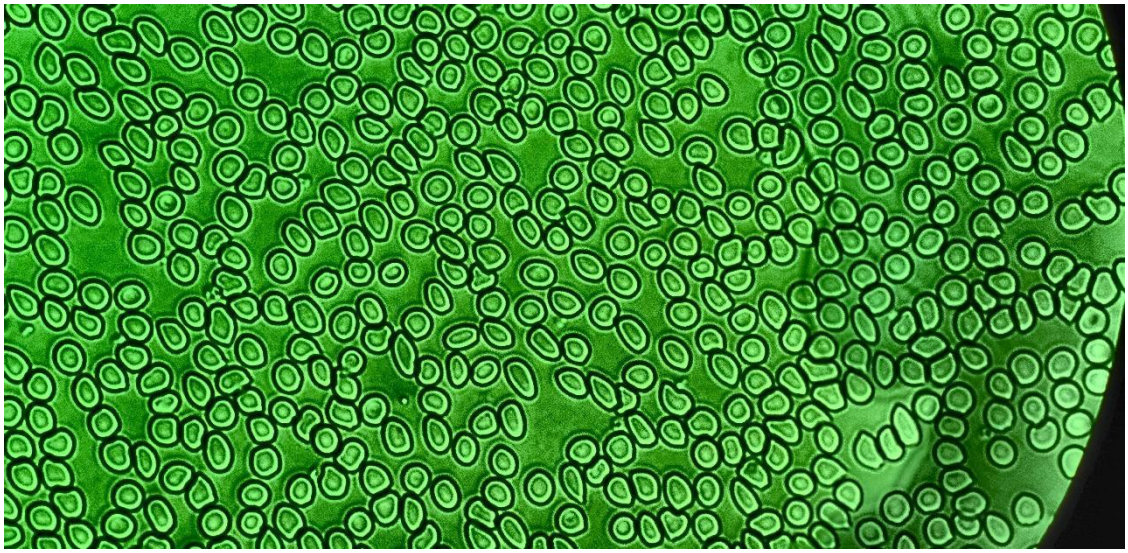
Inholland - University of Applied Sciences
Applied Mathematics & Data Science

Thesis

Machine Learning Approaches in Study of Multiple Sclerosis Disease Through Image Classification Models



Author: José Moreno Mendes (578148)
Supervisor: Frank Brandse

May 16, 2025

Management Summary

This thesis project, conducted in collaboration with RR Mechatronics and Inholland University of Applied Sciences, explored machine learning for detecting Multiple Sclerosis (MS) using blood cell images. The goal of this project was to assess the feasibility of using machine learning models to support early and accessible MS diagnosis, a field with limited existing research.

The research aimed to identify which image features in blood cells are most indicative of MS. To do so, the study focused on various machine learning models, addressing key questions on preprocessing strategies, model selection, performance optimization, evaluation methods and the application of explainable AI.

A mixed-methods approach was used, combining qualitative analysis of image features with quantitative model evaluation. Performance was assessed using metrics such as precision, recall and F1-score. All experiments were conducted using Python-based tools and a validated dataset from Southampton General Hospital.

Among the tested models, the pretrained VGG-19 model achieved a validation accuracy of up to 90%, showing strong potential for differentiating MS from healthy controls. Performance was further enhanced through techniques such as regularization and hyperparameter tuning. A reliable preprocessing pipeline, including data augmentation and normalization, also played a key role in boosting model performance. Explainable AI tools like SHAP and LIME provided valuable insights into the model's decision-making process.

This research demonstrates that deep learning, when combined with robust preprocessing and explainable AI, offers promising support for MS diagnosis via blood cell images. The findings underscore the potential of pretrained models like VGG-19 to contribute to early, accessible and transparent diagnostic tools.

Despite promising results, the study was limited by a small dataset of 314 images and a constrained 20-week timeline. Future research should focus on expanding the dataset, incorporating advanced techniques like image segmentation and leveraging high-performance computing to enhance model accuracy and generalizability.

Acknowledgements

I would like to express my sincere gratitude to all those who have supported and guided me throughout the course of this thesis project.

First and foremost, I wish to thank **Frank Brandse**, my internship supervisor, for his dedicated support, critical insights and thoughtful feedback during this research process. I am also grateful to **Marya Butt**, my company supervisor, for her valuable advice and continued encouragement. Furthermore, I would like to acknowledge the constructive consultations held with **Jan de Zoeten**, whose input has been instrumental in shaping the direction of this work.

My appreciation also goes to **Vera Hollink** and **Harald Drillenburg** for their thoughtful advice and helpful reflections, which contributed meaningfully to the refinement of my research.

I am also thankful to **RR Mechatronics** for granting me access to the dataset and for their trust in me to conduct this research within their organization.

On a more personal note, I would like to thank my **family** for their unconditional moral support, patience and practical assistance throughout this journey. In particular, I am profoundly grateful to my partner, **Sanne**, for helping me persevere during the most demanding moments, for taking on additional responsibilities at home and for her constant understanding and encouragement during the writing process. Finally, to my children, **Saskia** and **Julian**, who are the very reason I embarked on this academic journey.

Acronym and Keyword Explanation

- **AI** – Artificial Intelligence
- **ANN** – Artificial Neural Networks
- **AUC** – Area Under the Curve
- **CLAHE** – Contrast Limited Adaptive Histogram Equalization
- **CNNs** – Convolutional Neural Networks
- **DDSS** – Data Driven Smart Society
- **DIE** – Digital Image Processing
- **HC** – Healthy Cells – blood cells from healthy individuals
- **JPEG** – Joint Photographic Experts Group
- **K-Fold CV** – K-Fold Cross-Validation
- **LIME** – Local Interpretable Model-Agnostic Explanations
- **MS** – Multiple Sclerosis
- **MSp** – Multiple Sclerosis Patients – blood cells from individuals affected by MS
- **PILLOW** - Python Imaging Library
- **RIC-TOI** – Research & Innovation Centre for Technology and Computer Science
- **ROI** – Region of Interest
- **ROC** – Receiver Operating Characteristics
- **ReLU** - Rectified Linear Unit
- **RGB** – Red, Green and Blue
- **SHAP** – Shapley Additive Explanations
- **VGG-19** – Visual Geometry Group with 19 layers
- **XAI** – Explainable Artificial Intelligence
- **Xception** – Extreme Inception

Table of contents

Table of figures

1. Introduction

In the field of medical care, errors or incorrect diagnoses can have life-threatening consequences. The Lorrca instrument, developed by RR Mechatronics, is designed to analyse blood cells and provide indicative features related to various diseases (RR Mechatronics, 2024c). However, the device currently lacks the reliability and comprehensiveness required for accurately identifying indicators of Multiple Sclerosis. In response to this limitation, RR Mechatronics enlisted the expertise of Inholland's Data Driven Smart Society research group to support the development of a solution to this problem.

1.1 Context

Inholland University of Applied Sciences is an educational institution that hosts a variety of research departments engaged in diverse projects. One such department is the Data Driven Smart Society (DDSS), which operates within the Research & Innovation Centre for Technology and Computer Science (RIC-TOI). DDSS primarily focuses on data analysis, with a particular emphasis on practice-oriented research in the field of artificial intelligence (AI), including neural networks. The department is involved in a range of applied research projects, including sustainable agriculture and horticulture, preventive healthcare initiatives and the design of future-resilient urban environments. DDSS maintains active partnerships with numerous external organizations, including RR Mechatronics (Inholland, 2025).

Founded in 1986 and headquartered in Zwaag, the Netherlands, RR Mechatronics specializes in the development and distribution of advanced diagnostic instruments for the healthcare sector. The company designs various devices that assist doctors and laboratories in analysing blood samples. These instruments measure key factors in the blood, such as the rate at which red blood cells settle (known as the erythrocyte sedimentation rate), the characteristics of red blood cells and the process of preparing special substances (called reagents) that are used to perform blood tests (RR Mechatronics, 2024a).

In addition to its product offerings, RR Mechatronics provides comprehensive services such as installation, maintenance and user training to support the effective operation of its equipment. Given the critical importance of accuracy and reliability in healthcare diagnostics, the company's commitment to high-quality and dependable instrumentation is central to its mission. RR Mechatronics serves a global customer base, primarily composed of laboratories that specialize in the detection and diagnosis of various medical conditions (RR Mechatronics, 2024a).

An additional key activity of RR Mechatronics involves investing in research and development initiatives in collaboration with external organizations. These research efforts

encompass a range of topics, including red blood cell (RBC) membrane disorders, infectious diseases, erythrocyte enzyme deficiencies, among others. The company's research and development team collaborates with international partners such as academic research groups, clinical laboratories, pharmaceutical companies and universities, including Inholland University of Applied Sciences (RR Mechatronics, 2024b).

At present, RR Mechatronics is engaged in a joint project with Inholland's Data Driven Smart Society (DDSS), focusing on the application of machine learning techniques in the study of Multiple Sclerosis (MS). MS is a chronic autoimmune disorder that affects the central nervous system, characterized by inflammation and degradation of the myelin sheath surrounding nerve fibers, as well as progressive neurodegeneration (Naji, Mahdaoui, Klevor, & Kissani, 2023). From a biomedical perspective, early and effective intervention using disease-modifying therapies has been shown to significantly improve outcomes for individuals diagnosed with MS (Dobson & Giovannoni, 2018).

RR Mechatronics aims to position itself as a pioneer in the field of medical diagnostics by developing an innovative approach to detecting Multiple Sclerosis (MS) through the analysis of blood cells. At present, no other entity worldwide is utilizing this technique to diagnose MS, making RR Mechatronics' initiative both groundbreaking and highly promising. By leveraging advanced technologies to analyse blood cell characteristics, the company seeks to offer a non-invasive, efficient

and potentially more accessible method of diagnosing MS, a condition that traditionally requires more complex and invasive diagnostic procedures. Through this innovative approach, RR Mechatronics strives to revolutionize the early detection of MS, improving patient outcomes and expanding the scope of diagnostic capabilities in the medical field.

1.2 Main research question and subquestions

1.2.1 Main research question

What features in blood images are most indicative of multiple sclerosis machine learning classification when comparing image recognition models?

1.2.2 Subquestions

1. What preparations should be made during the preprocessing phase?
2. Which machine learning models are suitable for image classification tasks?
3. Which techniques are suitable to optimize model performance?
4. How can model performance be evaluated and compared?
5. How can explainable AI techniques identify visual features or regions in blood images?

2. Methodology

2.1 Research Strategy

This project will conduct an empirical research approach, utilizing both qualitative and quantitative methods to gather data and conduct experiments aimed at addressing the main research question and its sub-questions. The qualitative component will focus on assessing image attributes that are not represented numerically. In contrast, the quantitative component will involve the application of statistical measures such as precision, accuracy, F1-score and recall for the evaluation and optimization of machine learning models.

Additionally, exploratory research will be conducted through a comprehensive literature review using various academic databases. This will support the investigation of relevant topics, including preprocessing techniques, classification models, model optimization and explainable artificial intelligence methods. Predictive research will also be undertaken to explore the potential for predicting the presence of Multiple Sclerosis (MS) based on blood cell imagery through experimental analysis.

Explanatory research will be applied within the context of explainable AI to identify and interpret visual features or regions of interest within the blood cell images. Furthermore, deductive research will be employed by drawing on general theoretical insights and testing hypotheses to generate empirical evidence and formulate conclusions. The deductive experiments will be implemented using the Python programming language.

The Multiple Sclerosis (MS) blood cell dataset was collected in October 2023 at Southampton General Hospital in England. The data collection process was supervised by Professor Dr. Ian Galea and Dr. Carmen Jacob. All blood smear slides were prepared by Jan de Zoeten, Research Director at RR Mechatronics, with the specific aim of minimizing inter-individual variability.

Following consultations with the client representative Marya Butt (company supervisor) and Jan de Zoeten, it was concluded that the acquired blood cell image data are sufficient to meet the objectives of this project. Consequently, no additional data collection will be undertaken.

2.2 Research Design

2.2.1 Subquestion 1 - What preparations should be made during the preprocessing phase?

To address this subquestion, exploratory research will be conducted through an extensive literature review and examination of online documentation, including that of relevant Python libraries such as TensorFlow a framework that provides tools for implementing machine learning models. The research will focus on identifying and analysing preprocessing techniques applicable to Multiple Sclerosis (MS) blood cell images. These techniques may include image enhancement, data augmentation and data normalization.

Following the exploratory phase, deductive research will be employed to apply the selected preprocessing techniques to the dataset. This will be implemented using the Python programming language, utilizing various specialized libraries for image data preprocessing. Finally, explanatory research will be conducted to provide a rationale for the selection and application of specific preprocessing methods.

2.2.2 Subquestion 2 - Which machine learning models are suitable for image classification tasks?

To address this subquestion, exploratory research will be conducted through a literature review and online resources. In the context of this development project, machine learning models capable of performing binary classification tasks on image datasets will be investigated. Multiple models will be selected, evaluated and compared. Additionally, explanatory research will be undertaken to justify the selection of specific models.

2.2.3 Subquestion 3 - Which techniques are suitable to optimize model performance?

The aim of this subquestion is to identify the optimal hyperparameters for a given algorithm. To address this question, exploratory, predictive and explanatory research will be conducted. In the exploratory phase, hyperparameter tuning techniques will be explored through a comprehensive literature review and the documentation of relevant libraries, such as TensorFlow and Scikit-Learn, will be examined.

Subsequently, predictive research will be carried out on the blood cell images by applying the techniques identified in the exploratory phase. During the deductive research phase, the Python programming language will be employed to conduct experiments and the most optimal hyperparameters will be selected based on their performance when evaluated with the validation dataset. Finally, explanatory research will be conducted to provide a rationale for the selection of the chosen hyperparameters.

2.2.4 Subquestion 4 - How can model performance be evaluated and compared?

To address this subquestion, exploratory, predictive and explanatory research will be undertaken. Initially, a study will be conducted to identify appropriate statistical evaluation methods for binary classification tasks applied to blood cell images. In the predictive phase, performance evaluation will be carried out on the test dataset to assess the algorithms performance with new, unseen data. For this task, the Python programming language will be utilized.

Finally, explanatory research will be conducted to provide a detailed description of how the evaluation experiment was executed.

2.2.5 Subquestion 5 - How can explainable AI techniques identify visual features or regions in blood images?

To address this subquestion, both exploratory and explanatory research will be conducted. During the exploratory phase, a literature review will be undertaken to gather relevant information on the topic. Additionally, the application of explainable AI techniques using the Python programming language will be explored through deductive research.

Furthermore, explanatory research will be carried out to explain how machine learning models can identify, predict and interpret visual features or regions of interest within the blood images.

3. Results and analysis

3.1 Subquestion 1 - What preparations should be made during the preprocessing phase?

To enhance the computational efficiency of machine learning algorithms, various preprocessing techniques are commonly applied to filter and refine the data (Amato & Di Lecce, 2023). The blood cell images related to Multiple Sclerosis (MS) represent raw data captured through microscopy. Consequently, preprocessing procedures are crucial for improving the quality of the images. Techniques such as data exploration, digital image processing, data augmentation and data splitting will be employed to optimize the dataset for further analysis.

3.1.1 Data Exploration

The objective of this phase is to gain a comprehensive understanding of the dataset. The blood cell dataset comprises two classes: HC (representing healthy cells) and MSp (representing blood cells affected by Multiple Sclerosis). The dataset includes images from 16 patients, with 7 classified under HC and 9 under MSp. In total, the dataset contains 314 blood cell images: 110 from HC and 204 from MSp. Figure 1 provides a visual representation of the dataset distribution, highlighting the class imbalance.

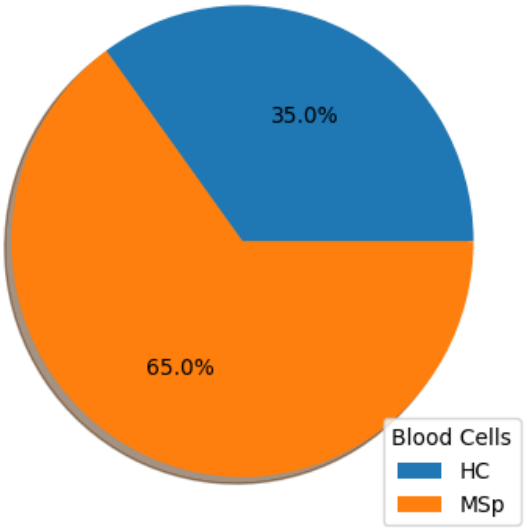


Figure 1: Blood Cells Distribution

To conduct further analysis of the images, the Pillow Python library (PILLOW) is employed. All images are rectangular in shape, represented as 2D arrays, with dimensions of 1960 pixels in width (columns) and 4032 pixels in height (rows). Additionally, the image files are stored in JPEG format and utilize the RGB color model.

3.1.2 Digital Image Processing

Image Enhancement

One of the most effective tools for image enhancement is the examination of the image histogram, which provides valuable information regarding the distribution of RGB levels. The histogram is computed for each individual color space and then combined to generate the overall histogram of the image. This histogram plots the image's brightness levels against the number of pixels corresponding to each level. The shape and position of the histogram contain crucial information regarding the image's contrast and brightness (Umbaugh, 2023).

Histogram modification aims to adjust pixel intensities by either stretching, shrinking, or shifting them in order to enhance the image's representation. To facilitate this process, the Python libraries OpenCV and Matplotlib are utilized to apply and visualize the histogram distribution of the original image, as well as the images after histogram equalization and Contrast-Limited Adaptive Histogram Equalization (CLAHE).

Figure 2 demonstrates the enhancement in image contrast by stretching the pixels intensity. The histograms indicate a significant number of pixels with an intensity of zero (black pixels). In the first image, the intensity of the blue channel predominantly ranges from 0 to 75, the red channel from 50 to 90 and the green channel from 150 to 200. In contrast, the second image, after the application of histogram equalization, shows a more uniform distribution of pixel intensities across the range of 0 to 250. On the other hand, the CLAHE technique also results in a well-distributed intensity range, with a reduction in the number of extreme pixel values, leading to a slight improvement in image contrast when compared to the other images.

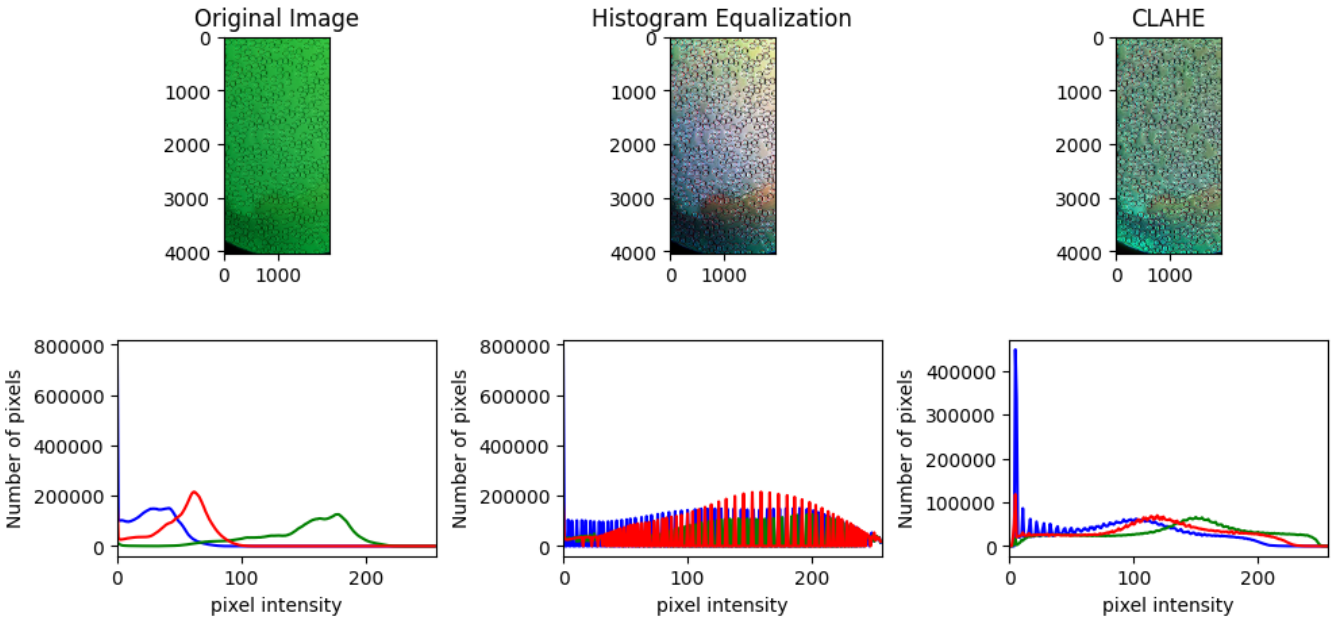


Figure 2: Histogram Modification

Image Compression

Image compression is the process of reducing the size of image data files while preserving the essential information (Umbaugh, 2023). The original blood cell images have dimensions of 1960 pixels (width) and 4032 pixels (height). To standardize the image size, padding is applied, adding extra pixels to create a square image of 4032 by 4032 pixels. Given the difficulty of consistently estimating the background color across the blood cell image samples, black was chosen as the padding color (Chennamsetty, 2018). This step is implemented in a way that ensures the resized image remains centered, a process known as loss-less conversion, which preserves the features of the original image (Shahnaz & Mollah, 2023). Additionally, it improves edge detection and minimizes data loss at the image borders (Islam, Assaduzzaman, & Zahid Hasan, 2024).

Subsequently, all images were resized to 224 × 224 pixels prior to being input into the models. This resizing was applied because most pretrained architectures are designed to accept this input size and are trained on resized images, which makes them robust to such preprocessing (Keras, 2025b). To minimize the loss of the image details, interpolation method used in this process is INTER_AREA, which resamples the image using pixel area relations (OpenCV, 2025). For this task, the Python programming language and the OpenCV library are employed to compress the images.

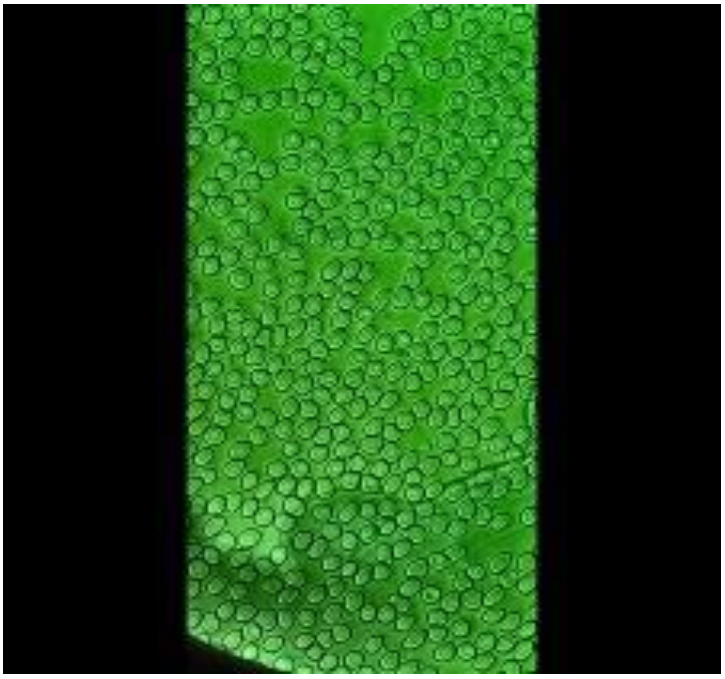


Figure 3: Image compression

3.1.3 Data Splitting

In machine learning, it is essential to partition the dataset into three subsets: the training set, the validation set and the test set. The training set is used by the models to learn patterns in the data, the validation set is employed for fine-tuning the models to improve performance and the test set is used for final evaluation after model development (Liu, 2020).

For this research, the dataset was shuffled to ensure generalization and the division was made based on the number of images and the distribution of patients. The test set consists of 84 images, with 42 images from each class, including two patients from the Multiple Sclerosis (MSp) class (MSp-J41 and MSp-J42) and one patient from the Healthy Control (HC) class (HC-J01). The training set includes the remaining patients, comprising 230 images, 162 from the MSp class and 68 from the HC class.

The validation set is used in conjunction with k-fold cross-validation, where the training data is split into 'K' folds and the model is evaluated on each fold in turn. This approach offers a significant advantage, particularly in cases with small sample sizes, such as inverse inference problems (Scikit-learn, 2025b). See figure 4 for a visual representation.

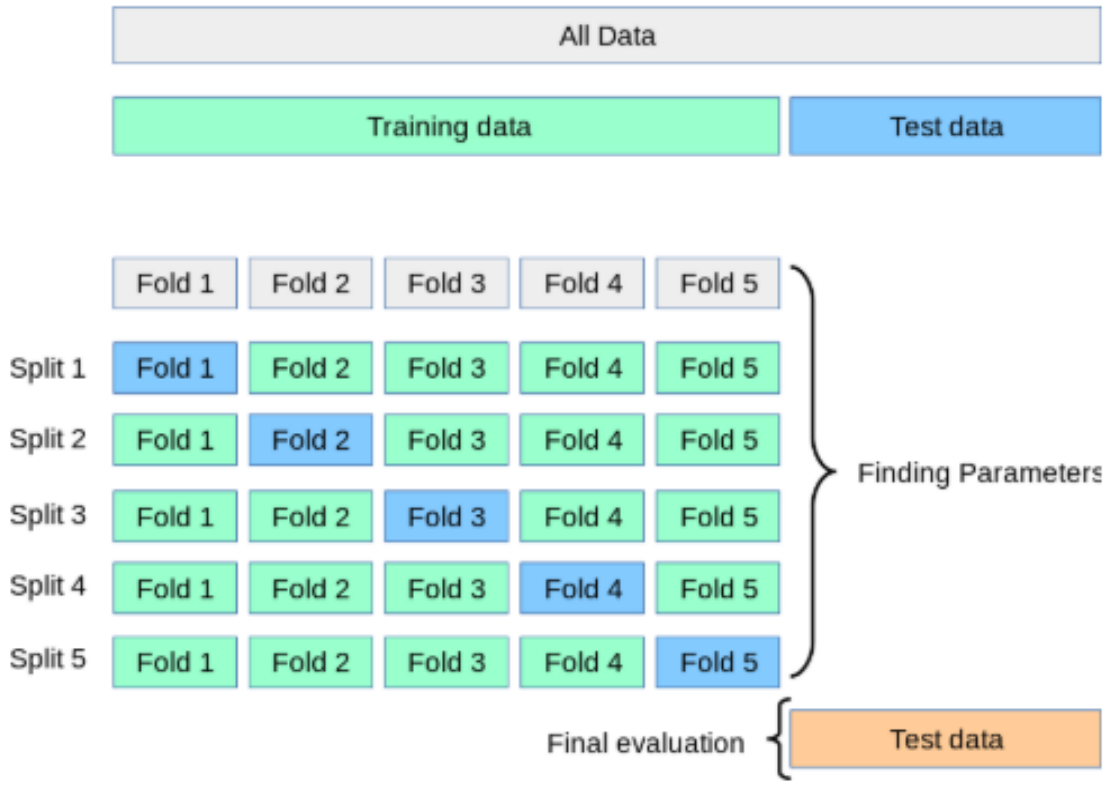


Figure 4: Cross Validation

3.1.4 Data Augmentation

Data augmentation involves generating additional training data by applying random transformations to the existing samples, thus producing new images that are realistic. This technique is employed to prevent overfitting, a common issue that arises when there are insufficient training samples, which prevents the model from generalizing effectively to new, unseen data (Chollet, 2021).

Given the limited size of the blood cell dataset, data augmentation is utilized to expand the training set. Various augmentation techniques are applied, including image flipping, cropping, Gaussian blur, contrast normalization, additive Gaussian noise and rotation.

Data augmentation is often regarded as a form of preprocessing specific to the training set (Goodfellow, Bengio, & Courville, 2016). Consequently, this technique is applied exclusively to the training dataset, increasing the number of instances from 230 to 648, with 324 images per class. This process has effectively balanced the dataset through oversampling of the Healthy Control (HC) class (Thabtah, 2020).

3.1.5 Data Normalization

Before being input into the neural network, data must be appropriately preprocessed into floating-point tensors (Chollet, 2018). To achieve this, the pixel values of the blood cell dataset are rescaled using the Min-Max Scaler, which transforms the pixel values from the range [0, 255] to the range [0, 1], as illustrated in Figure 5.

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Figure 5: Min-Max Scaler Formula

3.1.6 Label Encoding

This research focuses on binary classification of blood cells, with one class representing healthy blood cells (HC) and the other representing blood cells affected by Multiple Sclerosis (MSp). Machine learning algorithms, with a few exceptions, typically require numerical values for processing (Liu, 2020). Therefore, it is necessary to encode the classes using label encoding. As a result, the HC class is represented by the value 0, while the MSp class is represented by the value 1. The label encoding process is carried out using Python and the Scikit-learn library.

3.2 Subquestion 2 - Which machine learning models are suitable for image classification tasks?

Deep learning, a subfield of machine learning, involves algorithms inspired by the structure and functioning of the human brain, known as Artificial Neural Networks (ANNs) (Chen, Engkvist, Wang, Olivecrona, & Blaschke, 2018). Deep learning techniques are extensively applied across various domains, including natural language processing, time series analysis, regression and classification tasks. In particular, deep learning has demonstrated exceptional performance in handling image data.

Convolutional Neural Networks (CNNs) represent a specialized type of neural network designed to process data with a known, grid-like topology, such as image data, which can be conceptualized as a two-dimensional grid of pixels (Goodfellow, Bengio, & Courville, 2016). Furthermore, images are typically represented as rank-4 tensors with the shape (samples, height, width, channels), where each pixel is described by a vector of values across the different "channels" (Chollet, 2021).

CNNs consist of multiple layers and as the number of hidden layers in artificial neural networks increases, the network's complexity also increases, typically resulting in improved performance. Common types of layers within CNNs include the Convolutional Layer, Pooling Layer, Fully Connected Layer, Dense Layer, among others (Molchanov, Tyree, Karras, Aila & Kautz, 2017).

Following preprocessing and the application of data augmentation techniques to address class imbalance, the blood images dataset contains only 732 images. Deep learning algorithms generally require large amounts of data to train models effectively, enabling them to produce accurate generalizations across different classes. A widely adopted and highly effective strategy for applying deep learning to small image datasets is the use of a pretrained model. A pretrained model refers to a model that has been previously trained on a large dataset, typically for a large-scale image classification task (Chollet, 2021).

3.2.1 Transfer Learning

Numerous pretrained classification models are widely employed in deep learning applications. In the present study, the performance of VGG-19, Xception and InceptionV3 is evaluated on the blood cell dataset. These specific pretrained models were selected due to their demonstrated high accuracy in medical-related research domains, including brain big data classification, skin cancer detection, diabetic retinopathy classification (Jena, Nayak & Saxena, 2021) and white blood cells classification (Katar, 2022).

In this section, the pretrained models are loaded without their three fully connected layers. This approach makes it possible to construct and add a new top layer specifically designed to fit the features of the blood cell dataset. Moreover, the convolutional layers are frozen to prevent their weights from being updated during training, as allowing such modifications could alter the pretrained feature representations.

An empirical study was conducted to compare the performance of different models on the blood cell dataset. During the experiments, specific constraints were applied to ensure consistency across all models. For example, each model produced an output consisting of one flatten layer followed by one dense layer, with a sigmoid activation function applied to the final dense layer. The use of a sigmoid activation function is recommended for binary classification tasks, as it transforms raw output scores into probabilities between 0 and 1.

All models were trained and validated using cross-validation, with the validation set comprising 20% of the training set, which contained a total of 648 images. Furthermore, the models weights were initialized using ImageNet, a large-scale database containing over 14 million images and more than 100,000 classes (ImageNet, 2025).

Additionally, all models were trained for 30 epochs, where one epoch represents a complete iteration over the entire input (x) and output (y) datasets. The batch size was set to 16, indicating the number of samples processed before each gradient update. The Adam optimizer was employed which is a stochastic gradient descent method that relies on adaptive estimation of first-order and second-order moments.
The loss function used was binary cross-entropy, which calculates the cross-entropy metric between the true labels and the predicted outputs. Finally, binary accuracy was selected as the evaluation metric, as it is particularly suitable for binary classification tasks where labels are either 0 or 1 (Keras, 2025a).

Xception

Xception is a convolutional neural network architecture based on depth wise separable convolutional layers. It fully decouples the mapping of spatial correlations and cross-channel correlations within the feature maps. The model consists of 36 convolutional layers that extract features from the input images, organized into 14 modules, where each module except the first and last is connected through linear residual connections (Phukan & Gupta, 2022).

The Xception model demonstrated a performance comparable to that of InceptionV3, achieving 85% binary accuracy on the validation set and reaching 100% accuracy on the training set by the final epoch. This result indicates that the model was overfitting on the blood cell dataset. Furthermore, the validation loss did not follow the trend of the training loss, maintaining a gap of approximately 0.5%, which provides additional evidence of overfitting. See figures 6 and 9.

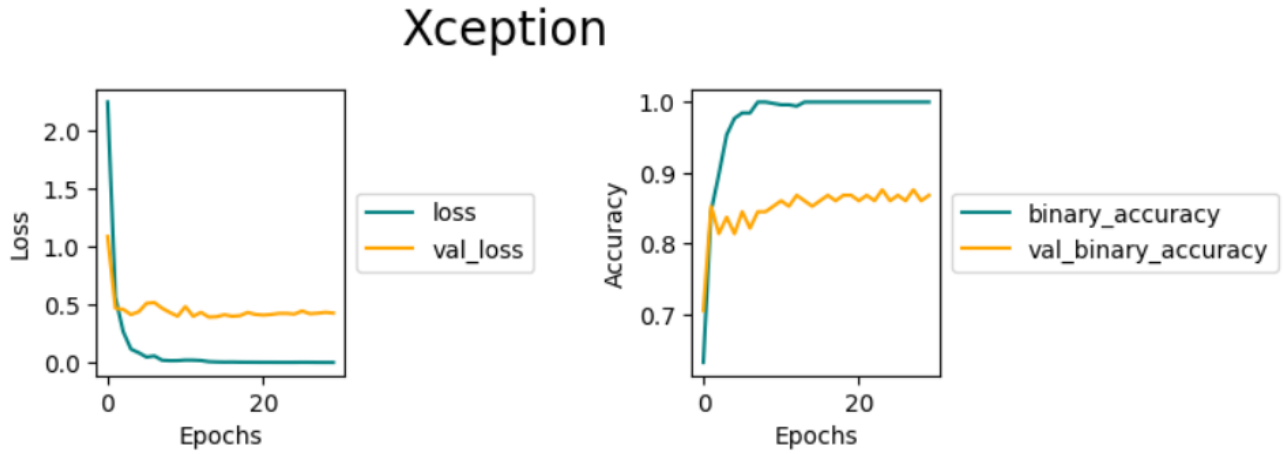


Figure 6: Xception performance

VGG-19

VGG-19 is a convolutional neural network consisting of 19 layers, including 16 convolutional layers and 3 fully connected layers used for image classification. It is a widely adopted model for image classification tasks, primarily due to its use of multiple 3×3 filters in each convolutional layer (Bansal, Kumar, Sachdeva & Mittal, 2021).

Among all models evaluated, VGG-19 achieved the highest performance, with a binary accuracy of 87%. Nevertheless, the model also exhibited signs of overfitting. On the other hand, the training loss function for VGG-19 showed superior behaviour compared to the other models, continuing to decrease over the epochs without plateauing at a constant level.

The validation loss demonstrated generally good performance, despite some fluctuations across different epochs. See figures 7 and 9.

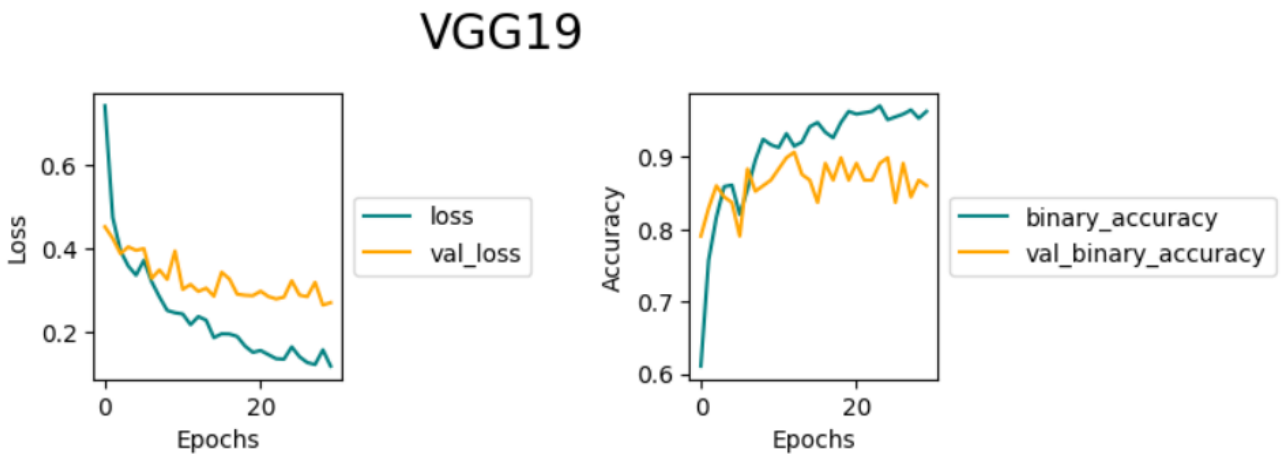


Figure 7: VGG-19 Performance

InceptionV3

InceptionV3 is a factorized convolutional model designed to reduce the number of connections and parameters without compromising the model's efficiency. It is constructed using standard convolutional components, such as convolutional layers, pooling layers, dropout layers and other commonly used elements (Jena, Nayak, & Saxena, 2021).

In the experiment conducted with the blood cell dataset, the model achieved a mean binary accuracy of 84% on the validation set, resulting in the worse model analysed. In addition, the model exhibited signs of overfitting, meaning that it performed better on the training set than on the validation set, characterized by high variance and low bias. Evidence of overfitting is also visible in the validation loss results, where the loss function plateaus and becomes constant. See figures 8 and 9.

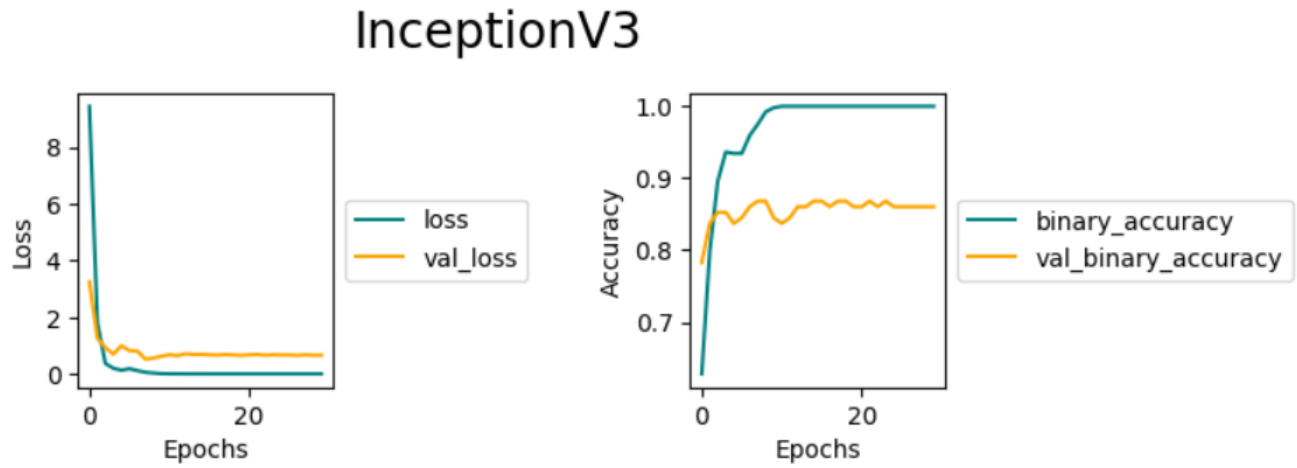


Figure 8: Inception V3 Performance

Model	Validation Mean Binary Accuracy
Xception	0.85
VGG19	0.87
InceptionV3	0.84

Figure 9: Models binary accuracy

3.3 Subquestion 3 -Which techniques are suitable to optimize model performance?

This section focuses on refining the most effective pretrained model identified in the previous subsection VGG-19, by addressing the balance between optimization and generalization. *Optimization* involves adjusting the model to achieve optimal performance on the training dataset, while *generalization* refers to the model's ability to maintain high performance when applied to previously unseen data (Chollet, 2021).

Neural networks are composed of interconnected nodes, organized in layers, where each node typically computes a weighted sum of the outputs from the previous layer. The terms *weights* and *biases* denote the model's *parameters*, as they are learned directly from the data through training. In contrast, elements such as the number of nodes per layer and the number of layers are referred to as *hyperparameters*, as they are set prior to training and determined by the practitioner (Agrawal, 2021).

3.3.1 Regularization

Regularization refers to a set of techniques used to prevent a model from overfitting the training data (Tian & Zhang, 2022). Overfitting occurs when a model learns patterns that are too specific to the training set, which can reduce its ability to perform well on new, unseen data. Regularization works by limiting the model's complexity, thereby encouraging it to learn more generalizable patterns. The goal is to improve the model's performance and reliability when evaluated on validation and test datasets (Chollet, 2021).

Dropout

Dropout is a widely used regularization technique in neural networks aimed at reducing overfitting (Liu, Xu, Jin, Shen & Darrell, 2023). It functions by randomly deactivating a subset of output units within a layer during each training iteration, thereby preventing the model from becoming overly reliant on specific neurons (Salehin, & Kang, 2023; Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). In this study, three dropout layers were implemented, each placed after a dense layer in the network architecture. The dropout rates were determined using Bayesian optimization, with selected values ranging between 0.1 and 0.5.

One notable advantage of dropout is its computational efficiency. During training, the application of dropout introduces minimal overhead, requiring only O(n) operations per example per update. Specifically, the generation of n random binary values and their element-wise multiplication with the layer's output (Goodfellow, Bengio, & Courville, 2016).

L2 Regularization

L2 regularization, also known as weight decay, is a widely used technique to prevent overfitting by constraining the complexity of the network. It achieves this by penalizing large weight values, thereby encouraging the model to maintain smaller and more evenly distributed weights, which contributes to a more regular and generalizable model structure (Goodfellow, Bengio & Courville, 2016; Lewkowycz, & Gur-Ari, 2020)

In this study, L2 regularization was applied to the dense layers alongside dropout. The regularization coefficients were optimized using Bayesian optimization to determine the most effective weight constraints. The search space for the L2 regularization parameter was defined within the range of 0.0001 to 0.01.

$$\text{L2:} \quad Loss = error(y_{pred}, y_{true}) + \lambda \sum_i^n w_i^2$$

Figure 10: L2 Regularization formula

3.3.2 Hyperparameter tuning pretrained model

The primary objective of hyperparameter tuning in this study is to control overfitting, enhance validation accuracy and minimize validation loss. Bayesian optimization was selected as the preferred tuning strategy due to its ability to model the performance of previously evaluated hyperparameter configurations. By maintaining a record of past observations and learning from them, this method avoids exploring regions of the hyperparameter space that are likely to yield suboptimal results. As such, Bayesian optimization offers a more efficient and intelligent search process compared to exhaustive approaches like Grid Search which experiment all the combinations provided in order to obtain the best hyperparameters. This approach is not recommended when computational resources are limited (Agrawal, 2021).

When working with pretrained models, hyperparameter tuning primarily focuses on optimizing the hyperparameters of the custom top layers that are manually added to serve as the model's output layers. In contrast, the model's internal parameters such as weights and biases are automatically adjusted through the process of backpropagation. This optimization technique calculates the overall loss and propagates the error backward from the output layer to the input layer, determining the extent to which each parameter contributes to the loss and updating them accordingly.

In this phase, three dense layers and corresponding dropout layers were added to the output section of the model. The inclusion of these additional layers enhances the model's robustness by mitigating overfitting.

The hyperparameters selected for optimization include the number of nodes in the dense layers, which were set to 506, 252 and 63, respectively. The chosen optimizer is Adam, with a learning rate of 0.0001. The activation function applied to the dense layers is ReLU (Rectified Linear Unit), which introduces non-linearity while maintaining computational efficiency. The dropout rates for the three corresponding layers are set at 0.4, 0.2 and 0.2, respectively, to mitigate overfitting. In addition, L2 regularization was applied to all three dense layers, each with a regularization rate of 0.001. Bayesian optimization identified the optimal hyperparameters by evaluating various combinations on the validation set and selecting the best-performing hyperparameters.

To monitor model performance and prevent unnecessary training, an early stopping callback was implemented. This mechanism halts the training process if the validation loss does not improve over three consecutive epochs, thereby accelerating the hyperparameter tuning process and conserving computational resources.

3.3.3 Fine-tuning with pretrained model

An additional technique employed in this study is fine-tuning using a pretrained model. This method involves unfreezing the upper layers of the pretrained network, which correspond to more specialized features. Given this, it is generally beneficial to fine-tune only the top two or three layers of the convolutional base. The unfreezing of these layers, combined with the training of the newly added fully connected classifier, facilitates feature extraction while simultaneously optimizing the new components of the model (Chollet, 2021).

As a result of implementing Bayesian hyperparameter tuning and fine-tuning the pretrained VGG-19 model with feature extraction, the model achieved a validation binary accuracy of 90%, representing a notable improvement over the baseline. Additionally, the validation loss showed improvement, narrowing the gap between training and validation losses. In summary, overfitting was slightly reduced in comparison to the initial model. See figure 11.

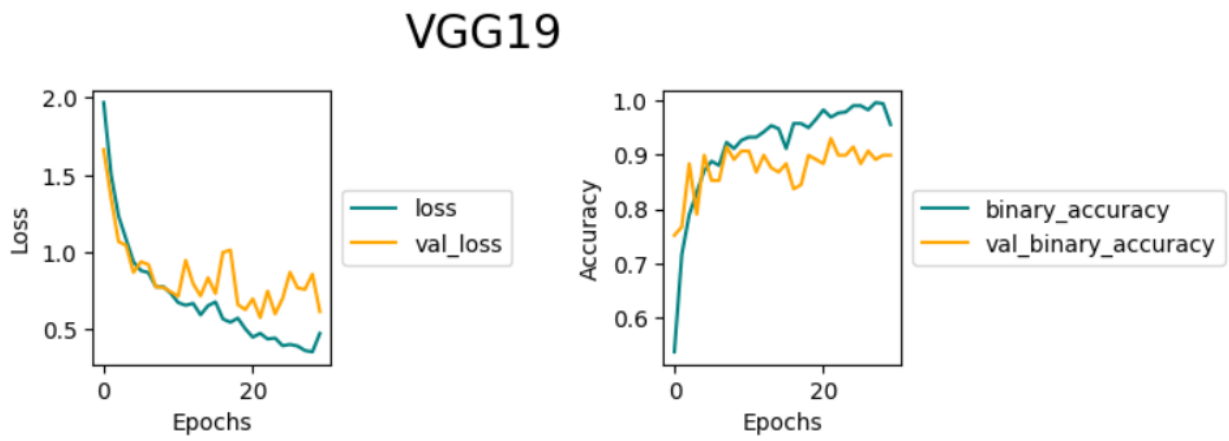


Figure 11: VGG-19 optimized

3.4 Subquestion 4 - How can model performance be evaluated and compared?

This phase of the research is dedicated to evaluate the performance of the optimized, pretrained VGG-19 model on a new dataset, the test set. To conduct this evaluation, several Python libraries are employed, including Scikit-learn, TensorFlow and Keras.

The initial evaluation involves the use of a confusion matrix, which summarizes the performance of the model by comparing predicted values with the true values of the test instances (Géron, 2019). Specifically, the model correctly identified 36 images as representing MS blood cells (label 1) and 29 images as HC blood cells (label 0). However, 6 images were incorrectly classified as HC blood cells when they were in fact MS blood cells, while 13 images were incorrectly predicted as MS blood cells despite being HC blood cells. See figure 12 and 13 below for a visual representation.

		Predicted	
		Negative	Positive
Actual	Negative	TN	FP
	Positive	FN	TP

TN = True Negative
FP = False Positive
FN = False Negative
TP =True Positive

Figure 12: Confusion Matrix

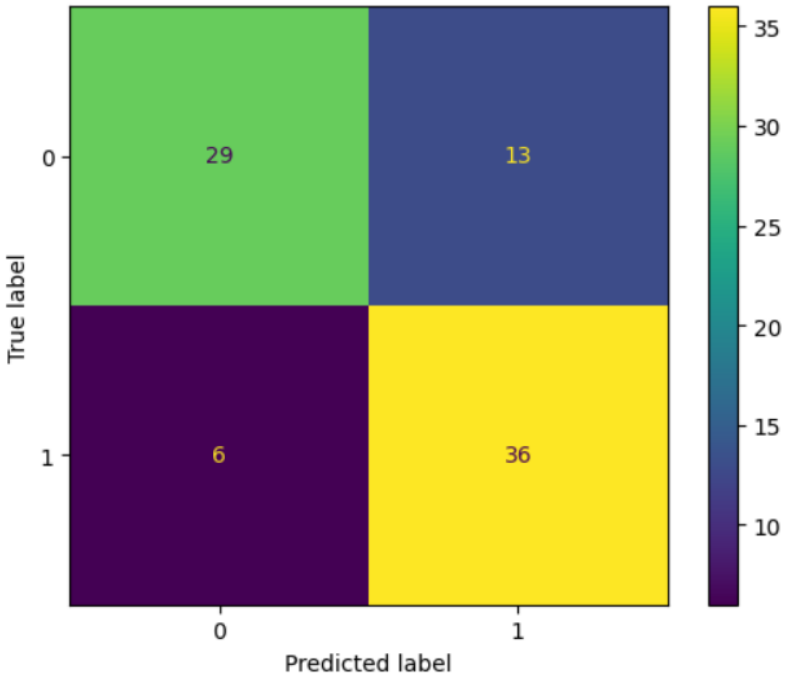


Figure 13: VGG-19 Confusion Matrix on Test Set

In addition to the values of true positives, true negatives, false positives and false negatives, several other informative metrics are commonly used to evaluate performance in balanced binary classification tasks. These metrics include Accuracy, Precision, Recall and F1-Score.

One such metric is accuracy, defined as the proportion of correctly predicted instances across the entire dataset (Zhao, Alzubaidi, Zhang, Duan & Gu, 2024). See figure 14. The pretrained VGG-19 model achieved an accuracy of 77% on the blood cell test set. *See figure 18.*

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

Figure 14: Accuracy formula

Furthermore, precision quantifies the proportion of positive predictions that are actually correct (Zhao et al., 2024). See figure 15. For the HC class, the precision was 83%, while for the MSp class it was 73%. See figure 18.

$$\text{Precision} = \frac{tp}{tp + fp}$$

Figure 15: Precision formula

In contrast, recall (also known as sensitivity) measures the proportion of actual positive cases that were correctly identified by the model. A higher recall value indicates a better performance of the method (Zhao et al., 2024). See figure 16. The recall for the HC class was 69%, whereas the MSp class achieved 86%. See figure 18.

$$\text{Recall} = \frac{tp}{tp + fn}$$

Figure 16: Recall formula

Finally, the F1 score provides a harmonic mean of precision and recall, offering a balanced measure of both metrics (Géron, 2019). See figure 17. The F1 score was 75% for the HC class and 79% for the MSp class. See figure 18.

$$F_1 = \left(\frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} \right) = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Figure 17: F1-score formula

To compute all the previously mentioned performance metrics, the classification report from the Scikit-learn library is utilized by supplying the predicted and true class values as input (Scikit-Learn, 2025a) . See figure 18. The support metric within this report indicates the number of instances belonging to each class, as well as the total number of instances in the dataset.

	precision	recall	f1-score	support
0	0.83	0.69	0.75	42
1	0.73	0.86	0.79	42
accuracy			0.77	84
macro avg	0.78	0.77	0.77	84
weighted avg	0.78	0.77	0.77	84

Figure 18: Metrics table

The Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) is a widely recommended metric for evaluating the performance of binary classification models. (Bradley, 1997). The ROC curve plots the true positive rate against the false positive rate across a range of classification thresholds, typically varying from 1 to 0. A given instance is classified as belonging to the positive class if its predicted probability exceeds the selected threshold; otherwise, it is classified as negative (Géron, 2019).

A higher AUC value indicates better model performance, as it reflects a greater ability to discriminate between the two classes. A diagonal line represents a model with no discriminative power (i.e., random guessing), while the blue curve represents the ROC of the model's predictions on the blood cell dataset. See figure 19.

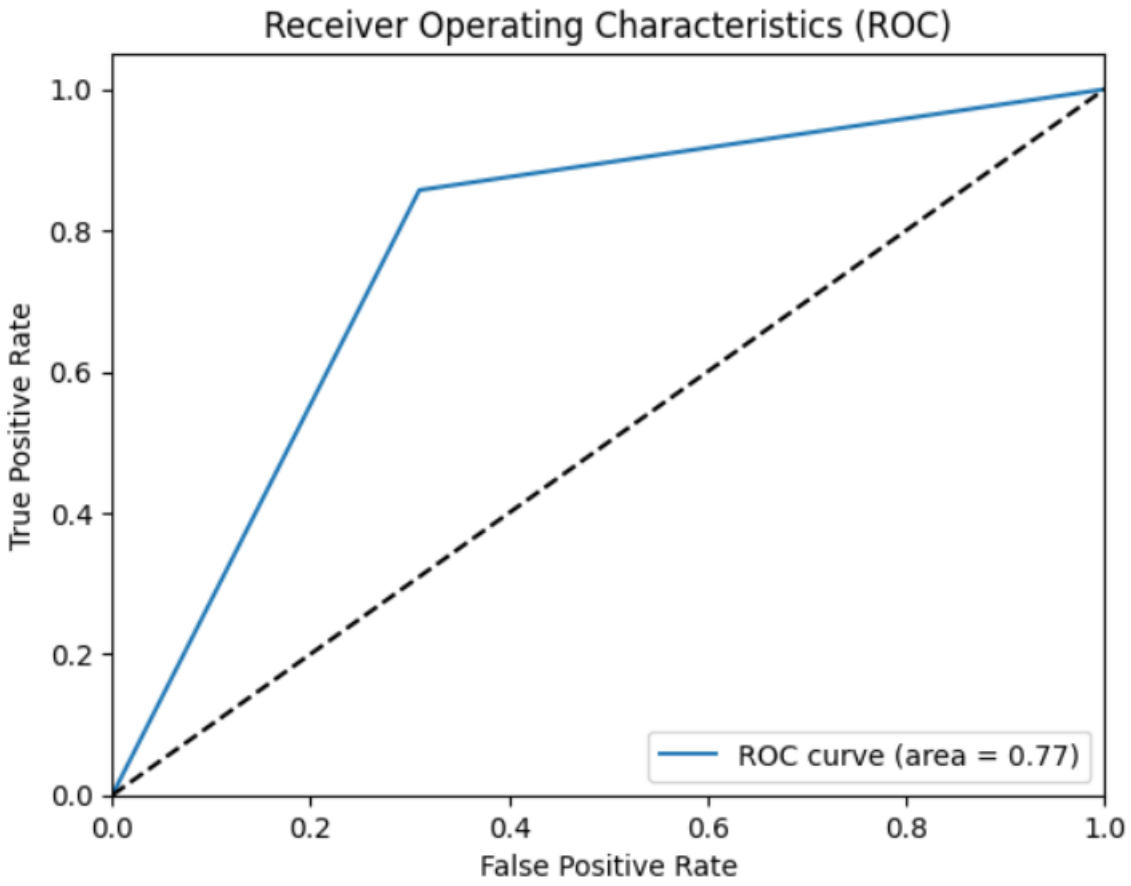


Figure 19: ROC and AUC

3.5 Subquestion 5 - How can explainable AI techniques identify visual features or regions in blood images?

Artificial Intelligence (AI) has been widely implemented across various sectors, including financial services, insurance, pharmaceuticals and, notably, healthcare. Within each of these domains, regulatory requirements play a significant role. In several industries, regulations require that AI models be explainable (Mishra, 2023). The need for explainability in AI systems is driven by several key factors:

- **Trust**: to gain user confidence in the model's predictions
- **Reliability:** to ensure that users can depend on the model's outputs
- **Regulatory:** to meet established regulatory and requirements
- **Adoption:** to support AI adoption among the users
- **Fairness:** to mitigate discriminatory outcomes and ensure fair predictions
- **Accountability**: to attribute responsibility for the model's decisions and outcomes.

One important way to meet these requirements is through the use of explanation methods that reveal how AI models make decisions. To support transparency and trust, it is crucial to apply both global and local explainable AI techniques. Together, these approaches offer a more complete understanding of model behaviour, both at the level of individual predictions and the overall decision logic (Ventura, Greco, Apiletti, & Malerba, 2022; Schrouff et al., 2021). The following sections explain the difference between global and local explanations and their relevance to understand visual features in medical images.

3.5.1 Model architecture

In healthcare applications, it is essential to maintain transparency about how machine learning models are built and how they make decisions. To support this, Appendix A presents a detailed overview of the VGG-19 architecture, showing each layer and its order. This information is provided before discussing the model's performance on the blood cell dataset, to give the reader a clear understanding of the model's structure.

3.5.2 Global explanation

Global explanation methods aim to clarify the overall behaviour and decision-making logic of a model across its entire input space. They provide insights into how the model processes data and makes predictions in general (Ventura et al., 2022).

In this experiment, the Shapley Additive Explanations (SHAP) Python library was used to interpret the predictions of a VGG-19 deep learning model. The model was trained to classify blood cell images as either indicative of Multiple Sclerosis (MS) or belonging to Healthy Controls (HC).

The experiment begins by loading a pretrained and optimized version of the VGG-19 model. Subsequently, a background data matrix well-known as the mask, is created, which matches the shape of the blood cell images and serves as a reference for SHAP value computation. The SHAP Explainer is then instantiated using the loaded model, the background mask and the algorithm parameter set to "auto," allowing SHAP to automatically select the most appropriate explanation algorithm based on the model and masker provided. Additionally, a custom function is defined to generate SHAP explanations for input images. This function utilizes the instantiated SHAP Explainer and accepts two arguments: the image to be explained and the maximum number of evaluations, which is set to 10,000 (SHAP, 2018).

Consequently, the SHAP Explainer was applied to analyse true positive predictions for both classes: Healthy Controls (HC), labeled as 0 and Multiple Sclerosis (MS), labeled as 1. Figure 20 presents a preprocessed blood cell image from the HC class alongside its corresponding SHAP feature importance visualization. Similarly, figure 21 displays the same information for a sample from the MSp class. In the SHAP visualizations, the color legend ranges from red to blue: red highlights regions that increase the model's predicted probability for the given class, while blue indicates regions that decrease it. These visualizations reveal how the model makes its classification prediction by assigning feature importance score to individual pixels in the image. Notably, the MS image shows more red-highlighted regions concentrated at the top and bottom, whereas the HC image exhibits a greater concentration of red areas in bottom as well and in the center. This suggests that the model relies on the bottom and in different spatial features when predicting each class. These visualizations offer valuable insights into the model's decision-making process and can support medical professionals in interpreting its predictions.

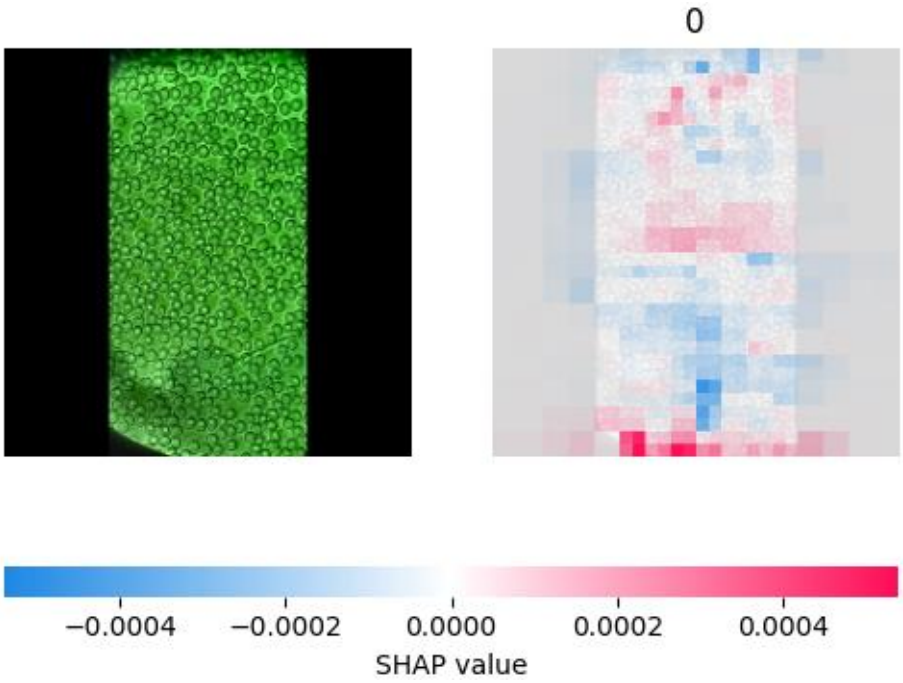


Figure 20: SHAP: HC (0)

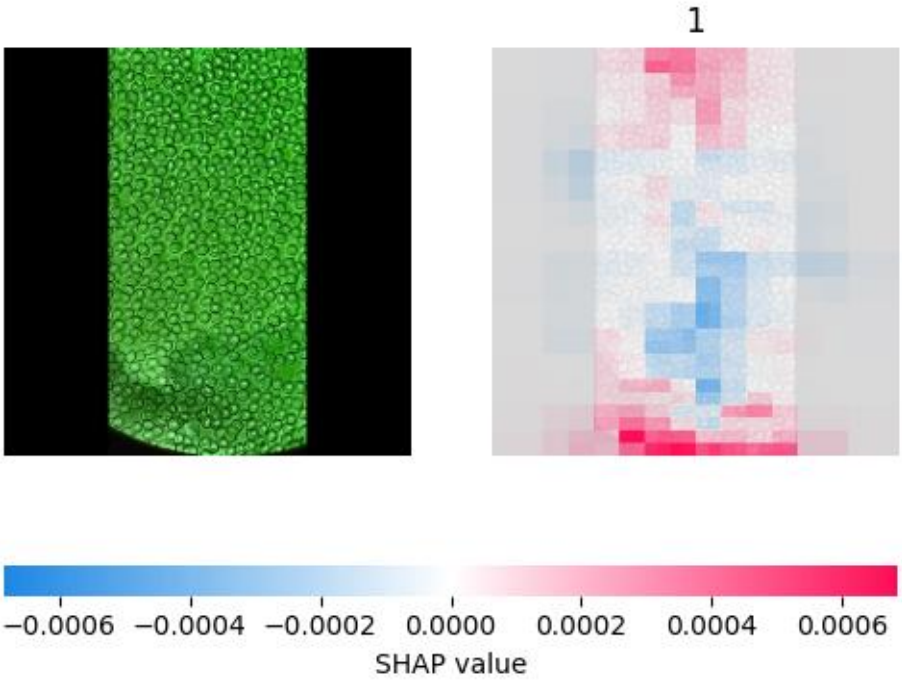


Figure 21: SHAP: MSp (1)

3.5.3 Local explanation

Local explanation methods focus on interpreting a model's decision for a specific input instance. They help to understand why the model made a particular prediction in a given case, which is particularly valuable in personalized applications such as medicine (Ventura et al., 2022).

For this task, the Local Interpretable Model-Agnostic Explanations (LIME) Python library was used. LIME is a method designed to generate local explanations for individual predictions made by a model. Two key characteristics of LIME are its use of an interpretable representation and its focus on local fidelity (Vimbi, Shaffi & Mahmud, 2024).

In this experiment, the same set of blood cell images used in the previous SHAP analysis was applied. The pretrained and optimized VGG-19 model was also loaded for use in the LIME framework. This ensures consistency across interpretability methods and allows for a comparative understanding of model behaviour.

The LIME Image Explainer was instantiated to explain the model's predictions on the image data. This method explains predictions by learning locally weighted linear models on data points sampled from the neighbourhood of the original image. These models provide interpretable approximations of the deep learning model's decision boundaries for each class (LIME, 2016). As a result, LIME generates segmented representations of the image and highlights the regions that are most influential for the classification decision. The explainer function takes as input: the image to be explained and the VGG-19 model.

The results show that the Region of Interest (ROI) identified by LIME is noticeably larger in images classified as Multiple Sclerosis (MS) compared to those classified as Healthy Controls (HC). See figures 22 and 23. This suggests that the model considers a broader set of features when identifying MS-related patterns, which may reflect the complexity or variability of visual markers in MS blood cell images.

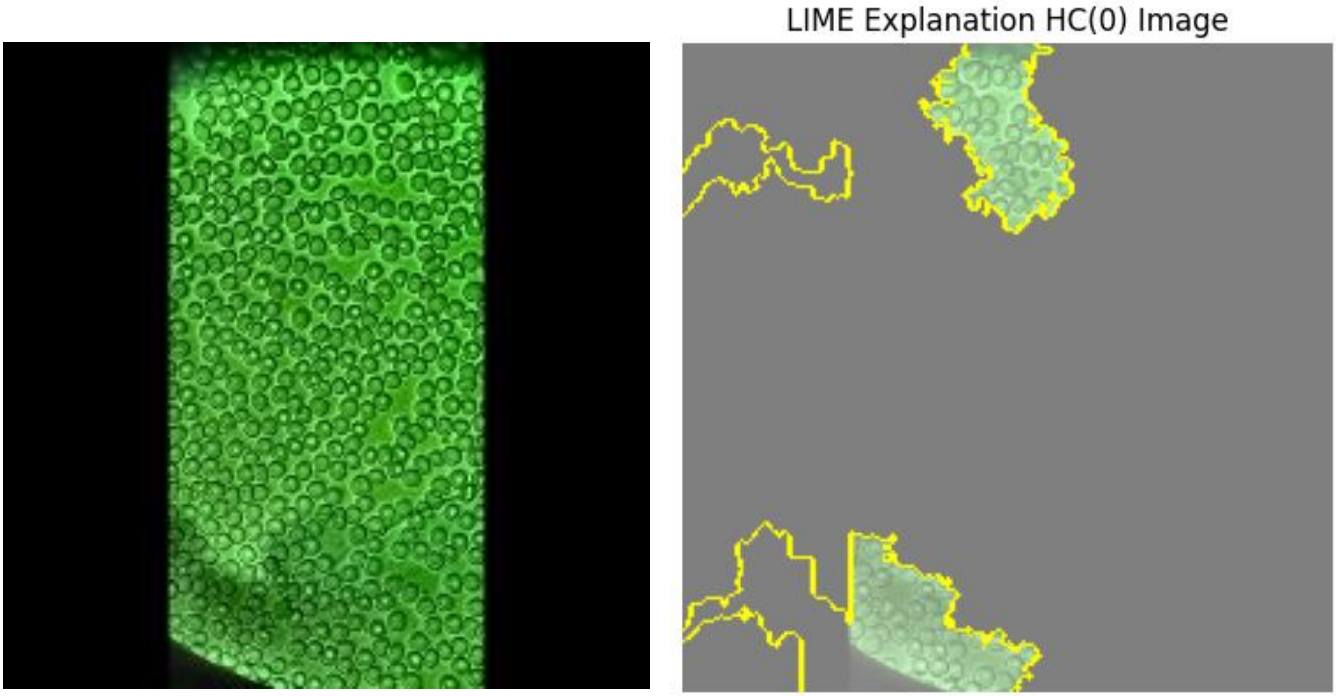


Figure 22: LIME: HC (0)

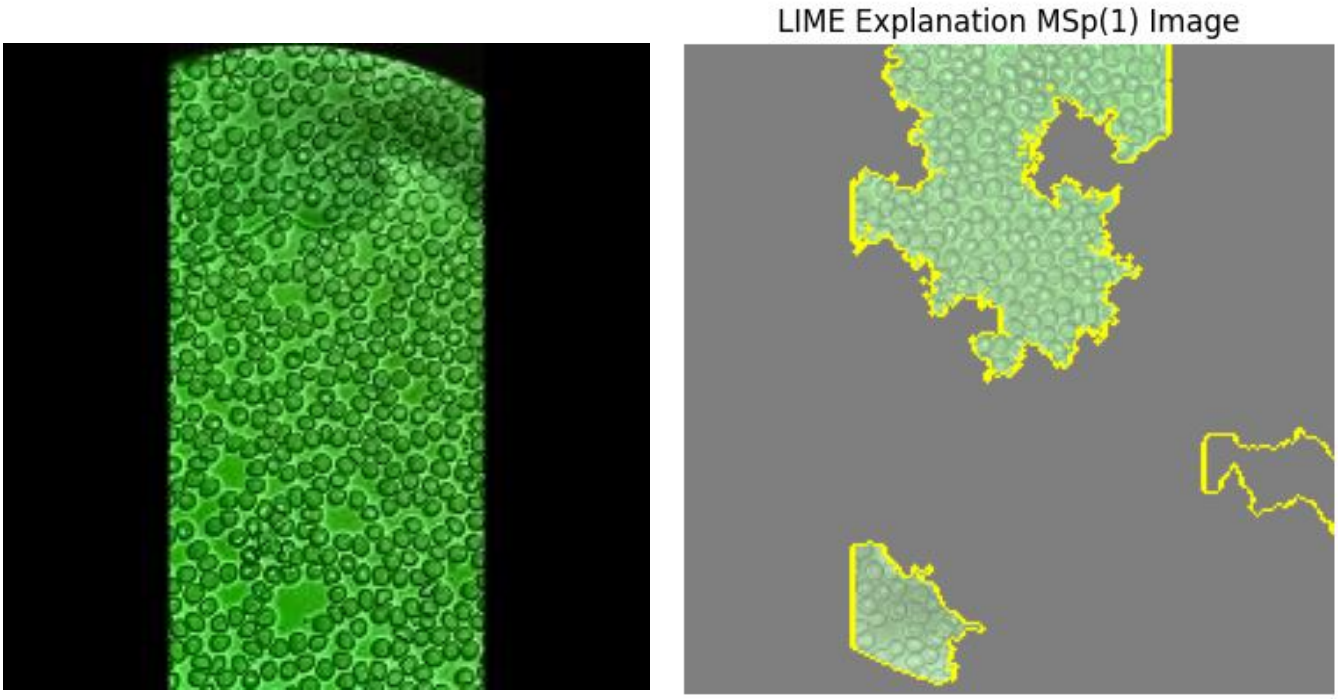


Figure 23: LIME: MSp (1)

4. Conclusion

This research investigates the use of machine learning for classifying blood cell images to support the diagnosis of Multiple Sclerosis (MS). The main research question is explored through a series of subquestions, each addressing an essential component of the machine learning pipeline, from preprocessing and model selection to evaluation and explanation. Below is a summary of the conclusions for each sub-question, followed by an overall conclusion that answers the main research question.

4.1 Subquestion 1

In conclusion, a comprehensive preprocessing strategy was essential to prepare the blood cell images for machine learning analysis. Through data exploration, digital image enhancement and compression, careful data splitting, augmentation, normalization and label encoding, the dataset was refined to improve both the quality and balance of the input data. These preprocessing steps collectively ensured that the models would be trained on standardized, high-quality data, thereby enhancing their potential accuracy and generalizability in distinguishing healthy blood cells from those affected by Multiple Sclerosis.

4.2 Subquestion 2

In summary, this study evaluated the suitability of different deep learning models for image classification tasks, with a focus on a blood cell dataset. The use of pretrained models specifically VGG-19, InceptionV3 and Xception. Proved to be an effective strategy for addressing the limitations of a small dataset. Among the models tested, VGG-19 achieved the highest validation accuracy at 87%, although all models exhibited varying degrees of overfitting. InceptionV3 and Xception each attained a validation accuracy of 84% and 85% respectively, but similarly demonstrated signs of overfitting, as reflected in their validation accuracy and validation loss trends. These findings highlight both the potential and challenges of applying transfer learning to small-scale medical image datasets, emphasizing the importance of techniques to mitigate overfitting in future research.

4.3 Subquestion 3

In this chapter, several techniques to optimize the performance of the pretrained VGG-19 model were explored, with a focus on balancing model optimization and generalization. Regularization strategies, such as dropout and L2 regularization, were employed to mitigate overfitting and enhance model generalization. Hyperparameter tuning using Bayesian optimization further refined the model's performance, ensuring efficient exploration of the hyperparameter space while minimizing computational overhead. Additionally, the fine-tuning of the pretrained model’s top layers facilitated the extraction of more specialized features, leading to improvements in validation accuracy and a reduction in validation loss. The

combination of these methods resulted in a 90% in validation accuracy and a slight reduction in overfitting. Overall, the strategies employed in this study contributed to a more robust and effective model, capable of generalizing the validation data.

4.4 Subquestion 4

In conclusion, the performance of the optimized, pretrained VGG-19 model was systematically evaluated using a range of well-established metrics for binary classification. The confusion matrix provided a foundational overview of the model's classification performance. In addition, evaluation metrics such as overall accuracy (77%), precision (83% for HC and 73% for MSp), recall (69% for HC and 86% for MSp) and F1-score (75% for HC and 79% for MSp) offered more detailed insights into the model's predictive strengths and limitations across both classes. The classification report generated via Scikit-learn facilitated the calculation of these metrics, including class-wise support. Moreover, the ROC curve and corresponding AUC value served as robust indicators of the model's overall discriminative capacity. Together, these evaluation tools enabled a comprehensive and comparative assessment of model performance on the blood cell test dataset.

4.5 Subquestion 5

This subquestion explored how explainable AI techniques can be used to identify visual features or regions in blood cell images. The application of both global (SHAP) and local (LIME) explanation methods to a VGG-19 deep learning model demonstrated their complementary value in understanding model predictions. SHAP provided insights into the general decision-making patterns of the model, revealing class-specific regions that contribute to the classification of blood cells as either indicative of Multiple Sclerosis (MS) or Healthy Controls (HC). In contrast, LIME focused on instance-level explanations, showing that the Region of Interest (ROI) in MS images tends to be broader, indicating a more complex feature space. Together, these techniques enhance the interpretability of AI-driven medical diagnostics by making the decision process more transparent and grounded in visual evidence.

4.6 Answering the main research question

What features in blood images are most indicative of multiple sclerosis machine learning classification when comparing image recognition models?

The primary aim of this thesis was to determine which visual features in blood images are most indicative of Multiple Sclerosis (MS) in machine learning classification when comparing image recognition models.

Based on the application of explainable AI techniques, the study identified that the features contributing to MS classification are distributed across distinct spatial regions within the images. SHAP, as a global explanation method, revealed consistent areas of high importance that differentiate MS from Healthy Controls (HC), while LIME, as a local method, showed that the Region of Interest (ROI) is generally broader in MS cases. This suggests that MS-related blood cell images involve more complex and spatially diverse patterns, which the model learns to associate with the disease.

Together, these insights demonstrate that deep learning models such as VGG-19 can effectively identify disease-indicative features when supported by interpretability methods. This integrated approach not only improves transparency in model decision-making but also contributes to the reliability of AI-based diagnostic tools.

5. Discussion

This chapter provides a comprehensive evaluation of the research findings, addressing the strengths and limitations of the study. Finally, recommendations for further research are proposed to build upon this study and enhance the understanding of the topic.

5.1 Strength of Research

This research effectively addresses a supervised machine learning binary classification task aimed at determining whether blood cell images indicate Multiple Sclerosis (MS) or represent healthy cells. The initial phase involved comprehensive preprocessing, which included data exploration, image enhancement, image compression, data normalization, data augmentation and label encoding. These preprocessing steps ensured that the image data were adequately prepared for model training.

Three well-established pretrained models, Xception, VGG19 and InceptionV3, were selected, all of which are frequently used in healthcare machine learning applications. Among these models, VGG19 demonstrated superior performance, based on analysis of both the training and validation sets and was chosen for further study. To address overfitting, various techniques such as dropout and L2 regularization were implemented. Additional improvements were achieved through Bayesian hyperparameter tuning and model fine-tuning, which enhanced both model performance and generalization on the training and validation sets.

The model was subsequently evaluated on an unseen test set, achieving an accuracy of 77%, though some signs of overfitting were still present. Finally, explainable AI techniques were applied to enhance the transparency of the model's decision-making process. This included providing a detailed model architecture and offering both global and local explanations through SHAP and LIME to clarify how the VGG19 model predicted MS or Healthy Control (HC) classes.

5.2 Limitations

This research was conducted over a relatively short time frame of 20 weeks, which posed a significant constraint given the complexity of predicting Multiple Sclerosis (MS) from blood cell images using deep learning techniques.

Furthermore, the scientific literature on this specific application is still in its early stages, resulting in a limited number of directly relevant sources. Consequently, this study had to rely on insights from related research domains, such as white blood cell image classification, to guide the methodology and interpretation.

Another notable limitation was the small size of the dataset, consisting of only 314 images, insufficient for training deep learning models from scratch. As a result, the research relied entirely on transfer learning with pretrained models.

Lastly, deep learning experiments, particularly those involving hyperparameter tuning, are computationally intensive and typically require high-performance GPUs. The use of a standard laptop for training significantly extended processing times and limited the scope of experimentation.

5.3 Recommendations for Further Research

A primary recommendation is to conduct further research to identify the specific indicators of Multiple Sclerosis (MS) in blood cell images. The dataset provided by RR Mechatronics included deformed cells annotated as indicative of MS. However, deformed cells were also observed in both the MS and Healthy Control (HC) classes, raising concerns about annotation accuracy and the clarity of diagnostic features. Therefore, establishing medically validated indicators of MS in blood cells is essential for improving model reliability and interpretability.

Additionally, as previously noted, deep learning models require large amounts of data to achieve optimal performance. Expanding the dataset for both MS and HC classes is critical to enhance the generalizability and reliability of the classification models.

Future studies should also consider integrating more advanced computer vision techniques, such as image segmentation, to better isolate and analyse relevant features within blood cell images. Lastly, leveraging additional computational resources, such as high-performance GPUs, would enable the use of more sophisticated hyperparameter optimization techniques such as Grid Search, which performs an exhaustive search across the hyperparameter space and could lead to further improvements in model performance.

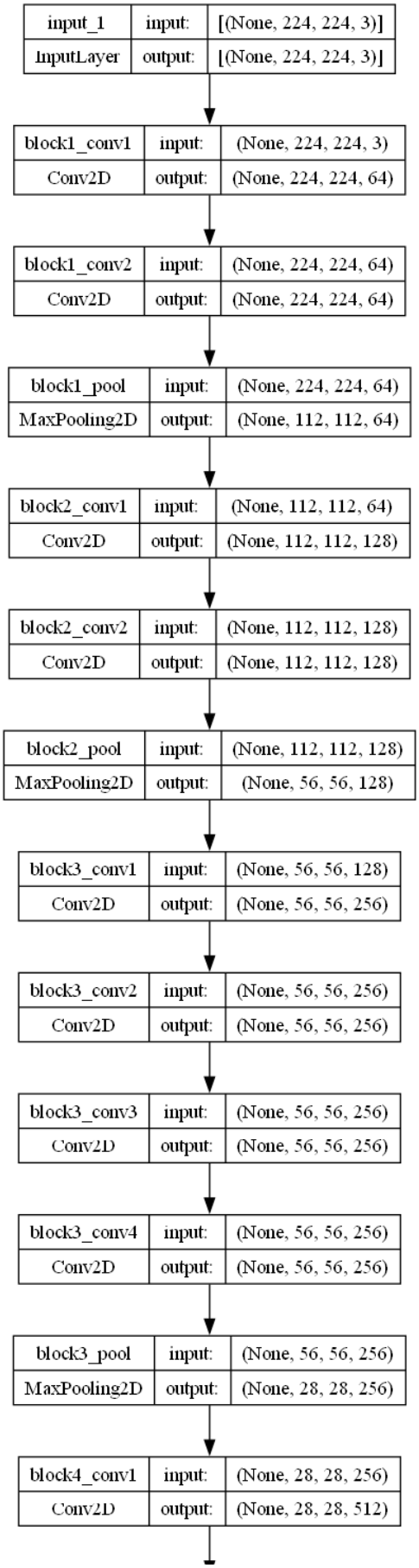
References

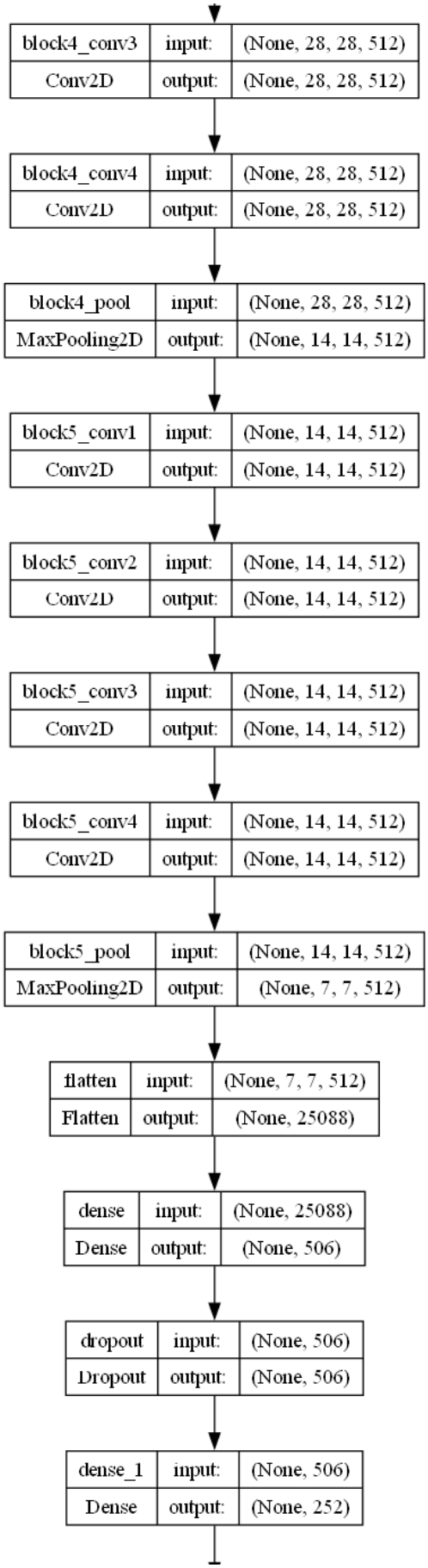
- Agrawal, T. (2021). *Hyperparameter Optimization in Machine Learning*. Apress.
- Amato, A., & Di Lecce, V. (2023). Data preprocessing impact on machine learning algorithm performance. *Open Computer Science, 13*(1). https://doi.org/10.1515/comp-2022-0278
- Bansal, M., Kumar, M., Sachdeva, M., & Mittal, A. (2021). Transfer learning for image classification using VGG19: Caltech-101 image data set. *Journal of Ambient Intelligence and Humanized Computing, 14*(4), 3609–3620. *https://doi.org/10.1007/s12652-021-03488-z*
- Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, *30(*7), 1145-1159. https://doi.org/10.1016/S0031-3203(96)00142-2
- Chen, H., Engkvist, O., Wang, Y., Olivecrona, M., & Blaschke, T. (2018). The rise of deep learning in drug discovery. Drug Discovery Today, *23*(6), 1241-1250. https://doi.org/10.1016/j.drudis.2018.01.039
- Chennamsetty, S.S., Safwan, M., Alex, V. (2018). Classification of Breast Cancer Histology Image using Ensemble of Pre-trained Neural Networks. In A. Campilho, F. Karray, B. ter Haar Romeny (Eds.), *Image Analysis and Recognition* (pp. 804-811). Springer
- Chollet, F. (2018). *Deep Learning with Python* (1st ed.). Manning Publications.
- Chollet, F. (2021). *Deep Learning with Python* (2nd ed.). Manning Publications.
- Dobson, R., & Giovannoni, G. (2018). Multiple sclerosis – a review. *European Journal of Neurology, 26*(1), 27–40 https://onlinelibrary.wiley.com/doi/pdf/10.1111/ene.13819
- Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly.
- Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning*. The Mit Press.
- ImageNet (2025). *ImageNet*. Retrieved on March 30, 2025 from https://www.image-net.org/
- Inholland (2025). *Data Driven Smart Society*. Retrieved on January 16, 2025 from https://www.inholland.nl/onderzoek/lectoraten/data-driven-smart-society/
- Islam, O., Assaduzzaman, M., & Zahid Hasan, M. (2024). An explainable AI-based blood cell classification using optimized convolutional neural network. *Journal of Pathology Informatics, 2(15).* https://doi.org/10.1016/j.jpi.2024.100389
- Jena, B., Nayak, G. K., & Saxena, S. (2021). Convolutional neural network and its pretrained models for image classification and object detection: A survey. *Concurrency and Computation: Practice and Experience, 34*(6). https://doi.org/10.1002/cpe.6767
- Keras (2025a). *Accuracy metrics*. Retrieved on April 7, 2025 from https://keras.io/api/metrics/accuracy_metrics/
- Keras, (2025b). *Keras Applications.* Retrieved on May 12, 2025 from https://keras.io/api/applications/
- Katar, Oguzhan., & Kilincer, I. F. (2022). Automatic Classification of White Blood Cells Using Pre-Trained Deep Models. *Sakaraya University Journal of Computer and Information Sciences, 5*(3). [Online publication]. Retrieved at April 10, from http://saucis.sakarya.edu.tr/en/download/article-file/2740488
- Lewkowycz, A., & Gur-Ari, G. (2020). *On the training dynamics of deep networks with L2 regularization.* Retrieved on April 20, 2025 from

https://proceedings.neurips.cc/paper/2020/file/32fcc8cfe1fa4c77b5c58dafd36d1a98-Paper.pdf
- LIME. (2016). *Lime Package*. Retrieved on May 4, 2025 from https://lime-ml.readthedocs.io/en/latest/lime.html#module-lime.lime_image
- Liu, Y. (2020). *Python Machine Learning By Example* (3rd ed.). Packt Publishing.
- Liu, Z., Xu, Z., Jin, J., Shen, Z., & Darrell, T. (2023). *Dropout reduces underfitting*. Retrieved on April 20, 2025 from https://arxiv.org/pdf/2303.01500
- Mishra, P. (2023). *Explainable AI Recipes*. Apress.
- Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2017). *Pruning convolutional neural networks for resource efficient inference.* Retrieved on April 2, from https://arxiv.org/pdf/1611.06440
- Naji, Y., Mahdaoui, M., Klevor, R., & Kissani, N. (2023). Artificial Intelligence and Multiple Sclerosis: Up-to-Date Review. *The Cureus Journal of Medical Science, 15(9).* https://pmc.ncbi.nlm.nih.gov/articles/PMC10581506/pdf/cureus-0015-00000045412.pdf
- OpenCV. (2025). *Geometric Image transformations*. Retrieved February 22, 2025 from https://docs.opencv.org/3.4/da/d54/group__imgproc__transform.html
- Phukan, A., & Gupta, D. (2022). EEG Based Emotion Classification Using Xception Architecture. In Nikhil Marriwala, C.C Tripathi, S. Jain, D. Kumar (Eds.), *Mobile Radio Communications and 5G Networks* (pp. 95-108). Singapore: Springer Nature
- RR Mechatronics (2024a). *Masters of measurement*. Retrieved January 6, 2025, from https://rrmechatronics.com
- RR Mechatronics (2024b). *R&D Projects*. Retrieved January 6 2025, from https://rrmechatronics.com/products/rd-projects/
- RR Mechatronics (2024c). *Lorrca Overview.* Retrieved January 6 2025, from https://rrmechatronics.com/products/lorrca-overview/
- Salehin, I., & Kang, D.K. (2023). A review on dropout regularization approaches for deep neural networks within the scholarly domain. *Electronics, 12*(14). https://doi.org/10.3390/electronics12143106
- Scikit-Learn (2025a). *Classification_report.* Retrieved April 21, 2025 from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html
- Scikit-Learn. (2025b). *Cross-validation: evaluating estimator performance.* Retrieved April 21, 2025 from https://scikit-learn.org/stable/modules/cross_validation.html
- Schrouff, J., Baur, S., Hou, S., Mincu, D., Loreaux, E., Blanes, R., Wexler, J., Karthikesalingam, A., & Kim, B. (2021). *Best of both worlds: local and global explanations with human-understandable concepts*. Retrieved on April 28 2025, from https://arxiv.org/pdf/2106.08641
- Shahnaz, M., & Mollah, A.F. (2023). On the Performance of Convolutional Neural Networks with Resizing and Padding. In S. Basu, D. K. Kole, A. K. Maji, D. Plewczynski, & D. Bhattacharjee (Eds.), *Proceedings of International Conference on Frontiers in Computing and Systems.* (pp 51-62). Singapore: Springer.
- SHAP. (2018). *shap.Explainer*. Retrieved at May 2, 2025 from https://shap.readthedocs.io/en/latest/generated/shap.Explainer.html
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research, 15*(1), 1929–1958. https://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

- Thabtah, F., Hammoud, S., Kamalov, F., Gonsalves, A., (2020). Data imbalance in classification: Experimental evaluation. *Information Sciences, 513*, 429-441. https://doi.org/10.1016/j.ins.2019.11.004
- Theckedath, D., Sedamkar, R. (2020). Detecting Affect States Using VGG16, ResNet50 and SE-ResNet50 Networks. *SN Computer Science, 79*(1). https://doi.org/10.1007/s42979-020-0114-9
- Tian, Y., & Zhang, Y. (2022). A comprehensive survey on regularization strategies in machine learning. *Information Fusion, 80,* 146-166. https://doi.org/10.1016/j.inffus.2021.11.005
- Umbaugh, S. E. (2023). *Digital image processing: Digital image enhancement, restoration and compression* (4th ed.). CRC Press.
- Ventura, F., Greco, S., Apiletti, D., & Malerba, D. (2022). Trusting deep learning natural-language models via local and global explanations. *Knowledge and Information Systems, 64*(7), 1863–1907.https://doi.org/10.1007/s10115-022-01690-9
- Vimbi, V., Shaffi, N., & Mahmud, M. (2024). Interpreting artificial intelligence models: a systematic review on the applications of LIME and SHAP in Alzheimer's disease detection. *Brain Informatics, 11*(10). https://doi.org/10.1186/s40708-024-00222-1
- Zhao, Z., Alzubaidi, L., Zhang, J., Duan, Y., & Gu, Y. (2024). A comparison review of transfer learning and self-supervised learning: Definitions, applications, advantages and limitations. *Expert Systems with Applications*, *242*, 122807. https://doi.org/10.1016/j.eswa.2023.122807

Appendix A:
VGG-19 architecture




block4_conv3 input: (None, 28, 28, 512)
Conv2D output: (None, 28, 28, 512)
block4_conv4 input: (None, 28, 28, 512)
Conv2D output: (None, 28, 28, 512)
block4_pool input: (None, 28, 28, 512)
MaxPooling2D output: (None, 14, 14, 512)
block5_conv1 input: (None, 14, 14, 512)
Conv2D output: (None, 14, 14, 512)
block5_conv2 input: (None, 14, 14, 512)
Conv2D output: (None, 14, 14, 512)
block5_conv3 input: (None, 14, 14, 512)
Conv2D output: (None, 14, 14, 512)
block5_conv4 input: (None, 14, 14, 512)
Conv2D output: (None, 14, 14, 512)
block5_pool input: (None, 14, 14, 512)
MaxPooling2D output: (None, 7, 7, 512)
flatten input: (None, 7, 7, 512)
Flatten output: (None, 25088)
dense input: (None, 25088)
Dense output: (None, 506)
dropout input: (None, 506)
Dropout output: (None, 506)
dense_1 input: (None, 506)
Dense output: (None, 252)