Computer Methods and Programs in Biomedicine 180 (2019) 105020

Contents lists available at ScienceDirect

Computer Methods and Programs in Biomedicine

journal homepage: www.elsevier.com/locate/cmpb

Recognition of peripheral blood cell images using convolutional neural networks☆

Andrea Acevedo[a,b], Santiago Alférez[a], Anna Merino[b,*], Laura Puigví[b], José Rodellar[a]

[a] Department of Mathematics Technical University of Catalonia Barcelona East Engineering School, Spain
[b] Biomedic Diagnostic Center, Clinic Hospital of Barcelona, University of Barcelona, Spain

ARTICLE INFO

Article history:
Received 10 April 2019
Revised 9 July 2019
Accepted 9 August 2019

Keywords:
Deep learning
Fine-tuning
Convolutional neural networks
Blood cell morphology
Blood cell automatic recognition

ABSTRACT

Background and objectives: Morphological analysis is the starting point for the diagnostic approach of more than 80% of hematological diseases. However, the morphological differentiation among different types of normal and abnormal peripheral blood cells is a difficult task that requires experience and skills. Therefore, the paper proposes a system for the automatic classification of eight groups of peripheral blood cells with high accuracy by means of a transfer learning approach using convolutional neural networks. With this new approach, it is not necessary to implement image segmentation, the feature extraction becomes automatic and existing models can be fine-tuned to obtain specific classifiers.

Methods: A dataset of 17,092 images of eight classes of normal peripheral blood cells was acquired using the CellaVision DM96 analyzer. All images were identified by pathologists as the ground truth to train a model to classify different cell types: neutrophils, eosinophils, basophils, lymphocytes, monocytes, immature granulocytes (myelocytes, metamyelocytes and promyelocytes), erythroblasts and platelets. Two designs were performed based on two architectures of convolutional neural networks, Vgg-16 and Inceptionv3. In the first case, the networks were used as feature extractors and these features were used to train a support vector machine classifier. In the second case, the same networks were fine-tuned with our dataset to obtain two end-to-end models for classification of the eight classes of blood cells.

Results: In the first case, the experimental test accuracies obtained were 86% and 90% when extracting features with Vgg-16 and Inceptionv3, respectively. On the other hand, in the fine-tuning experiment, global accuracy values of 96% and 95% were obtained using Vgg-16 and Inceptionv3, respectively. All the models were trained and tested using Keras and Tensorflow with a Nvidia Titan XP Graphics Processing Unit.

Conclusions: The main contribution of this paper is a classification scheme involving a convolutional neural network trained to discriminate among eight classes of cells circulating in peripheral blood. Starting from a state-of-the-art general architecture, we have established a fine-tuning procedure to develop an end-to-end classifier trained using a dataset with over 17,000 cell images obtained from clinical practice. The performance obtained when testing the system has been truly satisfactory, the values of precision, sensitivity, and specificity being excellent. To summarize, the best overall classification accuracy has been 96.2%.

1. Introduction

Peripheral blood (PB) is a fluid that circulates through the blood vessels and contains three main cell types suspended in plasma: erythrocytes (red blood cells), leukocytes (white blood cells) and platelets (thrombocytes). Specifically, leukocytes are nucleated cells that play an important role in defense against infections. They are classified as granulocytes (segmented and band neutrophils, eosinophils and basophils), lymphocytes and monocytes [1]. Granulocytes represent 54–62% of the total leukocytes in PB. Its nucleus is divided into segments (from 2 to 5) joined by a thin filament of nuclear material. In the band neutrophils the nucleus is not segmented and usually the normal percentages in blood are between 0 and 6%. Eosinophils constitute 1–3%, basophils 0–1%, lymphocytes 25–33% and monocytes 3–10% of the leukocytes circulating in PB [2]. In several situations (infections, regenerative anaemias or others) it is possible to find in PB some immature granulocytes

☆ This document is a result of the research project funded by the Ministry of Science, Innovation and Universities. Project DPI 2015-64493-R.

* Corresponding author.
E-mail address: amerino@clinic.cat (A. Merino).

https://doi.org/10.1016/j.cmpb.2019.105020
0169-2607/

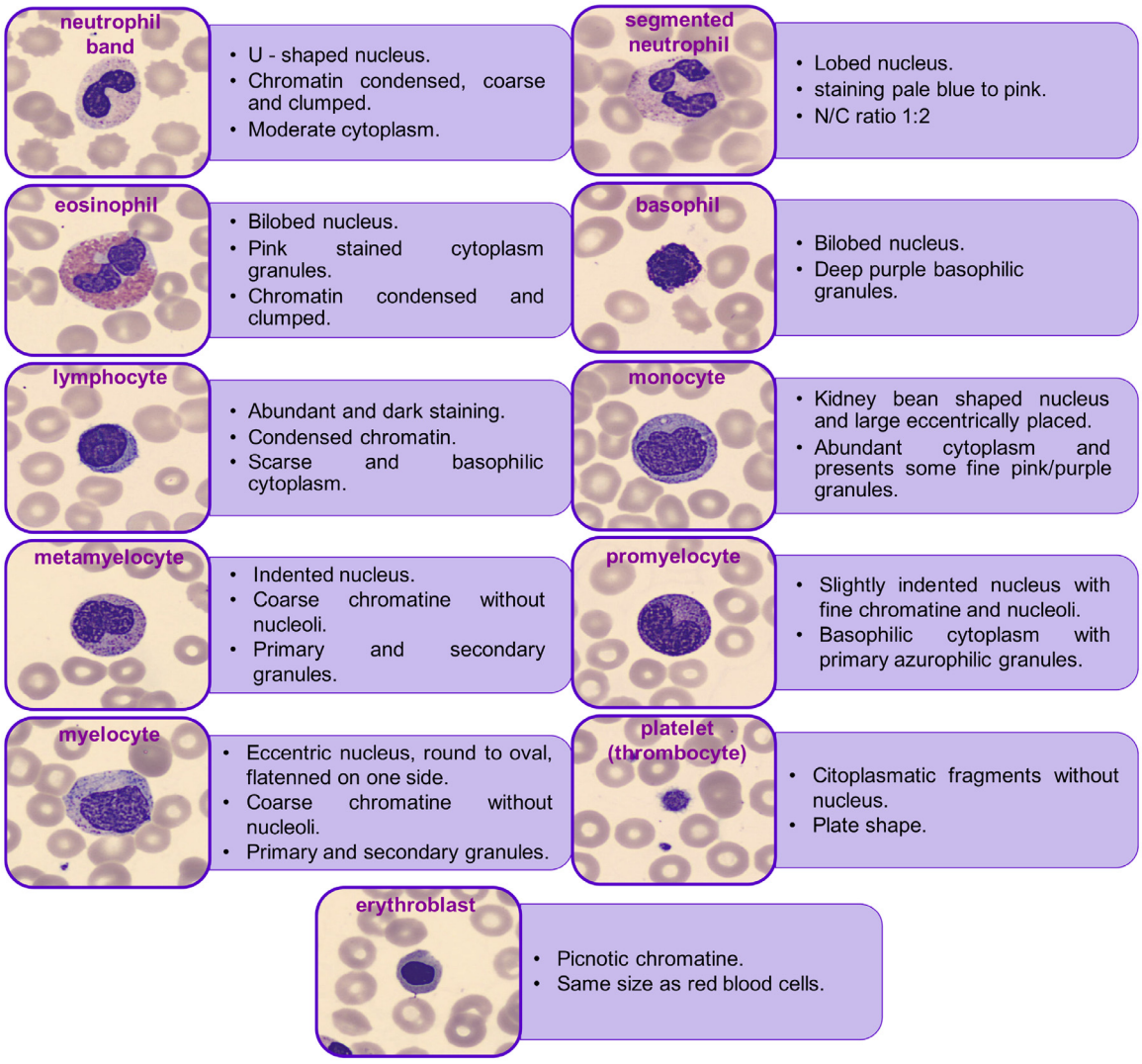


Fig. 1. Images of different types of leukocytes: neutrophil, eosinophil, basophil, lymphocyte, monocyte and immature granulocytes (metamyelocyte, myelocyte and promyelocyte), erythroblasts and platelets (thrombocytes) that can be found in peripheral blood and some of their most important morphological characteristics.

(IG) (promyelocytes, myelocytes and metamyelocytes) or erythroid precursors, such as erythroblasts. Examples of these types of cells and some of their most important morphological characteristics are shown in Fig. 1.

The morphological analysis of the different types of cells in PB is the starting point for the diagnostic approach of more than 80% of the hematological diseases. Currently, in the market there are several systems for the automated pre-classification of normal leukocytes [3–5]. These analyzers are based on a motorized microscope to acquire the images from a blood smear and algorithms that perform the pre-classification of the blood cell images. These systems have a high cost due to the complexity of the acquisition and the classification steps. Usually, the classification process using image analysis requires different steps, such as segmentation, feature extraction, selection and classification of the images using algorithms for pattern recognition [6–8]. The segmentation is considered the most critical step in the process, mainly due to the complex morphology and subtle differences in the wide class of blood cells. It has a direct effect on the accuracies of the classification results [9,10]. Trying to overcome some of the limitations of traditional computer vision techniques, new methods known as deep learning have emerged. This is a sub-field of machine learning, whose approach is learning representations from data using models called neural networks, which are structures arranged in layers [11] containing descriptive features. The depth of the model depends on how many layers make up the network. The most important characteristic of these networks is that engineers do not design these layers of features. Conversely, they are learned from data using different learning algorithms in which the segmentation of the images is not necessary. In recent years, different studies using one type of deep neural network known as convolutional neural networks (CNN) have shown good results in image classification [12].

The main objective of this work is to design a CNN-based system for the automatic classification of eight classes of peripheral blood cells: neutrophils, eosinophils, basophils, lymphocytes, monocytes, immature granulocytes (metamyelocytes, myelocytes and promyelocytes), platelets and erythoblasts. To our knowledge, a convolutional network has not been designed and implemented before for the recognition of this high number of blood cell types achieving an overall accuracy over 96% as obtained in this work. The practical clinical interest is twofold: first, they include the family of normal leukocytes and the cells more frequently observed in infections and regenerative anaemia; and second, the experience gained in this work will be the basis for further extensions of CNN to classify the broader classes of abnormal cells related to acute

Table 1
Dataset organized in eight classes: neutrophils, eosinophils, basophils, lymphocytes, monocytes, immature granulocytes, erythroblasts and platelets.

Cell type	Total of images by class	%
neutrophils	3,329	19.48
eosinophils	3,117	18.24
basophils	1,218	7.13
lymphocytes	1,214	7.10
monocytes	1,420	8.31
immature granulocytes (metamyelocytes, myelocytes and promyelocytes)	2,895	16.94
erythroblasts	1,551	9.07
platelets (thrombocytes)	2,348	13.74
Total	17,092	100

leukemia and lymphoma, such as myeloblasts, lymphoblasts, abnormal B or T lymphoid cells and dysplastic cells.

In the paper, we present the full training process and experimental test results considering two different architectures of CNN's already pre-trained with the ImageNet dataset [13]: Vgg-16 [14] and Inceptionv3 [15]. In a first step, a transfer learning approach is followed using the networks as feature extractors [16] and then, these features are used to train a support vector machine (SVM) classifier. In an alternative second stage, both networks are modified and tuned with our dataset of PB images to obtain two end-to-end models for classification of the eight classes of blood cells. This set includes more than 17,000 images obtained from blood smears of real patients, being among the largest sets in the literature to train a blood cell classifier.

After a summary of the most related work, the paper is organized as follows. Section 2 describes the data set used in the study and Section 3 describes the CNN methodological background. The proposed classification system is presented in Section 4, giving details on the architectures and the training procedures. The experimental assessment is described in Section 5 and their results and practical issues are then discussed in Section 6. Finally, Section 7 highlights the most relevant conclusions of this work.

1.1. Related work

The automatic classification of PB cells has been studied for several years in order to create tools that support the work on the clinical laboratories. The traditionally used technique is mainly based on the segmentation and extraction of handcrafted features that are then used together with pattern recognition tools for the classification of the different blood cells. In [17,18] the authors present a review of the state-of-the-art methods of leukocyte segmentation, feature extraction and classification published in the last two decades. As a result, the authors identified several aspects of improvement with respect to the classical approach, such as, the accuracy percentages obtained so far, and the lack of datasets containing a greater number of images to consider sufficiently reliable the classification models obtained. Due to the development of more efficient algorithms and methods related to machine learning, their use have become popular, more specifically those of deep learning and neural networks in the medical field [19–21]. In [22] the authors performed the classification of five types of normal PB cells using two different approaches: first, the traditional segmentation, extraction and classification steps using a SVM and principal components analysis (PCA) for feature selection; and second, the novel approach using a CNN and the entire image as input. Better performances (85%) were obtained with the CNN. In [23] Shahin et al. also proposed the use of CNN for the classification of five types of PB cells. They performed fine-tuning of Alexnet and LeNet-5 networks achieving accuracies of 91.2% and 84.9% respectively. They also proposed a trained model with a dataset of 2,551 images for the classification of five types of PB cells and reported a maximum testing accuracy of 96%.

2. Materials

A dataset of 17,092 digital images of normal PB cells was obtained from blood smears and stained with May Grünwald-Giemsa, which were compiled during the daily work in the Core Laboratory at the Hospital Clinic of Barcelona. The images were acquired using the analyzer CellaVision DM96 (RGB, 360×363 pixels).

As each image was properly identified and labeled by the clinical pathologists, the dataset was organized in eight groups or classes: neutrophils, eosinophils, basophils, lymphocytes, monocytes, immature granulocytes (including myelocytes, metamyelocytes and promyelocytes), erythroblasts and platelets. Table 1 details the number and percentages of the different types of blood cell images.

In order to balance the dataset, we performed data augmenting to increment the number of images and obtain the same number per group. Individual images were modified applying randomly transformations as vertical and horizontal flips, rotation from 0 to 60 degrees and pixel value standardization [24]. Up to four different versions of each of the original images were obtained from the groups with less number of images (basophils and lymphocytes) as shown in Fig. 2.

Regarding the network architectures, we selected two different architectures of CNN's pre-trained with other images different from blood cells, named Vgg-16 [14] and Inceptionv3 [15]. The networks were trained using Keras and Tensorflow software libraries [24] and a Nvidia Titan XP GPU.

3. Methodological background

3.1. Overview

The main objective of this work is to design a system for the automatic classification of eight classes of peripheral blood cells using CNNs as the background tool for processing and extracting information. Images of peripheral blood cells are the input to the system and the output should be the correct label indicating the class of each cell image. Fig. 3 shows its general structure. We considere two alternative approaches to design the classifier. In the first case, the CNN is used as a feature extractor and then a Support Vector Machine (SVM) is trained with these features. This serves as an exploratory search to find out if the use of convolutional neural networks is suitable to solve our classification problem. The second approach is based on the concept of fine tuning, where the CNN is re-trained with peripheral blood cell images to obtain an end-to-end classifier. Our hypothesis is that if the results of the classification are good with the approach one, they may improve when implementing the second approach. In any case, the system's output is a set of eight probabilities (one per class), such

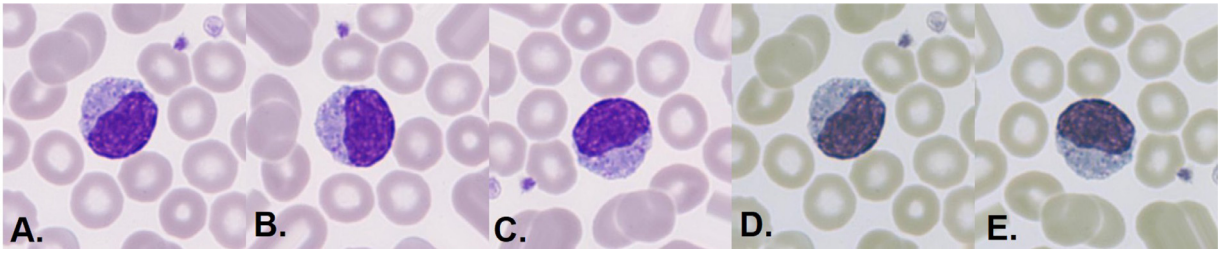


Fig. 2. Examples of some of the transformations applied to augment the number of training images. The lymphocyte (A) is the original image, (B) is a rotated version, (C) is a flipped and rotated image, (D) is a version with pixel values standardized and (E) is pixel standardized and a rotated image.

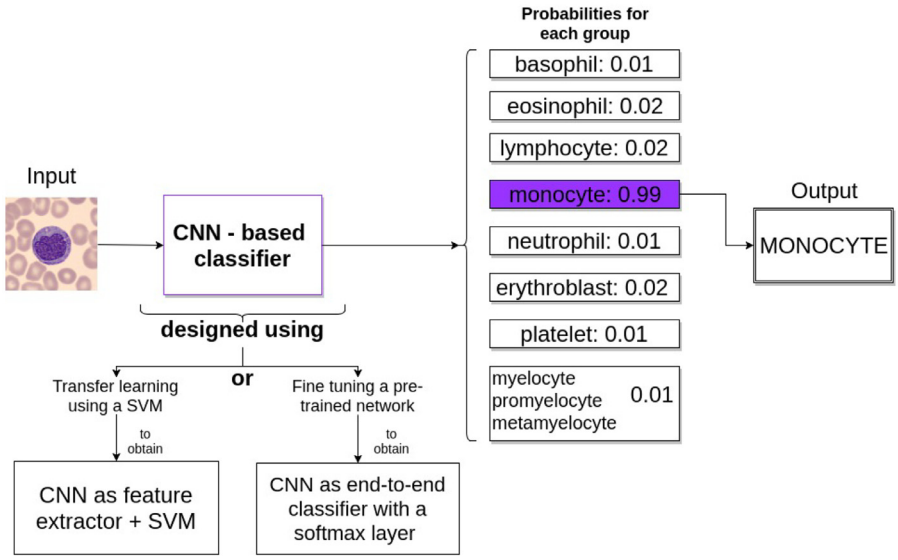


Fig. 3. Overall structure of the system for the classification of eight types of peripheral blood cells using convolutional neural networks as a principal tool for processing and extracting information from the images. *SVM: support vector machine. *CNN: Convolutional neural network.

that the highest probability indicates the group which the input image belongs to.

The purpose of using these two approaches is to take advantage of the knowledge of a CNN previously trained with a large dataset in a specific domain and use it for the classification of images in a different domain [25]. This helps to overcome the deficit of training examples and serve as an effective weight initialization technique for these deep neural networks [26].

CNN's are multilayered networks that can learn complex and high-dimensional features from a large collection of examples. Unlike traditional computer vision models where hand-crafted features are extracted and used to train a classifier, in this approach the model can learn to extract relevant features and use a fully connected structure for classification purposes. The use of CNNs for image classification presents many advantages, such as the efficient use of memory and the capacity of processing large images due to the use of connections of single neurons to specific portions of an image. In addition, these trained networks present invariance with respect to translation or distortions of the input image and can take advantage of the high correlation between nearby pixels of an image and combine them to identify simple patterns as edges and corners or more complex features. Based on the model presented by Lecun et al.in [27], Fig. 4 shows the basic layered structure of a CNN. The first layer is the input, which reads the pixels of the image. The following is the convolutional layer, where a number of convolution operations are performed over the image to extract the features. Its output is a set of feature maps. A CNN may contain one convolutional layer or a group of consecutive convolutional layers called convolutional blocks. After a convolutional layer or block, a pooling layer is usually placed to reduce the dimensions of the feature maps. Subsequent layers are combinations of convolutional and pooling layers to extract more complex features. After the convolutions and poolings, a set of fully connected layers is placed to combine all the features. Finally, the output is calculated in the last layer by applying a softmax operation to obtain the result of the classification as probability values.

3.2. Architectures for peripheral blood cells classification

As it was stated before, we implemented two CNNs as a base for the design of our classification system: Vgg-16 and Inceptionv3. The Vgg-16 is a deep convolutional neural network for object recognition designed and trained by the Visual Geometry Group (Vgg) of the Oxford University in 2014 [14]. Despite its around 160 million of parameters, this network is known for its good performance and simple architecture. On the other hand, the network Inceptionv3 was developed by the Google research team in 2015 and overcomes the high computational cost of convolutional networks as Vgg, introducing more complexity to the model by implementing convolutions in parallel, organized in modules called Inception [15]. This resulted into a more complex architecture but with less parameters, diminishing the execution time and making possible to implement the Inception network in cases where memory or computational capacity is limited.

The subsequent sections describe the main components of the system that we trained for PB cell classification.

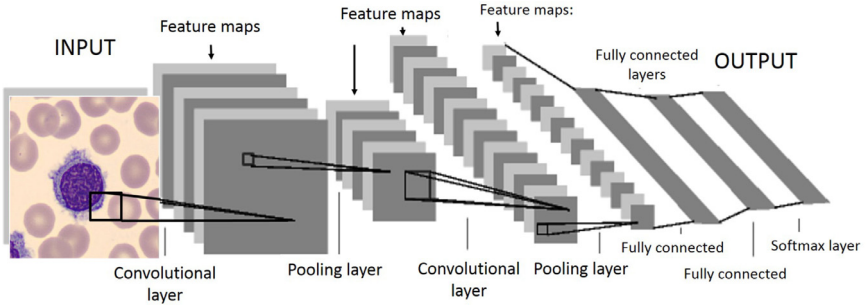


Fig. 4. Basic structure of a convolutional neural network [27].

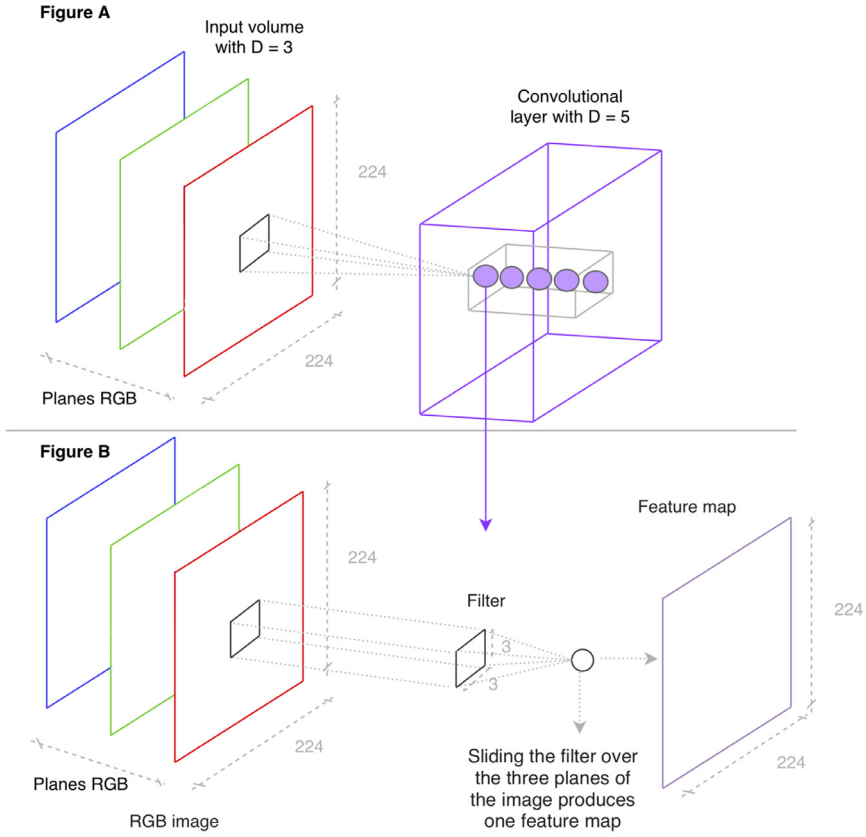


Fig. 5. Convolutional layer structure. Figure A presents the local connectivity of a neuron with a portion of the input image. Figure B is equivalent to Figure A but planes contain filters which are convoluted with the input image. The result of the convolution of each filter is a feature map [29].

3.2.1. The convolutional layer

The convolutional layer is the most important component in a CNN. It is a structure that transforms an input to an output through a convolution-based procedure. Both input and output are seen as a volume structure. The input volume is composed of planes with fixed height and width. The number of planes defines the depth (D). As illustrated in Fig. 5(A), when the input is a RGB image, we have $D = 3$ and each plane contains the corresponding component pixel matrix.

The convolutional layer has also a volume structure with a number N of planes, and each plane is defined by a number of filters (kernels) of small size (F). A typical case is, for instance, $F = 3$ so that the filter is an 3×3 matrix with weights. Let us consider the kth input plane with value matrix X_k. Consider also the corresponding filter in the jth plane of the convolutional layer, whose parameters (weights) are arranged in a matrix W_{kj}. The convolution of this filter sliding over the kth input plane produces a two-dimensional matrix $W_{kj} * X_k$. Fig. 5(B) illustrates an example of the convolution of a filter of size 3×3 with the input image. These matrices are summed for all the input planes and biased by a matrix B_j and the result is transformed by an activation function f as follows [28]:

$$Y_j = f\left(B_j + \sum_{k=1}^{D} W_{kj} * X_k\right); \quad j = 1, \ldots, N \tag{1}$$

The resulting matrix Y_j is called feature map. The final output of the convolutional layer is a volume whose planes are the N feature maps.

The rectifier linear unit (ReLU) is the activation function mostly used in the current CNN architectures. It converts all negative input values to zero while keeping all positive inputs the same, being defined as follows:

$$f(x) = max(0, x) \tag{2}$$

Table 2
Architecture of Vgg-16 (left) and Inceptionv3 (right).

Layers	Filter size	Number N of planes (filters)	Layers	Filter size	Number N of planes (filters)
2 × conv2D	3 × 3	64	conv2D	3 × 3	32
Max pooling: F = 2 and S = 2			conv2D	3 × 3	32
2 × conv2D	3 × 3	128	conv2D	3 × 3	64
Max pooling: F = 2 and S = 2			Max pooling: F = 3 and S = 2		
3 × conv2D	3 × 3	256	conv2D	1 × 1	80
Max pooling: F = 2 and S = 2			conv2D	3 × 3	192
3 × conv2D	3 × 3	512	Max pooling: F = 3 and S = 2		
Max pooling: F = 2 and S = 2			3 × inception modules		
3 × conv2D	3 × 3	512	4 × inception modules		
Max pooling: F = 2 and S = 2			2 × inception modules		
4096 nodes fully connected			Average pooling: F = 8		
4096 nodes fully connected			1000 nodes fully connected with softmax activation		
1000 nodes fully connected with softmax activation					

This ReLu function is applied to the output of a convolutional layer, and it helps to increase the learning speed of the network and the classification accuracy.

3.2.2. The pooling layer

After each convolutional layer or block of layers, a pooling layer is usually placed to reduce the feature map size. This reduction preserves important information while eliminating irrelevant details about the position of the features within the input images, reducing the sensitivity of the model to shifts and distortions.

Pooling is defined essentially by two parameters: (1) the size (F) of the squared portion of the feature map whose pixels are pooled in a single value, and (2) the stride (S), which indicates the steps along both width and height that pooling moves to cover the full map.

The Vgg-16 has in total five pooling layers with parameters $F = 2$ and $S = 2$. This means that the pooling operation is applied every 2×2 pixels with steps of two pixels over the whole feature map, keeping the maximum value of each frame (max pooling). The Inceptionv3 case is different because it has many more pooling layers. This network has five max pooling layers, four of them with $F = 3$ and $S = 2$ and one with $F = 3$ and $S = 1$. In addition, this network has eight average pooling layers with F=3 and S=1 and a last average pooling layer with F=8 and S=1. Unlike max pooling, the average pooling substitutes a frame of pixels by its average value.

Table 2 presents a summary of the implemented architectures of Vgg-16 (left) and Inceptionv3 (right). The Vgg-16 network is composed by 13 convolutional layers organized in blocks. Each block may contain two or three convolutional layers, whereas the Inceptionv3 is composed by five convolutional layers and 9 Inception modules [15], which contain between four and 10 convolutional layers each. The last layers of both networks are fully connected with 1000 nodes for the 1000 class classification of the Imagenet contest.

3.2.3. The fully connected and output layers

After the successive convolutions and pooling layers, a set of feature maps is obtained. They are a useful representation of the input images. At this point, it is possible to take two approaches to use these features. The first is to use them to train a different classifier such as SVM, k-NN, decision trees and others. The second possibility is to couple a block of fully connected layers to learn non-linear combinations of the features and to perform the final classification. Each neuron of a fully connected layer performs an input-output operation defined as follows:

$$Z = f\left(b + \sum_{i}^{m} \omega_i x_i\right) \tag{3}$$

where x_i are the input features, ω_i are the neuron weights, m is the number of features, and b is a bias parameter. Function f is the same activation function described in Eq. (2). The output Z is a new feature that serves as input to the next fully connected layer. The output layer is the last layer of the network. It is also a fully connected configuration but the activation function used is a softmax expressed as follows:

$$f(y_j) = \text{softmax}(y_j) = \frac{e^{(y_j)}}{\sum_{k=1}^{8} e^{(y_k)}} \quad for\ j = 1, \ldots, 8 \tag{4}$$

This function is applied to the result of each node of the output layer to obtain the results as a probability distribution. In our design, we have eight output neurons because we aim to classify eight classes of cell images.

4. Design and training of the proposed system

In the next sections we describe the two approaches taken to design and train our classifier: the transfer learning using a pre-trained CNN as a feature extractor and the fine tuning of a pre-trained CNN to build an end-to-end classifier.

4.1. Transfer learning using a CNN as feature extractor

In this approach the dataset was split into two subsets: training and testing. We selected 15,173 (88%) images for the training set and 1,919 (12%) images for the test set. The training set was augmented by applying random transformations, as explained in Section 2, to obtain a new training set with a total of 4,608 images per class. To extract the features of the training set, we used the Vgg-16 and Inceptionv3 with their original weights of the Imagenet contest. From Vgg-16, the fully connected layer and the output layer were removed in order to extract 4,096 features for each image. From Inceptionv3, the fully connected and the output layers were removed to extract 2,048 features per image.

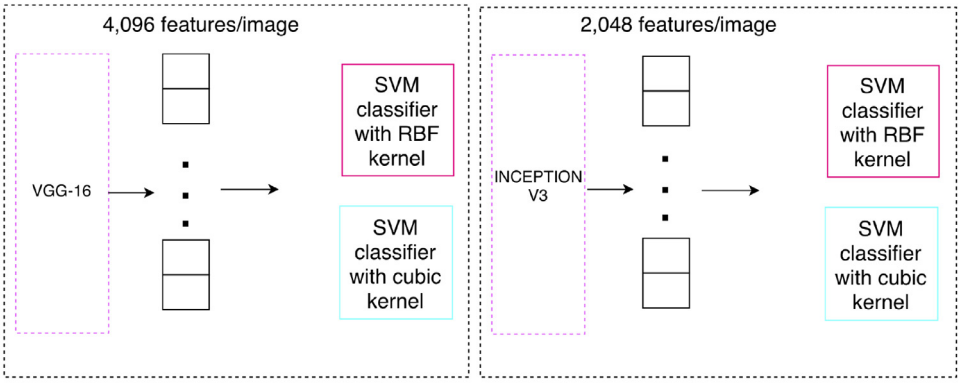


Fig. 6. Transfer learning. 4,096 features per image from the Vgg-16 and 2,048 features per image from the Inceptionv3 were extracted.

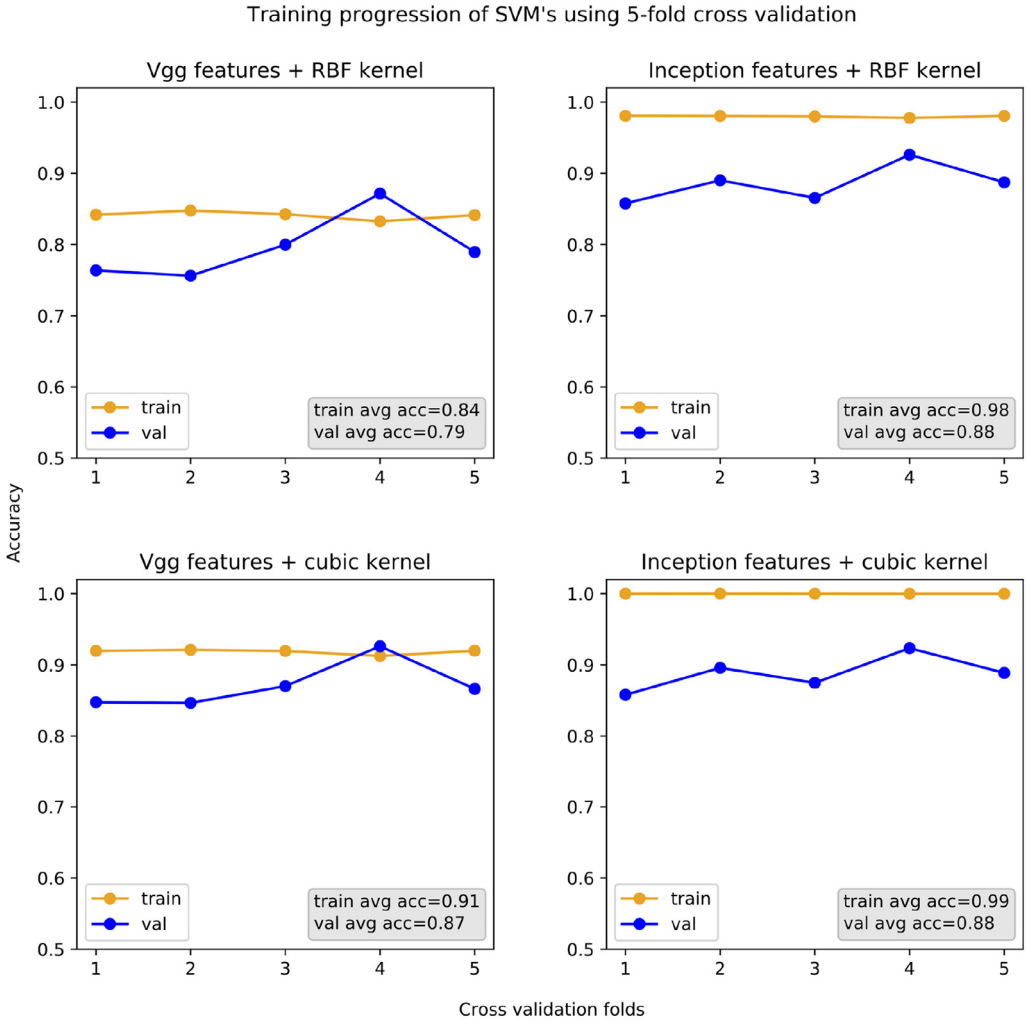


Fig. 7. Training and validation accuracies progression for the training of four clasifiers: (1) use of Vgg features to train a SVM with RBF kernel, (2) use of Inception features to train a SVM with RBF kernel, (3) use of Vgg features to train a SVM with cubic kernel and (4) use of Inception features to train a SVM with cubic kernel.

We selected two types of SVM kernels : (1) a radial basis function (RBF) kernel, which we have implemented in previous works with satisfactory results [10]; and (2) a cubic polinomial kernel. See Fig. 6.

To train the SVM's, we implemented a 5-fold cross validation, in which the training is performed through five iterations where the training set is divided automatically into five subsets. In each iteration, one subset is used as a validation set and the other four are used for training. The accuracy of the model is averaged over all the iterations. The training was implemented using Keras and Scikit-learn software libraries [30].

In Fig. 7 we present the training and validation accuracies for each iteration of the resulting four models. In view of these results, we decided to select the SVM with cubic kernel trained with both Vgg-16 and Inceptionv3 features. Validation average accuracies were 0.87 and 0.88, respectively. The

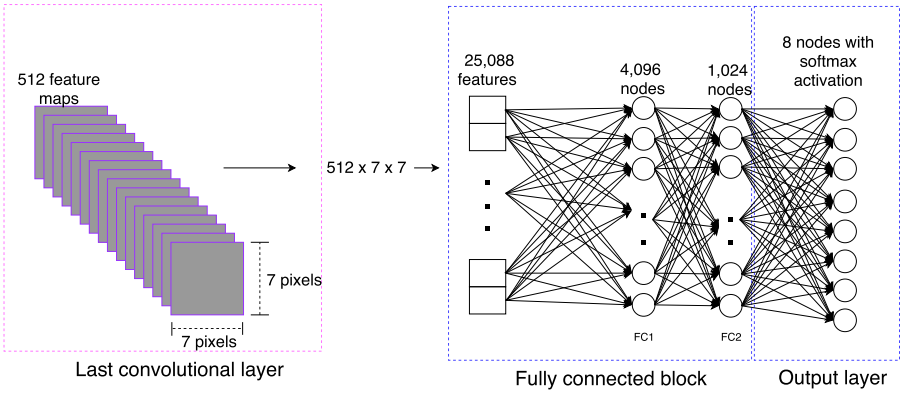


Fig. 8. Fully connected block and output layer designed for the Vgg-16.

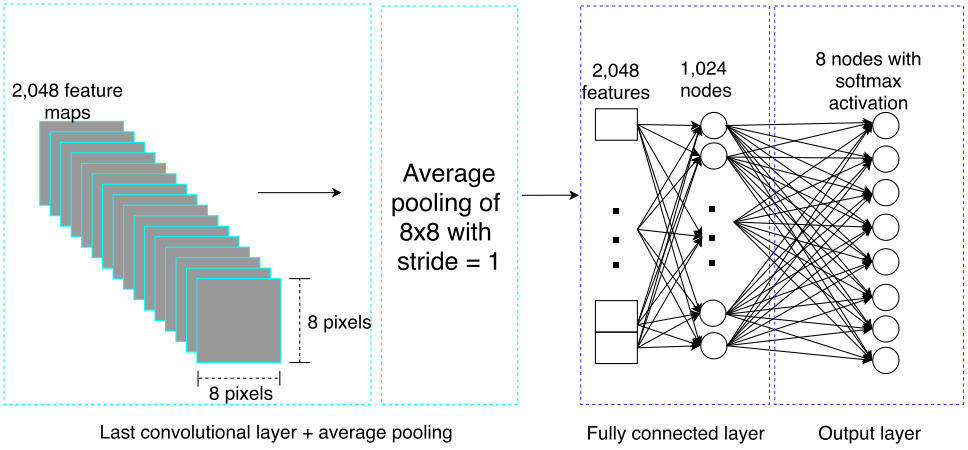


Fig. 9. Fully connected block and output layers designed for the Inceptionv3.

resulting models will be used later on in the experimental assessment.

4.2. Transfer learning using fine-tuning

The second approach, fine-tuning, is a type of transfer learning in which a CNN that has been trained for another task is modified, re-training some layers and adapting them to new data [31]. This approach is usually implemented with CNNs like Vgg-16 and Inceptionv3, which have large quantity of parameters because training them from scratch (weights initialized randomly) can affect their ability to generalize and may result into low accuracies or overfitting [32]. As the deeper layers of a CNN extract more dataset-specific features [33], the fine-tuning is usually performed over these layers to modify the representations accordingly to the new classes.

In this second approach we split the former training set, used for the first approach in Section 4.1, into a training set with 11,077 images (64%) and a validation set with 4,096 images (24%). The test set with 1,919 images (12%) remained the same.

A total of 512 feature maps were obtained from the Vgg-16 and converted to an array of 25,088 features as illustrated in Fig. 8. These features are the input to the fully connected layers, the first layer with 4,096 nodes and the second with 1,024 nodes. To obtain the final classification, a third fully connected layer with eight nodes was configured, one node for each class.

The design of the fully connected layer for the Inceptionv3 is shown in Fig. 9. A number of 2,048 feature maps of 8×8 pixels were obtained from Inceptionv3 convolutional blocks and were converted to an array of 2,048 features using average pooling. These features are fed to a fully connected layer of 1,024 nodes and the output goes to a final layer of eight nodes, one by each class.

The design of the fully connected layers of both networks was done empirically, starting from the original design and changing the number of layers and nodes to obtain better accuracies. The objective was to design a customized fully connected block maintaining a balance between performance and training time.

4.2.1. Training using the backpropagation algorithm

During the training stage, the neural network classifies each training image based on the features extracted. After each image is classified, the difference between the manual label assigned by the pathologists (ground truth) and the prediction made by the network is measured using a loss function. For this case, it was implemented a categorical cross-entropy function defined as follows:

$$L = -\sum_{j=1}^{n} y_j \log(\hat{y}_j) \tag{5}$$

where $\hat{y}_j$ is the probability of the predicted label and depends of the weights (ω) and biases of the network, y_j is the ground truth and n is the number of classes. During the training, the goal is to minimize the loss function calculating its gradient (∇L) at certain point and updating the weights and biases of the network iteratively. The gradient is calculated backwards through the network, computing first the gradient of the weights of the final layer and finishing with the gradients of the weights of the first layer. The computations from one layer are reused in the computation of the gradient of the previous layer. The weights are updated according

Table 3
Validation accuracies of Vgg-16 (*values in italic*) and Inceptionv3 changing two parameters: (1) The number of training images per group and (2) the number of convolutional blocks trained. The accuracies values in bold represent the models selected for testing.

images/group \ blocks trained	1	2	4	6	8
512	*92*	*93.7*	*93.5*		
	92.7	92	91.8	92.7	91
1,024	*93*	*95*	*95*		
	92.3	93	93.2	94	94
2,048	*93.2*	*95*	*95.5*		
	93	94	94	95	95
4,096	*94.5*	*95*	***95.4***		
	93	94.7	**95.2**	96	96

to a hyperparameter called learning rate (α) with the following relationship:

$$\omega := \omega - \alpha \nabla L \tag{6}$$

The learning rate is an empirical parameter that is selected based on experience. The lower the learning rate, the slower is found the minimum of the loss function. We selected a small value of learning rate ($\alpha = 10^{-4}$) to ensure the result convergence and train the model in an acceptable time. The training was performed through 100 iterations (epochs). In each epoch, all the images from the training and validation set are passed through the network and the training validation accuracy and loss are obtained. Although different CNN's architectures may require different number of training epochs, we selected a value of 100 because both models reached accuracy plateau for this value. Our target performance parameter was the accuracy, so that we saved the models that obtained the highest accuracy during the training. This is the number of images correctly classified over the total number of images.

Two important hyperparameters were selected: the number of training images and the number of convolutional blocks to train. To do this, we carried out several tests in which we varied: (1) the number of training images per group: 512, 1,024, 2,048 and 4,096 images; and (2) the number of convolutional blocks fine tuned: last block, last two, last four, last six and last eight blocks, respectively The validation accuracies are presented in Table 3. In the case of Vgg-16, the last 1, 2 and 4 blocks were trained (values in italic). From the results, it can be observed that the best models, with higher accuracies, are those whose last four and last six blocks were trained with 2,048 and 4,096 images per class.

Based on the results in Table 3, we selected the models with the last four convolutional blocks tuned using 4,096 images per class (accuracies in bold). Fig. 10 shows the training and validation progression of the accuracy and loss for these two models. In addition, the training and validation maximum accuracy values (train max acc, val max acc) and training and validation minimum loss values (train min loss, val min loss) are presented.

For Vgg-16, the maximum validation accuracy (0.96) was obtained in epoch 80. For Inceptionv3, the maximum accuracy (0.95) was obtained in epoch 78. The weights obtained in these epochs determined the final models. It should be noted that the training and validation curves are close and present a similar performance. This means that it can be assured that the tuned models do not present overfitting.

The fine-tuning and all tests were performed using Keras and Tensorflow software libraries.

In the next section, we present the results of a test assessment of the selected models, including the confusion matrices and some performance parameters.

5. Experimental assessment

The objective of this work was the classification of eight classes of peripheral blood cells with high accuracy. For this assessment, we used the test set of images that were not used for training. For each experiment, a confusion matrix was calculated to obtain performance parameters as follows:

$$Sensitivity = \frac{TP}{TP+FN}; \quad Specificity = \frac{TN}{TN+FP}; \quad Precision = \frac{TP}{TP+FP} \tag{7}$$

where for each class:

TP = True positive. Value in the principal diagonal of the target class in the confusion matrix.

TN = True negative. Is the sum of all rows and columns excluding the target class column and row.

FP = False positive. Is the sum of the column values of the target class, excluding the true positive.

FN = False negative. Is the sum of the row values of the target class, excluding the true positive.

Since this is a multi-class classification, we took a "one vs all" approach where the performance parameter is calculated for each class, considering the target class as the positive and the rest of the classes as negative We also calculated the sensitivity, specificity and precision average for each model and presented them in a bar plot along with the standard deviation for a general overview of each model performance.

5.1. Results of transfer learning using a CNN as feature extractor

Fig. 11 presents the confusion matrices for the classification of the test set using the SVM classifiers with cubic kernel and trained with Vgg-16 features (model a) and Inceptionv3 features (model b).

The overall test accuracies were 87.4% for model (a) and 90.5% for model (b). The principal diagonal of the matrix contains the true positive rates for each group (% of images correctly classified). The rows of the matrices are true values and the columns are the predicted values. For example, the confusion matrix for the results using a SVM trained with Vgg-16 features shows that 99.22% of the images corresponding to platelets were correctly classified.

With respect to the true positive rates for each class using the model a, we obtained the highest values for platelet and neutrophil classes (>98%). Other classes as eosinophil, lymphocyte, basophil, inmature granulocyte (ig) present values greater than 85%. Monocyte and erythroblast present the lower TPRs with 76.17% and 76.56% respectively. The overall accuracy with model (a) is 87.4%. Model (b) presents similar performance to model (a) but, in general, with higher true positive rates and consequently with higher overall accuracy (90.5%).

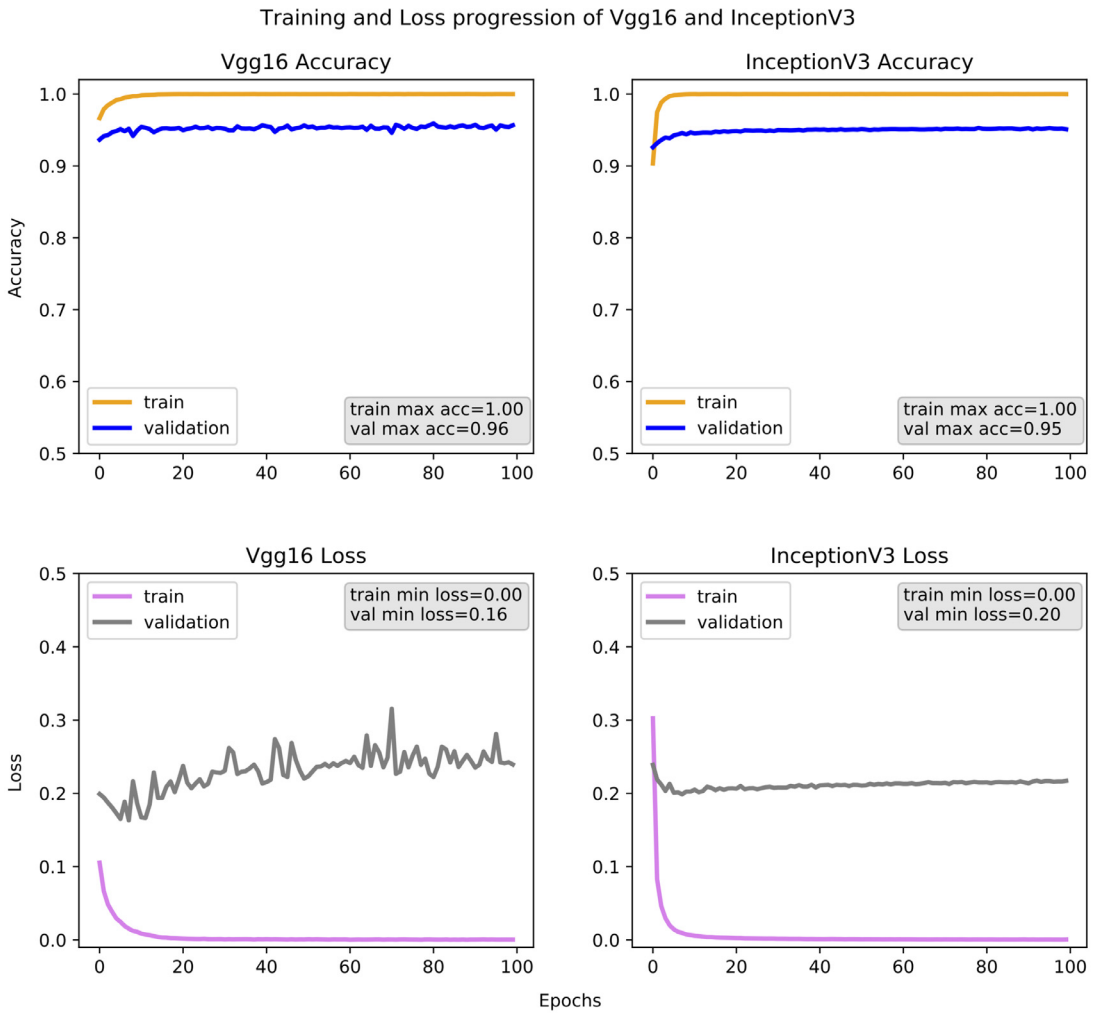
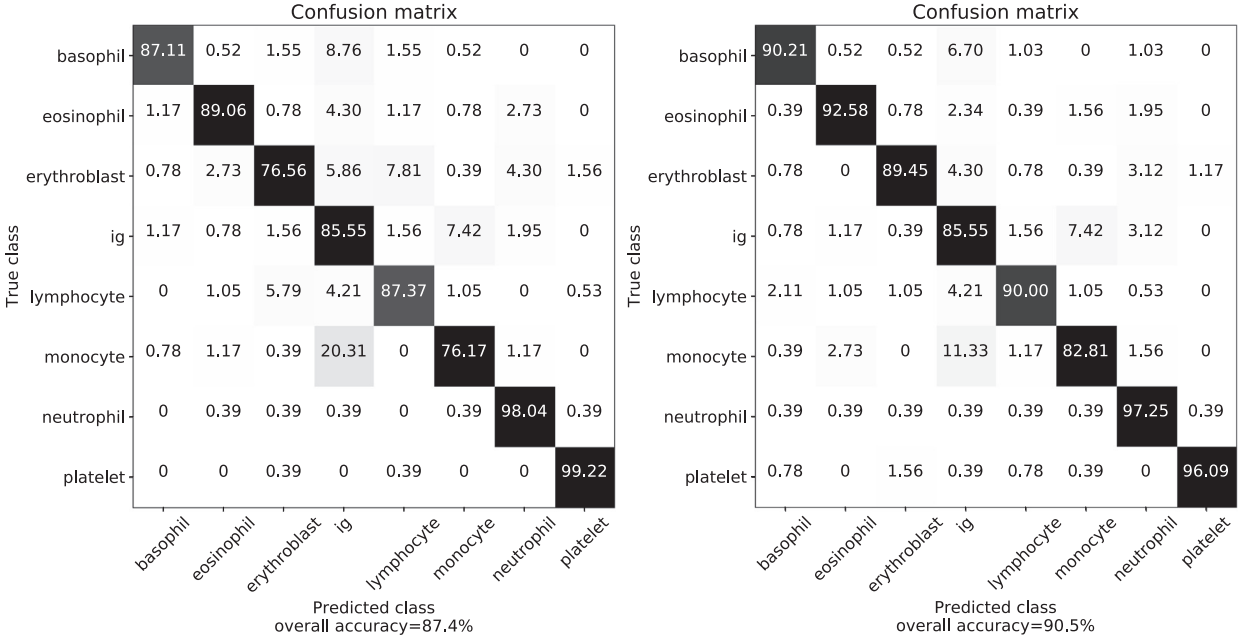


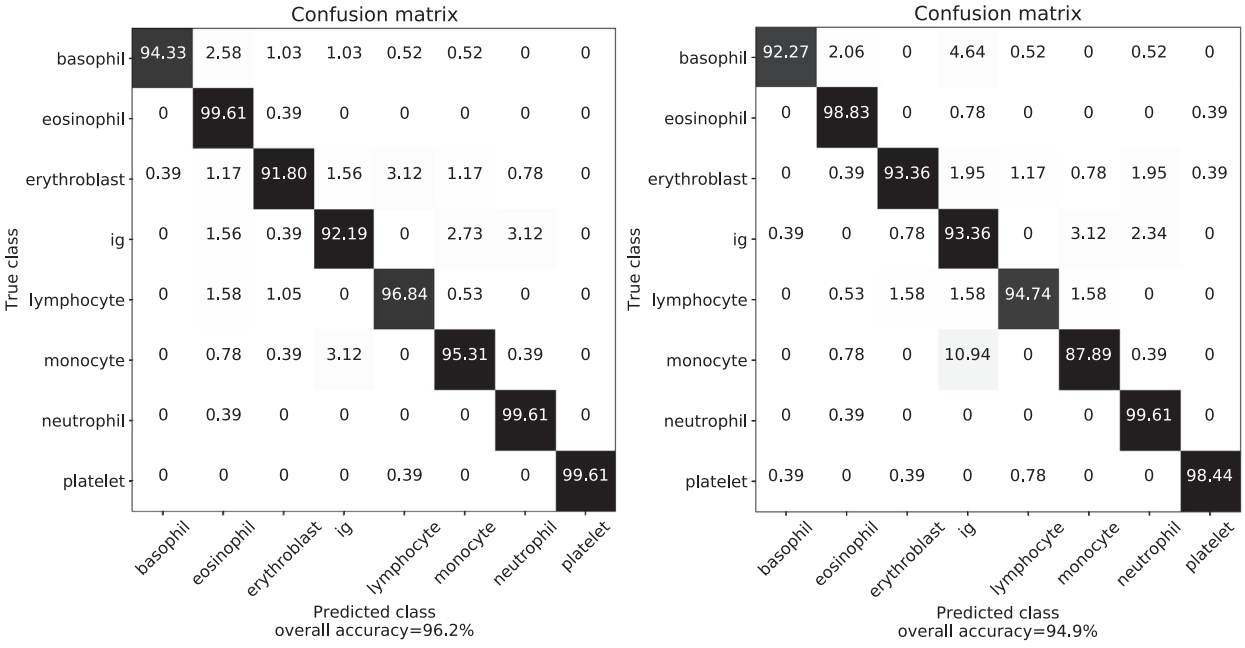
Fig. 10. Progression of the accuracy and loss during the fine tuning of the last four convolutional blocks of Inceptionv3 and Vgg-16. Each network was trained for 100 epochs.



(a) SVM trained with Vgg-16 features.

(b) SVM trained with Inceptionv3 features.

Fig. 11. Test assessment confusion matrices of SVM classifiers with cubic kernel. Values are in percentage.



(a) Vgg-16 based model

(b) InceptionV3 based model

Fig. 12. Confusion matrices of the test assessment of tuned models based on Vgg-16 and Inceptionv3. Rows represent the true values and columns represent the predicted values. The principal diagonal (grey) are the true positive rates. All values are in percentages.

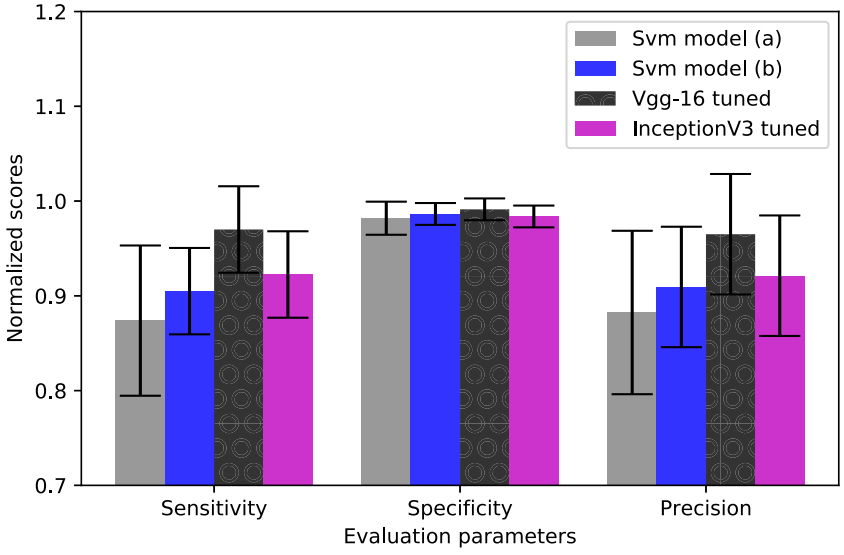


Fig. 13. Average sensitivity, specificity and precision for the four classification models tested.

5.2. Results of fine tuning

The confusion matrices in Fig. 12 show the results of the test assessment of the best models based on Vgg-16 and Inceptionv3 networks selected at the training stage. The model based on Vgg-16 presents very high TPRs for neutrophils, eosinophils and platelets (99.61%) classes. TPR for the rest of classes is over 90%. The Inceptionv3 based model presents a similar performance to Vgg-16, with the highest TPRs for the same three groups: neutrophils (99.61%), eosinophils (98.83%) and platelets (98.44%). An important difference between the two models lies in the results for the monocyte class (95.31% for Vgg-16 and 87.89% for Inceptionv3). Regarding the overall accuracies, the fine-tuned Vgg-16 is 96.2% and Inception 95%.

The bar plots in Fig. 13 present average sensitivity, specificity and precision for the four models tested. We obtained higher sensitivity and precision values with the fine-tuning approach in comparison with the first experiment of transfer learning. Specificity values are similar for all models. From these results, we can state that by implementing fine tuning to pre-trained models it is possible to obtain new models for classification with very high accuracies. This is particularly noticeable in this case of classification of peripheral blood cell images, where characteristics of the different classes are very subtle, such as the chromatin clumping in

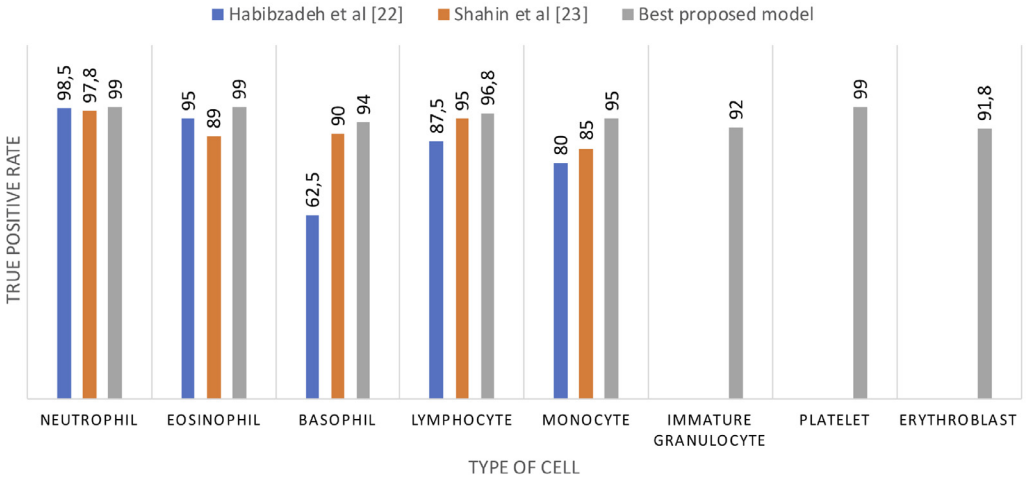


Fig. 14. True positive rates for each individual blood cell type, comparing the proposed Vgg-16 best model with two other previous published works.

neutrophils and eosinophils, and the variations in the nucleus shape: spherical, oval, u-shaped, kidney bean shaped, bilobed and others.

6. Discussion

The problem of automatic classification of the different cells circulating in peripheral blood has been addressed for some years now within the usual machine learning framework with the following steps: preprocessing, segmentation, feature extraction, selection and classification. Specifically, for the classification of leukocytes, different approaches have been used such as Fuzzy logic [34], Learning Vector Quantization [35], k-nearest neighbors algorithms (k-NN), Bayes classifiers [36], multi class support vector machines [37], multilayer perceptrons [37–39] among others.

The traditional framework requires to segment and extract features manually, this means that all possible variations in the morphology of normal and abnormal cells circulating in blood cells should be considered, which is truly difficult [40]. Specially, segmentation presents many challenges due to the complex morphology of the cells, the variability of the blood smear staining and the general conditions of the images acquisition [35]. All of these aspects affect the results of the accuracies of the classification models [34,35,38,41]. In addition, the small amount and low quality of the blood cell images available also affect the robustness of the models [36,39].

Our group has been working in developing image-based screening methods for automatic classification of abnormal lymphoid cells using the traditional framework [42–44]. In this work, we have addressed the use of new emerging deep learning techniques for the recognition of other groups of cells that are also circulating in peripheral blood, such as neutrophils, eosinophils, basophils, lymphocytes and monocytes, immature granulocytes (metamyelocytes, myelocytes and promyelocytes) and others as erythoblasts and platelets. As training a CNN from scratch requires large amounts of data and a lot of expertise to ensure convergence, fine-tuning arises as an alternative to train a model that can outperform a CNN trained from scratch [45].

In this work, two pre-trained networks of different architectures, Vgg-16 and Inceptionv3 have been used. They were trained for the ImageNet contest with images of very different objects with respect to blood cells. The test accuracy of 96% obtained with our fine-tuned Vgg-16 is higher than the one previously reported [22,23].

To perform a more detailed comparison with these works, Fig. 14 shows the true positive rate for each individual cell group. Our fine-tuned Vgg-16 used as an end-to-end classifier outperforms the results of the other models for all the five normal types of cells. In addition, our model classifies with high accuracy other types of cells circulating in peripheral blood such as platelets, erythroblast and immature granulocytes.

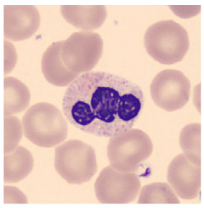
To accomplish this, two strategies have been implemented with the aim to improve the performance of the classifiers. The first one was the application of data augmentation to the training set to increase its size and balance the number of images per class. This strategy allows to obtain a more robust model [46]. The second strategy was the use of open source software that allows easy implementation of the classifier and the reproduction of the results presented here. Another important aspect to remark is the number of images and their high quality used to perform this work. The dataset acquired, with more than 17,000 images, is among the largest used so far to train a blood cell classifier. This allowed to obtain a robust model, avoiding overfitting [22].

An additional difference of this work with previous publications [22,23,41], and one of the main contributions, is that our tuned models succeeded in classifying eight types of blood cells. To our knowledge, this high number of blood cell groups automatically recognized in this work using CNNs has not been reported in the literature. The practical interest is twofold: first, they include the family of normal leukocytes and the cells more frequently observed in infections and regenerative anaemia; and second, the lessons learned in this work will be the basis for further extensions of CNN to classify the broader classes of leukemic and dysplastic cells.

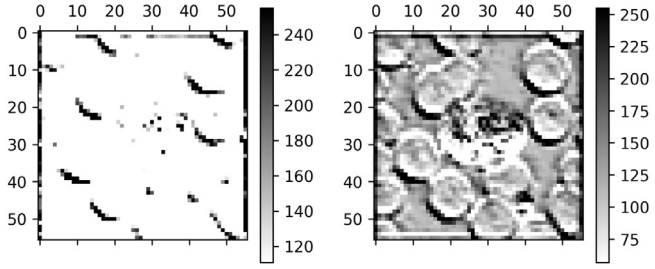
The remaining subsections discuss some insights on the convolutional features, some interpretations about the classification results and a few practical implementation issues.

6.1. Visualization of intermediate feature maps

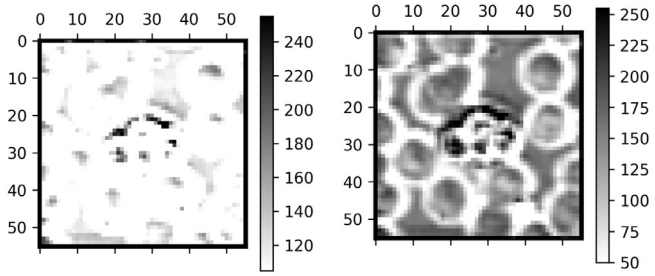
Displaying the feature maps generated by intermediate convolutional layers of a CNN may be useful to get an idea of what features the network extracts. With this purpose, we fed both our trained Vgg-16 model and the original Vgg-16 Imagenet model with a cell image selected from the test set. For comparison, we extracted the same feature maps from the same convolutional layers as illustrated in Fig. 15. Part (a) shows the original image (neutrophil). Fig. 15(b) contains the feature map 38 of 256 from the convolutional block 3, layer 1. From the feature map generated by the original CNN (left), it can be noted that at this deepness of the



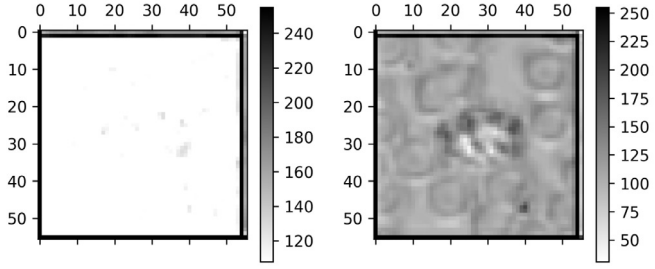
(a) Input image selected (Neutrophil)



(b) Feature map 38 from convolutional block 3, layer 1.



(c) Feature map 27 from convolutional block 3, layer 2.



(d) Feature map 54 from convolutional block 3, layer 3.

Fig. 15. Feature maps from Vgg-16 with Imagenet weights (left) compared with feature maps from Vgg-16 fine tuned for the classification of eight classes of peripheral blood cells.

network, it barely detects diagonal lines of the input image. This was expected since, at this depth, the network still detects dataset-specific features and the original Vgg-16 model was trained with images of everyday objects such as animals, plants, people, which are very different from the images of blood cells. Instead, our own fine-tuned network is able to detect information about the cell, its nucleus, its texture and the red cells around. In our problem of blood cell classification, subtle morphological differences, such as the roundness of the nucleus, the chromatin texture and the presence or absence of granules in the cytoplasm, can make the difference between cell types.

In Fig. 15(c), the feature map 27 of 256 from the convolutional block 3, layer 2, shows that at this stage our network is capable of detecting the red cells from the background, while the feature map from the original network does not capture almost any information from the image. Fig. 15(c) shows another case where the improvement in the detection of important characteristics of the cells is visually interpretable. In each case, the network detects different types of characteristics assigning different degrees of importance depending on the training performed.

It is important to notice that, although all layers of the network extract important information about the input images, not all the feature maps can be visually interpreted. Deeper layers contain more abstract information about the specific class to which the image belongs and further iterations produce smaller maps with features more difficult to observe and interpret.

In the daily work, pathologists perform a visual qualitative analysis of the cell features based on their experience and knowledge. These qualitative features have direct relationship with quantitative features that can be extracted directly from the images using classical machine learning pipelines and mathematical formulations. For example, shape characteristics expressed qualitatively by the pathologist as the size of the cell or the size of the nucleus may be expressed quantitatively using nucleus/cytoplasm ratio, the perimeter of the nucleus, the perimeter of the cell and cytoplasmic external profile. Regarding texture, qualitative features like chromatin clumping or cytoplasmic granulation among others are quantitatively described using formulations based on granulometry [6,8,43]. On the other hand, convolutional neural networks can extract features without image segmentation and are robust models that can achieve state-of-the-art accuracies as we achieved in this paper, but the features extracted have not direct relationship with mathematical formulations and are of limited understanding. Further developments would be worth to explore to translate them into visually interpretable patterns [47].

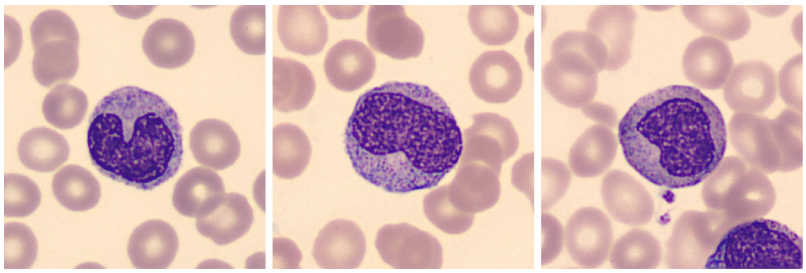
6.2. Detailed analysis of the confusion matrix

Another way to understand the behavior of the final models developed in this work is to perform an analysis of the results presented in the confusion matrix of Fig. 12. Both models Vgg-16 and Inceptionv3 present the highest true positive rates (TPR) for the neutrophils, eosinophils and platelets. The remaining blood cells also present high TPR but with some classification errors, which are worth to discuss in light of the morphological similarities between cell types to understand the model behavior and establish future lines of improvement.

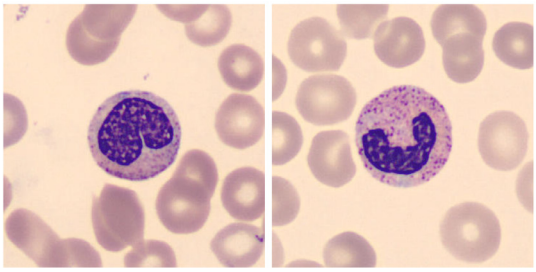
For example, the Vgg-16 model classifies 3.12% of monocytes as immature granulocytes as observed in Fig. 12(a). These two classes come from the same precursor cell, the myeloblast and therefore, have common morphological characteristics such as their size: monocytes are large cells (12–20 µm) and have fine chromatin as well as the promyelocytes (15–25 µm). In addition, the nucleus of the monocyte can be kidney-shaped or be indented as well as the nucleus of the metamyelocyte. A clear difference between monocytes and immature granulocytes (ig) is the texture of the cytoplasm, since the monocyte has some granulation but not as basophilic as that of the immature group (promyelocytes, myelocytes and metamyelocytes). Examples of these cells are presented in Fig. 16(a).

On the other hand, 3.12% of immature granulocytes (ig) were classified as neutrophils, as seen in Fig. 12(a). This may be due to morphological similarities, mainly in the nucleus shape, between the metamyelocytes and the neutrophils band since the latter are an intermediate state in the process of maturation of the metamyelocyte to segmented neutrophil. See Fig. 16(b).

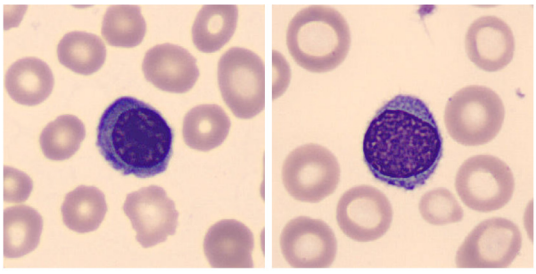
Regarding the erythroblasts, 3.12% of their images were classified as lymphocytes, see Fig. 12(a). Both cells exhibit common patterns. For instance, they have similar size, round nucleus and condensed chromatin as displayed in Fig. 16(c).



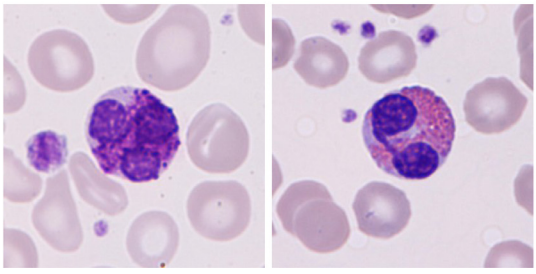
(a) Examples of monocyte (left), promyelocyte (center) and metamyelocyte (right).



(b) Examples of metamyelocyte (left) and neutrophil(right).



(c) Examples of erythroblast (left) and lymphocyte (right).



(d) Examples of basophil (left) and eosinophil (right).

Fig. 16. Examples of cell types where the final models present classification errors.

Finally, 2.58% of basophils were classified as eosinophils. These two groups share some morphological characteristics such as the similar size and the lobed shape of the nucleus. In addition, the basophil granules are larger and darker and cover the entire nucleus and cytoplasm. See Fig. 16(d).

6.3. Practical implementation issues

There are no mathematical expressions or well-established rules to define the number of images to properly train a CNN, how many epochs should be required or how many layers should be tuned. Therefore, most of the model hyperparameters in this work were selected empirically. We performed different tests varying these hyperparameters to know how they affect the final classification results. From our experience, we concluded that using a minimum of 1,000 images per group, a number of epochs around 80 and tuning the last two or four blocks of our proposed CNN's it was possible to obtain satisfactorily high accuracies (96% for Vgg-16).

It is interesting to analyze how the performance of the models change depending on the balance of the training set. In this respect, we performed fine-tuning of both CNNs, Vgg-16 and InceptionV3, with the original unbalanced dataset to compare the results with the balanced dataset presented in this paper. Fig. 17 shows the validation accuracies during the training in both cases. They are higher when the networks are trained with the balanced dataset (>0.95), as expected. Another observable feature is that training with the unbalanced dataset produces many fluctuations in the Vgg-16 validation accuracy. This effect may be due to the fact that the unbalanced validation set is too small. By increasing the number of images and balancing the dataset, it is observed that these fluctuations decrease.

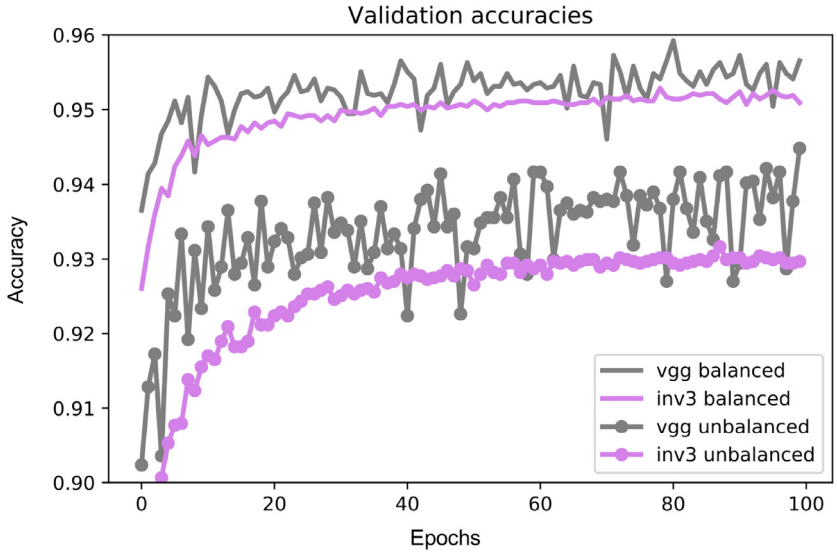


Fig. 17. Validation accuracies of Vgg-16 and Inceptionv3 trained with balanced and unbalanced (original) dataset.

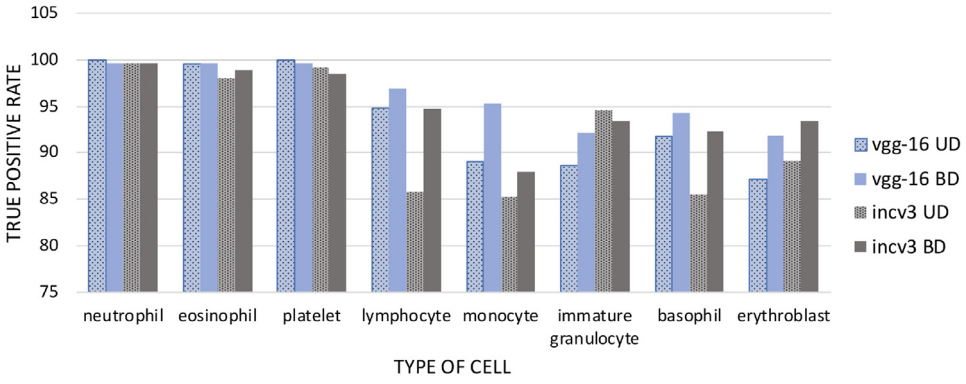


Fig. 18. True positive rate by type of cell obtained with Vgg-16 and Inceptionv3 (incv3) trained with the unbalanced dataset (UD) and the balanced dataset (BD).

We also performed a testing of the networks when trained with the unbalanced data set. In comparison to the balanced data, the overall test accuracy reduces from 96.2% to 93.9% for Vgg-16, while reduces from 94.9% to 92.5% for Inceptionv3. To give more details, Fig. 18 shows the true positive rate values for each type of cell. Two main details may be highlighted:

1. The true positive rates of the groups with higher number of images, eosinophil, neutrophil and platelet, remain at very high values in both training cases, balanced and unbalanced dataset.

2. The true positive rates for the rest of the groups with smaller number of images (lymphocyte, monocyte, immature granulocyte, basophil and erythroblast) decrease for both networks when are trained with the unbalanced dataset.

To summarize, the dataset quality is essential. It should not only have a sufficient number of images, but these images should be of good quality and be properly labeled to be able to observe morphological characteristics that can lead towards a diagnosis. In general, medical data is scarce and difficult to label. In our inter-disciplinary research group we have experienced pathologists able to perform a manual labelling that is also confirmed through other complementary tests if needed. On the other hand, images were all stained with the same standard May Grünwald–Giemsa technique used in the clinical practice.

7. Conclusion

The main contribution of this paper is a classification scheme involving a convolutional neural network trained to discriminate among eight classes of cells circulating in peripheral blood. Starting from a state-of-the art general architecture, we have established a fine tuning procedure to develop an end-to-end classifier trained using a dataset with over 17,000 cell images obtained from clinical practice.

The performance obtained when testing the system has been truly satisfactory, the values of precision, sensitivity and specificity being excellent. To summarize, the overall classification accuracy has been 96.2%.

The focus of this research has been on the group of normal leukocytes and the cells more prevalent in regenerative anaemia and infections, which cover a clinically relevant spectrum of blood cells. This is a necessary first step to develop a broader CNN-based classification system covering other types of blood cells circulating in peripheral blood in different acute leukemias and lymphomas, such as myeloblasts, lymphoblasts, abnormal B or T lymphoid cells and dysplastic cells.

Declaration of Competing Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

References

[1] H. Walker, W. Hall, J. Hurst, Clinical Methods: The History, Physical, and Laboratory Examinations, Butterworths, 1990.
[2] A. Merino, Manual de Citología de Sangre Periférica, Grupo Acción Médica, Madrid, 2005.
[3] The easyCell assistant, 2019, http://www.medicacorp.com/.
[4] Vision Hema, 2019, http://hema.wm-vision.com/.
[5] Cellavision, Cellavision DM9600, 2019, http://www.cellavision.com/.
[6] L. Puigví, A. Merino, S. Alférez, A. Acevedo, J. Rodellar, New quantitative features for the morphological differentiation of abnormal lymphoid cell images from peripheral blood, J. Clin. Pathol. 70 (12) (2017) 1038–1048, doi:10.1136/jclinpath-2017-204389.
[7] J. Rodellar, S. Alférez, A. Acevedo, A. Molina, A. Merino, Image processing and machine learning in the morphological analysis of blood cells, Int. J. Lab. Hematol. 40 (2018) 46–53. doi:10.111/ijlh.12818.
[8] A. Merino, L. Puigví, L. Boldú, S. Alférez, J. Rodellar, Optimizing morphology through blood cell image analysis, Int. J. Lab. Hematol. 40 (S1) (2018) 54–61, doi:10.1111/ijlh.12832.
[9] X. Zheng, Y. Wang, G. Wang, J. Liu, Fast and robust segmentation of white blood cell images by self-supervised learning, Micron 107 (2018) 55–71, doi:10.1016/j.micron.2018.01.010.
[10] S. Alférez, A. Merino, A. Acevedo, L. Puigví, J. Rodellar, Color clustering segmentation framework for image analysis of malignant lymphoid cells in peripheral blood, Med. Biol. Eng. Comput. (2019), doi:10.1007/s11517-019-01954-7.
[11] F. Chollet, Deep Learning With Python, 1, Manning Publications Company, 2015, doi:10.1017/CBO9781107415324.004.
[12] Y. Lecun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444, doi:10.1038/nature14539.
[13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, L. Fei-Fei, Imagenet large scale visual recognition challenge, Int. J. Comput. Vision 115 (3) (2015) 211–252, doi:10.1007/s11263-015-0816-y.
[14] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations (ICRL), 2015, pp. 1–14, doi:10.1016/j.infsof.2008.09.005.
[15] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 2818–2826, doi:10.1109/CVPR.2016.308. http://arxiv.org/abs/1512.00567.
[16] S. Hoo-Chang, H.R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, R.M. Summers, Deep convolutional neural networks for computer-aided detection : Cnn architectures, dataset characteristics and transfer, IEEE Trans. Med. Imaging 35 (5) (2016) 1285–1298, doi:10.1109/TMI.2016.2528162.
[17] M. Saraswat, K. Arya, Automated microscopic image analysis for leukocytes identification: a survey, Micron 65 (2014) 20–33.
[18] J. Rawat, H. Bhadauria, A. Singh, J. Virmani, Review of leukocyte classification techniques for microscopic blood images, in: Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on, IEEE, 2015, pp. 1948–1954.
[19] A. Kumar, J. Kim, D. Lyndon, M. Fulham, D. Feng, An ensemble of fine-tuned convolutional neural networks for medical image classification, IEEE J. Biomed. Health Inf. 21 (1) (2017) 31–40, doi:10.1109/JBHI.2016.2635663.
[20] D. Raví, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, G.-Z. Yang, Deep learning for health informatics, IEEE J. Biomed. Health Inf. (2017), doi:10.1109/JBHI.2016.2636665.
[21] F. Cabitza, G. Banfi, Machine learning in laboratory medicine: waiting for the flood? Clin. Chem. Lab. Med. (CCLM) 56 (4) (2018) 516–524, doi:10.1515/cclm-2017-0287.
[22] M. Habibzadeh, A. Krzyżak, T. Fevens, White blood cell differential counts using convolutional neural networks for low resolution images, in: International Conference on Artificial Intelligence and Soft Computing, Springer, Springer Berlin Heidelberg, 2013, pp. 263–274.
[23] A.I. Shahin, Y. Guo, K.M. Amin, A.A. Sharawi, White blood cells identification system based on convolutional deep neural learning networks, Comput. Methods Programs Biomed. 0 (2017) 1–12, doi:10.1016/j.cmpb.2017.11.015.
[24] F. Chollet, et al., Keras, 2015, https://keras.io.
[25] M. Oquab, L. Bottou, I. Laptev, J. Sivic, Learning and transferring mid-level image representations using convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1717–1724, doi:10.1109/CVPR.2014.222.
[26] H. Chougrad, H. Zouaki, O. Alheyane, Deep convolutional neural networks for breast cancer screening, Comput. Methods Programs Biomed. 157 (2018) 19–30, doi:10.1016/j.cmpb.2018.01.011.
[27] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[28] Y. LeCun, K. Kavukcuoglu, C. Farabet, et al., Convolutional networks and applications in vision., in: Proceedings of 2010 IEEE International Symposium on Circuits and Systems, 2010, 2010, pp. 253–256, doi:10.1109/ISCAS.2010.5537907.
[29] Cs231n: Convolutional neural networks for visual recognition, 2019, http://cs231n.github.io/convolutional-networks/. Accessed: 2019-01-12.
[30] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, et al., Api design for machine learning software: experiences from the scikit-learn project, (2013), arXiv:1309.0238v1.
[31] V. Sze, Y.H. Chen, T.J. Yang, J.S. Emer, Efficient processing of deep neural networks: a tutorial and survey, Proc. IEEE 105 (12) (2017) 2295–2329, doi:10.1109/JPROC.2017.2761740.
[32] A. Géron, Hands-on Machine Learning with Scikit-Learn and TensorFlow : Concepts, Tools, and Techniques to Build Intelligent Systems, O'Reilly Media, Incorporated, 2019.
[33] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks? in: Advances in Neural Information Processing Systems, 27, 2014, pp. 3320–3328, doi:10.1109/IJCNN.2016.7727519. http://arxiv.org/abs/1411.1792.
[34] Q. Wang, L. Chang, M. Zhou, Q. Li, H. Liu, F. Guo, A spectral and morphologic method for white blood cell classification, Opt. Laser Technol. 84 (2016) 144–148, doi:10.1016/j.optlastec.2016.05.013.
[35] N. Ramesh, B. Dangott, M.E. Salama, T. Tasdizen, Isolation and two-step classification of normal white blood cells in peripheral blood smears, J. Pathol. Inf. 3 (2012), doi:10.4103/2153-3539.93895. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3327044/
[36] J. Prinyakupt, C. Pluempitiwiriyawej, Segmentation of white blood cells and comparison of cell morphology by linear and naïve bayes classifiers, BioMed. Eng. Online 14 (1) (2015) 1–19, doi:10.1186/s12938-015-0037-1.
[37] S.H. Rezatofighi, H. Soltanian-Zadeh, Automatic recognition of five types of white blood cells in peripheral blood, Comput. Med. Imaging Graphics 35 (4) (2011) 333–343, doi:10.1016/j.compmedimag.2011.01.003.
[38] S. Nazlibilek, D. Karacor, T. Ercan, M.H. Sazli, O. Kalender, Y. Ege, Automatic segmentation, counting, size determination and classification of white blood cells, Meas. J. Int.Meas. Confederation 55 (2014) 58–65, doi:10.1016/j.measurement.2014.04.008.
[39] M.C. Su, C.Y. Cheng, P.C. Wang, A neural-network-based approach to white blood cell classification, Sci. World J. 2014 (1) (2014), doi:10.1155/2014/796371.
[40] Meiyin Wu, Li Chen, Image recognition based on deep learning, in: 2015 Chinese Automation Congress (CAC), 2015, pp. 542–546, doi:10.1109/CAC.2015.7382560.
[41] A. Mathur, A. Tripathi, M. Kuse, Scalable system for classification of white blood cells from leishman stained blood stain images, J. Pathol. Inf. 4 (2) (2013) 15, doi:10.4103/2153-3539.109883.
[42] S. Alférez, A. Merino, L. Bigorra, L. Mujica, M. Ruiz, J. Rodellar, Automatic recognition of atypical lymphoid cells from peripheral blood by digital image analysis, Am. J. Clin. Pathol. 143 (2) (2015) 168–176, doi:10.1309/AJCP78IFSTOGZZJN.
[43] S. Alférez, A. Merino, L. Bigorra, J. Rodellar, Characterization and automatic screening of reactive and abnormal neoplastic b lymphoid cells from peripheral blood, Int. J. Lab. Hematol. 38 (2) (2016) 209–219, doi:10.1111/ijlh.12473.
[44] L. Bigorra, S. Alférez, A. Merino, J. Rodellar, Feature analysis and automatic identification of leukemic lineage blast cells and reactive lymphoid cells from peripheral blood cell images, J. Clin. Lab. Anal. 31 (2) (2017) e22024, doi:10.1002/jcla.22024.
[45] N. Tajbakhsh, J.Y. Shin, S.R. Gurudu, R.T. Hurst, C.B. Kendall, M.B. Gotway, J. Liang, Convolutional neural networks for medical image analysis: full training or fine tuning? IEEE Trans. Med. Imaging 35 (5) (2016) 1299–1312, doi:10.1109/TMI.2016.2535302.
[46] A. Mikolajczyk, M. Grochowski, Data augmentation for improving deep learning in image classification problem, in: 2018 International Interdisciplinary PhD Workshop (IIPhDW), 2018, pp. 117–122, doi:10.1109/IIPHDW.2018.8388338.
[47] Z. Qin, F. Yu, C. Liu, X. Chen, How convolutional neural networks see the world — a survey of convolutional neural network visualization methods, Math. Found. Comput. 1 (2) (2018) 149–180, doi:10.3934/mfc.2018008.