Computers in Biology and Medicine 116 (2020) 103542

Contents lists available at ScienceDirect

Computers in Biology and Medicine

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

Comparing convolutional neural networks and preprocessing techniques for HEp-2 cell classification in immunofluorescence images

Larissa Ferreira Rodrigues [a,b,*], Murilo Coelho Naldi [a,c], João Fernando Mari [b]

[a] Departamento de Informática, Universidade Federal de Viçosa (UFV), Viçosa, MG, Brazil
[b] Instituto de Ciências Exatas e Tecnológicas, Universidade Federal de Viçosa (UFV), Rio Paranaíba, MG, Brazil
[c] Departamento de Computação, Universidade Federal de São Carlos (UFSCar), São Carlos, SP, Brazil

ARTICLE INFO

Keywords:
Convolutional neural networks
HEp-2 cells
Staining pattern classification
Preprocessing
Data augmentation
Hyperparameters
Fine-tuning

ABSTRACT

Autoimmune diseases are the third highest cause of mortality in the world, and the identification of an anti-nuclear antibody via an immunofluorescence test for HEp-2 cells is a standard procedure to support diagnosis. In this work, we assess the performance of six preprocessing strategies and five state-of-the-art convolutional neural network architectures for the classification of HEp-2 cells. We also evaluate enhancement methods such as hyperparameter optimization, data augmentation, and fine-tuning training strategies. All experiments were validated using a five-fold cross-validation procedure over the training and test sets. In terms of accuracy, the best result was achieved by training the Inception-V3 model from scratch, without preprocessing and using data augmentation (98.28%). The results suggest the conclusions that most CNNs perform better on non-preprocessed images when trained from scratch on the analyzed dataset, and that data augmentation can improve the results from all models. Although fine-tuning training did not improve the accuracy compared to training the CNNs from scratch, it successfully reduced the training time.

1. Introduction

Autoimmune pathologies arise when healthy cells and tissues are mistakenly attacked and destroyed by the immune system [1–3]. Cases related to autoimmune disease have undergone a significant increase in recent decades, making them the third most frequent cause of mortality, followed by cardiovascular disease and cancer [4]. There is no known cure for autoimmune diseases, and their treatment consists of alleviating symptoms using anti-inflammatory and immunosuppressive drugs [5]. Although most autoimmune diseases present similar symptoms, they are still difficult to diagnose, and most methods for this are expensive.

Visual analysis of staining patterns in immunofluorescence (IIF) images taken from human epithelial cells (HEp-2) is a procedure that can be used to identify autoimmune diseases. IIF is a multiplex technique which allows the detection of many different nuclear and cytoplasmic patterns [6]. In this procedure, a slide is designed to hold the HEp-2 cells substrate, and the serum of a subject is added. Following this, four steps are carried out: (i) image acquisition; (ii) mitotic cell recognition; (iii) classification of fluorescence intensity; and (iv) recognition of the staining pattern. The importance of the last step lies in the fact that each staining pattern may indicate a specific autoimmune disease depending on the patient's clinical history [6–8].

Analysis of HEp-2 cells by IIF has been carried out both manually and automatically over the decades, but these processes are very subjective and time-consuming [9–11]. Hand-engineered methodologies are unable to handle the increasing volumes of cell data that are generated on a daily basis, making them increasingly prohibitive. The most promising technique for fulfilling this need for automated and scalable methods is the development of computer-aided diagnosis systems based on image processing and machine learning techniques [12,13].

Digital image processing and machine learning are essential for a large number of computer-aided diagnosis applications [14,15]. Every day, new IIF images of HEp-2 cells are created worldwide, generating large datasets. Solutions based on computerized image analysis typically have a low cost and are much more democratic, as the data and results of analysis may be shared and reprocessed around the world [16,17]. This characteristic makes such techniques financially attractive, especially for emerging countries [18]. The accumulation of vast amounts of data, and especially in the form of images, has allowed deep learning techniques to become successful. In particular, convolutional neural networks (CNNs) have demonstrated effective results in general visual recognition tasks, motivating their broad application in several research fields [19,20]. Their success is due to their ability to extract high-level hierarchical representations of the data through

* Corresponding author at: Departamento de Informática, Universidade Federal de Viçosa (UFV), Viçosa, MG, Brazil.
E-mail addresses: larissa.f.rodrigues@ufv.br (L.F. Rodrigues), naldi@ufscar.br (M.C. Naldi), joaof.mari@ufv.br (J.F. Mari).

https://doi.org/10.1016/j.compbiomed.2019.103542
Received 25 April 2019; Received in revised form 12 November 2019; Accepted 12 November 2019
Available online 20 November 2019

multi-stage image processing. When combined with powerful prepossessing strategies, CNNs are capable of identifying several autoimmune diseases such as systemic lupus erythematosus, autoimmune hepatitis, rheumatoid arthritis, multiple sclerosis, and diabetes [8].

Recently, Gao et al. [21] presented a method for classifying HEp-2 cells based on a deep CNN inspired by the classical LeNet-5 architecture [22]. This was a pioneering method for state-of-the-art techniques of HEp-2 classification with CNNs. Inspired by [21], Rodrigues et al. [23] applied the same experimental methodology to compare three CNN architectures using different preprocessing strategies. However, the validation methodology used in [21] and [23] is a simple hold-out technique [24], which consists of randomly assigning data points to the training and test sets. This approach has a certain probability of creating two biased sets, which may result in abnormal CNN models, i.e., CNNs that do not behave like the majority of CNNs trained with this data. As a consequence, such CNNs may achieve feasible but abnormal accuracy results, which are unlikely to happen in real scenarios. Moreover, the values of the hyperparameters also influence the results for each CNN, and need to be fine-tuned for each data type and application scenario.

This work is an extension of the studies presented in [23] and [25]. Unlike this previous research, we compare five CNN architectures, LeNet-5 [22], AlexNet [19], Inception-V3 [26], VGG-16 [27], and ResNet-50 [28], using six different preprocessing strategies. A k-fold cross-validation technique is adopted [24] to achieve a better estimate of the accuracy of the studied CNN architectures. In k-fold cross-validation, the original data are randomly partitioned into k subsamples (folds) of similar sizes, and these are iteratively selected as training and validation data, for a total of k times. Finally, these k sets of results are averaged to produce a single estimation, which is more robust than the results from the hold-out method.

Automatic feature extraction is the main strength of solutions based on CNNs, since they do not require handcrafted feature extraction. However, identifying optimal hyperparameters may be a challenge, as there is no optimum method for the selection of hyperparameters such as the number of hidden units, weight decay, and others. In this work, we adopt the tree of Parzen estimators (TPE) method to optimize these hyperparameters. The appropriate pre-processing of HEp-2 cell images is another limitation of using CNNs, since the real impact of pre-processing based on contrast enhancement on the performance of a CNN is unknown.

This paper identifies a suitable method based on CNNs for automatically classifying HEp-2 cells in immunofluorescence images that addresses the previously mentioned limitations. Its main contributions are: (i) a comparison of the performance of five CNN architectures in terms of classification accuracy and training time; (ii) an exploration of these CNNs via training from scratch and fine-tuning; (iii) an investigation of the impact of strategies for preprocessing images based on combinations of contrast improvement, image normalization, and data augmentation.

Furthermore, a novel contribution of this paper is an assessment of the impact of preprocessing methods based on contrast enhancement when compared with computer-assisted diagnosis of autoimmune diseases without preprocessing. The results show that for the HEp-2 dataset, the majority of the analyzed CNNs perform better when trained without preprocessing and when combined with data augmentation. Our results are also very close to those of a state-of-the-art classification method presented in the literature for the automatic classification of HEp-2 cells. We believe that the work presented here can help healthcare workers to choose a correct classification method for identifying and managing autoimmune conditions and the diseases that cause them.

The remainder of this paper is organized as follows: Section 2 surveys related work; Section 3 describes background concepts related to materials and methods; Section 4 presents and discusses the results; and finally, conclusions and future work are presented in Section 5.

2. Related work

Perner et al. [29] were among the first to propose an automatic system for classifying HEp-2 image cells. These authors extracted several textural features from 12 quantized versions of the original images, and classified them using decision trees. The same approach was adopted by Sack et al. [30] to identify positive fluorescence in a set of immunofluorescence patterns. However, few images were available for training the decision tree model, which was symbolic but imprecise compared to the most recent deep learning approaches.

Several studies have been proposed for classifying HEp-2 cells with hand-crafted feature extraction (i.e., using a non-automated user-based process). Nosaka and Fukui [31] proposed a method for classifying HEp-2 cell images based on a new type of local binary pattern feature and support vector machines (SVMs). Stoklasa et al. [32] classified HEp-2 cells using a K-NN classifier with Haralick features, local binary patterns, SIFT, surface description, and a granulometry-based descriptor. Shen et al. [33] proposed a bag of visual words (BoVW) model of features based on intensity order pooling with SVM classification. Taalimi et al. [34] used multi-modal dictionary learning with sparse representation, and image patches were used to calculate the features while training the classifier.

Manivannan et al. [13] utilized several local descriptors and incorporated multiresolution local high-order statistical features. Based on the scale-invariant descriptor, Gragnaniello et al. [35] encoded the features into the BoVW model to classify HEp-2 cell images. Kastaniotis et al. [36] proposed a feature encoding process that incorporated sparse coding, and used a vector of hierarchically aggregated residuals, SIFT descriptors and different classifiers such as SVM and K-NN. Ensafi et al. proposed a variety of approaches, including a non-parametric Bayesian model [37], a superpixel technique using sparse codes of image patches [38], and SIFT and SURF features with a BoVW model to classify HEp-2 cells with SVM [39]. Faria et al. [40] introduced a simple combination of SIFT and SURF by stacking the key point descriptors in a single matrix, and used a multi-layer perceptron for classification. However, non-automated feature extraction can be a burdensome task, especially for large datasets.

With advances in technology and the increase in the computational resources and datasets available, deep CNN models now significantly outperform models based on handcrafted feature extraction in general visual recognition tasks. Malon et al. presented in [41], were among the first to adopt a CNN to classify HEp-2 cells in images by applying contrast enhancement, standardized at 100 x 100 pixels, and by considering only the green component. Unlike the work in [41], our study examines a variety of preprocessing methods and CNN architectures, and achieves better results.

Recently, Gao et al. [21] have proposed a method for the automatic classification of HEp-2 cells using a CNN that shares the basic architecture of LeNet-5 [22]. One advantage of a CNN is the ability to automatically learn features during training, which is not available in the work cited above. Experiments using one instance of each image achieved an accuracy of 88.58%, and this improved to 96.76% after data augmentation using rotated copies of images of the original data. However, this study used a simple hold-out validation technique [24], which consists of randomly assigning data points to the training and test sets. Although hold-out is a classical validation technique, there is a certain probability of building biased training and validation sets, which may result in abnormal CNN models (i.e., CNNs that behave unlike most CNNs trained with the same type of data). Other validation techniques such as cross-validation schemes (e.g. k-fold or leave-one-out) [24] can be considered. These techniques require more computer power or a longer processing time, but deliver greater reliability.

Using the methodology presented in [21], Rodrigues et al. [23] used AlexNet and GoogLeNet models for the analysis and classification of HEp-2 cell images, in addition to LeNet-5. The results showed that neither of these preprocessing strategies significantly improved the

accuracy of the CNNs when trained with fixed (pre-defined/default) values of their hyperparameters, and that when data augmentation is considered, contrast enhancement followed by data centralization is important in order to achieve good results. This study used the hold-out validation method adopted in [21], giving rise to the same problem of biased sets and causing the CNNs to fit non-representative (abnormal) samples, resulting in unexpected values of accuracy.

The authors of [25] present an extension of the work published in [23], where a k-fold cross-validation technique is adopted in order to better estimate the accuracy of the studied CNN architectures with empirically tuned hyperparameters. Unlike in [23], in this study we consider new validation techniques with k-fold hyperparameter optimization using TPE, assess the influence of different preprocessing approaches on each of the CNN architectures, and evaluate and analyze training from scratch and fine-tuning.

Li and Shen [42] adopted different network architectures for shallow and deep layers to extract and fuse features for HEp-2 cell image classification, using preprocessing based on contrast enhancement and data augmentation to generate new rotated and shifted images. Gupta et al. [43] used AlexNet architecture in combination with one-class support vector machines (OC-SVMs) to address the task of classification of cells in mitosis. In contrast, we solve the task of staining pattern classification in this paper.

Most recently, Lei et al. [44] proposed a method based on ResNet-50 using deeply supervised networks with cross-modal transfer learning to classify HEp-2 cell images. They found that the number of labeled medical images available to them was small, causing overfitting of the model. They therefore pre-trained the network using the International Conference on Pattern Recognition (ICPR) 2012 dataset, and applied the trained CNN to ICPR2016-Task1. The success of cross-modal-based transfer learning in the ResNet-50 network is based on the idea that the expression characteristics in the low-level representation and high-level features are similar. Transfer learning gave similar or superior results in most of the reported experiments. However, cross-modal techniques require a significant similarity between training datasets, which may not be possible in some application scenarios. The techniques we present in this work do not require additional datasets, and we consider a broader range of CNNs.

3. Methodology

The main goal of this paper is to evaluate the performance of different types of CNNs to classify microscopy images of HEp-2 cells. More precisely, we assess the influence on the performance of CNNs of the use of different preprocessing strategies, data augmentation, and fine-tuning training. An extensive study was carried out using k-fold cross-validation techniques and the optimization of hyperparameter values. Fig. 1 illustrates the steps of the methodology adopted here.

3.1. Image dataset

The dataset used was taken from a contest entitled "Performance Evaluation on Indirect Immunofluorescence Image Analysis Systems" hosted at ICPR 2014[1] [45]. It contains 13,596 images of HEp-2 cells, each with a single, centered cell in evidence, categorized into one of six classes: centromere (2741 images), golgi (724 images), homogeneous (2494 images), nucleolar (2598 images), nuclear membrane (2208 images), and speckled (2831 images). To illustrate the dataset, one image from each class is presented in Fig. 2. The ICPR committee does not provide the testing set used to evaluate the results of the competition.

[1] Available in: http://nerone.diem.unisa.it/hep2-benchmarking/dbtools/.

3.2. Hyperparameter optimization

Hyperparameters define the topology of the network, its architecture, the details of its training, and the optimization algorithms used [46]. In short, the choice of values for the hyperparameters is crucial to the performance of the CNN. This choice may be made empirically by testing different values and fine-tuning until the algorithm gives a "satisfactory" performance. Since the optimal values are unknown, any definition of a level of satisfaction using this methodology is subjective, and requires previous knowledge of the algorithm and the application scenario. Alternatively, it is possible to represent the choice of hyperparameter values as an optimization problem, where the hyperparameters are considered as the decision variables, and the validation error of the trained model is the objective function to be minimized.

The fine-tuned hyperparameters used to optimize the performance of the CNNs in this work are as follows:

- Number of hidden units (n_h): This is the number of neurons in the fully connected hidden layers. A small n_h value may lead the training to under-fit the model to the data, while a high n_h may cause the model to over-fit the data and increase the processing time required [47]. The larger and more complex the dataset, the higher the number of neurons needed to obtain the relevant information [46].
- Weight decay: The weight decay is a penalty term imposed on overweight values that did not significantly influence the correct classification during training. The idea is to minimize these values while keeping the values of other relevant connections relatively high. This directly influences the regularization of the λ coefficient and reduces network overfitting.
- Learning rate: The learning rate defines the levels of adjustment to the weight connections and network topology that are applied in each training cycle. A low learning rate permits surgical fine-tuning of the model to the training data at the expense of a higher number of training cycles and longer processing time. A high learning rate permits the model to learn more quickly, but may sacrifice its accuracy due to a lack of precision in the adjustments. This value is usually set at 0.01, but in some cases it is valuable to fine-tune this parameter, particularly in order to improve the runtime when using SGD [46].
- Momentum: The momentum is responsible for reducing noise and oscillations in the high-curvature regions of the loss function generated by the SGD. By default, its value is set to $\mu = 1$, but in some situations, fine-tuning this hyperparameter may lead to improved results [48].
- Nesterov Momentum: Unlike traditional momentum, this imposes a correction factor on the current gradient velocity when it is evaluated, which can result in significantly faster convergence to the optimum during learning [47,49].

3.2.1. Tree of Parzen Estimators

The TPE algorithm [50] is a type of sequential model-based optimization (SMBO) used to fine-tune a set of hyperparameters x in order to optimize the accuracy of the model based on previous observations [51]. It is suited to applications where the evaluation of the cost function is expensive [50], as it iterates between previous models, acquiring the knowledge needed to choose the next setup values investigated. The TPE algorithm gives an improvement over the classic hyperparameter optimization methods, and allows the model to learn from the training history and to generate increasingly good estimations for the next set of parameters [50,52].

TPE builds a probabilistic model $p(x|y)$, where x is the set of hyperparameters to be fine-tuned and y is the quality index to be optimized,

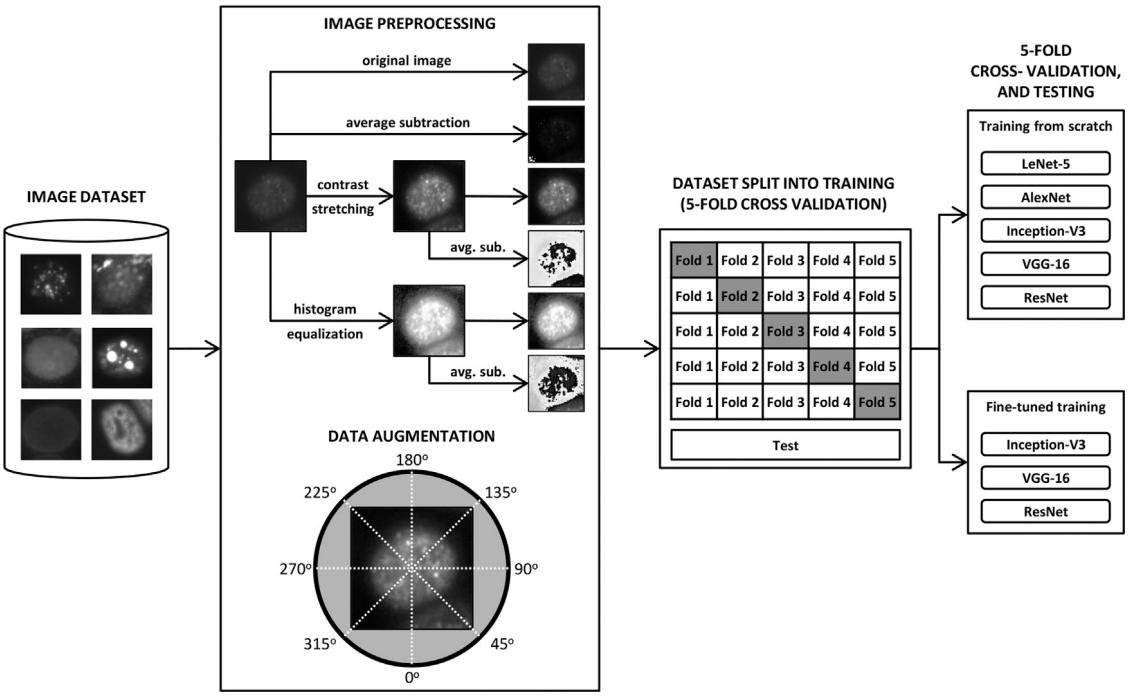


Fig. 1. Steps of proposed method.

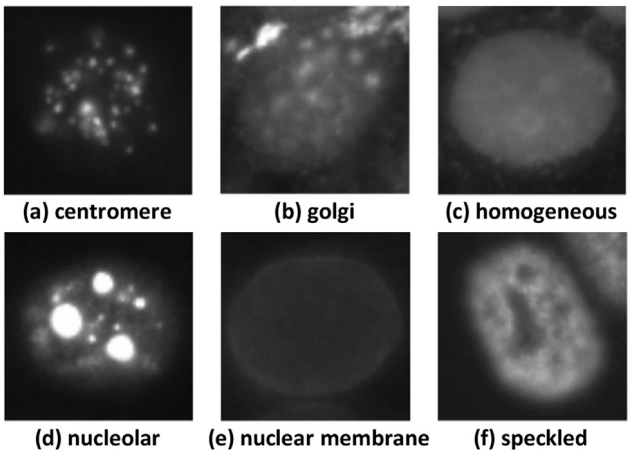


Fig. 2. Examples of image instances for each class of dataset (ICPR 2014).

assuming non-parametric distributions instead of the prior distribution of x. The calculation of $p(x|y)$ is defined in Eq. (1):

$$p(x|y) = \begin{cases} \ell(x), \text{if } y < y^* \\ g(x), \text{if } y \geq y^* \end{cases} \quad (1)$$

where $\ell(x)$ is the density function of the observations with better results, and $g(x)$ is the density function formed by the remaining observations. The TPE algorithm chooses y^* to be some quantile γ of the observed values so that $p(y < y^*) = \gamma$, but no specific model for $p(y)$ is necessary. The hyperparameters x and y represent the associated quality index.

After an initial value estimation is made using a random search, two Gaussian mixed models are built for each TPE iteration $\ell(x)$, which is estimated based on the best-evaluated observations, and $g(x)$, which is estimated based on the remaining observations. The objective is to find hyperparameter values x that minimize the ratio $\ell(x)/g(x)$. After each iteration, TPE returns the set of hyperparameters x with the greatest expectation improvement [50,53].

3.3. Image preprocessing

All images were resized to 224 × 224 pixels, the smallest size allowed for the input of the CNN models used in this work. This resizing was based on bilinear interpolation, and did not add or remove any information to the images as the images were increased. This was done before applying the preprocessing methods, and the images were used to train, validate, and test all the CNN models.

The brightness and contrast of the HEp-2 cell images vary greatly. Preprocessing techniques for reducing this variance based on contrast enhancement and data centralization have been reported in the literature [21,31,32,40] [42]. We propose several image preprocessing strategies based on combinations of contrast improvement and image normalization in order to assess their impact on the classification results of each CNN model studied. These strategies combine contrast stretching, histogram equalization, and average image subtraction. In this paper, we also investigate the impact of no preprocessing, i.e., training the model on the original images.

- Contrast stretching: This is a linear transformation function for mapping the maximum intensity of an image to 1 and the minimum to 0, while the intermediary values are linearly mapped onto the interval [0,1] according to their original values [54].
- Histogram equalization: This uses a piecewise linear function based on the normalized image histogram, resulting in an image with uniform histogram values [54].
- Average subtraction: This process computes the average image of the training set, and subtracts it from each image in the dataset. The result is a training set with zero mean, and the mean of the whole dataset is also close to zero if the training set is representative [55].

Combinations of these preprocessing techniques resulted in six preprocessing strategies, as illustrated in Fig. 3. The leftmost image is the original from the dataset, which is defined as strategy (a) (i.e., no preprocessing). On the right-hand side, in (b), we see the result of the average subtraction strategy. Contrast stretching is used in (c), and average subtraction is added in (d). Finally, strategies (e) and (f) use histogram equalization with and without average image subtraction, respectively.

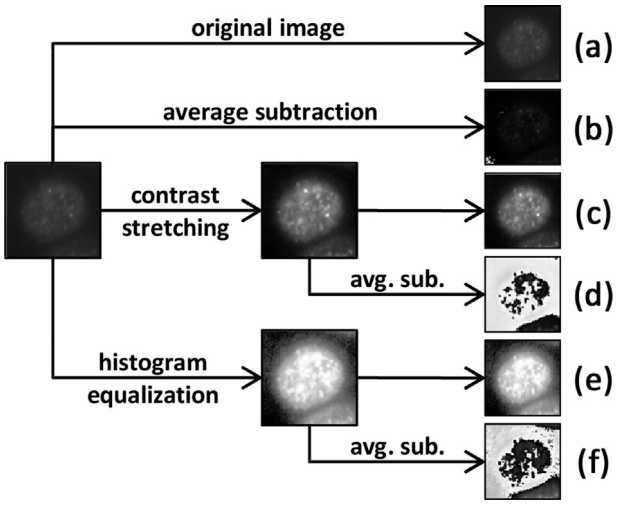


Fig. 3. Examples of contrast improvement of a HEp-2 cell image: (a) original image; (b) average subtraction; (c) contrast stretching; (d) contrast stretching with average subtraction; (e) histogram equalization; and (f) histogram equalization with average subtraction.

3.3.1. Data augmentation

The data augmentation step aims to increase the training data artificially, without introducing labeling costs [19].

In this study, we created new images by rotating each of the original images around its center through randomly chosen angles of between 0° and 360°.

3.4. Convolutional neural networks

A CNN is one category of deep learning methods, along with restricted Boltzmann machines (RBMs), sparse coding, and autoencoders. A CNN mainly consists of three types of neural layers: (a) convolutional layers; (b) pooling layers; and (c) fully connected layers [56]. These layers form a hierarchical architecture, with convolutional layers alternating with pooling layers and a fully-connected layer [22]. The details of the topology of CNNs are described below.

Convolutional layers: These perform convolution of the feature maps in the previous layer using a set of filters, each of which is intended to detect a given type of feature. A convolutional layer C^l is composed of a bank of K filters W_k^l and K bias b_k^l, for which $k \in [1, \ldots, K]$. The size of each filter W_k^l is $(F \times D)$, where F is the spatial stent of the filter and D is a hyperparameter from the input volume M^{l-1}. The input volume, M^{l-1}, is a stack of D feature maps resulting from the previous layer $l-1$, with size $(H \times W \times D)$. To generate the output volume M^l, the convolutional layer C^l convolves each slice $d \in [1, \ldots, D]$ of the input volume M^{l-1} with each of the $k \in [1, \ldots, K]$ filters W_k^l. The resulting convolutions from each filter are summed and added to the bias. The resulting K 2D feature maps are stacked in an output volume M^l. Eq. (2) shows the formation of one M_k^l slice of the output volume M^l.

$$M_k^l = \sum_{d=1}^{D} M_d^{l-1} * W_d^l + b_k^l \tag{2}$$

After the convolution layer, an element-wise activation function is applied. Generally, a REctified Linear Unit (RELU), $f(M) = \max(0, M)$, is preferred over other common activation functions such as sigmoid and hyperbolic tangent, which tend to result in much faster training in very deep architectures [57].

Pooling layers: Following a convolution layer, pooling layers are usually responsible for reducing the size of the feature maps based on certain criteria, such as maximum or average pooling in accordance with the size of the pooling region. A pooling layer with pooling size 2×2 reduces a path with 2×2 pixels to one pixel, by selecting the maximum or the average pixel in max-pooling or average-pooling, respectively [58]. Maximum pooling results in faster convergence and better generalization in most cases [58].

Fully connected layers: The last layers of a CNN convert the 2D feature maps into a 1D feature vector. This feature vector forms the input to a number of fully connected layers, performing a traditional inner product computation. The last layer generally consists of a softmax classifier. The softmax layer h has n neurons, one for each of the n classes in the dataset, and their output $\tilde{y}_j$ is the probability that the input image belongs to class j [21].

The fully connected layers contain about 90% of the total parameters in a CNN, and are responsible for most of the computational cost during training [56]. With this in mind, Inception V3 (GoogLeNet), one of the architectures considered in this work, adopts a strategy to reduce the number of connections in the fully connected layers [26,59].

3.4.1. Optimizers for training

A set of optimization algorithms may be employed to perform training of the CNN models, i.e., to adjust the connection parameters W in order to minimize an objective function (here, the loss function). In this work, we consider the cross-entropy function, which is a common choice in the literature for training deep learning models [47,60].

The cross-entropy function computed over a set of training samples x_j is based on a classification that considers the parameters W, $f(x_j, W)$ and the known classes y_j shown in Eq. (3):

$$\mathcal{L}(W) = \frac{1}{n} \sum_{j=1}^{N} \ell(y_j, f(x_j; W)) \tag{3}$$

The minimization of $\mathcal{L}(W)$ may be done by an optimization algorithm such as gradient descent (GD) [61], which computes partial derivatives in the space parameters to find the set of parameter values W that locally minimizes the cost function [60].

However, using GD with back-propagation is impracticable in deep learning applications, due to the need to compute the gradient for each instance. To overcome these performance problems, variations of the GD algorithm have been proposed, such as stochastic GD (SGD) [22], AdaGrad [62], Adam [63], RMSprop [64], and others. In this work, we chose to apply SGD.

SGD is one of the most popular algorithms for parameter optimization of deep learning models. It is based on a GD approximation using batches of randomly selected data samples, instead of computing the gradient for each object of the dataset. This work uses SGD with momentum (both traditional and Nesterovs') [22,47], as discussed in Section 3.2.

3.4.2. CNN architectures

Five CNN architectures were selected for experimental comparison based on their past performance in image classification tasks. These are described below:

- *LeNet*: This approach was initially developed by Yan LeCun et al. [22] for character recognition. LeNet-5 was one of the first successful applications of CNNs, and is composed of three convolutional layers, two pooling layers, one fully connected layer and a softmax layer [21,22].
- *AlexNet*: Proposed by Krizhevsky et al. [19], AlexNet won the ILSVRC 2012 competition [65]. It is composed of five convolutional layers, three pooling layers, two fully connected layers and a softmax layer [66]. AlexNet adopts dropout connections to reduce overfitting in the fully connected layers [67], and ReLU as the activation function that resides in the convolutional and fully connected layers.

- *Inception-V3 (GoogLeNet)*: GoogLeNet won the ILSVRC-2014 competition (classification and detection tracks), and was the first model to introduce the idea that the layers in CNNs do not need to be executed sequentially. GoogLeNet is a particular instance of the Google Inception architecture, which includes Inception modules interpolated with convolution, pooling, and fully connected layers. With 22 layers, GoogLeNet is a very deep architecture that combines convolutional layers of size 1×1, 3×3 and 5 × 5 with max-pooling layers of size 3 × 3. Similar to AlexNet, GoogLeNet applies dropout regularization to the ReLU activation and fully connected layer functions in the convolutional layers [26,59].
- *VGG-16*: The VGG network [27] won the identification and classification tasks in the ILSVRC 2014 competition. It was the first architecture to replace the larger filters, i.e., those that incur a more substantial computational cost, by large sequences of convolutional filters of size 3 × 3. There are several variations of the VGG model, and the VGG-16 and VGG-19 are particularly notable. In the present study, we use only the VGG-16 architecture, due to its simplicity.
- *ResNet-50*: ResNet [28] won the ILSVRC 2015 competition [65], and utilizes a residual connection to address the degradation problem by training a very deep network. It is composed of a series of residual blocks, each composed of several stacked convolutional layers. ResNet has a lower convergence loss without overfitting, and the residual connection can accelerate the convergence of deep layers. This architecture is about 20 times deeper than AlexNet, and eight times deeper than VGG [68].

Fig. 4 illustrates the main blocks in the CNN models considered in this paper: (a) LeNet-5; (b) AlexNet; (c) Inception-V3; (d) VGG-16; and (e) ResNet-50.

3.5. Training strategies

We consider two training strategies:

- *Training from Scratch*: In this approach, the neural network is trained from the very beginning, initializing all parameters randomly. During training, the values of the parameters are learned directly from the dataset in all layers, resulting in a complete training. Although this approach tends to achieve good results, it needs the dataset to be large and requires high computational effort to train the model [47].
- *Fine-Tuning*: This is a practical approach that is commonly used to train deep learning models [47] in which the network is previously trained for a classification task using a huge dataset such as ImageNet [65] or CIFAR-10 [69]. The values of the parameters (weights) learned for the initial layers of the network are then frozen, and the top layers are trained on the dataset of interest, with the aim of learning the more complex structures of the data [60]. In this work, we employ fine-tuning training for the three deepest CNN models on the ImageNet dataset: Inception-V3, VGG-16, and ResNet-50. We then freeze the initial layers of the models, while the top layers are trained using a very low learning rate. The first 197 layers of Inception-V3 are frozen, while the three initial blocks of convolutional layers are frozen for VGG-16, and all initial layers except for the last layer of the residual block for ResNet-50. The frozen and free training layers are illustrated in Fig. 4 for each CNN model.

3.6. Model evaluation

In order to compare the CNNs and the preprocessing strategies proposed in this work, we use the mean class accuracy (MCA) [70], which consists of the average of the per class accuracies defined in Eq. (4):

$$MCA = \frac{1}{n}\sum_{k=1}^{n} CCR_k, \tag{4}$$

where CCR_k is the accuracy of classification of the kth class, and n is the number of classes. The same evaluation criterion was used in the HEp-2 ICPR 2014 competition, from where our dataset was obtained.

Table 1
Hyperparameter search space used for optimization.

Hyperparameter	Distribution	Value
No. of Hidden Units n_h	Categorical	$x \in \{256, 512, 1024, 2048, 4096\}$
Weight Decay	Categorical	$x \in \{1e^{-6}, 1e^{-5}, 5e^{-4}, 5e^{-3}\}$
Learning Rate	Categorical	$x \in \{0.01, 0.001\}$
Momentum	Uniform	$x \in [0, 1]$

4. Results and discussion

All experiments were performed on a machine with an Intel i5 3.00 GHz processor, 16 GB RAM, and a GPU NVIDIA GeForce GTX Titan Xp with 12 GB memory. All experiments were programmed using Python 3.5, the Keras[2] 2.0 [71] library with TensorFlow[3] 1.4.1 [72] with CUDA version 8.0 and cuDNN 6.0. The operating system was Ubuntu 16.04.2 LTS. The hyperparameter optimization algorithm TPE was drawn from the Hyperas[4] library [52,73], version 0.1.

Our cross-validation strategy is illustrated in Fig. 5. The 13,596 images in the dataset were randomly sampled and partitioned into six stratified sets (folds). Five folds were used for training and one for testing. At each iteration of the cross-validation, one of the folds was selected for validating the trained model, and the others were used for training. A cross-validation strategy is more robust to outliers and eventual overfitting, generating results that are more reliable. Most prior studies of CNNs do not consider this strategy, probably due to the high amount of time needed to perform cross-validation, and split the data randomly into training, validation and testing groups (for example in the work by Gao et al. [21] and Rodrigues et al. [23]).

4.1. Hyperparameters estimation

To fine-tune the values of the set of hyperparameters in order to optimize the CNNs, we applied the TPE algorithm, as described in Section 3.2.1. The hyperparameter space search is summarized in Table 1 and the best values for each hyperparameter returned by the TPE algorithm are presented in Table 2, based on a CNN architecture after 30 epochs and five evaluations on the search space. These values were adopted for all CNN models of the experiments with networks trained from scratch.

To fine-tune values for the set of hyperparameters to optimize the CNNs, we applied the Tree of Parzen Estimators (TPE) algorithm, as described in Section 3.2.1. The hyperparameter space search is depicted in Table 1 and the best values for each hyperparameter returned by the TPE algorithm are presented at Table 2, considering CNN architecture after 30 epochs and 5 evaluations on the search space. These values were adopted for all CNN models of the experiments with networks trained from scratch.

[2] http://keras.io.
[3] http://tensorflow.org.
[4] http://hyperopt.github.io/hyperopt.

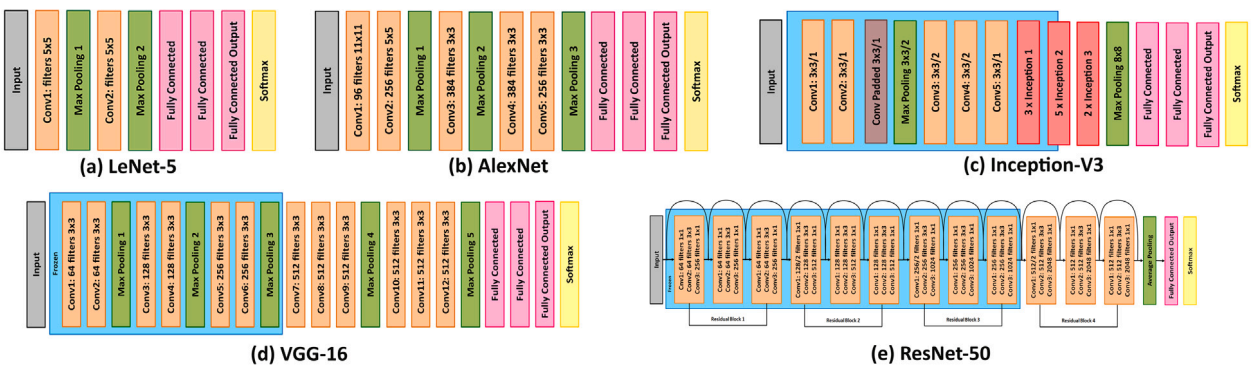


Fig. 4. Architectures of the different CNNs evaluated in this work. Blue boxes indicate the frozen layers in the CNNs to which we applied fine-tuning.

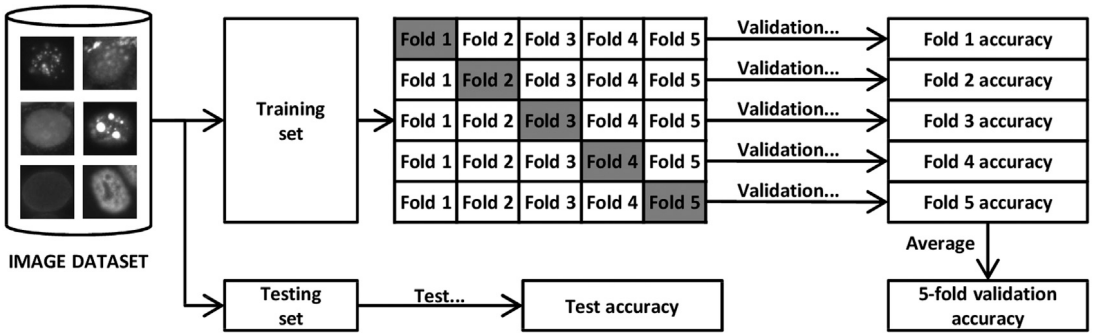


Fig. 5. Graphical representation of experimental setup with k-fold cross-validation.

Table 2
Hyperparameters optimized for each CNN with tree-structured Parzen estimator.

CNN	Hyperparameter			
	No. of hidden units	Decay	Learning rate	Momentum
LeNet-5	2048 e 2048	$1e^{-6}$	0.01	0.337626380941364
AlexNet	512 e 2048	$5e^{-4}$	0.01	0.437116259431842
Inception-V3	2048	$5e^{-4}$	0.01	0.437116259431842
VGG-16	2048	$1e^{-6}$	0.01	0.337626380941364
ResNet-50	2048	$1e^{-6}$	0.01	0.337626380941364

4.2. Assessing the impact of preprocessing

Table 3 presents the average 5-fold cross-validation accuracies and processing times for each CNN architecture, with the different preprocessing and normalization strategies. It can be seen that the best results were achieved with the original images, i.e., without contrast enhancement techniques. The LeNet-5, AlexNet, and ResNet-50 models gave better results by using average subtraction on the original images, while Inception-V3 and VGG-16 achieved the best results for the original dataset without data centralization. It is important to note that for Inception-V3 and VGG-16, the values of accuracy obtained without average subtraction are very close to those obtained with average subtraction. In terms of the best accuracy values for each architecture, Inception-V3 achieved the best performance with 96.69% accuracy, followed by AlexNet with 94.33%, VGG-16 with 92.63%, ResNet-50 with 90.73% and LeNet-5 with 90.27%.

In order to assess the values of loss and accuracy during training and validation, the evolution of these values is shown graphically in Fig. 6 for the first iteration of the k-fold. It is important to note that the validation scores follow the training scores, although some variation may be observed in the early epochs, especially for AlexNet and ResNet-50. This suggests that the training did not overfit the data, thus retaining the generalization property of the CNNs.

4.3. Assessing the impact of data augmentation

The datasets resulting from the preprocessing strategies were submitted to a data augmentation procedure based on random rotations of between 0° and 360° around the centers of the images, as described in Section 3.3.1. To evaluate the impact of this procedure on the accuracy of the CNNs, they were all trained from scratch. As shown in Table 4, data augmentation improved the accuracy for most of the CNN models, regardless of the preprocessing strategy adopted. The exception was ResNet-50, for which the accuracy score was reduced by 27.38% when histogram equalization with average subtraction was applied. After training with the augmented datasets, all architectures achieved their best results when fed with images without preprocessing, i.e., the original images, except for ResNet-50, which performed better with contrast stretching. The best results were achieved by AlexNet with 98.00% accuracy, closely followed by Inception-V3 with 97.89%, both of which were trained with original images. It is worth noticing that contrast stretching over the augmented data was the best strategy for ResNet-50, achieving a mean accuracy of 96.03% over the 5-fold validation.

Table 3
5-fold validation accuracy and average training time for each CNN model for the six preprocessing strategies.

CNN	Original	Original + Avg. Sub.	Contrast Stretching	Contrast Stretching + Avg. Sub.	Hist. Eq.	Hist. Eq. + Avg. Sub.
LeNet-5	89.57 21.45 min.	**90.27** 22.37 min.	84.01 21.44 min.	83.93 22.37 min.	69.60 21.54 min.	70.84 22.48 min.
AlexNet	91.51 39.00 min.	**94.33** 39.00 min.	90.95 39.00 min.	91.17 39.00 min.	79.39 39.00 min.	76.02 39.00 min.
Inception-V3	**96.69** 117.00 min.	96.40 117.00 min.	92.49 117.00 min.	92.15 117.00 min.	91.43 117.00 min.	87.62 117.00 min.
VGG-16	**92.63** 140.00 min.	92.54 140.00 min.	88.70 140.00 min.	88.79 140.00 min.	78.39 140.00 min.	78.43 140.00 min.
ResNet-50	85.90 132.00 min.	**90.73** 132.00 min.	86.50 132.00 min.	80.81 132.00 min.	62.47 132.00 min.	77.05 132.00 min.

Table 4
5-fold validation accuracy and average training time for each CNN model, for six preprocessing strategies with data augmentation.

CNN	Original	Original + Avg. Sub.	Contrast Stretching	Contrast Stretching + Avg. Sub.	Hist. Eq.	Hist. Eq. + Avg. Sub.
LeNet-5	**92.94** 145.23 min.	90.09 145.62 min.	90.07 145.29 min.	90.66 145.80 min.	84.22 145.22 min.	84.88 145.79 min.
AlexNet	**98.00** 145.38 min.	96.92 146.00 min.	95.16 145.51 min.	96.16 145.86 min.	80.94 145.51 min.	78.55 146.01 min.
Inception-V3	**97.89** 151.08 min.	97.88 151.67 min.	96.19 151.32 min.	95.56 151.86 min.	88.92 151.42 min.	89.83 151.82 min.
VGG-16	**97.17** 152.48 min.	97.12 152.66 min.	94.93 152.43 min.	95.57 152.88 min.	89.35 152.47 min.	89.68 152.75 min.
ResNet-50	88.00 153.00 min.	88.13 153.51 min.	**96.03** 153.14 min.	92.09 153.56 min.	59.94 153.16 min.	49.67 153.62 min.

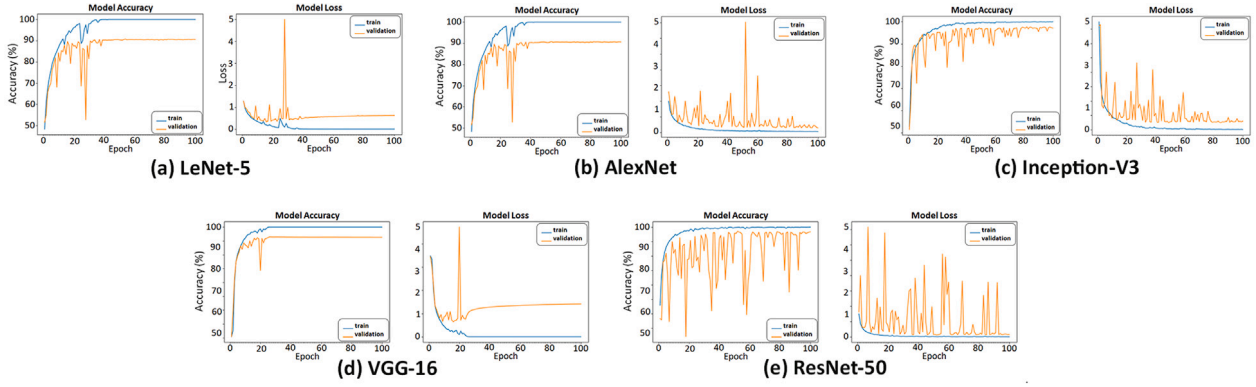


Fig. 6. Charts showing the evolution of accuracy and loss values for the validation set.

4.4. Comparison of architectures

In order to compare the five CNN models trained from scratch, the best preprocessing strategy for each model was used, based on the highest mean accuracy at the validation stage, as shown in Tables 3 and 4. For LeNet, AlexNet, Inception-V3, and VGG-16, the augmented dataset without preprocessing was chosen, while for ResNet-50, the augmented dataset with contrast stretching was chosen. New models were then trained on the whole training set (the five folds) and tested using the remaining fold. The values of the mean validation accuracy, test accuracy, and training time for each CNN model in these scenarios are presented in Table 5. As a check on the learning behavior of the filters embedded in the convolutional layers, Fig. 7 also shows some of the feature maps acquired in selected convolutional layers of the CNNs used. Most of the features behaved as texture extractors and edge detectors. It can be seen that the spatial information of the HEp-2 cell was preserved in the feature maps, which become more and more abstract at higher network layers.

Inception-V3 achieved the best testing accuracy of 98.28%, followed by VGG-16 and ResNet-50 with 96.82% and 96.38%, respectively. The worst test score of 94.35% was obtained by both AlexNet and LeNet-5, although AlexNet had the best validation accuracy, as shown in Table 5. This indicates that AlexNet was not able to generalize the test data as well as it could the validation data, probably due to over-fitting.

The confusion matrices for each CNN model are presented in Table 6, and illustrate several aspects of the classification problem investigated in this work. Note that except for Inception-V3, the CNNs misclassified a considerable proportion of the homogeneous cells (and other types) as speckled; the latter was one of the most misclassified types of cell for all CNNs, and this may require attention in future investigations of HEp-2 cells. Other less relevant misclassifications occurred, and varied with the CNN model.

Table 5
5-fold average accuracy and test accuracy for each CNN model, using the best preprocessing strategy.

CNN	Validation accuracy (%)	Test accuracy (%)
LeNet-5 (original + data augmentation)	92.94	94.35
AlexNet (original + data augmentation)	98.00	94.35
Inception-V3 (original + data augmentation)	97.89	98.28
VGG-16 (original + data augmentation)	97.17	96.82
ResNet-50 (contrast stretching + data augmentation)	96.03	96.38

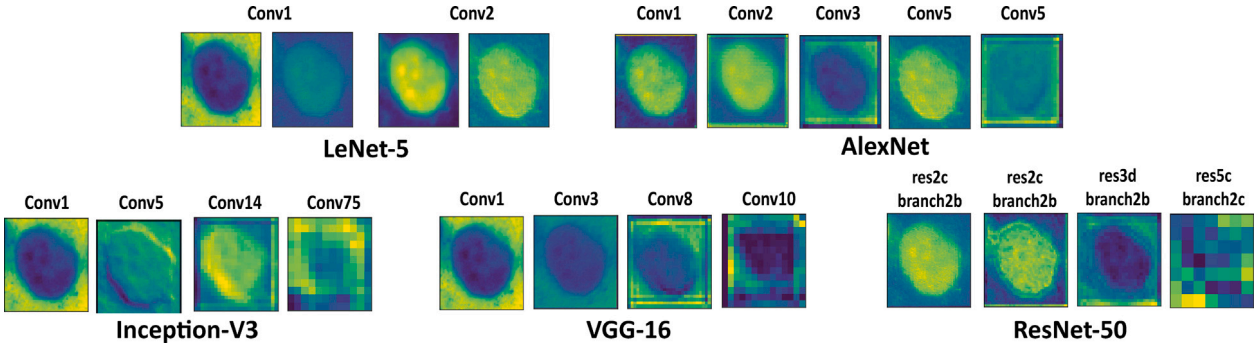


Fig. 7. Feature maps for some of the convolutional layers of the CNNs.

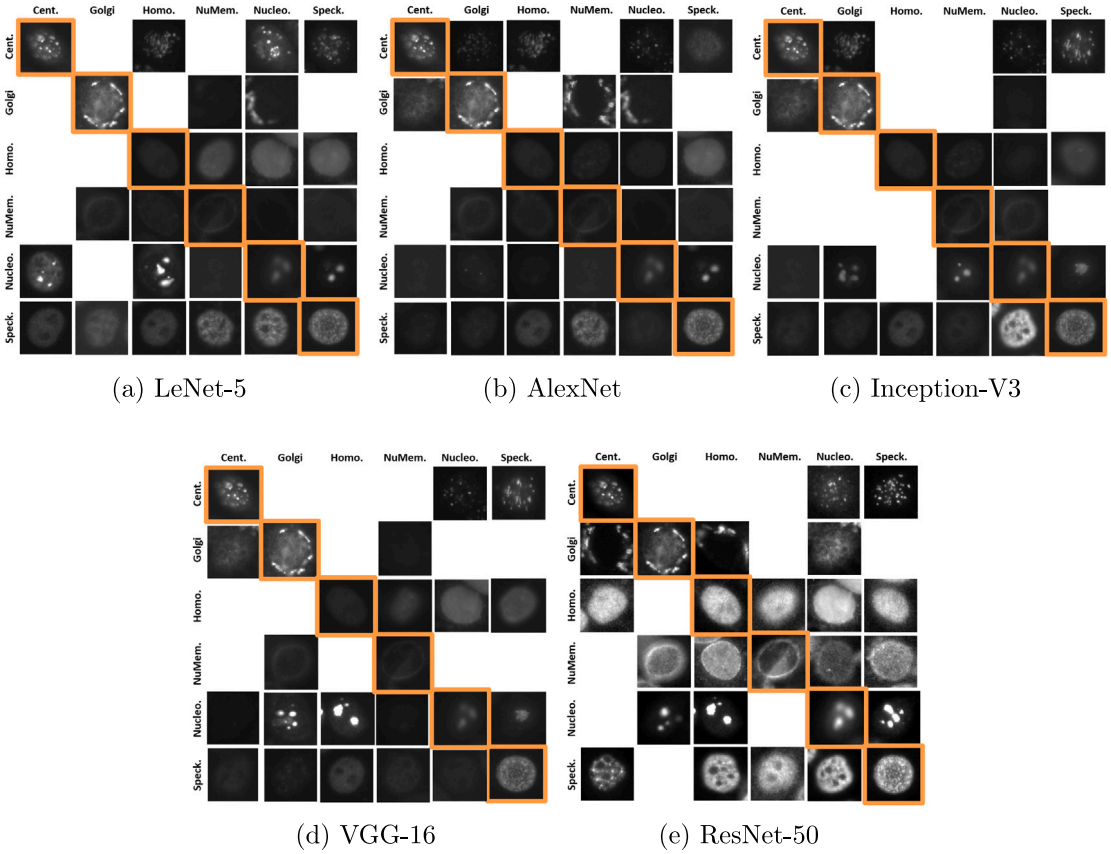


Fig. 8. Examples of correctly classified and misclassified images in accordance with the confusion matrices.

Several samples for each CNN model tested are shown in Fig. 8 allowing for a visual analysis. The rows represent the "labele" class, the columns represent the predicted class and the main diagonals are examples of correctly classified cells. White squares represent results without misclassifications.

Note that the dataset shows some imbalance between the classes: the smallest class (Golgi) is about 25% of the size of the largest (Speckled). However, the performance obtained in our study for the Golgi class was very similar to that for the other classes, as reflected in the number of true positives for this class in the confusion matrices.

Table 6
Confusion matrix of the best results for each CNN model.

	Cent.	Golgi	Homo.	NuMem.	Nucleo.	Speck.
(a) LeNet-5 (original + data augmentation).						
Cent.	**93.22**	0.00	0.22	0.00	3.06	3.50
Golgi	0.00	**96.69**	0.00	1.65	1.65	0.00
Homo.	0.00	0.00	**91.57**	1.69	0.72	6.02
NuMem.	0.00	0.54	0.54	**98.10**	0.54	0.27
Nucleo.	0.92	0.00	0.46	0.46	**94.69**	3.46
Speck.	1.48	0.64	1.69	0.64	1.48	**94.07**
(b) AlexNet (original + data augmentation)						
Cent.	**94.97**	0.22	0.22	0.00	2.41	2.19
Golgi	1.65	**94.21**	1.65	1.65	0.83	0.00
Homo.	0.00	0.00	**95.66**	1.45	0.72	2.17
NuMem.	0.00	1.63	2.17	**95.38**	0.27	0.54
Nucleo.	0.92	0.23	1.62	0.23	**94.46**	2.54
Speck.	1.91	0.21	4.24	0.85	1.06	**91.93**
(c) Inception-V3 (original + data augmentation)						
Cent.	**99.12**	0.22	0.00	0.00	0.22	0.44
Golgi	2.48	**96.69**	0.00	0.00	0.83	0.00
Homo.	0.00	0.00	**98.55**	0.72	0.24	0.48
NuMem.	0.00	0.00	0.00	**99.46**	0.54	0.00
Nucleo.	0.23	0.23	0.00	0.23	**98.61**	0.69
Speck.	0.64	0.21	1.91	0.64	0.21	**96.40**
(d) VGG-16 (original + data augmentation)						
Cent.	**98.91**	0,00	0.00	0.00	0.66	0,44
Golgi	2.48	**94,21**	0.00	3.31	0.00	0.00
Homo.	0.00	0.00	**94.46**	2.41	0.72	2.41
NuMem.	0,00	0.27	0.00	**99.73**	0.00	0.00
Nucleo.	0.46	0.23	0.23	0.23	**98.15**	0.69
Speck.	1.69	0.00	2.75	0.85	0.64	**94.07**
(e) ResNet-50 (contrast stretching + data augmentation)						
Cent.	**94.53**	0.00	0.00	0.00	4.38	1.09
Golgi	0.83	**95.04**	0.83	0.00	3.31	0.00
Homo.	0.24	0.00	**97.83**	0.24	0.24	1.45
NuMem.	0.00	1.36	1.09	**96.74**	0.27	0.54
Nucleo.	0.00	0.46	0.23	0.00	**98.61**	0.69
Speck.	0.21	0.00	3.81	0.21	0.85	**94.92**

These results therefore demonstrate that the imbalance in the dataset did not bias the classification performance.

4.5. Training from scratch vs. transfer learning

In addition to training the CNN models from scratch, we also fine-tuned previously trained models in order to compare their performance, as described in Section 3.5. For this fine-tuning, the Inception-V3, VGG-16, and ResNet-50 architectures were initialized using parameters from models trained on the large ImageNet dataset [65]. Following this, the last layer (Softmax) was replaced with a new one with six neurons, in order to adapt the CNN to our classification problem (which has six classes). We froze all connection and topology values in the first 197 layers of Inception, the three initial blocks of the convolutional layers in VGG-16, and all initial layers of the residual block, except for the last one, from ResNet (as described in Section 3.5 and illustrated in Fig. 4(c), (d) and (e)). The CNNs were then retrained by updating the top layers during the fine-tuning process and using a reduced learning rate based on values from the literature [46].

Table 7 shows the average accuracies and processing times for the 5-fold cross-validation using fine-tuning training for each preprocessing strategy. VGG-16 achieved the best accuracy with contrast stretching and average subtraction (96.07%) when fine-tuning was adopted. Compared with the full training results in Table 3, fine-tuning improved the accuracy values for almost all CNNs, and especially for VGG-16 with histogram equalization, for which the accuracy was improved by 15.25 points. The exception was Inception-V3. For all CNN models, preprocessing strategies with and without average subtraction gave very similar accuracies.

To assess the impact of data augmentation on fine-tuning training, we carried out experiments with fine-tuning and data augmentation, and the results are presented in Table 8. The best accuracy values were obtained when VGG-16 was trained using contrast stretching and average subtraction, followed by ResNet trained with original images, and finally by Inception-V3 using original images with average subtraction. The values of accuracy for the experiments done with and without average subtraction were very close. When we compare the results of fine-tuning training with and without data augmentation (Table 7), data augmentation improved the accuracies in almost all cases, except for VGG-16 trained with original images. When trained with data augmentation and average subtraction, the accuracy of Inception-V3 and ResNet-50 increased by 2.46 and 2.33 percentage points, respectively. The accuracy of the VGG-16 model (with contrast stretching and average image subtraction) increased by 1.24 percentage points compared with fine-tuning training without data augmentation. Fine-tuning generated significant improvements for the ResNet-50 model for all preprocessing strategies, compared to complete training and data augmentation (Table 4).

To compare the CNNs trained with fine-tuning, we selected the best preprocessing strategies for each model and ran a new training process for the whole training set. We then assessed the accuracy for the testing set, in a similar way to the procedure described in the previous sections.

Since the best results when training with fine-tuning (Tables 7 and 8) were obtained with data augmentation, we used this technique again for all CNNs. The resulting values are presented in Table 9. In terms of the accuracy of the test set, VGG-16 trained with contrast stretching and average subtraction achieved the best results, achieving 96.69%.

As shown in Tables 7, 8, and 9, the fine-tuning training improved the results only when data augmentation was not applied, and the best results for fine-tuning training were obtained with data augmentation. However, when we used data augmentation procedures, the best results were achieved by training the models from scratch. Table 11 in Appendix summarizes the results of all of the experiments presented in this paper.

4.6. Comparison with literature

The best result achieved in this study for the test accuracy is compared with other state-of-art work in the literature. The best accuracy in our work was obtained with Inception-V3, trained from scratch using augmented data without preprocessing or average subtraction, which scored 98.28% (as shown in Table 5). The best results reported in the literature are presented in Table 10 for the same HEp-2 dataset, which was created for ICPR 2014. It can be seen that our best accuracy score is close to the best state-of-the-art technique reported in the literature.

5. Conclusion

The results presented in this work allow us to conclude that for the classification of HEp-2 cell images, the majority of the CNNs studied here obtained better results when trained from scratch, using augmented datasets and with no other preprocessing strategy. Using 5-fold cross-validation, the best preprocessing strategies when training a CNN from scratch were the use of original images with data augmentation for LeNet-5, AlexNet and Inception-V3, while the best results for ResNet-50 were obtained using contrast stretching with data augmentation. We evaluated the CNN models trained with the best preprocessing strategies when using k-fold cross-validation on the test set, and the best result was achieved by Inception-V3, with an accuracy of 98.28%, followed by VGG-16 with 96.82%, and ResNet-50 with 96.38%. LeNet-5 and AlexNet both obtained the lowest accuracy of 94.35%.

Table 7
5-fold validation accuracy and training time for each CNN model training with fine-tuning.

CNN	Original	Original + Avg. Sub.	Contrast Stretching	Contrast Stretching + Avg. Sub.	Hist. Eq	Hist. Eq. + Avg. Sub.
Inception-V3	90.50 67.08 min.	**90.55** 66.87 min.	90.01 67.33 min.	90.00 66.84 min.	85.78 67.02 min.	85.87 66.93 min.
VGG-16	94.32 76.29 min.	94.25 76.44 min.	95.97 76.72 min.	**96.07** 76.75 min.	93.64 77.20 min.	93.57 77.23 min.
ResNet-50	93.31 61.77 min.	**93.37** 61.67 min.	92.89 61.87 min.	92.95 61.83 min.	90.21 61.78 min.	90.11 61.72 min.

Table 8
5-fold validation accuracy and training time for each CNN model trained with fine tuning and data augmentation.

CNN	Original	Original + Avg. Sub.	Contrast Stretching	Contrast Stretching + Avg. Sub.	Hist. Eq.	Hist. Eq. + Avg. Sub.
Inception-V3	92.97 149.18 min.	**93.01** 149.71 min.	90.48 149.47 min.	90.48 149.79 min.	87.15 149.27 min.	87.32 149.81 min.
VGG-16	93.32 150.11 min.	93.39 150.73 min.	97.24 150.39 min.	**97.31** 150.94 min.	95.85 150.23 min.	96.15 150.91 min.
ResNet-50	**95.91** 150.39 min.	95.70 150.97 min.	94.06 150.53 min.	94.08 151.18 min.	92.63 150.61 min.	92.69 151.09 min.

Table 9
5-fold validation accuracy and test accuracy for each CNN model trained by fine-tuning, using the best preprocessing strategies.

CNN	Validation accuracy (%)	Test accuracy (%)
Inception-V3 (original + avg. sub. + data augmentation)	93.01	92.71
VGG-16 (contrast stretching + avg. sub. + data augmentation)	97.31	96.69
ResNet-50 (original + data augmentation)	95.91	95.50

Table 10
Highest accuracy of other classification methods using the HEp-2 dataset from ICPR 2014.

Method	Accuracy (%)
Gragnaniello et al. (2016) [35]	82.60
Manivannan et al. (2016) [13]	87.10
Kastaniotis et al. (2017) [36]	82.30
Gao et al. (2017) [21]	88.58
Gao et al. (2017) (with data aug.) [21]	96.76
Faria et al. (2018) [40]	84.97
Lei et al. (2018) [44]	98.42
Our work (2019)(Inception-V3, no preprocessing + data aug.)	**98.28**

Although fine-tuning technique appears to be a promising approach, it did not achieve higher accuracy values than the CNNs trained from scratch, especially when data augmentation was used. It could be used in scenarios with previously trained networks and where only a short time is available for training, as fine-tuning techniques require less time to train the unfrozen layers. The best results were achieved by VGG-16 using data augmentation and contrast stretching and average subtraction, with an accuracy of 96.69% for the testing set.

A combination of the proposed methods may be useful, allowing health agents to choose the best strategy to use in the automatic classification of HEp-2 cells and to identify autoimmune conditions by revealing the diseases that cause them. This work also represents an important step in the investigation of different preprocessing techniques, as multiple CNN architectures were compared and the best result of 98.28% was very close to the highest accuracy score presented in the literature; this result was obtained with Inception-V3 trained from scratch using augmented data with no other preprocessing technique.

In future work, the following avenues may be considered: (a) studying further optimization algorithms and their impact with a more extensive set of hyperparameters; (b) testing more data augmentation strategies, such as vertical and horizontal flips, rotations through random angles, random crops, variation in the intensity of the images, etc; (c) applying new normalization strategies, such as standard deviation normalization and ZCA whitening; (d) fusion between different CNNs using different topologies; and (e) acquiring and testing the proposed strategies with other datasets of images containing HEp-2 cells and other types of cells, in order to verify that our findings hold for similar datasets.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We gratefully acknowledge the support of NVIDIA Corporation, Brazil with the donation of the TITAN Xp GPU used for this research. We would like to thanks CAPES, Brazil, FUNARBE, Brazil and FAPEMIG, Brazil (Grant number CEX - APQ-02964-17) for financial support. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. We would also like to thank Dr. Ricardo R. Oliveira Neto and Prof. Dr. John Sessions for English language review, and Prof. Dr. Jugurta Lisboa Filho for the support.

Appendix

See Table 11.

Table 11
Summary of all experiments presented in this paper, with 5-fold cross-validation accuracy, training time, and test accuracy (between parentheses).

		Original		Original + Avg. Sub.		Contrast stretching		Contrast Stretching + Avg. Sub.		Histogram Equalization		Histogram Equalization + Avg. Sub.	
			D.A.		D.A.		D.A.		D.A.		D.A.		D.A.
From Scratch	LeNet-5	89.57 21.45 min.	**92.94** **145.23 min.** **(Test: 94.35)**	90.27 22.37 min.	90.09 145.62 min.	84.01 21.44 min.	90.07 145.29 min.	83.93 22.37 min.	90.66 145.80 min.	69.60 21.54 min.	84.22 145.22 min.	70.84 22.48 min.	84.88 145.79 min.
	AlexNet	91.51 39.00 min.	**98.00** **145.38 min.** **(Test: 94.35)**	94.33 39.00 min.	96.92 146.00 min.	90.95 39.00 min.	95.16 145.51 min.	91.17 39.00 min.	96.16 145.86 min.	79.39 39.00 min.	80.94 145.51 min.	76.02 39.00 min.	78.55 146.01 min.
	Inception-V3	96.69 117.00 min.	**97.89** **151.08 min.** **(Test: 98.28)**	96.40 117.00 min.	97.88 151.67 min.	92.49 117.00 min.	96.19 151.32 min.	92.15 117.00 min.	95.56 151.86 min.	91.43 117.00 min.	88.92 151.42 min.	87.62 117.00 min.	89.83 151.82 min.
	VGG-16	92.63 140.00 min.	97.17 152.48 min.	92.54 140.00 min.	97.12 152.66 min.	88.70 140.00 min.	94.93 152.43 min.	88.79 140.00 min.	95.57 152.88 min.	78.39 140.00 min.	89.35 152.47 min.	78.43 140.00 min.	89.68 152.75 min.
	ResNet-50	85.90 132.00 min.	88.00 153.00 min.	90.73 132.00 min.	88.13 153.51 min.	86.50 132.00 min.	**96.03** **153.56 min.** **(Test: 96.38)**	80.81 132.00 min.	92.09 153.56 min.	62.47 132.00 min.	59.94 153.16 min.	77.05 132.00 min.	49.67 153.62 min.
Fine-Tuning	Inception-V3	90.50 67.08 min.	**92.97** **149.18 min.** **(Test: 92.71)**	90.55 66.87 min.	93.01 149.71 min.	90.01 67.33 min.	90.48 149.47 min.	90.00 66.84 min.	90.48 149.79 min.	85.78 67.02 min.	87.15 149.27 min.	85.87 66.93 min.	87.32 149.81 min.
	VGG-16	94.32 76.29 min.	93.32 150.11 min.	94.25 76.44 min.	93.39 150.73 min.	95.97 76.72 min.	**97.24** **150.39 min.** **(Test: 96.69)**	96.07 76.75 min.	97.31 150.94 min.	93.64 77.20 min.	95.85 150.23 min.	93.57 77.23 min.	96.15 150.91 min.
	ResNet-50	93.31 61.77 min.	**95.91** **150.39 min.** **(Test: 95.50)**	93.37 61.67 min.	95.70 150.97 min.	92.89 61.87 min.	94.06 150.53 min.	92.95 61.83 min.	94.08 151.18 min.	90.21 61.78 min.	92.63 150.61 min.	90.11 61.72 min.	92.69 151.09 min.

References

[1] A. Davidson, B. Diamond, Chapter 3 - general features of autoimmune disease, in: N.R. Rose, I.R. Mackay (Eds.), The Autoimmune Diseases (Fifth Edition), fifth ed., Academic Press, Boston, 2014, pp. 19–37, http://dx.doi.org/10.1016/B978-0-12-384929-8.00003-4.

[2] K. Watanabe, V. Karuppagounder, R. Sreedhar, M. Harima, S. Arumugam, Chapter 12 - kampo medicines for autoimmune disorders: Rheumatoid arthritis and autoimmune diabetes mellitus, in: S. Arumugam, K. Watanabe (Eds.), Japanese Kampo Medicines for the Treatment of Common Diseases: Focus on Inflammation, Academic Press, 2017, pp. 103–110, http://dx.doi.org/10.1016/B978-0-12-809398-6.00012-3.

[3] A. Zhernakova, S. Withoff, C. Wijmenga, Clinical implications of shared genetics and pathogenesis in autoimmune diseases, Nature Rev. Endocrinol. 9 (11) (2013) 646–659, http://dx.doi.org/10.1038/nrendo.2013.161.

[4] J.-F. Bach, The effect of infections on susceptibility to autoimmune and allergic diseases, New Engl. J. Med. 347 (12) (2002) 911–920, http://dx.doi.org/10.1056/NEJMra020100.

[5] L. Chatenoud, Precision medicine for autoimmune disease, Nat. Biotech. 34 (9) (2016) 930–932, News and Views.

[6] A.S. Wiik, M. Høier-Madsen, J. Forslid, P. Charles, J. Meyrowitsch, Antinuclear antibodies: A contemporary nomenclature using HEp-2 cells, J. Autoimmunity 35 (3) (2010) 276–290 (Special Issue - In Honour of Professor Haralampos M. Moutsopoulos), http://dx.doi.org/10.1016/j.jaut.2010.06.019.

[7] P.L. Meroni, P.H. Schur, ANA screening: an old test with new recommendations, Ann. Rheum. Dis. 69 (8) (2010) 1420–1422, http://dx.doi.org/10.1136/ard.2009.127100.

[8] K. Egerer, D. Roggenbuck, R. Hiemann, M.-G. Weyer, T. Büttner, B. Radau, R. Krause, B. Lehmann, E. Feist, G.-R. Burmester, Automated evaluation of autoantibodies on human epithelial-2 cells as an approach to standardize cell-based immunofluorescence tests, Arthritis Res. Therapy 12 (2) (2010) 1–9, http://dx.doi.org/10.1186/ar2949.

[9] R. Tozzoli, C. Bonaguri, A. Melegari, A. Antico, D. Bassetti, N. Bizzaro, Current state of diagnostic technologies in the autoimmunology laboratory, Clin. Chem. Lab. Med. 51 (1) (2013) 129–138.

[10] M.J. Fritzler, The antinuclear antibody test: last or lasting gasp? Arthritis Rheum. 63 (1) (2011) 19–22.

[11] D.H. Solomon, A.J. Kavanaugh, P.H. Schur, Evidence-based guidelines for the use of immunologic tests: antinuclear antibody testing, Arthritis Care Res. 47 (4) (2002) 434–444.

[12] P. Foggia, G. Percannella, A. Saggese, M. Vento, Pattern recognition in stained HEp-2 cells: Where are we now? Pattern Recognit. 47 (7) (2014) 2305–2314, http://dx.doi.org/10.1016/j.patcog.2014.01.010.

[13] S. Manivannan, W. Li, S. Akbar, R. Wang, J. Zhang, S.J. McKenna, An automated pattern recognition system for classifying indirect immunofluorescence images of HEp-2 cells and specimens, Pattern Recognit. 51 (2016) 12–26, http://dx.doi.org/10.1016/j.patcog.2015.09.015.

[14] L. Nanni, S. Ghidoni, S. Brahnam, Ensemble of convolutional neural networks for bioimage classification, Appl. Comput. Inform. (2018) http://dx.doi.org/10.1016/j.aci.2018.06.002.

[15] L. Nanni, S. Brahnam, S. Ghidoni, A. Lumini, Bioimage classification with handcrafted and learned features, IEEE/ACM Trans. Comput. Biol. Bioinform. 16 (3) (2019) 874–885, http://dx.doi.org/10.1109/TCBB.2018.2821127.

[16] K. Doi, Computer-aided diagnosis in medical imaging: Historical review, current status and future potential, Comput. Med. Imaging Graph. 31 (4) (2007) 198–211, Computer-aided Diagnosis (CAD) and Image-guided Decision Support, http://dx.doi.org/10.1016/j.compmedimag.2007.02.002.

[17] G. Dougherty, Digital Image Processing for Medical Applications, Cambridge University Press, 2009, http://dx.doi.org/10.1017/CBO9780511609657.

[18] Q. Wu, F. Merchant, K. Castleman, Microscope Image Processing, Academic Press, 2010.

[19] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, vol. 25, Curran Associates, Inc., 2012, pp. 1097–1105.

[20] A.S. Razavian, H. Azizpour, J. Sullivan, S. Carlsson, CNN Features off-the-shelf: An astounding baseline for recognition, in: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPRW '14, IEEE Computer Society, Washington, DC, USA, 2014, pp. 512–519, http://dx.doi.org/10.1109/CVPRW.2014.131.

[21] Z. Gao, L. Wang, L. Zhou, J. Zhang, HEp-2 cell image classification with deep convolutional neural networks, IEEE J. Biomed. Health Inf. 21 (2) (2017) 416–428, http://dx.doi.org/10.1109/JBHI.2016.2526603.

[22] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324, http://dx.doi.org/10.1109/5.726791.

[23] L.F. Rodrigues, M.C. Naldi, J.F. Mari, Exploiting convolutional neural networks and preprocessing techniques for HEp-2 cell classification in immunofluorescence images, in: 2017 30th SIBGRAPI Conference on Graphics, Patterns and Images, SIBGRAPI, 2017, pp. 170–177, http://dx.doi.org/10.1109/SIBGRAPI.2017.29.

[24] P.A. Devijver, J. Kittler, Pattern Recognition: A Statistical Approach, Prentice-Hall, 1982.

[25] L.F. Rodrigues, M.C. Naldi, J.F. Mari, HEp-2 cell image classification based on convolutional neural networks, in: 2017 Workshop of Computer Vision, WVC, 2017, pp. 13–18, http://dx.doi.org/10.1109/WVC.2017.00010.

[26] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2016, pp. 2818–2826, http://dx.doi.org/10.1109/CVPR.2016.308.

[27] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, CoRR arXiv:abs/1409.1556.

[28] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2016, pp. 770–778, http://dx.doi.org/10.1109/CVPR.2016.90.

[29] P. Perner, H. Perner, B. Müller, Mining knowledge for HEp-2 cell image classification, Artif. Intell. Med. 26 (1–2) (2002) 161–173, Medical Data Mining and Knowledge Discovery, http://dx.doi.org/10.1016/S0933-3657(02)00057-X.

[30] U. Sack, S. Knoechner, H. Warschkau, U. Pigla, F. Emmrich, M. Kamprad, Computer-assisted classification of HEp-2 immunofluorescence patterns in autoimmune diagnostics, Autoimmun. Rev. 2 (5) (2003) 298–304, http://dx.doi.org/10.1016/S1568-9972(03)00067-3.

[31] R. Nosaka, K. Fukui, HEp-2 cell classification using rotation invariant co-occurrence among local binary patterns, Pattern Recognit. 47 (7) (2014) 2428–2436, http://dx.doi.org/10.1016/j.patcog.2013.09.018.

[32] R. Stoklasa, T. Majtner, D. Svoboda, Efficient k-NN based HEp-2 cells classifier, Pattern Recognit. 47 (7) (2014) 2409–2418, http://dx.doi.org/10.1016/j.patcog.2013.09.021.

[33] L. Shen, J. Lin, S. Wu, S. Yu, HEp-2 image classification using intensity order pooling based features and bag of words, Pattern Recognit. 47 (7) (2014) 2419–2427, http://dx.doi.org/10.1016/j.patcog.2013.09.020.

[34] A. Taalimi, S. Ensafi, H. Qi, S. Lu, A.A. Kassim, C.L. Tan, Multimodal dictionary learning and joint sparse representation for HEp-2 cell classification, in: N. Navab, J. Hornegger, W.M. Wells, A.F. Frangi (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, Springer International Publishing, Cham, 2015, pp. 308–315.

[35] D. Gragnaniello, C. Sansone, L. Verdoliva, Cell image classification by a scale and rotation invariant dense local descriptor, Pattern Recognit. Lett. 82, Part 1 (2016) 72–78, Pattern recognition Techniques for Indirect Immunofluorescence Images Analysis, http://dx.doi.org/10.1016/j.patrec.2016.01.007.

[36] D. Kastaniotis, F. Fotopoulou, I. Theodorakopoulos, G. Economou, S. Fotopoulos, HEp-2 cell classification with vector of hierarchically aggregated residuals, Pattern Recognit. 65 (2017) 47–57, http://dx.doi.org/10.1016/j.patcog.2016.12.013.

[37] S. Ensafi, S. Lu, A.A. Kassim, C.L. Tan, Sparse non-parametric Bayesian model for HEP-2 cell image classification, in: 2015 IEEE 12th International Symposium on Biomedical Imaging, ISBI, 2015, pp. 679–682, http://dx.doi.org/10.1109/ISBI.2015.7163964.

[38] S. Ensafi, S. Lu, A.A. Kassim, C.L. Tan, Accurate HEp-2 cell classification based on sparse coding of superpixels, Pattern Recognit. Lett. 82 (2016) 64–71, Pattern recognition Techniques for Indirect Immunofluorescence Images Analysis, http://dx.doi.org/10.1016/j.patrec.2016.02.007.

[39] S. Ensafi, S. Lu, A.A. Kassim, C.L. Tan, Accurate HEp-2 cell classification based on sparse bag of words coding, Comput. Med. Imaging Graph. 57 (2017) 40–49, Recent Developments in Machine Learning for Medical Imaging Applications, http://dx.doi.org/10.1016/j.compmedimag.2016.08.00.

[40] L.C. de Faria, L.F. Rodrigues, J.F. Mari, Cell classification using handcrafted features and bag of visual words, in: 2018 Workshop de Visão Computacional, WVC, 2018, pp. 68–73.

[41] P. Foggia, G. Percannella, P. Soda, M. Vento, Benchmarking HEp-2 cells classification methods, IEEE Trans. Med. Imaging 32 (10) (2013) 1878–1889, http://dx.doi.org/10.1109/TMI.2013.2268163.

[42] Y. Li, L. Shen, HEP-net: a smaller and better deep-learning network for HEp-2 cell classification, Comput. Methods Biomech. Biomed. Eng. Imaging Vis. 7 (3) (2018) 266–272, http://dx.doi.org/10.1080/21681163.2018.1449140.

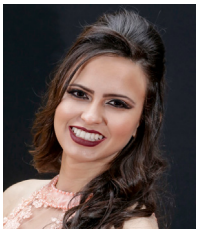
[43] K. Gupta, A. Bhavsar, A.K. Sao, Detecting mitotic cells in HEp-2 images as anomalies via one class classifier, Comput. Biol. Med. (2019) 103328, http://dx.doi.org/10.1016/j.compbiomed.2019.103328.

[44] H. Lei, T. Han, F. Zhou, Z. Yu, J. Qin, A. Elazab, B. Lei, A deeply supervised residual network for HEp-2 cell classification via cross-modal transfer learning, Pattern Recognit. 79 (2018) 290–302, http://dx.doi.org/10.1016/j.patcog.2018.02.006.

[45] P. Hobson, B.C. Lovell, G. Percannella, M. Vento, A. Wiliem, Classifying anti-nuclear antibodies HEp-2 images: A benchmarking platform, in: 2014 22nd International Conference on Pattern Recognition, 2014, pp. 3233–3238, http://dx.doi.org/10.1109/ICPR.2014.557.

[46] Y. Bengio, Practical recommendations for gradient-based training of deep architectures, in: Neural Networks: Tricks of the Trade: Second Edition, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 437–478, http://dx.doi.org/10.1007/978-3-642-35289-8_26.

[47] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, http://www.deeplearningbook.org.
[48] B. Polyak, Some methods of speeding up the convergence of iteration methods, USSR Comput. Math. Math. Phys. 4 (5) (1964) 1–17, http://dx.doi.org/10.1016/0041-5553(64)90137-5.
[49] I. Sutskever, J. Martens, G. Dahl, G. Hinton, On the importance of initialization and momentum in deep learning, in: Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28, ICML'13, JMLR.org, 2013, pp. III–1139–III–1147.
[50] J.S. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter optimization, in: J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, K.Q. Weinberger (Eds.), Advances in Neural Information Processing System, vol. 24, Curran Associates, Inc., 2011, pp. 2546–2554, URL http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf.
[51] F. Hutter, H.H. Hoos, K. Leyton-Brown, Sequential model-based optimization for general algorithm configuration, in: C.A.C. Coello (Ed.), Learning and Intelligent Optimization, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 507–523.
[52] J. Bergstra, D. Yamins, D.D. Cox, Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures, in: Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28, ICML'13, JMLR.org, 2013, pp. I–115–I–123.
[53] D.R. Jones, A taxonomy of global optimization methods based on response surfaces, J. Global Optim. 21 (4) (2001) 345–383, http://dx.doi.org/10.1023/A:1012771025575.
[54] R.C. Gonzalez, R.E. Woods, Digital Image Processing, third ed., Prentice Hall, 2007.
[55] Y.A. LeCun, L. Bottou, G.B. Orr, K.-R. Müller, Efficient backprop, in: Neural Networks: Tricks of the Trade, Springer, 2012, pp. 9–48.
[56] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, M.S. Lew, Deep learning for visual understanding: A review, Neurocomputing 187 (2016) 27–48, Recent Developments on Deep Big Vision, http://dx.doi.org/10.1016/j.neucom.2015.09.116.
[57] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[58] D. Scherer, A. Müller, S. Behnke, Evaluation of pooling operations in convolutional architectures for object recognition, in: K. Diamantaras, W. Duch, L.S. Iliadis (Eds.), Artificial Neural Networks – ICANN 2010, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 92–101.
[59] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2015, pp. 1–9, http://dx.doi.org/10.1109/CVPR.2015.7298594.
[60] M.A. Ponti, L.S.F. Ribeiro, T.S. Nazare, T. Bui, J. Collomosse, Everything you wanted to know about deep learning for computer vision but were afraid to ask, in: 2017 30th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials, SIBGRAPI-T, 2017, pp. 17–41, http://dx.doi.org/10.1109/SIBGRAPI-T.2017.12.
[61] A. Cauchy, Méthode générale pour la résolution des systemes d'équations simultanées, C. R. Sci. Paris 25 (1847) (1847) 536–538.
[62] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Mach. Learn. Res. 12 (2011) 2121–2159.
[63] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, CoRR abs/1412.6980 arXiv:1412.6980.
[64] T. Tieleman, G. Hinton, Lecture 6.5 - rmsprop: Divide the gradient by a running average of its recent magnitude, COURSERA: Neural Netw. Mach. Learn. 4 (2) (2012) 26–31.
[65] J. Deng, W. Dong, R. Socher, L.J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255, http://dx.doi.org/10.1109/CVPR.2009.5206848.
[66] H. Shin, 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 learning, IEEE Trans. Med. Imaging 35 (5) (2016) 1285–1298, http://dx.doi.org/10.1109/TMI.2016.2528162.
[67] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[68] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, T. Chen, Recent advances in convolutional neural networks, Pattern Recognit. 77 (2018) 354–377, http://dx.doi.org/10.1016/j.patcog.2017.10.013.
[69] A. Krizhevsky, Learning Multiple Layers of Features from Tiny Images, Tech. Rep., University of Toronto, 2009.
[70] B.C. Lovell, G. Percannella, M. Vento, A. Wiliem, Performance Evaluation of Indirect Immunofluorescence Image Analysis Systems, Tech. Rep., ICPR Workshop, 2014.
[71] F. Chollet, et al., Keras, 2015, https://keras.io.
[72] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, Tensorflow: Large-scale machine learning on heterogeneous systems, 2015, Software available from tensorflow.org.
[73] J. Bergstra, D. Yamins, D.D. Cox, Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms, in: Proceedings of the 12th Python in Science Conference, Citeseer, 2013, pp. 13–20.



Larissa F. Rodrigues received the B.Sc. degree in Computer Information Systems from the Federal University of Viçosa, Rio Paranaíba/MG, Brazil, in 2016, and the M.Sc. degree in Computer Science from the Federal University of Viçosa (UFV), Viçosa/MG, Brazil, in 2018. Her research interests include Image Processing, Computer Vision and Machine Learning.

Murilo C. Naldi received the B.Sc., M.Sc. and Ph.D. degrees in Computer Science in, respectively, 2004, 2006, and 2011, from the University of São Paulo (USP), Brazil. He is a fellow of the Brazilian Computing Society since 2005, working in several research groups. Currently, he is an Associate Professor at the Federal University of São Carlos (UFScar), with main research interests in Data Mining and Analytics, Machine Learning, and Evolutionary Computation.

João F. Mari is currently an Adjunct Professor at the Federal University of Viçosa (UFV), Brazil. He received the B.Sc. degree in Computer Science from UNIVEM, Brazil, in 2004, and the M.Sc. and Ph.D. degrees in Computer Science from Federal University of São Carlos (UFSCar) in 2007 and 2015, respectively. His research interests include Image and Signal Processing, Computer Vision, and Machine Learning.