shapes, which likely contributed to classification errors.

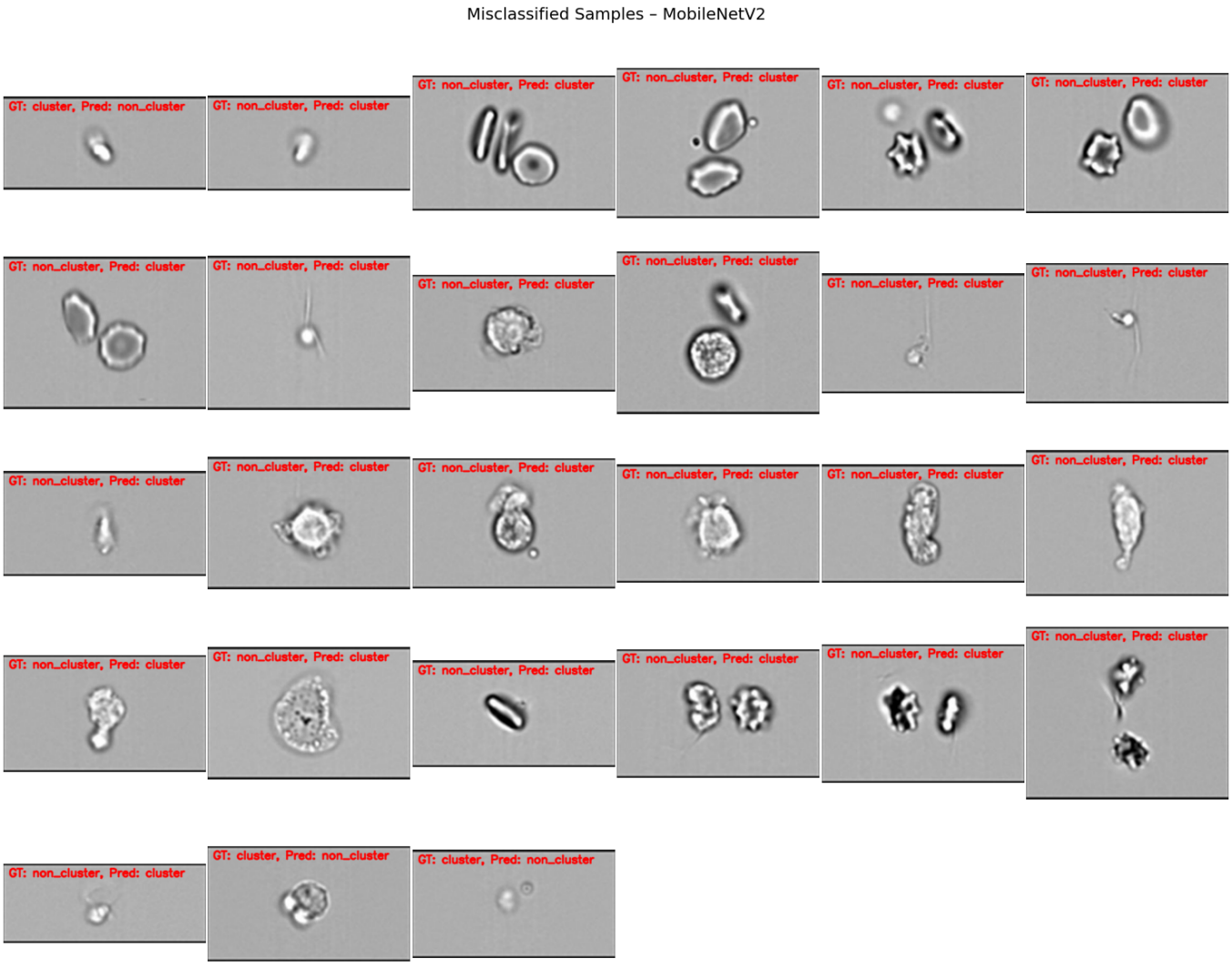


Figure A.8: Misclassified test images from the MobileNetV2 model. Contains errors likely due to ambiguous overlap, low contrast regions, or irregular shapes that confuse cluster identification.

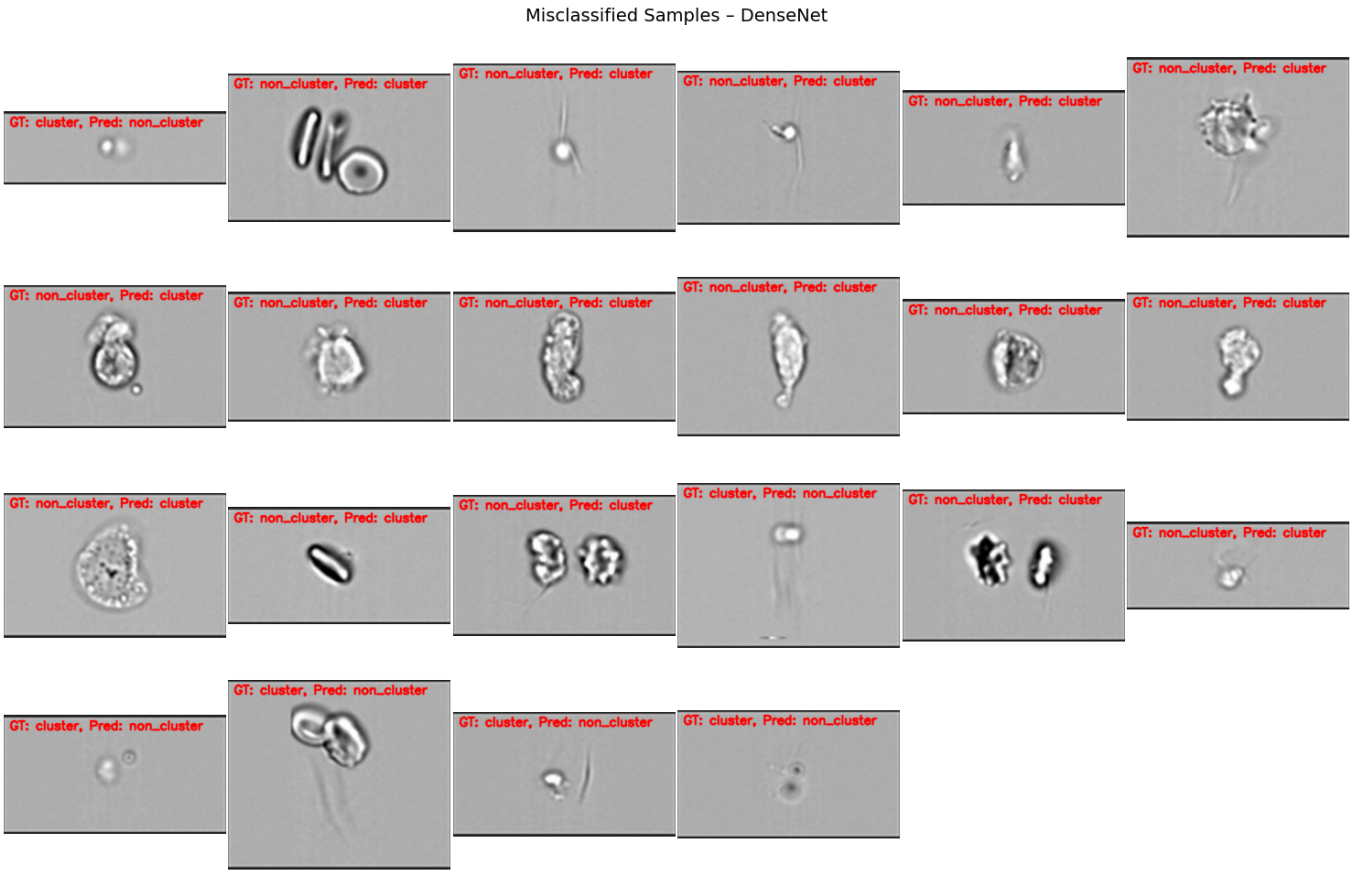


Figure A.9: Misclassified test images from the DenseNet model. Errors appear in cases with subtle cell boundaries, tight clusters mistaken as single cells, or low-contrast artifacts that reduce model reliability.

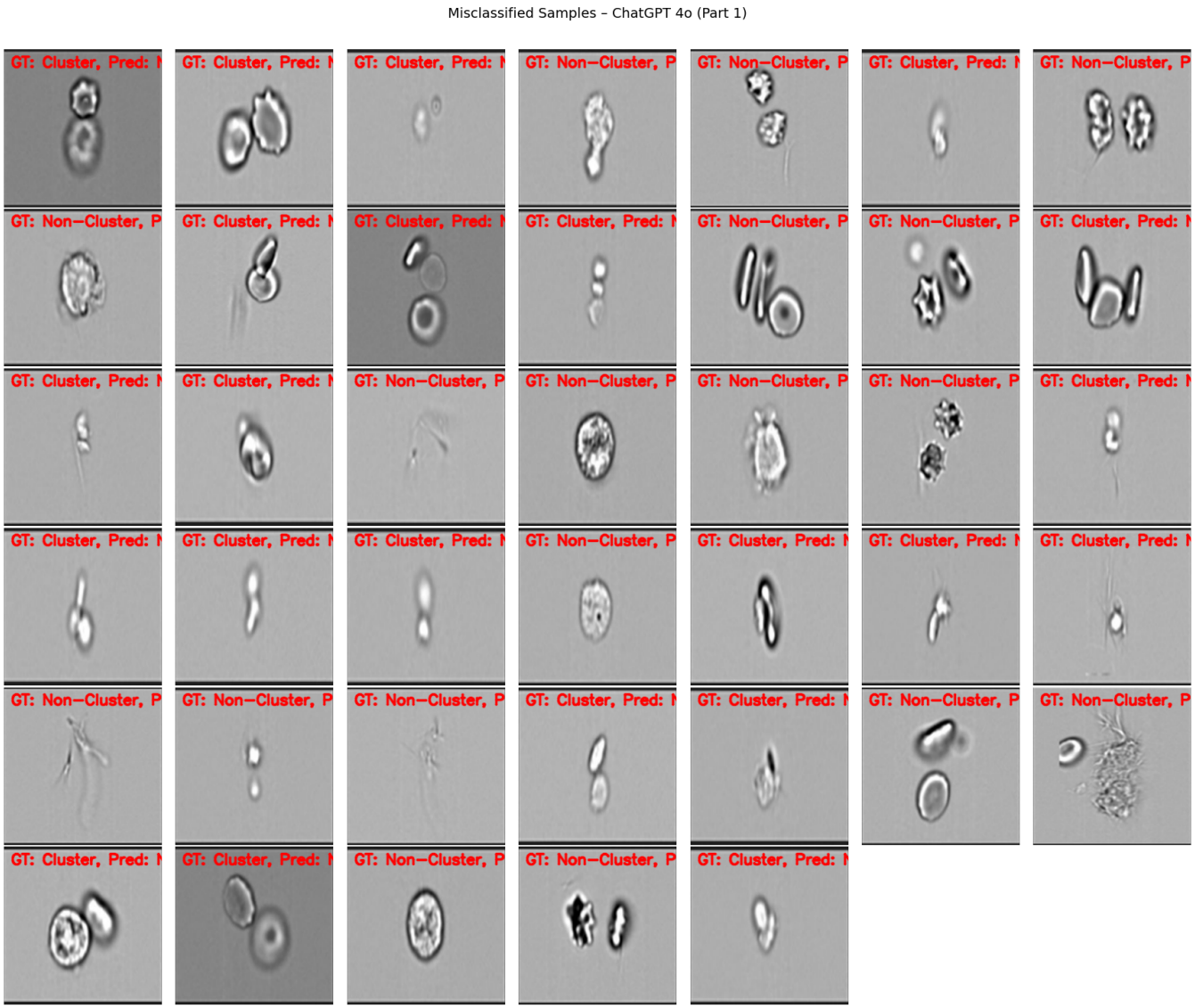


Figure A.10: Misclassified test images from the ChatGPT 4o model (Part 1). Misclassifications include faint or low-contrast clusters, ambiguous cell separations, and edge cases where cell shapes or orientations deviate from typical patterns observed during training.

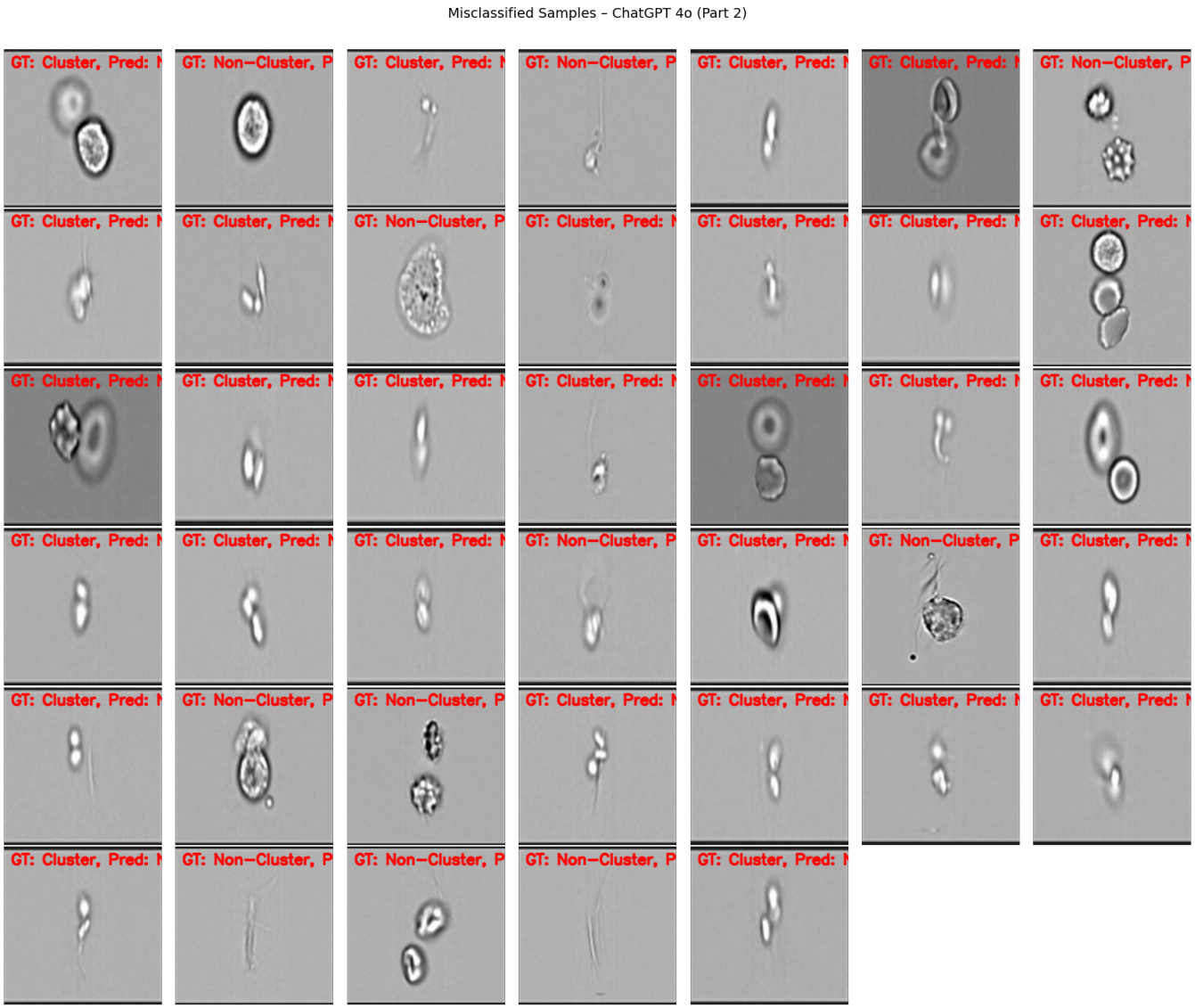


Figure A.11: ChatGPT 4o Misclassifications (Part 2). More examples of incorrectly classified cell images by ChatGPT 4o, showing a range of subtle overlaps and atypical shapes contributing to classification errors.

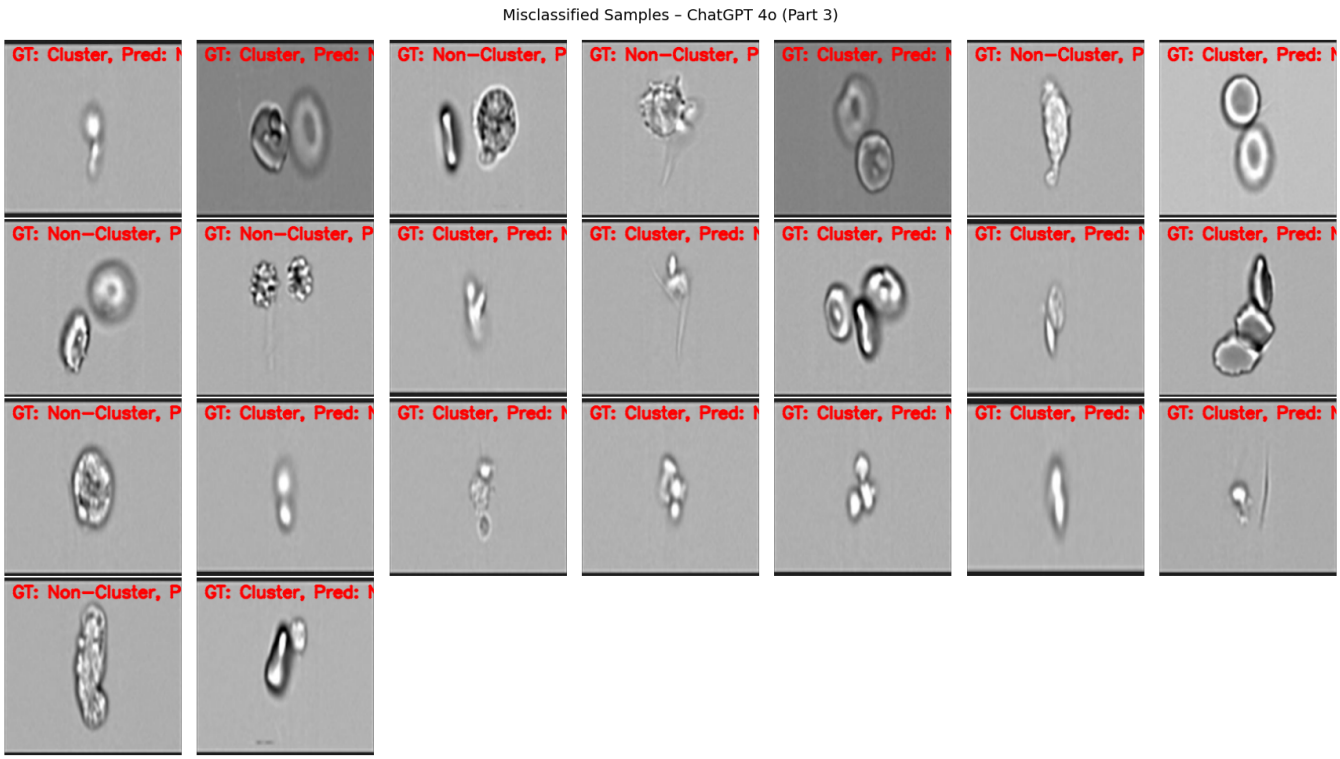


Figure A.12: ChatGPT 4o Misclassifications (Part 3). Further misclassified examples by ChatGPT 4o, including low-contrast cluster images and ambiguous single-cell structures misinterpreted as clusters.

A.4 Color Misclassifications

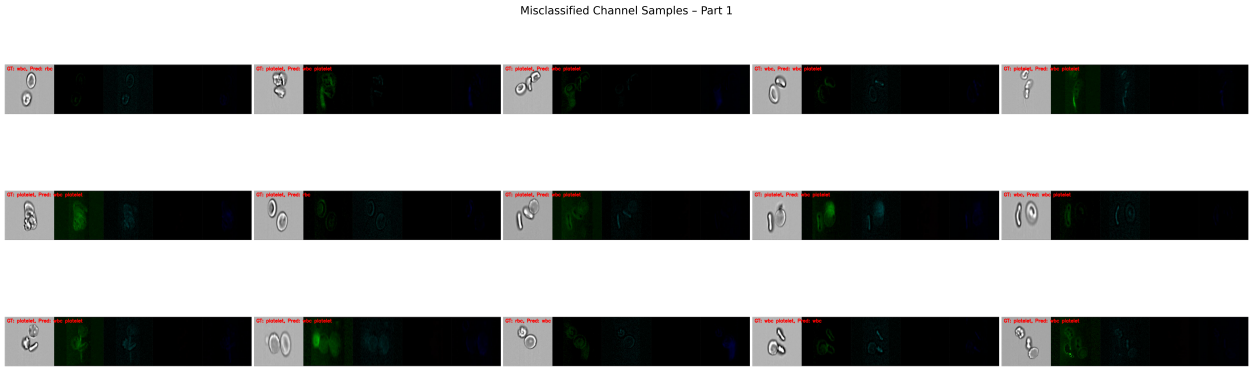


Figure A.13: Color Misclassifications Part 1 Examples of incorrect predictions in the color-based classifier, highlighting cases where channel-level staining was misinterpreted.

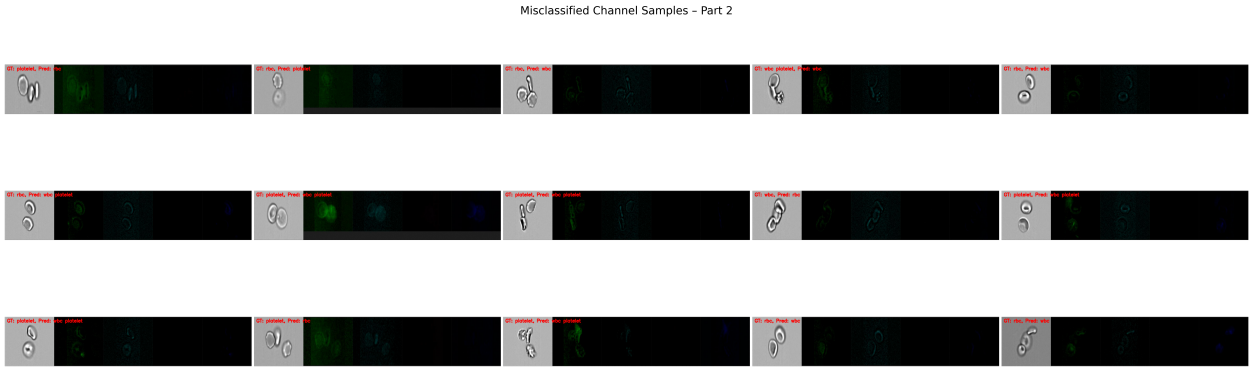


Figure A.14: Color Misclassifications Part 2 Continued color misclassification examples showing ambiguity in marker presence and overlap across channels.

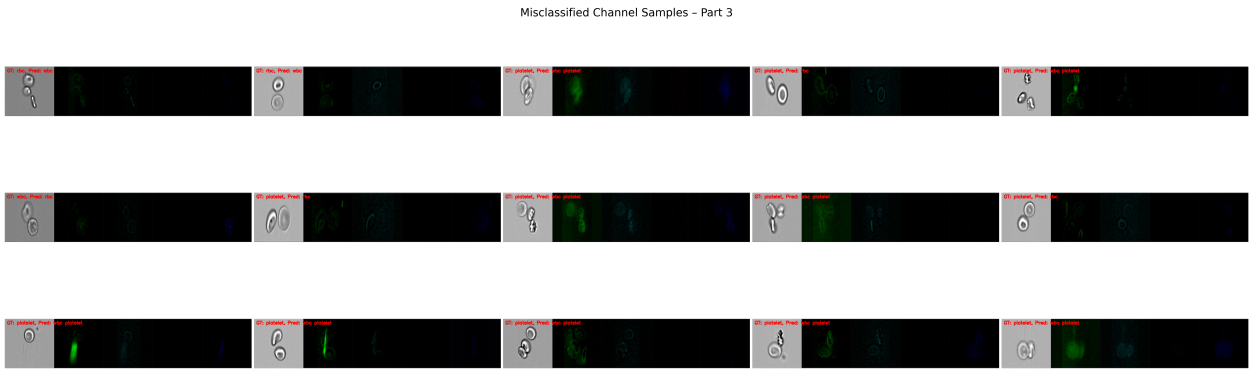


Figure A.15: Color Misclassifications Part 3 Additional color classification errors with overlapping fluorescence signals or weak staining contributing to incorrect labels.

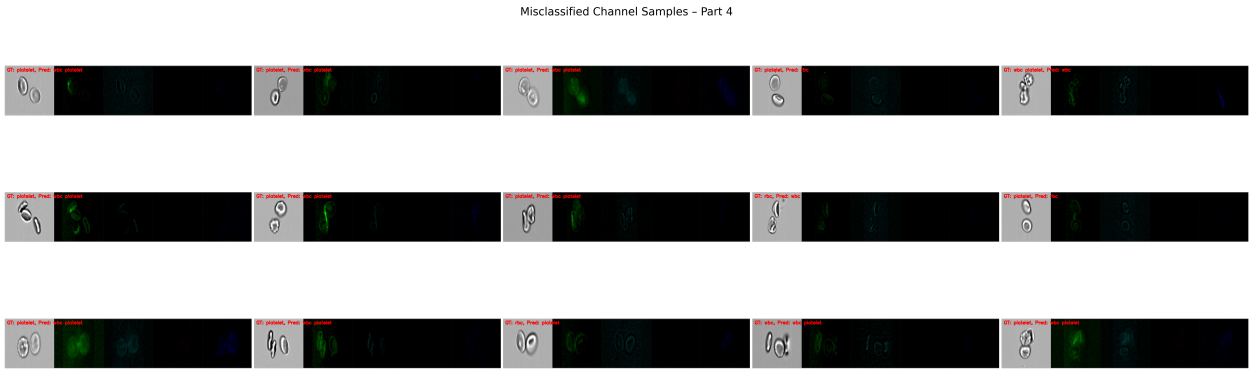


Figure A.16: Color Misclassifications Part 4 Examples of color classification errors where marker interpretation was affected by weak or ambiguous staining patterns.

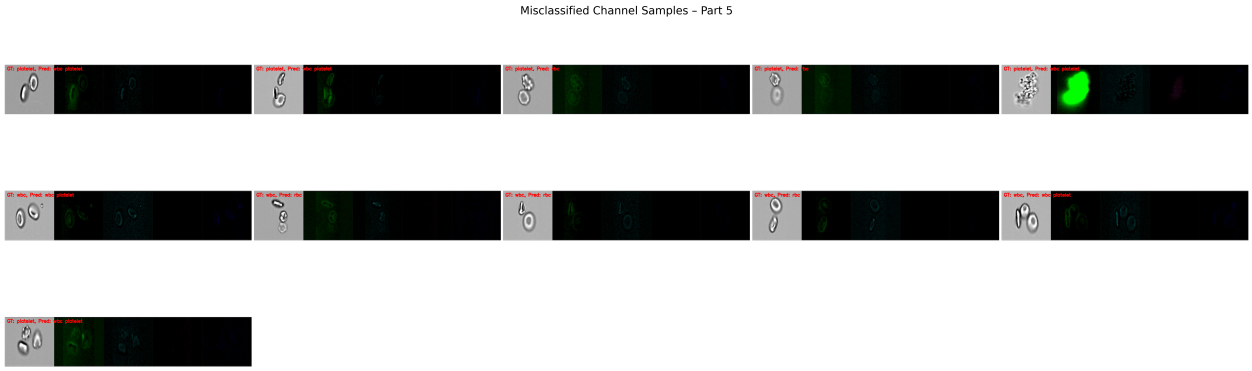


Figure A.17: Color Misclassifications Part 5 Final set of color misclassified samples, including borderline or low-contrast cases challenging for threshold-based detection.

Bibliography

[1] Tang N, Li D, Wang X, Sun Z. Abnormal coagulation parameters are associated with poor prognosis in patients with novel coronavirus pneumonia. J Thromb Haemost. 2020;18:844-847.

[2] Hanff, Thomas C., et al. "Thrombosis in COVID-19." American journal of hematology 95.12 (2020): 1578-1589.

[3] Jayarangaiah, Apoorva, et al. "COVID-19-associated coagulopathy: an exacerbated immunothrombosis response." Clinical and Applied Thrombosis/Hemostasis 26 (2020): 1076029620943293.

[4] Dorken-Gallastegi, Ander, et al. "Circulating cellular clusters are associated with thrombotic complications and clinical outcomes in COVID-19." Iscience 26.7 (2023).

[5] Marcos-Jubilar, María, Ramón Lecumberri, and José A. Páramo. "Immunothrombosis: molecular aspects and new therapeutic perspectives." Journal of Clinical Medicine 12.4 (2023): 1399.

[6] Butenas, Saulius et al. “Tissue factor in coagulation: Which? Where? When?.” Arteriosclerosis, thrombosis, and vascular biology vol. 29,12 (2009): 1989-96. doi:10.1161ATVBAHA.108.177402

[7] Li, Yangxin, et al. "Inflammasomes as therapeutic targets in human diseases." Signal transduction and targeted therapy 6.1 (2021): 247.

[8] Brinkmann, Volker, et al. "Neutrophil extracellular traps kill bacteria." science 303.5663 (2004): 1532-1535.

[9] Bonner, W. A., et al. "Fluorescence activated cell sorting." Review of Scientific Instruments 43.3 (1972): 404-409.

[10] McKinnon, Katherine M. “Flow Cytometry: An Overview.” Current protocols in immunology vol. 120 5.1.1-5.1.11. 21 Feb. 2018, doi:10.1002/cpim.40

[11] Schneck, Emmanuel, et al. "Flow Cytometry-Based Quantification of Neutrophil Extracellular Traps Shows an Association with Hypercoagulation in Septic Shock and Hypocoagulation in Postsurgical Systemic Inflammation—A Proof-of-Concept Study." Journal of Clinical Medicine 9.1 (2020): 174.

[12] Heestermans, Marco, et al. "Immunothrombosis and the role of platelets in venous thromboembolic diseases." International Journal of Molecular Sciences 23.21 (2022): 13176.

[13] Verschoor, Chris P., et al. "An introduction to automated flow cytometry gating tools and their implementation." Frontiers in immunology 6 (2015): 380.

[14] Liu, Peng, et al. "Comprehensive evaluation and practical guideline of gating methods for high-dimensional cytometry data: manual gating, unsupervised clustering, and auto-gating." Briefings in Bioinformatics 26.1 (2025): bbae633.

[15] Eslami, Mohammed, et al. "AutoGater: a weakly supervised neural network model to gate cells in flow cytometric analyses." Scientific Reports 14.1 (2024): 23581.

[16] Fisch, Lukas, et al. "GateNet: A novel neural network architecture for automated flow cytometry gating." Computers in Biology and Medicine 179 (2024): 108820.

[17] Sriphum, Wiwat. FLOPTICS: A novel automated gating technique for flow cytometry data. Diss. University of Southampton, 2023.

[18] Luo, Shaobo, et al. "Machine-learning-assisted intelligent imaging flow cytometry: A review." Advanced Intelligent Systems 3.11 (2021): 2100073.

[19] Monaghan, Sara A., et al. "A machine learning approach to the classification of acute leukemias and distinction from nonneoplastic cytopenias using flow cytometry data." American journal of clinical pathology 157.4 (2022): 546-553.

[20] Cohen, Anat, et al. "Label-free imaging flow cytometry for cell classification based on multiple interferometric projections using deep learning." Advanced Intelligent Systems 6.1 (2024): 2300433.

[21] Lewis, Joshua E., et al. "Automated deep learning-based diagnosis and molecular characterization of acute myeloid leukemia using flow cytometry." Modern Pathology 37.1 (2024): 100373.

[22] Cheng, Fu-Ming, et al. "Deep learning assists in acute leukemia detection and cell classification via flow cytometry using the acute leukemia orientation tube." Scientific Reports 14.1 (2024): 8350.

[23] Cheng, Zhangkai J., et al. "Artificial intelligence reveals the predictions of hematological indexes in children with acute leukemia." BMC cancer 24.1 (2024): 993.

[24] Hybel, Trine Engelbrecht, et al. "Imaging Flow Cytometry and Convolutional Neural Network-Based Classification Enable Discrimination of Hematopoietic and Leukemic Stem Cells in Acute Myeloid Leukemia." International Journal of Molecular Sciences 25.12 (2024): 6465.

[25] Vora, Nilay, et al. "Deep learning-enabled detection of rare circulating tumor cell clusters in whole blood using label-free, flow cytometry." Lab on a Chip 24.8 (2024): 2237-2252.

[26] Becht, Etienne, et al. "High-throughput single-cell quantification of hundreds of proteins using conventional flow cytometry and machine learning." Science advances 7.39 (2021): eabg0505.

[27] Lippeveld, Maxim, et al. "Classification of human white blood cells using machine learning for stain-free imaging flow cytometry." Cytometry Part A 97.3 (2020): 308-319.

[28] Park, Seongcheol, et al. "A simple approach to biophysical profiling of blood cells in extranodal NK/T cell lymphoma patients using deep learning-integrated image cytometry." BMEMat (2024): e12128.

[29] Lemieux, Madeleine E., et al. "Detection of early-stage lung cancer in sputum using automated flow cytometry and machine learning." Respiratory research 24.1 (2023): 23.

[30] Rosenberg, Carina A., et al. "Exploring dyserythropoiesis in patients with myelodysplastic syndrome by imaging flow cytometry and machine-learning assisted morphometrics." Cytometry Part B: Clinical Cytometry 100.5 (2021): 554-567.

[31] Wilkins, Malcolm F., et al. "Comparison of five clustering algorithms to classify phytoplankton from flow cytometry data." Cytometry: The Journal of the International Society for Analytical Cytology 44.3 (2001): 210-217.

[32] Li, Yueqin, et al. "Deep cytometry: deep learning with real-time inference in cell sorting and flow cytometry." Scientific reports 9.1 (2019): 11088.

[33] Eulenberg, Philipp, et al. "Deep learning for imaging flow cytometry: cell cycle analysis of Jurkat cells." bioRxiv (2016): 081364.

[34] Gupta, Anindya, et al. "Deep learning in image cytometry: a review." Cytometry Part A 95.4 (2019): 366-380.

[35] Liu, Chao, et al. "High-content video flow cytometry with digital cell filtering for label-free cell classification by machine learning." Cytometry Part A 103.4 (2023): 325-334.

[36] Bini, Lorenzo, et al. "FlowCyt: A Comparative Study of Deep Learning Approaches for Multi-Class Classification in Flow Cytometry Benchmarking." arXiv preprint arXiv:2403.00024 (2024).

[37] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[38] Ma, Li, et al. "Combining DC-GAN with ResNet for blood cell image classification." Medical & biological engineering & computing 58 (2020): 1251-1264.

[39] Zhu, Ziquan, et al. "RDNet: ResNet-18 with dropout for blood cell classification." International Work-Conference on the Interplay Between Natural and Artificial Computation. Cham: Springer International Publishing, 2022.

[40] Farooq, Muhammad, and Abdul Hafeez. "Covid-resnet: A deep learning framework for screening of covid19 from radiographs." arXiv preprint arXiv:2003.14395 (2020).

[41] Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." International conference on machine learning. PMLR, 2019.

[42] Batool, Amreen, and Yung-Cheol Byun. "Lightweight EfficientNetB3 model based on depthwise separable convolutions for enhancing classification of leukemia white blood cell images." IEEE access 11 (2023): 37203-37215.

[43] Ramamurthy, Karthik, et al. "A deep learning network for Gleason grading of prostate biopsies using EfficientNet." Biomedical Engineering/Biomedizinische Technik 68.2 (2023): 187-198.

[44] Ravi, Vinayakumar, Vasundhara Acharya, and Mamoun Alazab. "A multichannel EfficientNet deep learning-based stacking ensemble approach for lung disease detection using chest X-ray images." Cluster Computing 26.2 (2023): 1181-1203.

[45] Sandler, Mark, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[46] Akay, Metin, et al. "Deep learning classification of systemic sclerosis skin using the MobileNetV2 model." IEEE Open Journal of Engineering in Medicine and Biology 2 (2021): 104-110.

[47] Kumar, Indrajeet, and Jyoti Rawat. "Segmentation and classification of white blood SMEAR images using modified CNN architecture." Discover Applied Sciences 6.11 (2024): 1-18.

[48] Ragab, Mahmoud, et al. "COVID-19 identification system using transfer learning technique with mobile-NetV2 and chest X-ray images." Frontiers in Public Health 10 (2022): 819156.

[49] Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[50] Houssein, Essam H., et al. "Using deep DenseNet with cyclical learning rate to classify leukocytes for leukemia identification." Frontiers in Oncology 13 (2023): 1230434.

[51] Bozkurt, Ferhat. "Classification of blood cells from blood cell images using dense convolutional network." Journal of Science, Technology and Engineering Research 2.2 (2021): 81-88.

[52] Hasan, Najmul, et al. "DenseNet convolutional neural networks application for predicting COVID-19 using CT image." SN computer science 2.5 (2021): 389.

[53] Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018).

[54] Achiam, Josh, et al. "Gpt-4 technical report." arXiv preprint arXiv:2303.08774 (2023).

[55] Johnson, Olanrewaju Victor, Osamah Mohammed Alyasiri, and Olabisi Esher Johnson. "Image Analysis through the lens of ChatGPT-4." Journal of Applied Artificial Intelligence 4.2 (2023): 31-46.

[56] Ren, Zhiyuan, Yiyang Su, and Xiaoming Liu. "ChatGPT-powered hierarchical comparisons for image classification." Advances in neural information processing systems 36 (2024).

[57] Wu, Chenfei, et al. "Visual chatgpt: Talking, drawing and editing with visual foundation models." arXiv preprint arXiv:2303.04671 (2023).

[58] Kirillov, Alexander, et al. "Segment anything." Proceedings of the IEEE/CVF international conference on computer vision. 2023.

[59] Liu, Shilong, et al. "Grounding dino: Marrying dino with grounded pre-training for open-set object detection." European Conference on Computer Vision. Cham: Springer Nature Switzerland, 2024.

[60] Selvaraju, Ramprasaath R., et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization." Proceedings of the IEEE international conference on computer vision. 2017.

[61] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05). Vol. 1. Ieee, 2005.