ANNOTATION

Developing the field of medical imaging is an important task with the growing need for automated, fast and efficient diagnostics. Traditionally, blood cell counts are determined using a hemocytometer with the use of additional laboratory equipment and chemical compounds, but this method is time consuming and labor intensive. This paper investigates the application of machine learning methods to the task of blood cell identification and classification to increase the speed of recognition without degradation of quality.

ABSTRACT

The development of the field of medical imaging is an important task with the growing need for automated, fast and efficient diagnostics. Traditionally, the number of blood cells is determined using a hemocytometer using additional laboratory equipment and chemical compounds, but this method is time consuming and laborious. The paper studies the application of machine learning methods to the problem of identification and classification of blood cells to increase the recognition rate without compromising quality.

CONTENTS.

INTRODUCTION

Clinical blood analysis allows the detection of various diseases based on deviations from reference values of blood cellular components. In hematology, the ratio of different cell types is particularly important in the diagnosis of hematologic diseases: hemoblastoses (e.g., lymphomas and leukemias), anemias, and other pathologies. In addition to standard peripheral blood sampling, in case of significant deviations from the reference values of cellular composition indicators, red bone marrow cell analysis is also used, which allows for a more precise identification of the disease and its causes.

Differential cell counting in red bone marrow aspirate and peripheral blood smears is crucial for the classification of hematologic diseases. Manual cell counting in a Goryaev chamber is considered the standard, but it is labor-intensive, time-consuming, and leads to human error. Digital imaging of pathology based on machine learning algorithms is a promising new technology for this solution to the existing problems of manual cell analysis. In doing so, deep learning techniques can detect cells with high accuracy.

The purpose of the graduate qualification work is to implement machine learning methods to create a model for classifying blood and bone marrow cells, and to conduct experiments with blood and bone marrow cell images.

Job Objectives:

1. Analyze и select approaches к implementation classification methods.
2. To implement algorithms and methods of selected machine learning techniques for classification of blood and bone marrow cells.
3. Create a database with images of blood and bone marrow cells.
4. Conduct experiments to classify blood cells and hematopoietic cells of red bone marrow by analyzing their histological images by selected machine learning methods, and conduct experiments to identify the features that are used by the classification model.
5. To select methods with the best quality of recognition of blood cells and hematopoietic cells of red bone marrow by comparative analysis of methods.
6. Implement a client-server application and a graphical user interface to interact with the system.

1 PRE-PROJECT STUDY

1.1 Analysis of the subject area

Detecting and recognizing objects in data is a task that appears in various fields of science and engineering. Usually, this task is solved by statistical methods. Empirically, it is found that this approach provides the best results in most of the object detection tasks. This is due to the development of machine learning and data analysis, and especially the development of deep learning, in the last decade. Machine learning actively exploits the described approach, and is an interdisciplinary field that combines mathematical statistics, optimization theory, computer engineering, numerical methods and data engineering [1].

Machine learning methods are versatile, but require knowledge of the subject area in which the methods are used. Knowledge in the subject area is necessary to be able to obtain and make error-free objective conclusions from the results obtained after using the methods. The choice of features adequate to the subject area, the method of data collection, processing and storage allows to achieve the best results for solving the task at hand [2].

Data for object recognition, as for any other machine learning tasks, can contain both labeled and unlabeled objects. For labeled data, methods of learning with a teacher are used, and for unlabeled data, methods of learning without a teacher are used. There are also intermediate paradigms - methods with partial teacher involvement and methods of reinforcement learning.

In object recognition, the methods of supervised and unsupervised learning can be reduced to classification and clustering methods. Classification is a way of using an off-the-shelf trained machine learning model with a teacher to categorize data into predefined categories. Clustering is the use of machine learning models to discover clusters in which objects are most similar to each other.

1.2 Deep neural networks

Multi-label classification involves predicting two or more class labels. Unlike conventional classification tasks in which class labels are mutually exclusive, multi-label classification requires specialized machine learning algorithms that support the prediction of multiple mutually non-exclusive classes or "labels".

Deep neural networks are an example of an algorithm that inherently supports multi-label classification tasks. Neural network models for multi-label classification tasks can be easily defined and evaluated using various libraries.

Deep learning is part of a broader family of machine learning methods based on artificial neural networks with representational learning. Learning can be supervised, partially supervised or unsupervised.

Deep learning architectures such as deep neural networks, graph-based neural networks, recurrent neural networks, and convolutional neural networks have been applied in areas such as computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medicine, materials testing, and board games, where they have produced results comparable to, and in some cases superior to, those of experts [3].

Artificial Neural Networks (ANNs) have been inspired by information processing and distributed communication nodes in biological systems. That said, of course, ANNs have differences from

biological brain. In particular, neural networks tend to be static and symbolic, while the biological brain of most living organisms is dynamic and plastic [4].

ANN is based on a set of connected units or nodes, called artificial neurons, that loosely model neurons in the biological brain. Each connection, like synapses in the biological brain, can transmit a signal to other neurons. The artificial neuron that receives the signal then processes it and can signal to the neurons connected to it. "Signal" in a connection is a real number, and the output of each neuron is calculated by some nonlinear function of the sum of its inputs. The connections are called edges. Neurons and edges usually have weights that are adjusted as they are trained. The weight increases or decreases the strength of the signal in the connection. Neurons may have a threshold such that a signal is sent only if the combined signal crosses that threshold. Neurons are usually organized into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (input layer) to the last layer (output layer), possibly after passing through the layers multiple times (Figure 1).

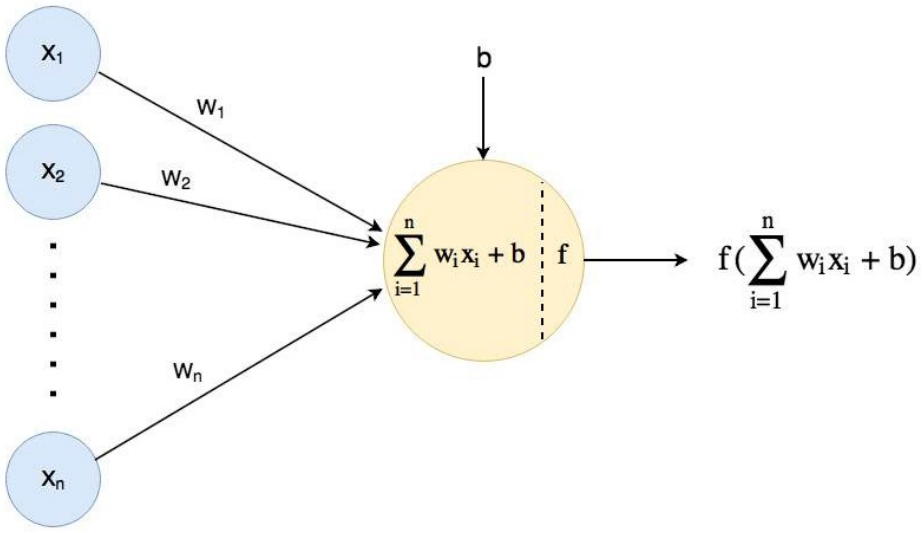


Figure 1: Illustration of the principle of operation of the activation function in one neuron

The adjective "deep" in deep learning refers to the use of multiple layers in a network. Early work has shown that a linear perceptron cannot be a universal classifier, but a network with a nonpolynomial activation function with a single hidden layer of unbounded width has the properties of a universal classifier.

Deep learning is a modern variation of machine learning methods that deals with an unlimited number of layers of bounded size, which allows for practical applications and optimal implementation, while maintaining theoretical generality in a variety of settings. In deep learning, layers are also allowed to be heterogeneous and deviate widely from biological connectionist models for the sake of efficiency, learnability and comprehensibility, from where the "structured" part originated [5].

In artificial neural networks, the activation function of a node determines the output of that node given an input or set of inputs. A standard integrated circuit can be thought of as a digital network of activation functions that can be "ON" (1) or "OFF" (0), depending on input. This is similar to a linear perseptron in neural networks. However, only nonlinear activation functions allow such networks to solve non-trivial problems using only a small number of nodes, and such activation functions are called nonlinearities.

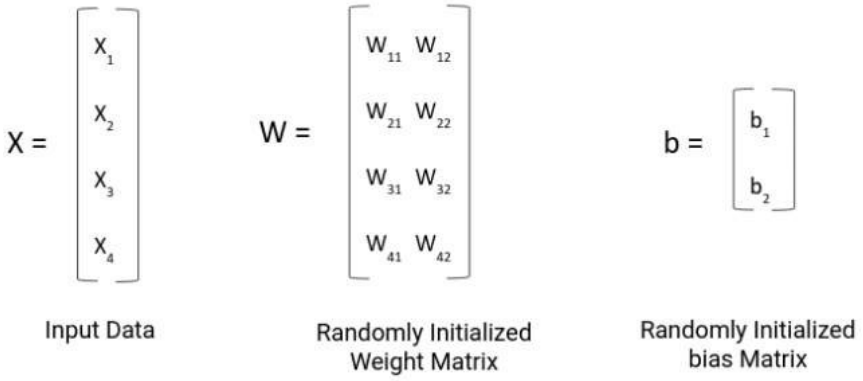


Figure 2: Conceptual representation of the structure used data

The most common activation functions can be divided into three categories: ridge functions, radial functions, and folding functions.

Figure 3 and 4 shows the activation function that is used the most in neural network practice, but this function sometimes produces suboptimal results.

$$\phi(\mathbf{v}) = \max(0, a + \mathbf{v}'\mathbf{b})$$

Figure 3: ReLu() function

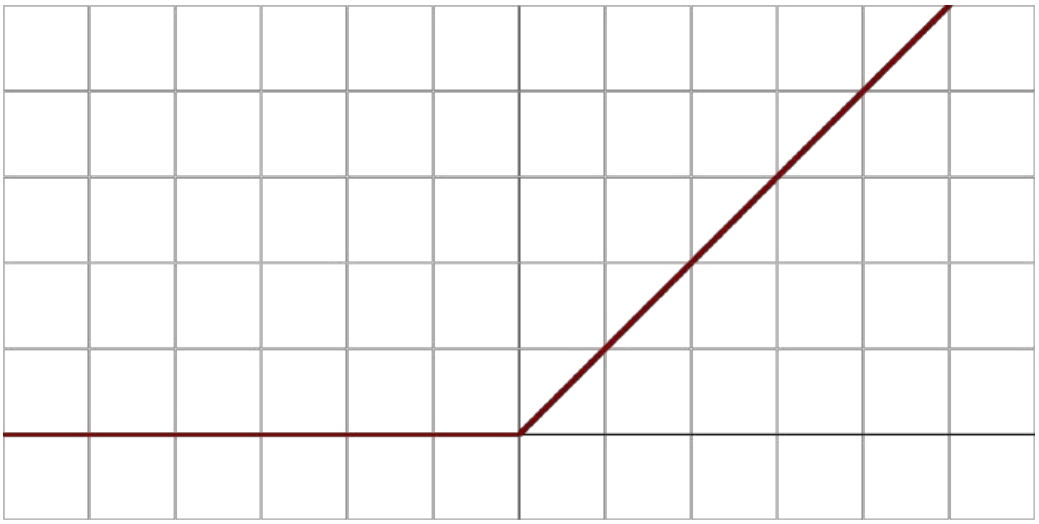


Figure 4: Graph of the ReLu() activation function

Training a model means finding good values for all weights and biases from labeled examples. In learning with a teacher, a machine learning algorithm builds a model by examining a set of examples X and trying to find a model that minimizes the loss in finding the true value of Y. In other words, a function h: x->y, called hypothesis minimization, is searched for. This process is called empirical risk minimization. Figure 5 shows the mathematical expression of empirical risk minimization. The function L() is the loss function (from the word loss). The function P() is the joint distribution of the set X and Y. Accordingly, the final expression is a search for the mathematical expectation of the difference between the predicted and true value [6].

$$R(h) = \mathbf{E}[L(h(x), y)] = \int L(h(x), y)\, dP(x, y).$$

Figure 5: Mathematical expression of the minimization of empirical Risks.

A loss is a penalty for a bad prediction. That is, the loss is a number that shows how bad the model's prediction was on a single example. If the model's prediction is perfect, the loss is zero; otherwise, the loss is larger. The goal of model training, as mentioned above, is to find a set of weights and biases that have low loss on average across all examples. For example, Figure 6 shows a high-loss model on the left and a low-loss model on the right.

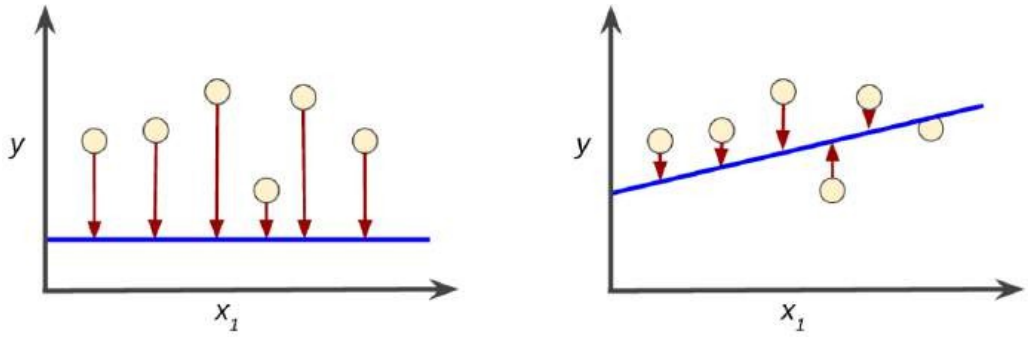


Figure 6. Effect of two loss functions with different levels losses

The best known loss function is the quadratic error function shown in

$$MSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}$$

Figure 7.

Figure 7: Quadratic loss function

The quadratic loss function is commonly used in linear regression problems, but is not universal, so there are a number of other different functions.

The key algorithm for training modern ANNs is the error back propagation algorithm. This algorithm achieves an optimal distribution of weights within the neural network by using optimization algorithms, such as the gradient descent algorithm or the stochastic gradient descent method. The results of the algorithm provide opportunities to find a separating hyperplane in a multidimensional feature space when classifying objects.

A neural network can be visualized as a combination of compositions of activation and matrix multiplication functions (Figure 8).

$$g(x) := f^L(W^L f^{L-1}(W^{L-1} \cdots f^1(W^1 x) \cdots))$$

Figure 8. Mathematical representation of ANN

Back propagation calculates the gradient in the weight space of the feed forward neural network with respect to the loss function. Figure 9 shows the mathematical expression of the error back propagation algorithm. The first expression is a scalar

product gradient functions losses of neuron *a* with derivative activation function from the result of addition of all values of the neuron last layer L. Further expressions represent the continuation of operations on subsequent ANN neurons [7].

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{BP1}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \tag{BP2}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \tag{BP3}$$

$$\frac{\partial C}{\partial w^l_{j\,k}} = a^{l-1}_k \delta^l_j \tag{BP4}$$

Figure 9. Mathematical expression of the error back propagation algorithm

1.3 Converged neural networks

In deep learning, convolutional neural network (CNN or ConvNet) is a class of deep neural network most commonly used for visual image analysis. They are also known as shift invariant or spatially invariant artificial neural networks (SIANN), based on an architecture of convolution kernels or common weight filters that slip over input objects and provide equivalent translation responses known as feature maps. Most convolutional neural networks are only equivariant, not invariant, with respect to translation. They have applications in image and video recognition, recommender systems, image classification, image segmentation, medical image analysis, natural language processing, brain-computer interfaces, and financial time series [8].

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, meaning that every neuron in one layer is connected to all neurons in the next layer. "Fully connectedness" of these networks makes them prone to overfetching data. Typical ways to regularize or prevent overfitting include: penalizing parameters during training (e.g., weight reduction) or restricting connectivity (missed connections, dropouts, etc.). CNNs take a different approach to regularization: they take advantage of the hierarchical data structure and collect patterns of increasing complexity using smaller and simpler patterns obtained in their filters (Figure 5).

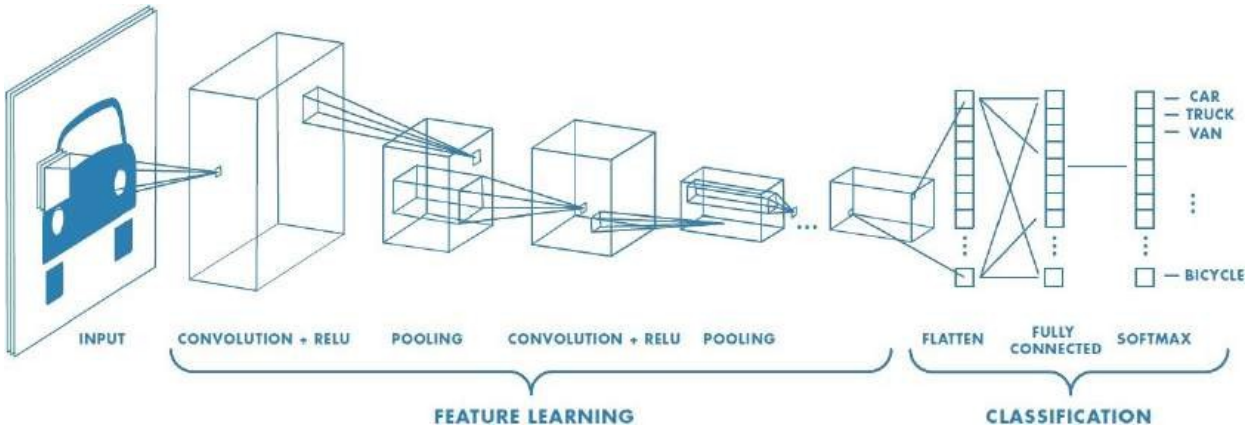


Figure 5. Conceptual model of convolutional neural network Basic operation в convolutional neural networks - operation convolution. This operation is described in Figure 6.

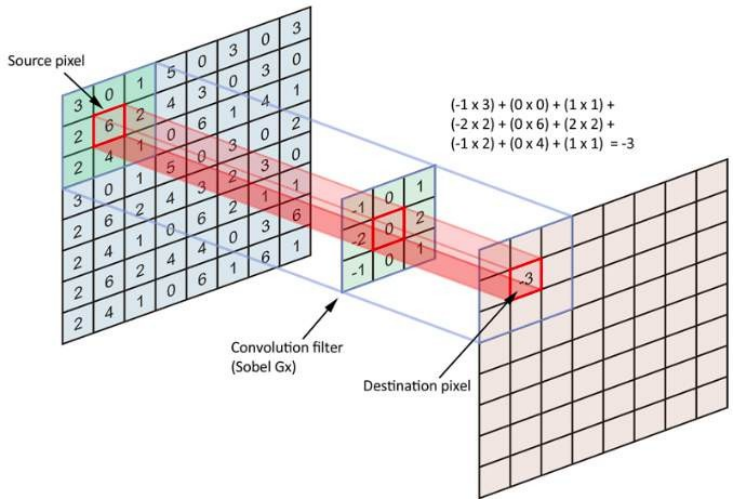


Figure 6: Logic of convolution operation on a matrix

As a result of using the convolution operation on each layer of the convolutional neural network, feature maps (Figure 7) appear at the output of each layer [9].

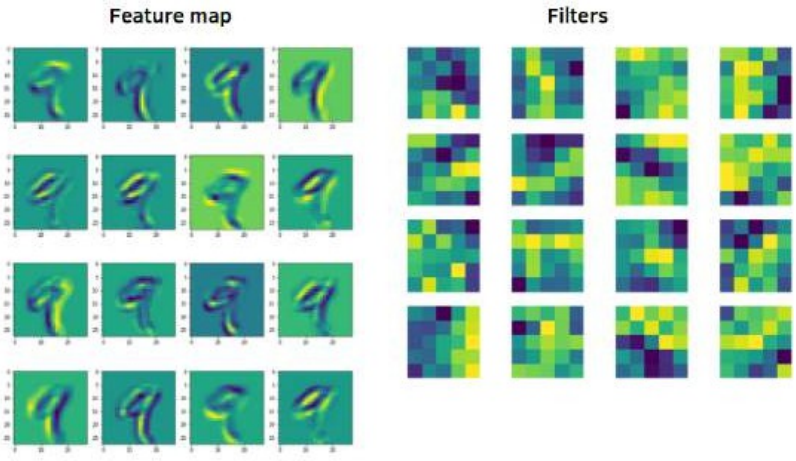
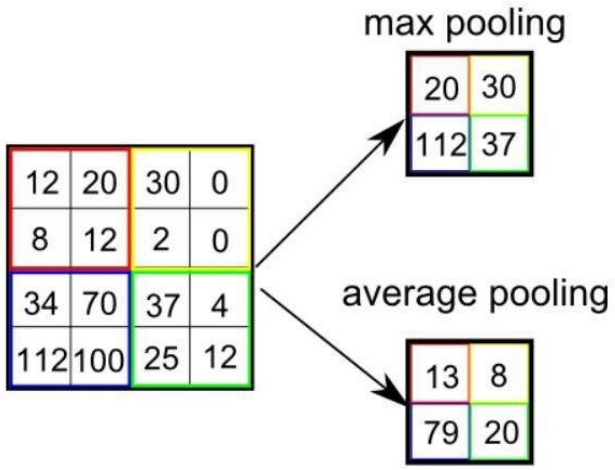


Figure 7: Result of using filters (convolutions) on an image with the number nine

Pooling is another basic operation in convolutional neural networks. This operation allows to reduce the dimensionality of data on the subsequent layers of the network, which improves performance and reduces the amount



of resources consumed (Figure 8).

Figure 8. Maximum pooling and average pooling

1.4 Machine learning in bone cell classification systems

brain

The classification accuracy of natural images has improved significantly over the past few years due to the increasing use of convolutional neural networks (CNNs) [10]. Meanwhile, this technology has also been applied to the interpretation of various medical images, including mitosis detection in breast cancer histological slices, skin cancer detection, mammogram evaluation, and cytologic classification in peripheral blood. However, successful use of CNNs for image classification usually depends on the availability of sufficient high-quality data - images and high-quality annotations - which can be difficult to access due to the costs associated with obtaining tags from medical experts. This is especially important in situations such as bone marrow cytomorphologic examination, where there is no technical gold standard and expert researchers are needed to label the data to train a neural network.

Based on CNNs, Kermani et al. created a diagnostic tool for screening patients with common treatable retinal diseases leading to blindness, which demonstrated classification performance comparable to that of human experts. Using CNNs, a correlation mapping between bone marrow cell image and classification can be established to realize efficient recognition of bone marrow morphology [11].

In 2017, a paper [12] was published in which the authors presented an approach to detect and automatically count red blood cells (red blood cells) in blood smear images. Romanowski-Wright stained blood smear images of healthy and cancer patients from Indian Medical College Kasturba and Atlas of Hematology were used for analysis. The authors did not provide information about the number of images, nor did they provide references to the original data sources, which tentatively downgrades the correctness of the developed approach [13], [14]. The obtained images were converted to the LAB color space. The authors motivated the conversion by the fact that the LAB space is closer to the perception of the human eye, but did not provide evidence about the relationship of this color space to the quality of detection and classification of any objects in the image. After changing the color space, the images were converted to grayscale. This conversion lowers the dimensionality of the feature space, thereby may lead to lower classification quality [15]. The authors also used image binarization based on Otsu thresholding method and equalization of each pixel values, which also reduces the dimensionality of the feature space and potentially reduces the accuracy of object detection and classification in the images. Image segmentation was performed using watershed transform method. The images were further filtered with a 2x2 median filter to reduce noise. An algorithm was used to cut out white blood cells on the acquired image area using the

k-medoids segmentation algorithm and extracted image properties. The circular Hough transform was used for erythrocyte counting.

The 2018 article [16] reviewed image processing and machine learning techniques for morphological analysis of blood cells: red blood cells, platelets and leukocytes. The authors of the article drew attention to the need to consider more specific blood cell types that depend on the degree of differentiation and the presence of pathology, and proposed an approach in which classification proceeds in several steps: from cell groups to specific cells of a given cell type. Many of the works reviewed in the article achieved high accuracy, and the main method was the support vector method. The authors also drew attention to the need for a unified repository of blood cell image data, which would improve the quality of blood cell detection and classification systems.

In [15], the authors propose a novel deep learning network to assist in the detection and classification of erythroid and myeloid cells in bone marrow biopsy histopathology images. To improve the detection and classification accuracy, the proposed neural network incorporates a novel neighborhood selection mechanism that considers the region surrounding the center of the detected cell.

In another paper [17], the authors compared classical feature selection using image processing methods and automatic selection based on AlexNet convolutional neural network [18]. The authors reported that the convolutional neural network selects the best features, which improves the final classification accuracy compared to classical methods.

In a 2019 paper [19], to detect and count blood cells: red blood cells, lymphocytes and platelets, the authors used the YOLO object detection model [20] and achieved an accuracy of 96%.

In the study, the authors used a cell detection algorithm based on a Faster Region-Based Convolutional Network (Faster Region-Based Convolutional Network).

Although CNNs have outperformed all classifiers relying on manually generated features in a wide range of tasks, the structure of their output data is usually not directly interpretable by humans. To address this shortcoming, various feature recognition methods have been developed based on which the model classifies objects. In this way, the features chosen by the network to classify images appear to be robust and robust against some label noise, which cannot be avoided in a data-driven method relying on expert annotations.

Thus, creating an efficient algorithm and training on a large number of images can break the dependence of bone marrow cell morphology analysis on human recognition and realize intelligent recognition.

2 MATERIALS AND METHODS

2.1 Materials

Hemocytoblast is a multipotent stem cell from which other cell types can differentiate [21].

All blood cells are divided into three lineages:

- Red blood cells, also called red blood cells, are cells that carry oxygen. Red blood cells function and enter the bloodstream. The number of reticulocytes, immature red blood cells, gives an estimate of the rate of erythropoiesis.
- Lymphocytes are the cornerstone of the adaptive immune system. They are derived from common lymphoid precursors. The lymphoid lineage consists of T cells, B cells, and natural killer cells. This is lymphopoiesis.
- Cells of myeloid origin, which include granulocytes, megakaryocytes, and macrophages, are derived from normal myeloid precursors and are involved in roles as diverse as innate immunity and blood clotting. This is myelopoiesis.
- Granulopoiesis (or granulocytopoiesis) is the maturation process of granulocytes, with the exception of mast cells, which are granulocytes with extramedullary maturation.
- Megakaryocytopoiesis is the hematopoiesis of megakaryocytes.

A dataset taken from open sources is used to analyze and develop the system. The first dataset is taken from the image archive,

Cancer Image Archive (Cancer Image Archive) is a set of images of different types of red bone marrow cells stained by Romanowsky and includes 170000 images of 21 cell types [10]. The dataset was collected from 961 patients at the Munich Leukemia Research Laboratory Institute (Figure 8). The second dataset is a peripheral blood cell image set consisting of 17092 images of 9 cell types. The set was collected at the Hospital Clinic of Barcelona (Figure 9).

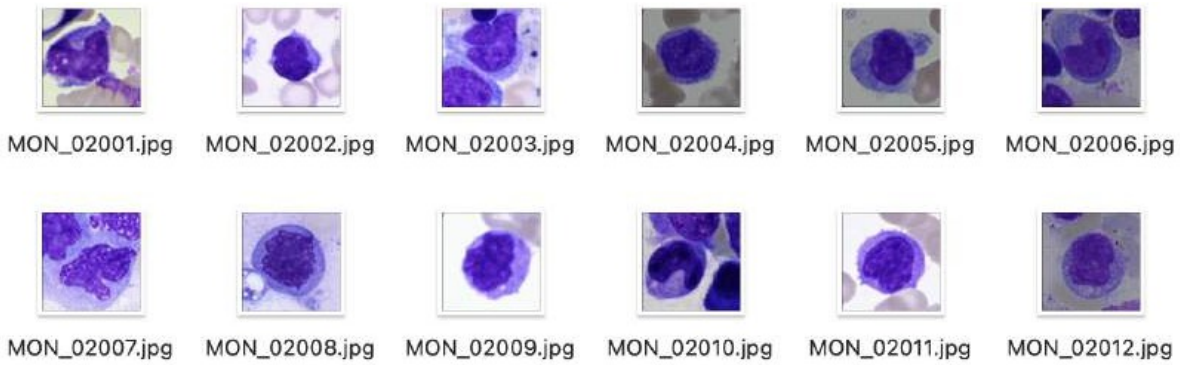


Figure 9: Images of monocytes from Cancer Image Archive

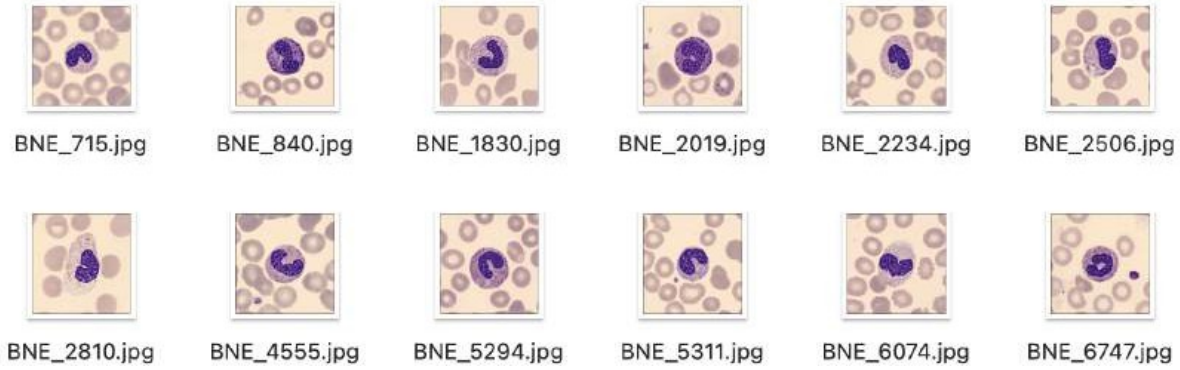


Figure 10. Images of neutrophils from the Barcelona Hospital Clinic dataset

All datasets were processed and prepared for further use. The result was a set with the following 10 classes:

- Neutrophils
- Monocytes
- Blasts
- Plasma cells
- Granulocytes
- Erythroblasts
- Platelets
- Lymphocytes
- Pathogenic cells (Leukemia, destroyed cells and abnormal cells)
- Other cells (unclassifiable cells)

2.2 Proposed methods and approaches

2.2.1 Deep learning framework

To make the right choice, it is necessary to compare the pros and cons of different solutions, evaluate their limits of capabilities and learn about the best use cases. TensorFlow was developed by Google and implemented in C++ [22]. A shell that can run on top of TensoFlow, Theano or CNTK. It supports a wide range of neural network layers such as convolutional layers, recurrent or dense. PyTorch is the successor to the Torch library written in Lua. The framework was developed by Facebook and has been used by Twitter, Salesforce, Oxford University and many other companies. The next framework, CNTK (Microsoft Cognitive Toolkit) is an open source deep learning framework designed to handle really big data with support for Python, C++, C# and Java [23]. Caffe is a framework implemented in C++ that has a Python interface. The framework supports CNNs and feed-forward networks, and is suitable for model training, image processing, and improving existing neural networks [24]. Theano is an open source project whose main developer is the machine learning group at the University of Montreal. On September 28, 2017, the project was announced to be discontinued after the 1.0 release, with a promise to maintain minimal support for one year [25].

Table 1 presents a comparative analysis of some of the most popular deep learning frameworks.

Свойство	Caffe	Theano	TensorFlow	PyTorch	CNTK	Keras
Базовый язык	C++	Python	C++	Pytho	C++	Python
API	C++ Python	Python	C++ Python	Python	C++, C# Python	Python
Многоядерные CPU	+	+	+	+	+	+
GPU	+	+	+	+	+	+
Распределенное обучение	+	-	+	+	+	+
Разработчик	Центр компьютерного зрения и обучения Беркли	Университет Монреаля	Google	Facebook	Microsoft	Франсуа Шолле
Открытые коды	+	+	+	+	+	+
Обученные сети	+	-	+	+	+	+

Table 1 - Comparative analysis of deep learning frameworks

Widespread practical use of neural networks is possible due to the availability of a large number of ready-made solutions for training deep neural networks, frameworks that allow fast and efficient development of new architectures, as well as due to the possibility of using modern multi-core processors, GPU computing gas pedals [26].

PyTorch is an open-source machine learning framework for the Python language based on Torch. It is used to solve various tasks: computer vision, natural processinglanguage . It is developed mainly by Facebook's artificial intelligence group. There is also an ecosystem built around this framework, consisting of various libraries developed by third-party teams: PyTorch Lightning and Fast.ai, which simplify the learning process

models, Pyro, a module for probabilistic programming, from Uber, Flair, for natural language processing and Catalyst, for training deep learning models [8].

PyTorch provides two basic high-level models:

- Tensor calculations (similar to NumPy) with advanced GPU acceleration support
- Deep neural networks based on autodiff system

Thus, this framework was found to be the most suitable for the aims and objectives of the work.

2.2.2 Pre-trained models on ImageNet

Transfer learning is a concept from the field of machine learning that represents the reuse of learning results from one domain to another. For example, knowledge gained from learning to recognize cars can be applied when trying to recognize trucks. From a practical perspective, reusing or transferring information from previously learned tasks to learn new tasks can significantly improve the sampling efficiency of a reinforcement learning agent. Transfer learning is flexible, allowing previously trained models to be directly used as pre-processing of feature extraction and integrated into entirely new models [11].

The ImageNet database is a project to build and maintain a massive database of annotated images designed to hone and test image recognition and machine vision techniques. As of 2022, the database has recorded about ten million URLs with images that have been manually annotated for ImageNet, with the annotations listing the objects caught in the image and rectangles with their coordinates [18].

Annotations at the level of images themselves indicate the presence or absence of an object of a given class (e.g., "there is a tiger in the picture" or "there are no tigers in the picture"). At the object level, the annotation includes a rectangle with the coordinates of the visible part of the object. ImageNet uses a variant of the WordNet semantic network to categorize objects that is quite detailed, e.g., dog breeds are represented by 120

classes. Each node in WordNet has hundreds or thousands of images mapped to it, but the average for 2016 is about 500 images. For 2022, there are 14,197,122 images in ImageNet, organized into 21,841 categories [27].

The Pytorch library has an extension for working with ImageNet pre-trained neural networks - torchvision. The extension includes a set of different models of convolutional neural networks with stored weights inside. In this paper we analyze the accuracy of the classifiers of the following models: ResNet [28], MobileNetV3 [29], EfficientNetV2 [30], CoAtNet [31].

2.2.3 Interpretive machine learning

Recently, attempts have been made to use neural networks, especially convolutional neural networks, for detection and classification in the medical field, however, these methods contain a black box problem common to all neural networks. To solve this problem, we use interpretive machine learning, a branch of machine learning that aims to determine the behavior of a neural network during training and operation of the neural network on the objects being analyzed. Interpretive machine learning methods bring predictability and interpretability to the detection and classification system. In addition, interpretive methods allow us to determine which features in the images are the most important, this is especially useful when analyzing the number and morphological state of blood and bone marrow cells, in other words, automated interpretation allows us to gain new knowledge about the structure of the analyzed objects and what a human needs to pay attention to in order to correctly assign the object to the right class [32].

With increasing model complexity and the resulting lack of transparency, methods for model interpretation become increasingly important. Model understanding is both an active area of research and an area of practical application of machine learning in various industries.

The interpretation algorithms are divided into three groups: primary attribution, layer attribution and neuron attribution, which are defined as follows:

- Primary attribution: evaluates the contribution of each input feature to the model outputs.
- Layer attribution: estimates the contribution of each neuron in a given layer to the model output.
- Neuron attribution: evaluates the contribution of each input feature to the activation of a particular hidden neuron.

Integrated gradients are the integral of the gradients with respect to the input data on the path from a given baseline to the input data. The integral can be approximated using the Riemann or Gauss-Lejandre sum quadrature rule. Formally, it can be described as follows:

$$\mathsf{IntegratedGrads}_i(x) ::= (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} \, d\alpha$$

Integrated gradients along the i-th dimension of the input data X. Alpha is the scaling factor. The equations are copied from the original article.

The cornerstones of this approach are two fundamental axioms, namely sensitivity and realization invariance.

The occlusion method is a perturbation-based approach to computing attribution that involves replacing each adjacent rectangular region with a given baseline/ethalon and computing the difference in results. For features located in multiple regions (hyperrectangles), the corresponding output differences are averaged to compute the attribution for that feature. Shading is most useful in

cases such as images where pixels in a continuous rectangular region are likely to be highly correlated.

2.2.4 YOLO object detection model

The modern field of object detection includes a set of concepts that must be defined before a model can be described. . .

Earlier architectures for object detection consisted of two separate stages, a region suggestion network that performs object localization and a classifier to detect object types in the suggested regions. These can be computationally very expensive and are therefore not suitable for real-world real-time applications. Single-shot detection encapsulates both localization and detection tasks into a single direct network operation, resulting in much faster detection when deployed on lighter hardware.

In image classification tasks, predictions are based on the final convolutional feature map, the smallest but deepest representation of the original image. In object detection, feature maps from intermediate convolutional layers can also be directly useful because they represent the original image at different scales. Consequently, a fixed-size filter working with different feature maps will be able to detect objects of different sizes.

Priors are pre-calculated fields defined at specific positions on specific object maps, with specific proportions and scales. They are carefully selected to match the characteristics of the object bounding boxes in the dataset.

Jacquard Index or Jacquard Overlap or Intersection over Unification (IoU) measures the degree or extent to which two blocks overlap.

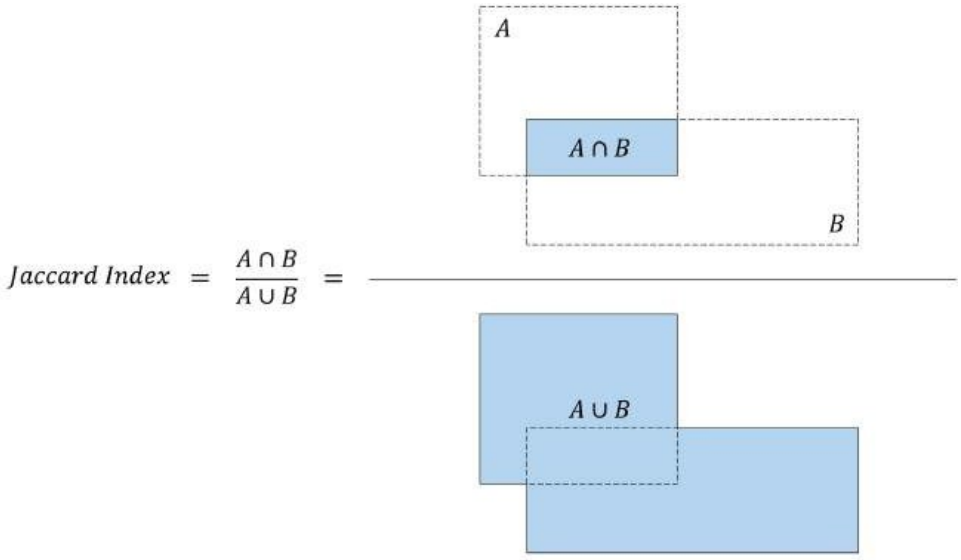


Figure 11: Jaccard index

YOLO is an object detection system. It was developed by Joseph Redmon. The advantage of YOLO over other architectures is speed. The YOLO family models are fast and outperform R-CNN (Region-Based Convolutional Neural Network) [33] and other models. This makes it possible to achieve real-time object detection [20].

At the time of first publication, YOLO has achieved an advanced mAP (Mean Average Precision) value compared to other systems such as R-CNN and DPM (Deformable Part Model). On the other hand, YOLO has difficulty in accurately localizing objects. However, the new version has made improvements in the speed and accuracy of the system.

Other architectures mainly used the sliding window method over the whole image and the classifier was used for a specific region of the image (DPM). Also, R-CNN used the region proposal method. The described method first creates potential bounding boxes. Then, a classifier is run on the regions bounded by the bounding boxes and the next removes repeated recognitions and refines the bounding boxes.

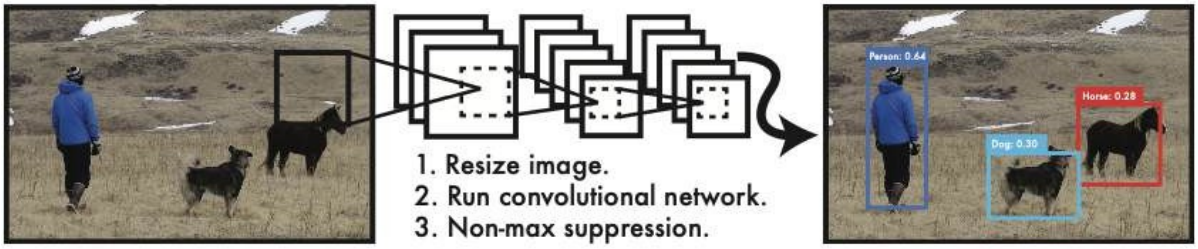


Figure 12. YOLO detection system

Processing images c YOLO represents is a sequence of steps (Figure 12). System:

(1) resizes the input image to 448× 448,

(2) runs a single convolutional network on the image

(3) limits the resulting detections to the validity of the model.

At any given location, several a priori values may overlap significantly. Consequently, the predictions arising from these a priori values may actually be duplicates of the same object. Non-max suppression (non-max supression) is a means of removing redundant predictions by suppressing all predictions except the prediction with the maximum score.

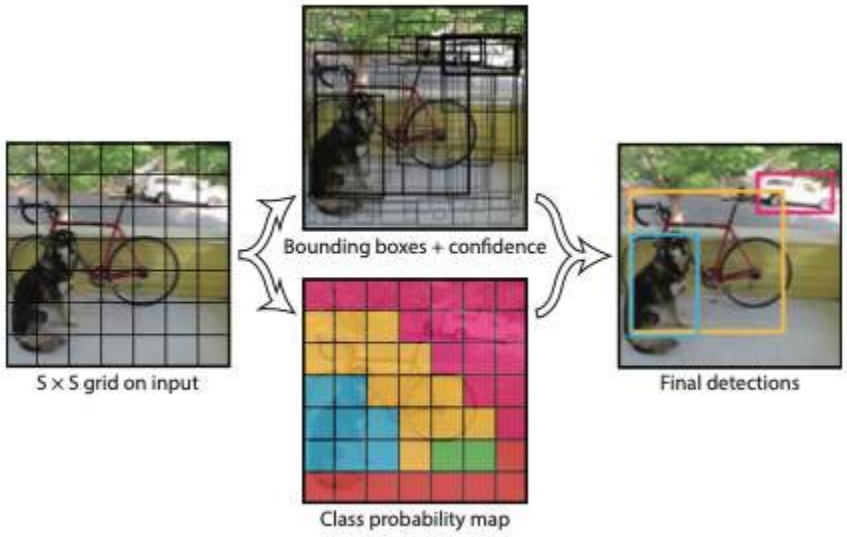


Figure 13. YOLO model

YOLO has reimagined the object detection problem into a regression problem. It goes from image pixels to the coordinates of bounding boxes and class probabilities. Thereby, a single convolutional network predicts multiple bounding boxes and class probabilities for the content of these regions (Figure 13).

The YOLO system models detection as a regression problem. The model divides the image into an S × S grid and for each grid cell predicts B bounding boxes, the confidence for these boxes, and the

C class probabilities. These predictions are encoded as tensor S× S× (B* 5 + C).

2.2.5 Client-server architecture

A client-server architecture is a computing or network architecture in which jobs or network loads are distributed between service providers, called servers, and service customers, called clients. The actual client and server are software programs. Usually these programs are located on different computing machines and communicate with each other over a computing network through network protocols, but they can also be located on the same machine. Server programs expect requests from client programs and provide them with their resources in the form of data (e.g., file transfer via HTTP, FTP, BitTorrent, streaming media, or database operations) or in the form of service functions (e.g., e-mail, instant messaging, or web browsing on the World Wide Web). Since one server program can execute requests from many client programs, it is placed on a dedicated computing machine, configured in a special way, usually in conjunction with other server programs, so the performance of this machine must be high. Because of the special role of such a machine in the network, the specifics of its hardware and software, it is also called a server, and the machines executing client programs are called clients, respectively [34].

Benefits:

- Absence duplication code duplication of the server program by client programs.

- Since all calculations are performed on the server, the requirements for the computers on which the client is installed are reduced.
- All data is stored on the server, which is usually much better protected than most clients. It is easier to organize authority control on the server to allow only clients with appropriate access rights to access the data.

Disadvantages:

- Server inoperability can make the entire network inoperable. An inoperable server should be considered a server whose performance is insufficient to serve all clients, as well as a server that is under repair, maintenance, and so on.
- Supporting the operation of this system requires a separate specialist - system administrator.
- High cost of equipment.

2.2 Implementation tools

The high-level programming languages used to implement the system are Python version 3.9 and JavaScript of the ECMAScript 2021 standard.

Python is a high-level interpreted general-purpose programming language. Its design philosophy emphasizes code readability using mandatory indentation. Python has dynamic typing and garbage collection. It supports several programming paradigms, including structured (especially procedural), object-oriented, and functional programming [8].

The client part of the application uses HTML markup language, CSS style language and standard JavaScript language library. JavaScript is the main programming language used to build interfaces for web applications. Facebook's React is used as the framework for the client part of the application.

The server side of the application uses the Python library FastAPI version 0.70. FastAPI is a modern, fast and high-performance web infrastructure for creating APIs based on standard Python type annotation. The system is based on the following standards: OpenAPI and JSON Schema.

The Docker virtualization tool is used to deliver a working application. The Docker system allows you to run programs in an isolated environment - a container, which provides opportunities for application portability.

The models were trained and analyzed using the standard PyTorch deep learning library version 1.10 and the torchvision convolutional neural networks extension version 0.12.

The Captum library version 0.5 is used for interpretive machine learning methods. Captum provides state-of-the-art algorithms, including integrated gradients, to give researchers and developers an easy way to understand which features affect model output. Captum helps machine learning researchers more easily implement interpretability algorithms that can interact with PyTorch models.

3 DESIGN SOLUTION

3.1 Conceptual model and algorithms

Logic classification represents set set of steps shown in Figure 14.

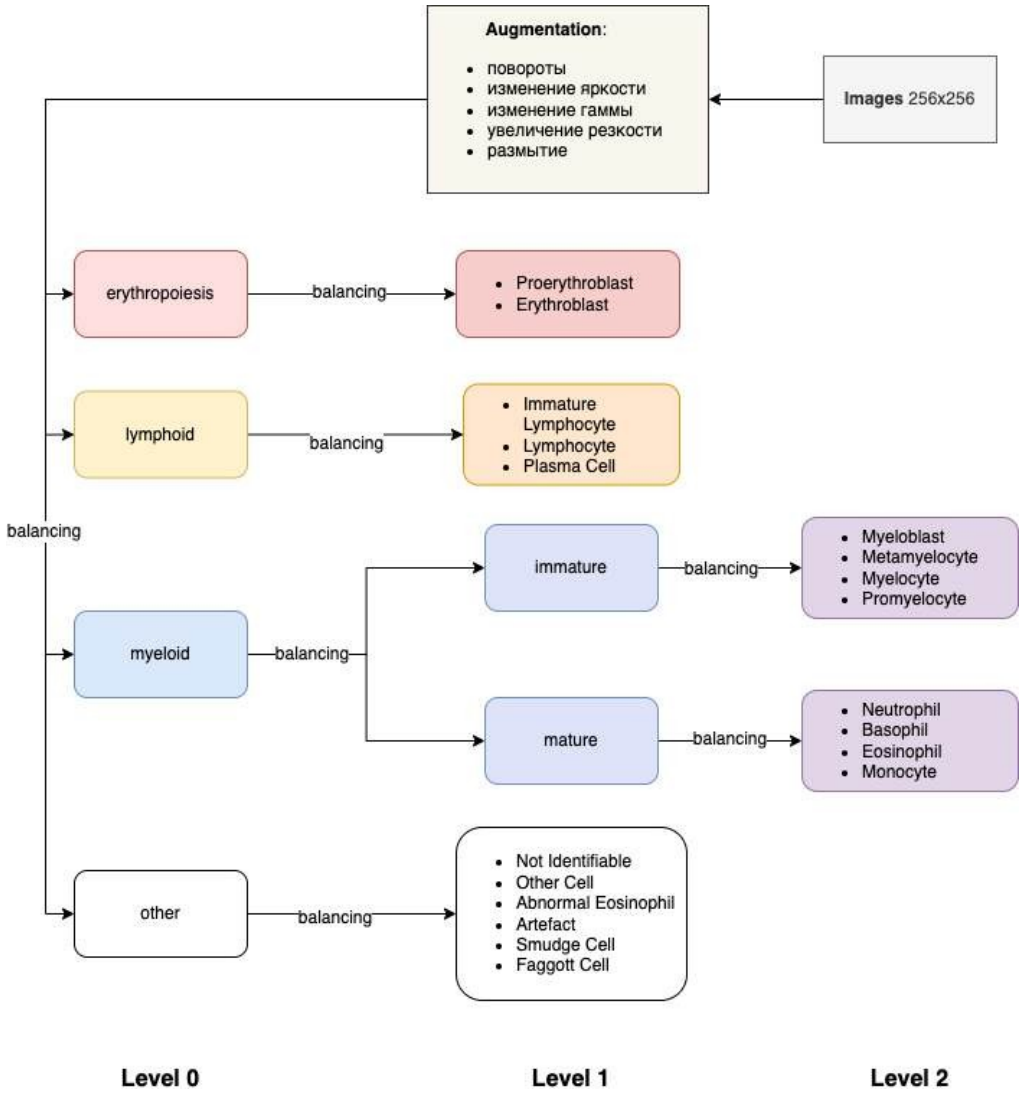


Figure 14. Conceptual model of model training

Before the model starts training, an augmentation process takes place that consists of random rotations, brightness changes, changes in

gamut, sharpening and blurring. This choice of augmentation format is due to the peculiarities of microscope operation [35].

The logic of the training process is a tree structure. Since the dataset can be unbalanced, it is important to use class balancing for this purpose, as it directly affects the quality of the classification [36]. For this purpose, the algorithm described in Figure 15 is used. A graph map of classes or cell types with respect to cell differentiation pathways is given as input (Figure 16). The map is a nested tree structure with the names of cell types at a given level. The algorithm traverses this map and at each level evaluates how many objects are at a given level of the tree and performs balancing based on a class with a minimum number of objects, not less than a given threshold, for example, a class with at least 1000 images. Each level prepares data for training, validation and training with the ratio of 8/1/1 respectively.

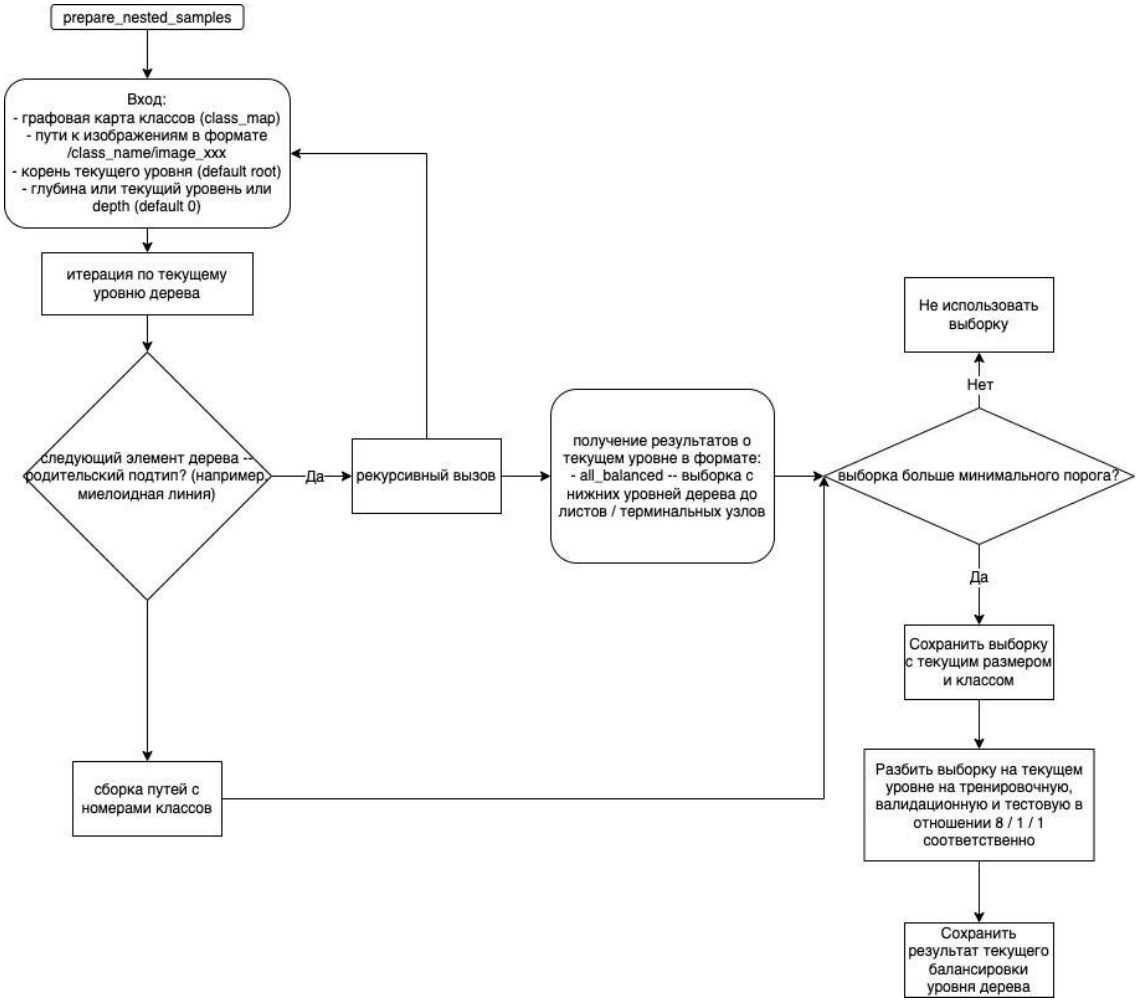


Figure 15. Algorithm for balancing the sample at each level of the cell differentiation description tree

```
CELL_TYPES = {
    'erythropoiesis': {
        'Proerythroblast': 'PEB',
        'Erythroblast': 'EBO',
    },
    'lymphoid': {
        'Immature Lymphocyte': 'LYI',
        'Lymphocyte': 'LYT',
        'Plasma Cell': 'PLM',
    },
    'myeloid': {
        'myeloid_immature': {
            'Myeloblast': 'BLA',
            'Metamyelocyte': 'MMZ',
            'Myelocyte': 'MYB',
            'Promyelocyte': 'PMO',
        },
        'myeloid_mature': {
            'Neutrophil': {
                'Band Neutrophil': 'NGB',
                'Segmented Neutrophil': 'NGS',
            },
            'Basophil': 'BAS',
            'Eosinophil': 'EOS',
            'Monocyte': 'MON',
        },
    },
    'abnormal': {
        'Not Identifiable': 'NIF',
        'Other Cell': 'OTH',
        'Abnormal Eosinophil': 'ABE',
        'Artefact': 'ART',
        'Smudge Cell': 'KSC',
        'Faggott Cell': 'FGC'
    }
}
```

Figure 16. Description of the tree of blood cell types and subtypes based on their differentiation pathways

Next, classification of cells into types and subtypes was performed. Subsequently, a new classifier was trained for each class based on the results of the previous classifier.

In the interpretation phase, already trained models are used. For interpretation, it is necessary to select one of the trained models and also to select an image to be used for interpretation. The selected model and images are then loaded

to the server and the user is prompted to select an interpretation method. After selection, the selected method is used for interpretation. At the output the user receives a processed image, which shows the features that are taken into account by the neural network during classification (Figure 17).

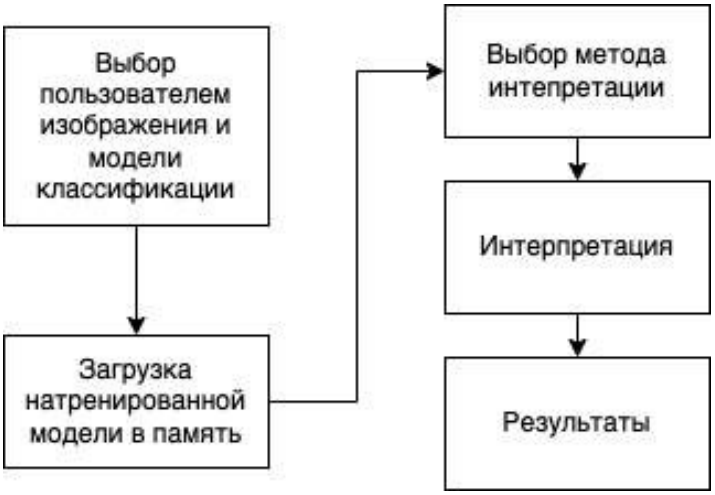


Figure 17. Conceptual model for interpreting the performance of the classification model for a selected image

3.2 Database model

A MySQL database is a set of elements (tables) structured to execute multiple users' queries simultaneously at a particular point in time. In essence, a MySQL database is a set of interrelated data and contains information about various entities. Figure 18 shows the MySQL database being designed for this classification work.

The "Cells" entity is necessary for storing and viewing information about cell type names and images. It contains fields: name (cell names), image (cell images), type (cell type).

The "ModelsTypes" entity is required to store and view information about the names of trained models. It contains the following fields: name (model name).

The "Models" entity is required to store and view information about trained models. It contains fields: name (model name), dump (saved model weights), type (cell types for this model).

The "CellTypes" entity is necessary for storing and viewing information about cell names. It contains fields: name (name of cell types), code (abbreviated name of cell types).

The "ModelsToCellTypes" entity is required to store and view information about the relationship between cell type names and their corresponding models. It contains fields: id_model (model identifier), id_cell_type (cell types identifier).

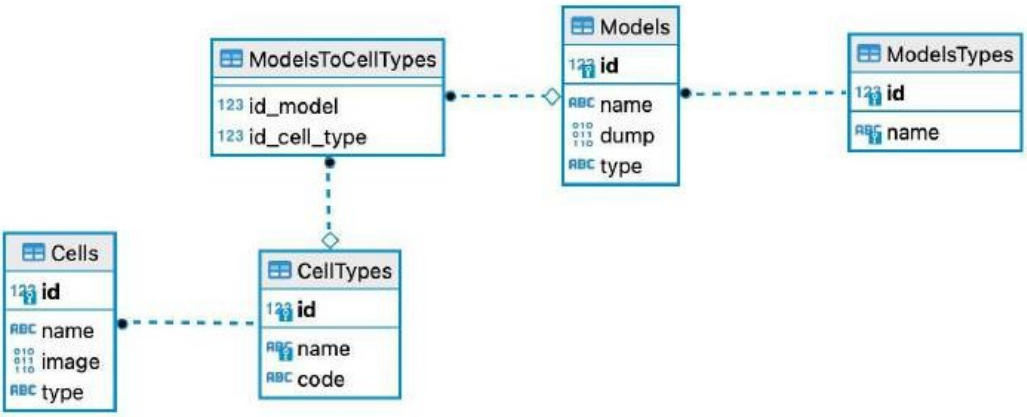


Figure 18. Data schema

3.3 System Architecture

The system is a client-server application, which consists of two main parts: server and client (Figure 19). The server part is implemented as software that allows accessing the server by address. Two input points can be accessed: model training and model testing. When training models, the user needs to upload data in a special format.

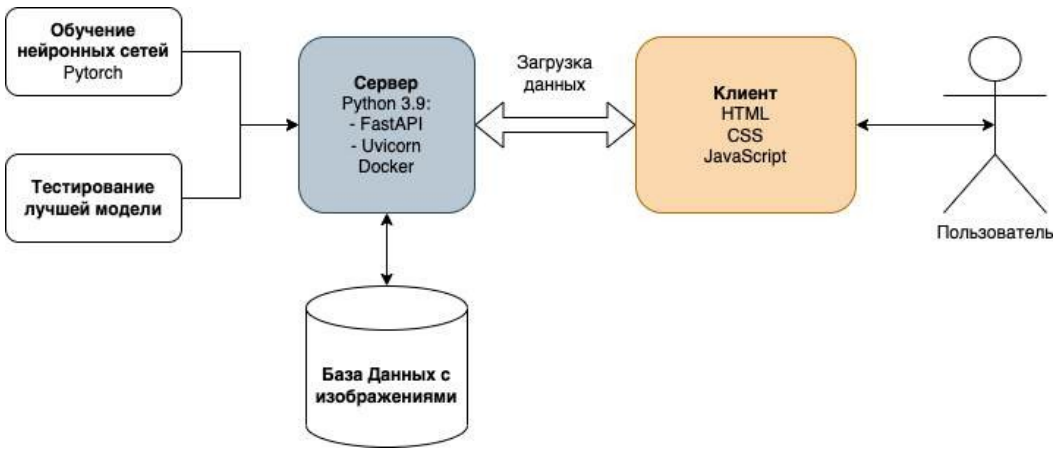


Figure 19: System architecture

3.5 Interaction with the user

User interaction takes place through a web browser. The interface is implemented on the basis of HTML, CSS and JavaScript for interaction with the server.

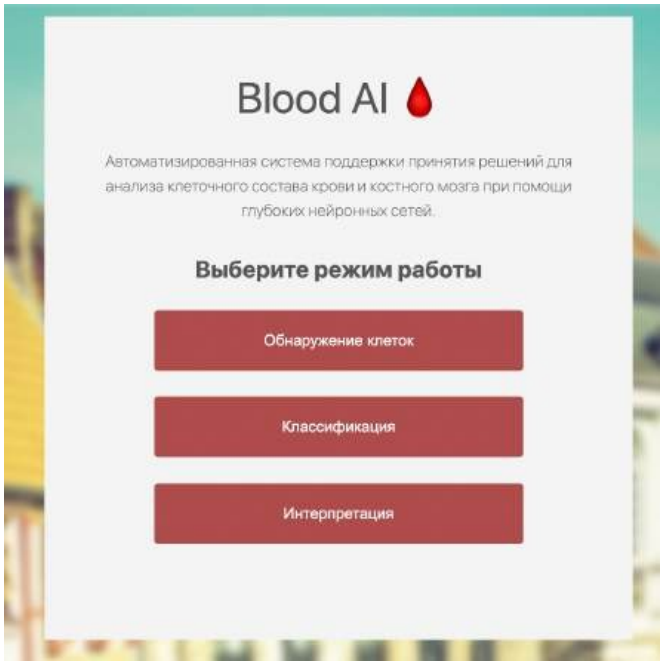


Figure 20. Program start page

Before use, the user is prompted to select one of the system modes: cell detection, classification, and interpretation (Figure 20).

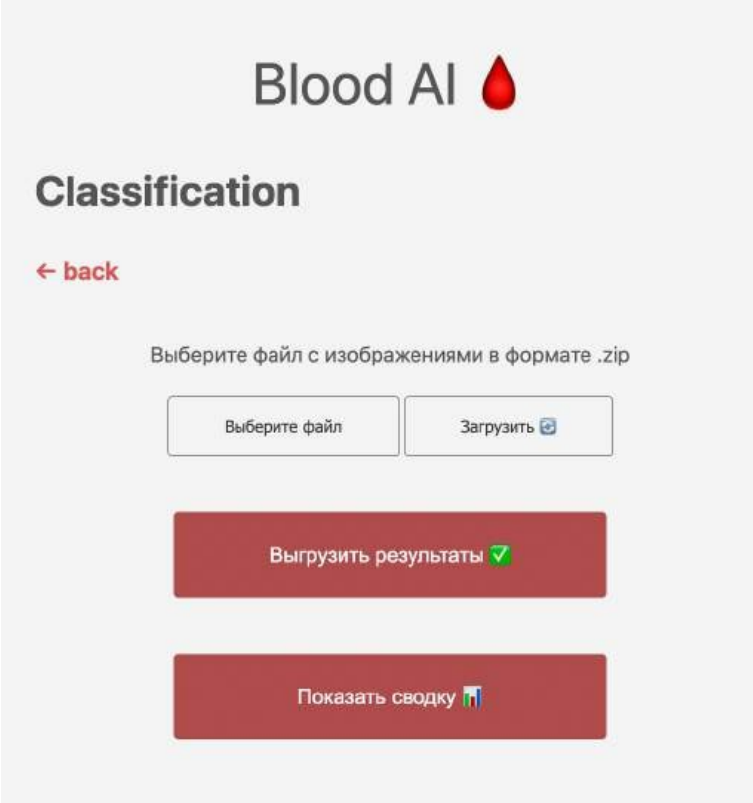


Figure 21. Page interface for classification

In Classification mode, the user is prompted to select an archive file with a .zip extension that is expected to contain blood and bone marrow cell images (Figure 21).

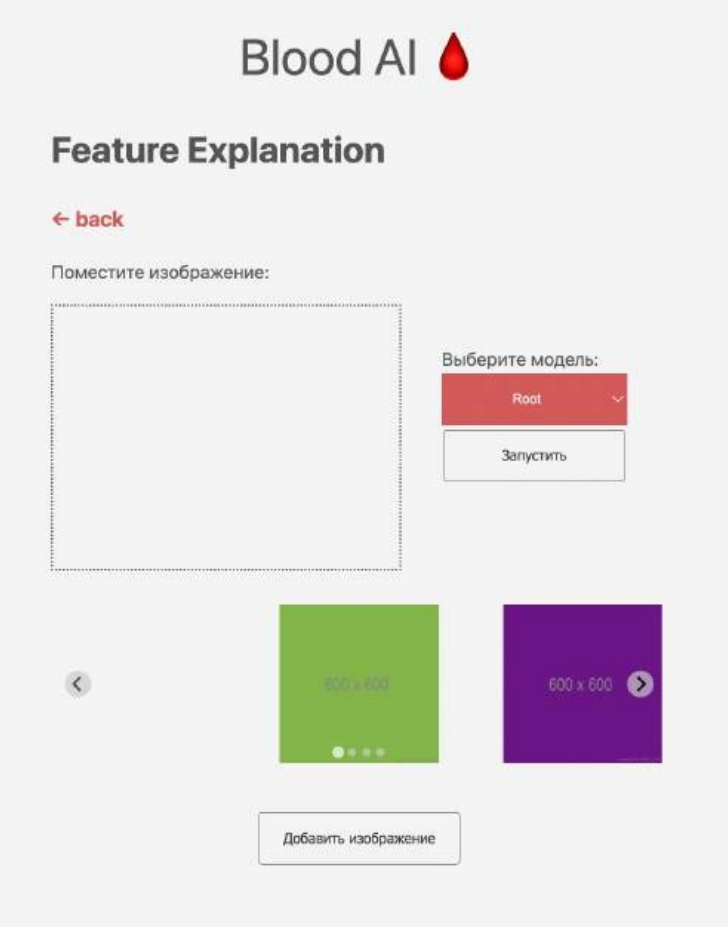


Figure 22: Page interface for feature interpretation For the operation of interpretive machine learning methods (Feature Explanation) the interface in Figure 22 is used. This requires the user to select one of the trained models at a particular classification level based on blood and bone marrow cell differentiation pathways, and to select the image to be transformed using interpretive learning techniques.

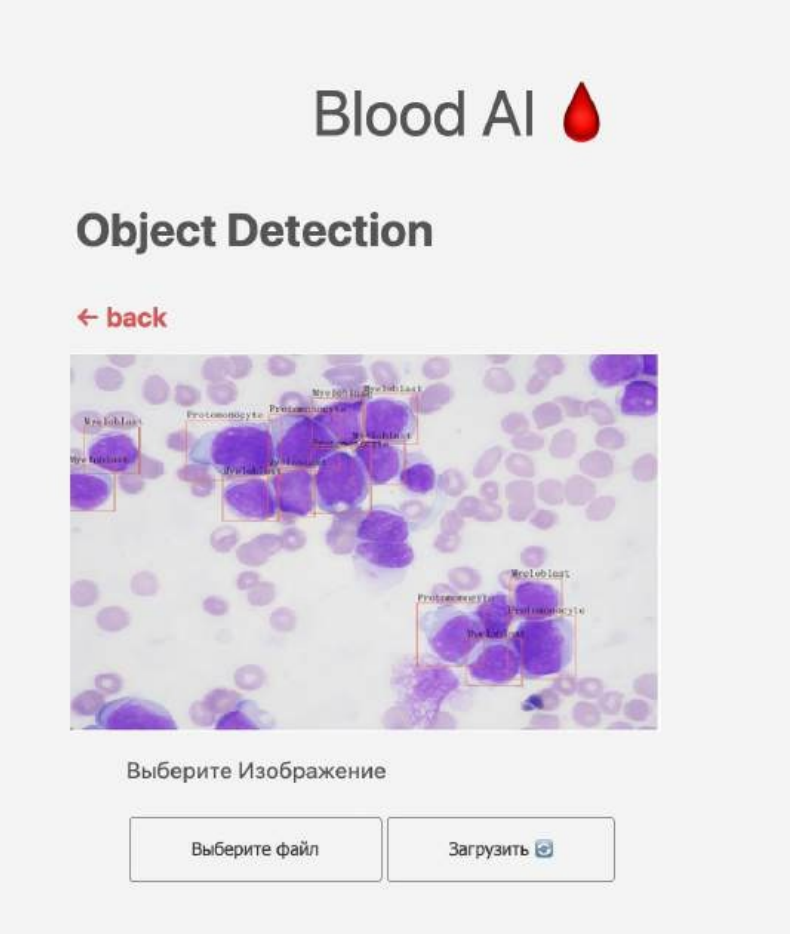


Figure 23. Page interface for cell detection

Figure 23 shows the operation interface of the cell detection algorithm on a slide with bone marrow cells.

4 EXPERIMENT AND RESULTS

4.1 Data preparation

The original dataset is a set of two sources. Each cell type has a different number of samples, which means that the samples are unbalanced, so it was decided to balance the samples based on the type with the smallest number of samples (Figure 24).

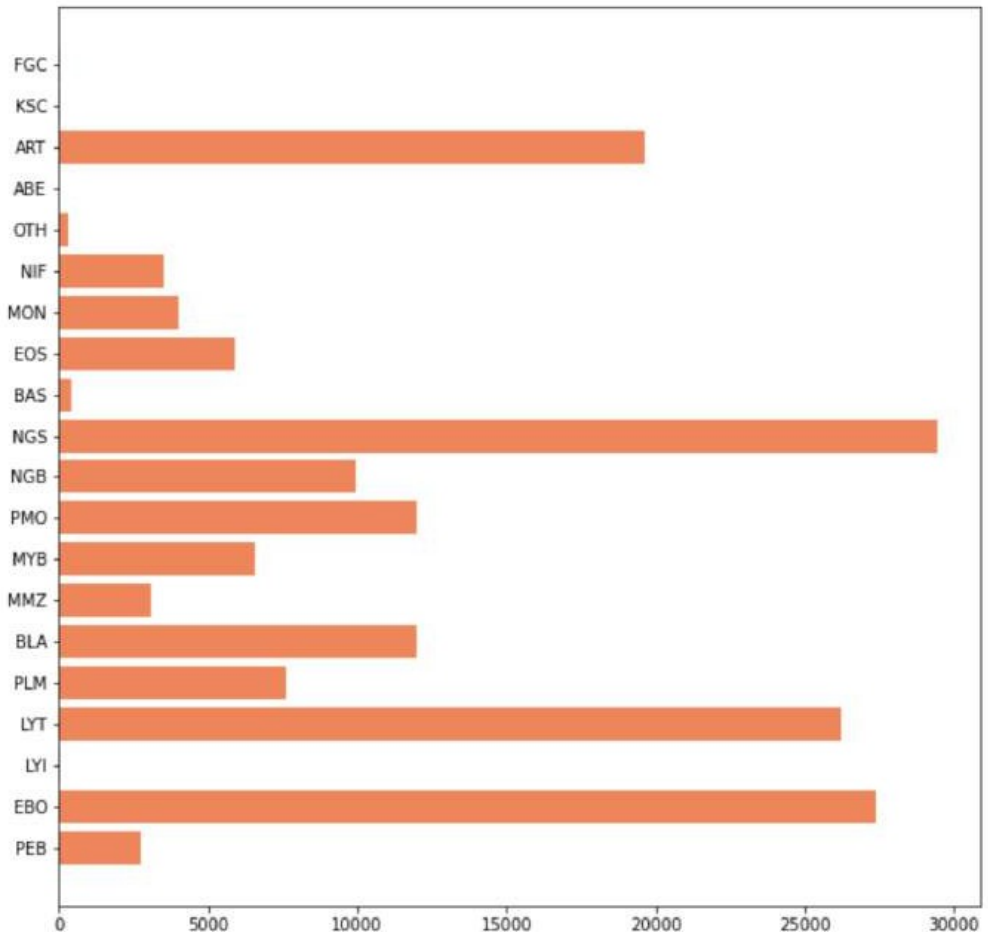


Figure 24. Unbalanced sample

```
[0, 1] erythropoiesis 2740
[0, 1] lymphoid 7629
[0, 1, 2, 3] myeloid_immature 3055
[0, 1] Neutrophil 9968
[0, 1, 2] myeloid_mature 4040
[0, 1] myeloid 12120
[0, 1] abnormal 3538
[0, 1, 2, 3] root 5480
```

Figure 25. Balancing results at each nesting level Figure 25 shows the result of the balancing algorithm at each

at the level of the cell classification tree.

Augmentation with the following parameters was used to expand the sample (Figure 26):

- random turns
- random gamma and contrast variations
- random variation in blur and sharpness

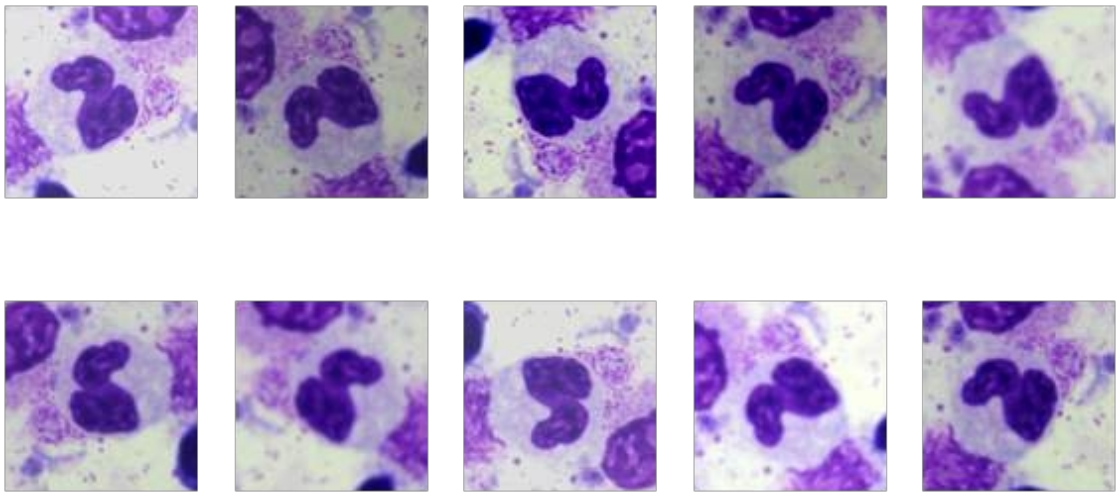


Figure 26. Augmented images

4.2 Experimental results

The following convolutional neural network architectures for classification have been selected for training:

- ResNetXt50 (2019)
- MobileNetV3 (2019)
- EfficientNetV2 (2019)
- CoAtNet (2020)

The models were trained in three modes: pure architecture without pre-training and augmentation, with pre-training, and with pre-training and augmentation with a number of epochs of 10. The averaged results for all classes and subclasses are depicted in Figure 27. It can be seen that the pre-trained models that were trained on the augmented data perform better in classification. This further confirms that transfer learning and artificially augmented sampling should be used to improve accuracy.

чистая модель	с предобучением	с предобучением и аугментацией	Случай
81.7	87.5	89.5	корень дерева классификации
92.1	95.6	97.2	эритропоэз
95.9	97.2	97.9	лимфоидная линия
85.2	88.8	92.2	миелоидная линия
70.9	78.5	82.2	незрелая миелоидная линия
92.4	95.2	96.8	миелоидная зрелая линия
72.8	88.3	89.4	нейтрофилы
87.1	89.8	90.0	патологичные клетки
84.7	90.1	91.9	все

Figure 27. Average accuracy results of classifiers on different levels

Figures 28, 29, 30 show the graphs of the classifier learning process for different classes for different cases as a ratio of the loss function results for the training and validation samples. It can be seen that overtraining occurs on models without pre-training and augmentation, but in the reverse case the results show a correct training process.

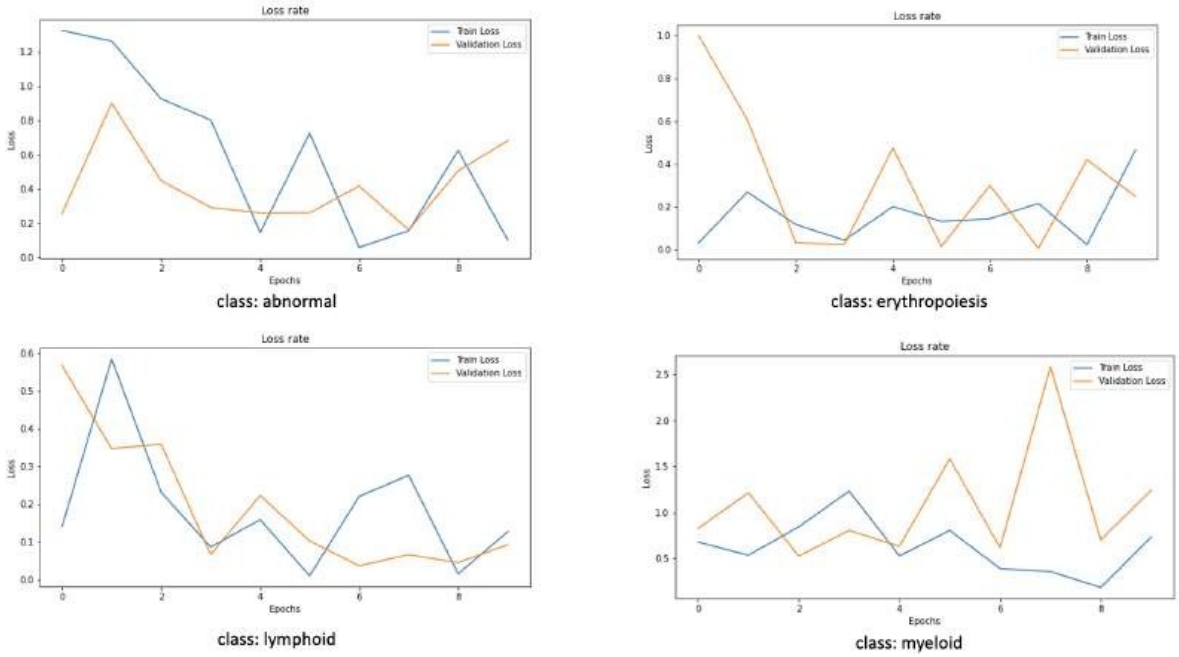


Figure 28. Learning process: without pre-training and without augmentation

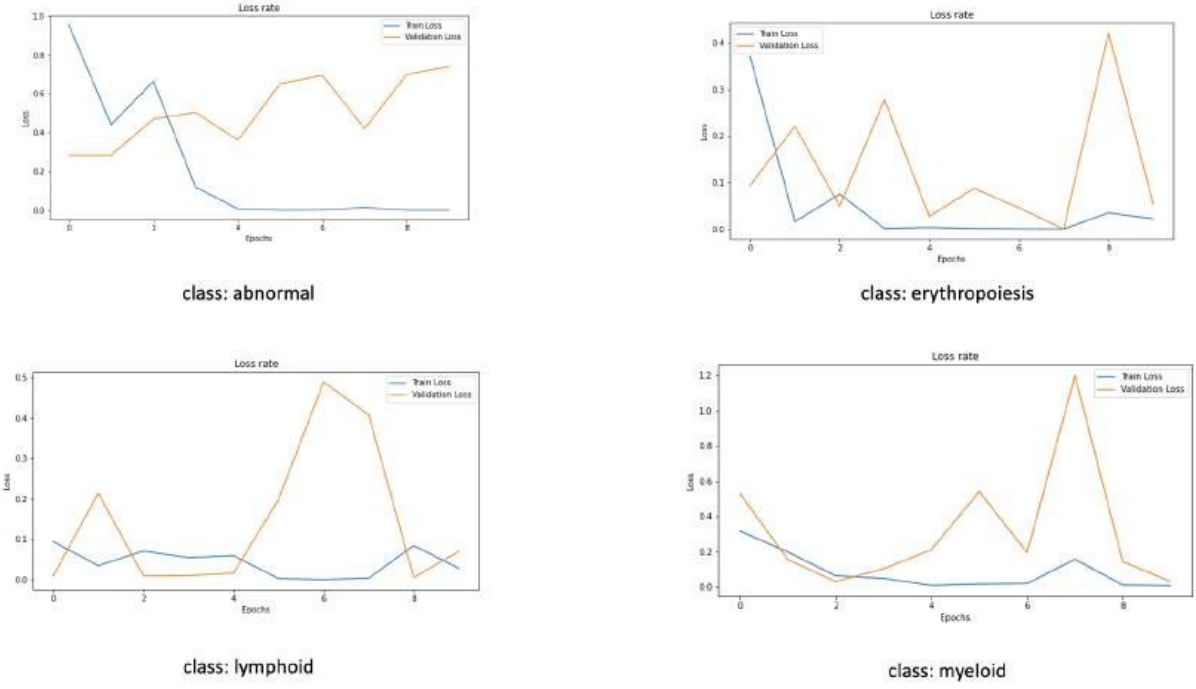


Figure 29. Learning process: with and without augmentation pre-training

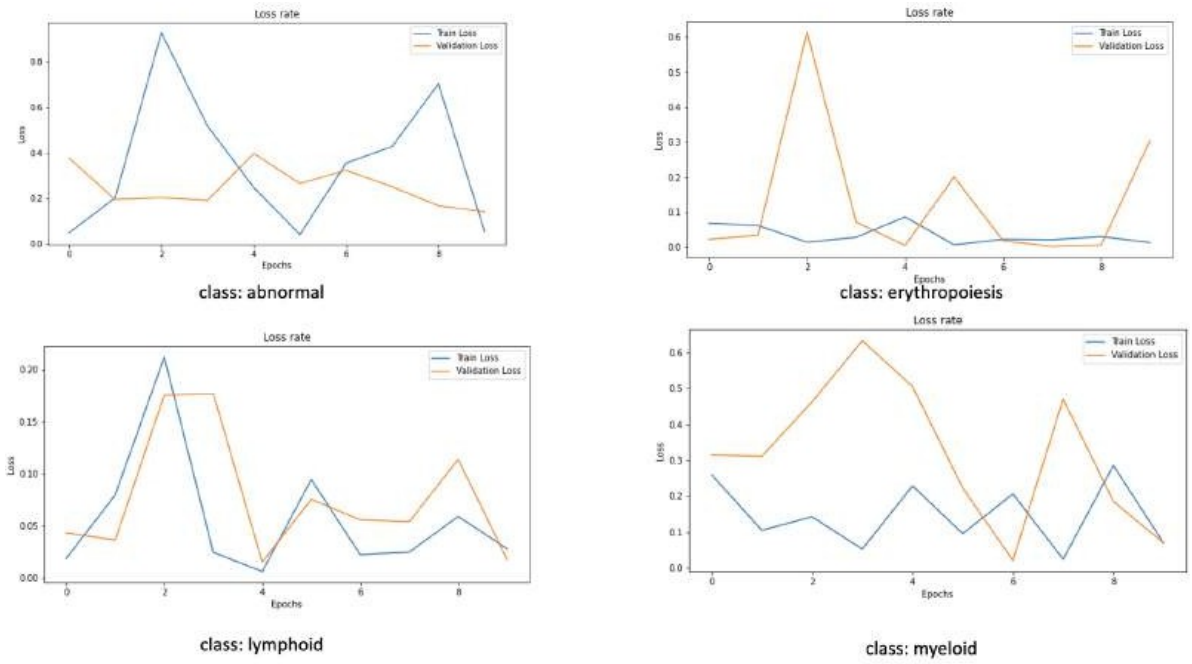


Figure 30. Learning process: with pre-training and with augmentation

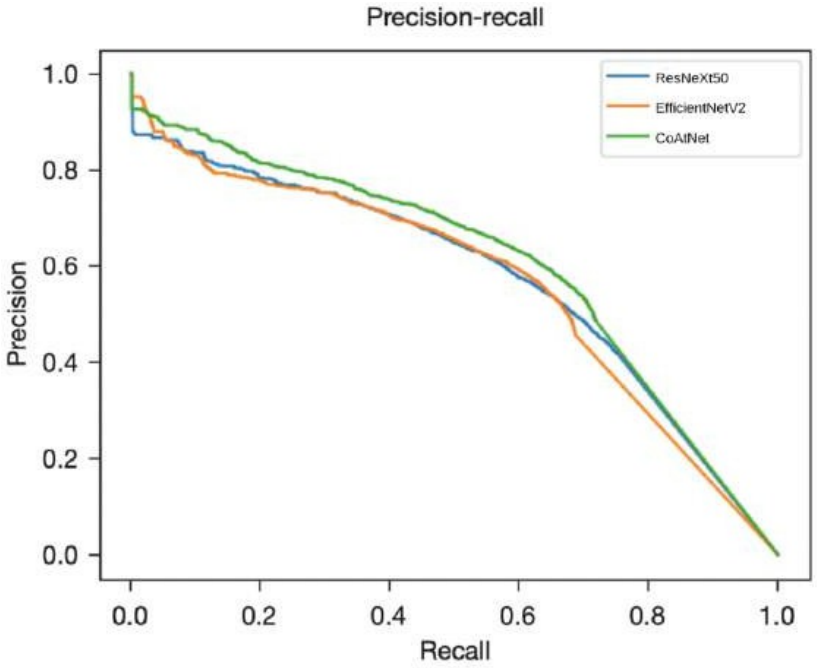


Figure 31. Precision (precision) and sensitivity (recall) for the best classification models

Figure 31 is a graph of the performance accuracy of the best classification models. It is important to note that the best results were shown by the CoAtNet model, which was also confirmed by the authors of this architectures at the ImageNet competition.

The Figure shows the results of two interpretive machine learning methods: the Occlusion Method and the Integrated Gradient Method. These images show that the model uses features that provide nuclei and the shape of the cell and nucleus.

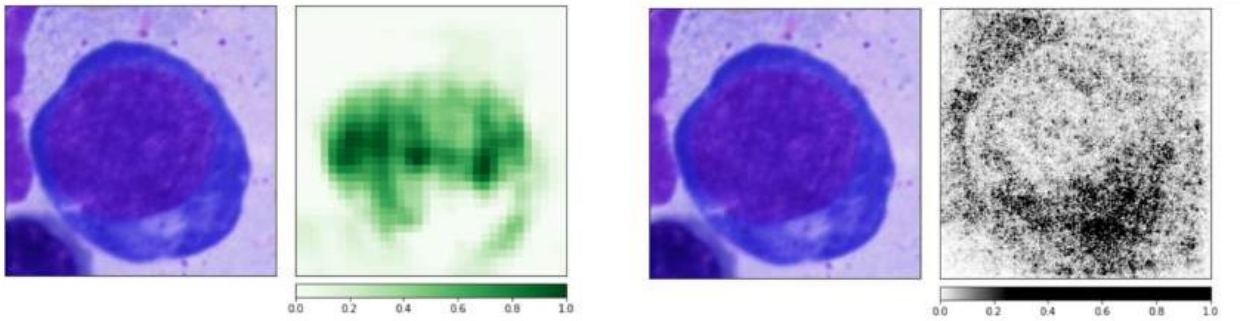


Figure 32. Result of the interpretive learning methods Figure 33 shows the result of the detection algorithm

on the slide images of blood cells and bone marrow cells. It can be seen that the model almost completely determines the position and type of red bone marrow cells.

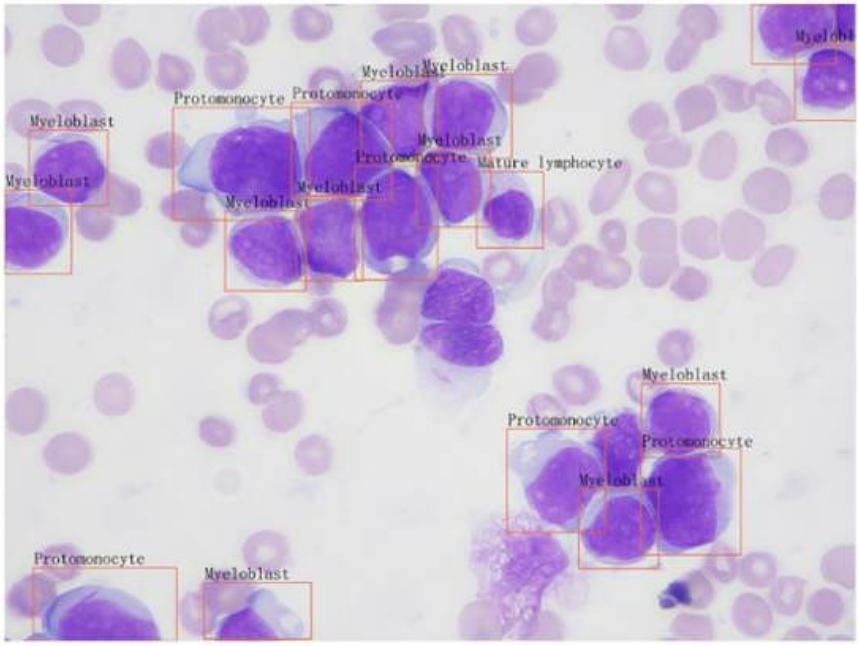


Figure 33. Result of the algorithm for detection of blood cells and bone marrow

CONCLUSION

The main result of this work is the implementation of machine learning algorithms and the creation of a system for detecting and classifying blood and bone marrow cells, as well as counting them in a sample and on an image slide.

In this paper, the approaches for implementing classification, interpretation and detection methods were analyzed and selected. For classification, the following convolutional neural network architectures were chosen: ResNetXt50, MobileNetV3, EfficientNetV2, CoAtNet. For interpretive machine learning, the following were chosen: the Occlusion method and the Integrated Gradients method. The YOLO algorithm was chosen for cell detection in the slides, as this model showed the best results. The selected algorithms and methods were implemented in the paper.

To work with the data correctly, a database with images of blood and bone marrow cells was created. To work with the data, a data schema was developed and a MySQL DBMS was selected to work with the schema and with the images.

In this paper, experiments were conducted to classify blood cells and red bone marrow hematopoietic cells by analyzing their histological images using selected machine learning methods and to conduct experiments to identify the features used by the classification model. In addition, results were obtained on the use of the YOLO model in the cell detection task. The classification and detection accuracy of the best selected model was 91% on 21 types of blood and bone marrow cells. The model was.

selected through a comparative analysis of the different models used in the context of augmentation and transfer learning.

To work with the system, a client-server application and a graphical user interface have been implemented. The user can use this system using a web browser.

The merits of the developed solution are as follows:

1. universality of classifiers, i.e. the possibility of expanding the number of classes and using the developed architecture for other cell types
2. scalability of the system and the ability to run on any platform thanks to Docker
3. large number of classes (21 classes) without loss of accuracy (91%)
4. possibility to trace the attributes to which the neural network "pays attention" due to the methods of interpretive learning Disadvantages and perspective of the developed solution:

1. The need for a productive GPU for new convolutional neural networks training
2. slow output of interpretive learning methods (~2-3 minutes for one image)
3. relatively low classification accuracy. Now it is 91%, i.e. about every 10th image can be misclassified, which is unacceptable in clinical practice. This problem can be solved by using more neural networks with more parameters, as well as by optimizing hyperparameters of neural networks, but all this requires high-performance hardware: GPU, CPU and RAM

REFERENCE LIST

[1] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning second edition*. MIT Press, 2018.

[2] Mehmed Kantardzic, *Data Mining. Concepts, Models, Methods, and Algorithms*. IEEE, 2020.

[3] U. Kose, O. Deperlioglu, and D. Jude Hemanth, *Deep Learning for Biomedical Applications*. 2021. [Online]. Available: https://www.routledge.com/

[4] Nikolenko Sergey, *Nikolenko Deep Learning*. 2018.

[5] A. Glassner, *Deep Learning*. 2020.

[6] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT Press, 2012.

[7] J. Berner, P. Grohs, G. Kutyniok, and P. Petersen, "The Modern Mathematics of Deep Learning," May 2021, [Online]. Available: http://arxiv.org/abs/2105.04026

[8] N. K. Manaswi, *Deep Learning with Applications Using Python*. Apress, 2018. doi: 10.1007/978-1-4842-3516-4.

[9] R. Szeliski, "Computer Vision: Algorithms and Applications," 2010. [Online]. Available: http://szeliski.org/Book/.

[10] C. Matek, S. Krappe, C. Münzenmayer, T. Haferlach, and C. Marr, "Highly accurate differentiation of bone marrow cell morphologies using deep neural networks on a large image data set," Blood, vol. Marr, "Highly accurate differentiation of bone marrow cell morphologies using deep neural networks on a large image data set," *Blood*, vol. 138, no. 20, pp. 1917-1927, Nov. 2021, doi: 10.1182/blood.2020010568.

[11] J. Liu *et al*, "A deep learning method and device for bone marrow imaging

cell detection," *Annals of Translational Medicine*, vol. 10, no. 4, pp. 208-208, Feb. 2022, doi: 10.21037/atm-22-486.

[12] V. Acharya and P. Kumar, "Identification and red blood cell automated counting from blood smear images using computer-aided system," *Medical and Biological*

Engineering and Computing, vol. 56, no. 3, pp. 483-489, Mar. 2018, doi: 10.1007/s11517-017-1708-9.

[13] E. J. Mascha and T. R. Vetter, "Significance, errors, power, and sample size: The blocking and tackling of statistics," *Anesthesia and Analgesia*, vol. 126, no. 2, pp. 691-698, 2018, doi: 10.1213/ANE.0000000000002741.

[14] D. J. Biau, S. Kernéis, and R. Porcher, "Statistics in brief: The importance of sample size in the planning and interpretation of medical research," *Clinical Orthopaedics and Related Research*, vol. 466, no. 9. Springer New York, pp. 2282-2288, 2008. doi: 10.1007/s11999-008-0346-9.

[15] A. Güneş, H. Kalkan, and E. Durmuş, "Optimizing the color-to-grayscale conversion for image classification," *Signal, Image and Video Processing*, vol. 10, no. 5, pp. 853-860, Jul. 2016, doi: 10.1007/s11760-015-0828-7.

[16] J. Rodellar, S. Alférez, A. Acevedo, A. Molina, and A. Merino, "Image processing and machine learning in the morphological analysis of blood cells," International Journal of Laboratory Hematology, vol. Merino, "Image processing and machine learning in the morphological analysis of blood cells," *International Journal of Laboratory Hematology*, vol. 40, pp. 46-53, May 2018, doi: 10.1111/ijlh.12818.

[17] R. B. Hegde, K. Prasad, H. Hebbar, and B. M. K. Singh, "Feature extraction using traditional image processing and convolutional neural network methods to classify white blood cells: a study," *Australasian Physical and Engineering Sciences in Medicine*, vol. 42, no. 2, pp. 627-638, Jun. 2019, doi: 10.1007/s13246-019-00742-9.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," p. 12, 2012, [Online]. Available: http://code.google.com/p/cuda-convnet/

[19] M. M. Alam and M. T. Islam, "Machine learning approach of automatic identification and counting of blood cells," *Healthcare Technology Letters*, vol. 6, no. 4, pp. 103-108, Jul. 2019, doi: 10.1049/htl.2018.5098.

[20] J. Redmon, S. Divvala, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Dec. 2016, vol. 2016-December, pp. 779-788. doi: 10.1109/CVPR.2016.91.

[21] C. M. Schürch, C. Caraccio, and M. A. Nolte, "Diversity, localization, and (patho)physiology of mature lymphocyte populations in the bone marrow," *Blood*, vol. 137, no. 22, pp. 3015-3026, Jun. 2021, doi: 10.1182/blood.2020007592.

[22] M. Abadi *et al*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," Mar. 2016.

[23] F. Seide and A. Agarwal, "CNTK," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2016,
pp. 2135-2135. doi: 10.1145/2939672.2945397.

[24] Y. Jia *et al*, "Caffe: Convolutional Architecture for Fast Feature Embedding," Jun. 2014.

[25] The Theano Development Team *et al*, "Theano: A Python framework for fast computation of mathematical expressions," May 2016.

[26] B. Farnham, S. Tokyo, B. Boston, F. Sebastopol, and T. Beijing, "Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems," 2021.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on*

Computer Vision and Pattern Recognition, Jun. 2009, pp. 248-255. doi: 10.1109/CVPR.2009.5206848.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Dec. 2015.

[29] A. Howard *et al*, "Searching for MobileNetV3," May 2019.

[30] M. Tan and Q. v. Le, "EfficientNetV2: Smaller Models and Faster Training," Apr. 2021.

[31] Z. Dai, H. Liu, Q. v. Le, and M. Tan, "CoAtNet: Marrying Convolution and Attention for All Data Sizes," Jun. 2021.

[32] C. Molnar, *Interpretable Machine Learning Interpretable Machine Learning A Guide for Making Black Box Models Explainable*. 2021. [Online]. Available: https://christophm.github.io/

[33] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," Nov. 2013.

[34] H. S. Oluwatosin, "Client-Server Model," *IOSR Journal of Computer Engineering*, vol. 16, no. 1, pp. 57-71, 2014, doi: 10.9790/0661-16195771.

[35] L. Perez and J. Wang, "The Effectiveness of Data Augmentation in Image Classification using Deep Learning," Dec. 2017.

[36] M. A. Lones, "How to avoid machine learning pitfalls: a guide for academic researchers," Aug. 2021, [Online]. Available: http://arxiv.org/abs/2108.02497