Optimizing data augmentation policies for convolutional neural networks based on classification of sickle cells

Matheus Vieira da Silva, Larissa Ferreira Rodrigues, João Fernando Mari
Instituto de Ciências Exatas e Tecnológicas
Universidade Federal de Viçosa - UFV
Caixa Postal 22 - 38.810-000 - Rio Paranaíba - MG - Brasil
Email: {matheus.v.silva, larissa.f.rodrigues, joaof.mari}@ufv.br

***Abstract*—Data augmentation is a key procedure in many image classification tasks, mainly to overcome the problem of small datasets. In this work, we exploit the data augmentation as a hyperparameter optimization approach. We tested our methods to classify erythrocytes to assist the diagnosis of sickle cell anemia. In this study, we proposed a data augmentation approach based on hyperparameter optimization to find the best augmentation policies through the Bayesian optimization algorithm. We also developed a convolutional neural network architecture from scratch and compared it with two classic architectures to classify sickle cell images. Our approach defines the best data augmentation solutions and sends those solutions to be carried out by CNN in the final training. All experiments were validated using a stratified five-fold cross-validation procedure, and our best result achieves 92.54% of accuracy. The results suggest the best augmentation policies defined with optimization allow us to obtain superior results than other strategies such as without data augmentation, several randomly defined image transformations, and only random rotations. As far as we know, our paper is the first to propose optimizing data augmentation policies in biomedical images leading to a better exploration of these strategies in several fields.**

***Keywords*—sickle cell; medical imaging; deep learning; data augmentation; Bayesian optimization.**

I. Introduction

Sickle cell anemia is an inherited disease caused by a single genetic mutation in the hemoglobin resulting in the abnormal hemoglobin S (HbS). Thus, the erythrocyte (red blood cell) assumes an irregular shape becoming a sickle cell. This format reduces the oxygen and blocking the blood vessels, which may cause stroke and other chronic complications. It is estimated that approximately 300,000 children are born with sickle cell anemia each year, making the disease a global health problem [1].

Visual analysis of blood smear is a procedure that can be used to identify sickle cell disease. However, this task is subjective, very time-consuming, and challenging in emerging countries, especially where the incidence of this disease is high [1] [2] [3].

Computer-aided diagnosis has been used for decades to identify patterns in medical images. The improvements in the hardware of computers are allowing us to train even more complex models to identify, classify and quantify anomalies in biomedical images [4] [5].

Recent advances in deep learning techniques, in particular Convolutional Neural Networks (CNNs), demonstrate that this approach learns complex structures from the data itself, without requiring handcrafted feature extraction [6]. Solutions based on CNNs have a low cost and can help healthcare workers diagnose several diseases, such as sickle cell disease. Also, the data and results may be shared and processed around the world [7].

Several studies have been proposed for classifying sickle cell using manually feature extraction [8] [9] [10] [11]. However, solutions based on CNNs are advantageous and allowing automatized the steps of feature extraction and require minimal preprocessing [12].

Although CNN appears to be a promising approach, data augmentation strategies may be necessary to deal with small datasets and overfitting issues. In this way, the choice of best data augmentation strategies is crucial to the classification performance. Usually, this choice is carried manually by testing different strategies based on random transformations until the model gives a "satisfactory" performance. Since the optimal data augmentation strategies are unknown, any definition of a level of satisfaction using this methodology is subjective and time-consuming [13].

In this work, we aim to automate the process of finding an effective data augmentation policy for cell classification tasks. We define each policy as a possible choice of augmentation (e.g., rotations, flips, color adjustments) and the magnitudes for each transformation. Furthermore, the contribution of this paper is an assessment of the data augmentation policies as an optimization problem, where the policies are considered decision variables and the accuracy of the trained model is the objective function to be maximized. We applied a Bayesian search algorithm [14] to identify data augmentation strategies according to the data. These operations are evaluated using three different CNNs architectures: AlexNet [15], LeNet-5 [16], and our custom architecture named Model A. As far as we know, our work is the first to introduce the optimization of data augmentation policies for biomedical image classification. Our results suggest that optimization improving the performance of all CNNs evaluated.

The remaining of this paper is organized as follows: Section II presents the related work. Section III describes the material and methods. Section IV presents and discusses the results.

Section V presents conclusions and future work.

II. Related Work

The state-of-the-art presents multiple systems dedicated to erythrocyte classification, involving machine learning with handcrafted feature extraction to differentiate among the different types of cells [17] [10] [11].

Recently, approaches using deep learning have been proposed. Xu et al. [18] proposed a 10-layer CNN to classify erythrocytes using 7,000 images categorized into five and eight classes. They considered k-fold cross-validation to validate the experiments and obtained for five and eight categories an average accuracy of 89.28% and 87.50%.

Qiu et al. [19] presented a structure to extract regions from the images containing the cells using a Region-based Convolutional Network (RCNN). They performed a multi-label classification using a pre-trained ResNet-50 architecture and a binary classification using Gradient Boosting Classifier. Finally, the method proposed by [19] obtained an overall accuracy of 72.2%.

Alzubaidi et al. [20] utilized a CNN architecture composed of 18 layers, ReLu as the activation function, and batch normalization. The Error-Correcting Output Codes (ECOC) were used to solve the classification problem using the SVM classifier, achieving 92.06% in terms of accuracy.

Most recently, Alzubaidi et al. [21] applied the same domain transfer learning in conjunction with SVM and data augmentation techniques to minimize the overfitting. They evaluated three datasets (main dataset, training with transfer learning, and testing) and developed three architectures with 40, 35, and 29 layers. The result obtained was of 99.98%, which is the best state-of-the-art score reported in the literature. However, for applications in a real-world scenario, it is impracticable to train a CNN with a dataset from the same domain.

The main difference between previous work to our work is the analysis of the data augmentation impact on the performance of a number of CNN architectures. We selected data augmentation policies automatically through a Bayesian search algorithm. We believe that our approach can contribute to identifying sickle cell disease, overcoming the costs of more data acquisition, and avoiding overfitting. Also, our method allows for finding the best augmentation operations to increase accuracy and is suitable to deal with the issue of the lack of training data for CNN architectures.

III. Material and Methods

As presented before, the main goal of this work is to automatically find the best data augmentation policies for training CNNs using the Bayesian optimization strategy. The proposed method was programmed using Python 3.6, the Keras 2.2.4[1] framework with TensorFlow 1.12.0, CUDA version 9.0 and cuDNN 7.1. The data augmentation optimization was drawn from the *deepaugment*[2] library. Also, we used *Numpy*, *OpenCV*, *Scikit-learn*, and *imgaug*[3] libraries. The workflow of the proposed method is summarized in Fig 1.

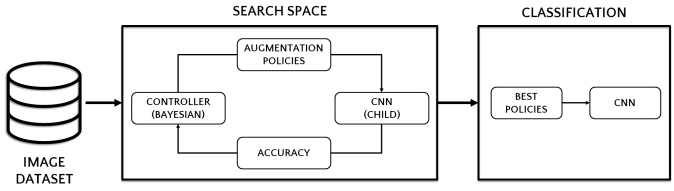

Fig. 1. Steps of the proposed method.

A. Image Dataset

The images used was taken from the *erythrocytesIDB*[4] dataset. It contains 626 images of erythrocytes, each with a single, centered cell in evidence, categorized as healthy (202 images), sickle cell (211 images), and with other deformation (213 images) [22].

When considering the erythrocytes classification based on CNNs, previous studies used images without any preprocessing [20] [21]. However, we performed some preliminary experiments considering the original dataset, and the results demonstrate that training from scratch with original images hurts the results, as the models reached an accuracy of less than 60%. Thus, all images were preprocessed using a simple segmentation procedure proposed by Rodrigues et al. [10]. In this approach, we segment each image using the global Otsu's threshold and morphological operations. Fig. 2 illustrates one image from each class resulting from preprocessing.

Fig. 2. Examples of image instances resulting from preprocessing for each class of *erythrocytesIDB* dataset: (a) healthy; (b) sickle cell; and (c) other deformation.

B. Augmentation policies

The proposal presented in this paper is inspired by [23], which is a framework to search for the best data augmentation policies automatically. These policies are composed of processing functions that will provide a training solution for a child CNN architecture. The term child CNN is the reference to CNN approved in the optimization tests, which uses the accuracy generated in the training step as feedback for the search algorithm.

The augmentation policies were adapted from the *deepaugment* library and are composed of two image processing operations and their respective magnitudes, A and B. The

[1] https://keras.io/
[2] https://pypi.org/project/deepaugment/
[3] https://imgaug.readthedocs.io/en/latest/
[4] Available in: http://erythrocytesidb.uib.es/

magnitude is an optimizable parameter that passed for a discretization process [24], with real values between 0 and 1. In this study, we consider optimizing twenty image processing operations available in the *imgaug* library.

C. CNN Model A

We designed a lightweight convolutional neural network with custom architecture, called Model A. As demonstrated by [25], lightweight models trained from scratch can achieve better results in medical images without the transfer learning technique, especially when the dataset is too small to train a deep network. In this way, when the transfer learning technique is not applied, data augmentation strategies are essential in the training step. Also, our architecture has a lower computational cost when compared to other architectures such as [26] and [27].

Our Model A is composed of six convolutional layers, three pooling layers, and one fully connected layer. Besides, this architecture adopts dropout connections to reduce overfitting and REctified Linear Unit (ReLU) activation to accelerate the training. Fig. 3 illustrates the Model A.

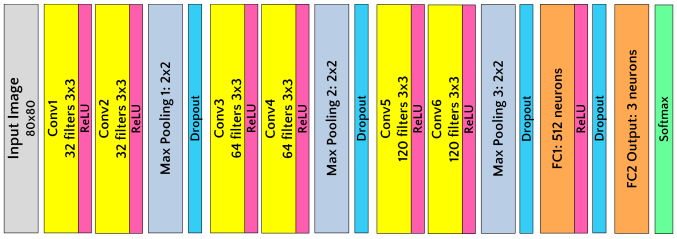

Fig. 3. Model A architecture.

The details of each layer are described below [12].

a) Convolutional layer: The convolutional layers perform the convolution operation in each previous layer to extract relevant features from the images, e.g., color and border. Eq. 1 summarizes the convolution operation.

$$Z_j^l = \sum_{i=1}^{I} W_i^l * A_i^{l-1} + B_j^l \tag{1}$$

Where Z_j^l is the output volume that contains the feature maps, W_i^l is a tensor containing the filters A_i^{l-1}. The Z is the previous layer. Lastly, is added the bias B_j^l and each layer Z has a ReLU activation function.

b) Pooling layer: The pooling layer reduces the size of the feature map. In this study, we apply the maximum-pooling technique. This operation calculates the maximum value of a region of the feature map to improve the generalization and the convergence speed of the model [28]. In our network Model A, we adopted max-pooling of size 2×2, reducing the number of pixels in half.

c) Fully connected layer: The last layer consists of a classic neural network that computes the scalar product between the input vector 1D and the weight vector and adds a bias. The input vector is the result of the 2D feature map converting [16]. Finally, the softmax activation function is applied in the last layer of the network to transform the units into probabilities [15].

d) Dropout: Dropout connections are applied to reduce overfitting In order to reduce overfitting, the dropout method is generally used in literature during training. This connection allows excluding some units and their respective connections avoiding an excessive adaptation of neurons [29].

The training step defines a loss function to calculate the model error and an optimizer for the optimization process that will update the weights using the back-propagation algorithm [30]. In this work, we chose to apply Adam [31] optimizer in conjunction with the Categorical Cross-Entropy (CCE) loss function, defined in Eq. 2.

$$CCE = -\frac{1}{N}\sum_{i=0}^{N}\sum_{j=0}^{J} y_j log(\hat{y}_j) + (1-y_j) log(1-\hat{y}_j) \tag{2}$$

Where $\hat{y}$ is the prediction of the model and the respective label is represented by y. For all three CNN architectures evaluated in this paper, the learning rate was set to 0.001 [32].

D. Classical CNN architectures

We selected two classical architectures for experimental comparison: LeNet-5 and AlexNet, both of which were trained from scratch and initializing the weights randomly.

a) LeNet-5: Was the first case of success of CNNs and was proposed by LeCun et al. [16] for character recognition. It is composed of seven layers: three convolutional layers, two pooling layers with average pooling, and two fully connected layers.

b) AlexNet: Proposed by Krizhevsky et al. [15] and won the ILSVRC 2012 [33]. It is composed of ten layers: five convolutional layers, three pooling layers, and two fully connected layers.

E. Controller

In this study, the controller uses the Bayesian algorithm instead of Reinforcement Learning applied by [23]. The Bayesian approach finds the best possible parameter setup faster than other strategies and presents a lower computational cost [34]. Initially is carried a priori sampling of the augmentation policies. These policies should maximize an objective function O to found the best parameters at each interaction based on a surrogate model built from the objective function O. Updating the magnitudes is performed according to the accuracy feedback obtained from the child CNN.

F. Configuration and Training

All images were randomly partitioned into a training set and testing set with the proportions 80% and 20%, respectively. After, we considered 20% of the training set as a validation set to search for the best augmentation policies. In the two stages, we used the same CNN: search space (with child CNN); and classification.

a) Search space: To find an optimized solution, we defined 300 interactions. In each iteration, the child CNNs are trained with three different samples with a N number of epochs and batch size of 64. The controller uses as feedback the accuracy provided by CNN to search the best possible augmentation policies from the dataset. Accuracy is a metric derived from the confusion matrix used to measure the performance of a model [35], as defined in Eq. 3.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

where TP is the True Positives, TN True Negatives, FP False Positives and FN False Negatives.

b) Classification: This step performs the classification of the cells applying the data augmentation policies according to the optimization defined in the previous step. The training set is used according to the first partitioning, i.e., 80% of the dataset. We adopted a k-fold cross-validation strategy [36], and the dataset was randomly partitioned into five stratified folds which are iteratively selected as training and test sets. Finally, we measured the averaged accuracy of five sets to produce a single estimation.

IV. Results and Discussion

The data augmentation optimization was performed on a machine with a GPU NVIDIA Titan XP with CUDA version 9.0, an Intel processor i5 3.00GHz, and 32 GB RAM. In order to accelerate the experiments, the classification steps were carried out in parallel in a machine with a GPU NVIDIA GeForce GTX 1080 TI under CUDA version 9.0, an Intel processor i5 3.00GHz, and 32 GB RAM. The random processes are locked using a fixed seed value to ensure the reproducibility of the experiments.

All images were resized to 80 $\times$ 80 pixels for Model A; 32 $\times$ 32 pixels for LeNet-5; and 224 $\times$ 224 pixels for AlexNet. Then, we performed the data augmentation policies using Bayesian optimization for each CNN evaluated.

We represent the choice of data augmentation operations as a hyperparameter optimization problem, where we extracted the optimized solution from the dataset for each CNN architecture. Table I presents the augment policies and their respective magnitudes for each CNN found through the Bayesian search optimization approach. It is important to note that for twenty operations available, only ten operations were selected since each policy is composed by two image processing operations and its respective magnitudes, there are five data augmentation policies in this study. During training, each image has 50% chance to be augmented by a policy and one of the five policies is selected randomly to augment the image.

The high computational cost is one of the biggest challenges when working with the optimization approach explored in this study because as the number of CNN parameters increases, the time required for training also increases significantly. However, our approach allows for better results in fewer evaluations than a grid search or random search [37]. The training of LeNet-5 was carried out in two hours and seven minutes, defining in 50 epochs of child CNN. The Model A, it took two hours with six epochs of child CNN. AlexNet, the most complex CNN, needed seven hours and fifty-four minutes for total training, with the number of epochs set at six. We defined the number of child CNN epochs empirically, and 100 epochs for the classification step.

Our results demonstrate that optimized data augmentation operations reduce the overfitting and do not require other regularization techniques. In order to assess the values of loss and accuracy during training and validation, the average values are shown graphically in Fig 4. The charts show each CNN's behavior in the training step, in which training losses and validation losses decrease with each iteration. This behavior suggests that the training did not overfit the data, thus evidencing the importance of cross-validation in our experiments allowing results less subject to randomness.

After defining the best data augmentation policies using Bayesian optimization, we test the trained models generated by each fold. Table II shows the test result obtained for each fold and the average result. These results demonstrate that our Model custom design achieved an average accuracy of 92.54%, and in some folds, obtained an accuracy above 96%, which is very close to the current state of the art [21] that did not consider k-fold cross-validation. Finally, the networks AlexNet and LeNet-5 obtained an average accuracy of 90.00% and 87.93%, respectively.

In addition, we compared our proposed approach with other three strategies: i): without data augmentation; ii) different operations (such as flips, rotations, zoom, whitening transformation, and others.); and iii) random rotations with angles steps of 10°.

To evaluate the impact of each strategy on the accuracy of the CNNs, they were all trained from scratch. To evaluate the impact of these strategies on the accuracy of the CNNs, they were all trained from scratch. As shown in Table III, optimized data augmentation improved the accuracy of all CNN models. Our results demonstrate that only applying several data augmentation strategies without considering Bayesian optimization reduces classification performance. In particular, the accuracy of Model A (with different operations) decreased by 49.21 percentage points compared with optimized data augmentation. It is worth noticing that optimized data augmentation is the best strategy, but in scenarios where only a short time is available for training, only random rotations should be considered as a data augmentation strategy.

V. Conclusion

This paper presents and evaluates an approach to search for the best data augmentation policies using Bayesian optimization. As far we know, our method is the first to introduce optimizing data augmentation policies for biomedical image classification, and the experimental results were very close to the state-of-the-art, reaching an accuracy of 92.54%.

We compared the data augmentation optimized using the Bayesian method with three other training strategies (without data augmentation, different data augmentation operations, and random rotations). Our results point out that Bayesian optimization overcomes the traditional empirically defined data augmentation methods. We also proposed a lightweight CNN architecture we called Model A. Our experimental results

TABLE I. THE BEST DATA AUGMENTATION POLICIES FOUND BY THE BAYESIAN OPTIMIZATION.

	Model A		AlexNet		LeNet-5	
	Augmentation Strategy	**Magnitude**	**Augmentation Strategy**	**Magnitude**	**Augmentation Strategy**	**Magnitude**
Policy A	brighten	0.18	invert	0.352	clouds	0.522
	gamma-contrast	0.665	brighten	0.566	vertical-flip	0.949
Policy B	clouds	0.18	dropout	0.15	gaussian-blur	0.785
	brighten	0.708	translate-y	0.054	rotate	0.928
Policy C	shear	0.027	vertical-flip	0.555	translate-y	0.146
	rotate	0.503	dropout	0.092	dropout	0.379
Policy D	clouds	0.266	emboss	0.857	clouds	0.814
	invert	0.891	rotate	0.626	add-to-hue-and-saturation	0.035
Policy E	brighten	0.726	additive-gaussian-noise	0.047	horizontal-flip	0.389
	gamma-contrast	0.611	dropout	0.571	vertical-flip	0.947

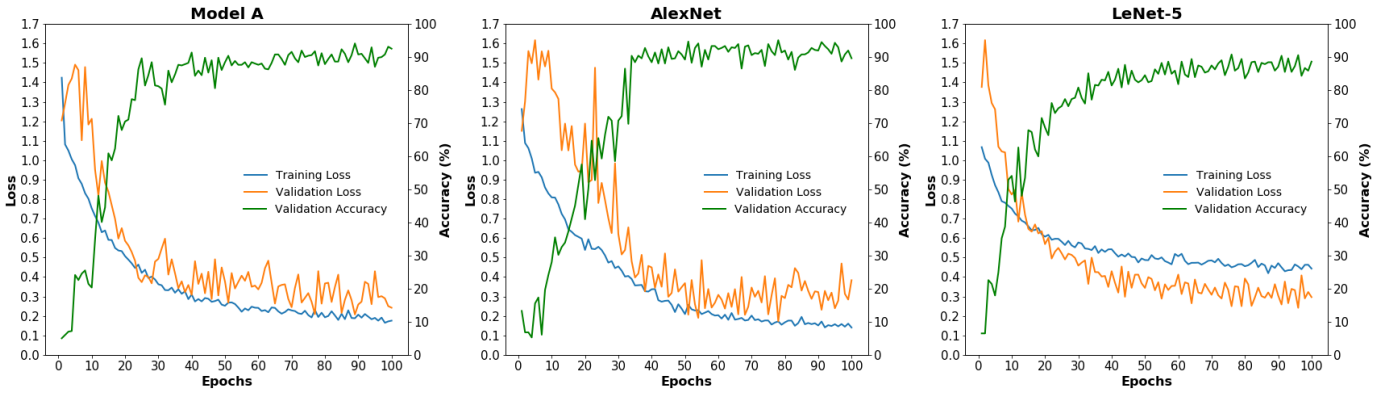

Fig. 4. Charts showing the average evolution of accuracy and loss values for the training and validation set.

TABLE II. 5-FOLD TEST AND AVERAGE ACCURACY FOR EACH CNN MODEL, USING THE BEST DATA AUGMENTATION POLICIES.

Fold	**Model A (%)**	**AlexNet (%)**	**LeNet-5 (%)**
1	94.44	91.27	86.51
2	96.03	90.48	88.89
3	95.24	88.89	89.68
4	91.27	89.68	86.51
5	85.71	89.68	88.10
Average	92.54	90.00	87.93

TABLE III. AVERAGE ACCURACY FOR EACH CNN EVALUATED CONSIDERING DIFFERENT DATA AUGMENTATION STRATEGIES.

	Data Augmentation Strategy			
CNN	**Optimized**	**Without**	**Different Operations**	**Random Rotations**
Model A	**92.54%**	89.68%	43.33%	90.48%
AlexNet	90.00%	87.14%	46.98%	88.10%
LeNet-5	87.93%	87.62%	63.33%	89.68%

demonstrate that combining optimized data augmentation policies and the custom-designed CNN architecture has significantly improved the performance of the sickle cell disease classification. Moreover, the Bayesian search speeds the training process when compared to grid and random search, while it reduces the number of trials.

As future work, we plan to test our proposed method for other biomedical images in order to verify that our findings hold for similar datasets. Moreover, we intend to perform benchmarking with the training technique based on transfer learning and evaluating further optimization algorithms.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the TITAN Xp GPU used for this research. This study was financed by Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (M. V. da Silva received a scholarship from PIBIC/CNPq). We would like to thanks CAPES and FAPEMIG (Grant number CEX - APQ-02964-17) for the financial support. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

REFERENCES

[1] Gregory J Kato, Frédéric B Piel, Clarice D Reid, Marilyn H Gaston, Kwaku Ohene-Frempong, Lakshmanan Krishnamurti, Wally R Smith, Julie A Panepinto, David J Weatherall, Fernando F Costa, et al. Sickle cell disease. *Nature Reviews Disease Primers*, 4(1):1–22, 2018.

[2] Frédéric B Piel, Martin H Steinberg, and David C Rees. Sickle cell disease. *New England Journal of Medicine*, 376(16):1561–1573, 2017.

[3] Gentil Claudino de Galiza Neto and Maria da Silva Pitombeira. Aspectos moleculares da anemia falciforme. *Jornal Brasileiro de Patologia e Medicina Laboratorial*, 39(1):51–56, 2003.

[4] James S Duncan and Nicholas Ayache. Medical image analysis: Progress over two decades and the challenges ahead. *IEEE transactions on pattern analysis and machine intelligence*, 22(1):85–106, 2000.

[5] Marleen De Bruijne. Machine learning approaches in medical image analysis: From detection to diagnosis, 2016.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[7] Livia Faes, Xiaoxuan Liu, Aditya Kale, Alice Bruynseels, Mohith Shamdas, Gabriella Moraes, Dun Jack Fu, Siegfried K Wagner, Christoph Kern, Joseph RE Ledsam, et al. Deep learning under scrutiny: performance against health care professionals in detecting diseases from medical imaging-systematic review and meta-analysis. 2019.

[8] Athira Sreekumar and Ashok Bhattacharya. Identification of sickle cells from microscopic blood smear image using image processing. *International Journal of Emerging Trends in Science and Technology*, 1(5):783–787, 2014.

[9] Shashi Bala and Amit Doegar. Automatic detection of sickle cell in red blood cell using watershed segmentation. *Int. J. Adv. Res. Comput. and Commun. Eng*, 4(6):488–491, 2015.

[10] Larissa Ferreira Rodrigues, Murilo Coelho Naldi, and João Fernando Mari. Morphological analysis and classification of erythrocytes in microscopy images. In *XII Workshop de Visão Computacional*, Campo Grande, MS, Brazil, 2016. WVC.

[11] Lucas Costa de Faria, Larissa Ferreira Rodrigues, and João Fernando Mari. Cell classification using handcrafted features and bag of visual words. In *2018 Workshop de Visão Computacional (WVC)*, pages 68–73, Nov 2018.

[12] M. A. Ponti, L. S. F. Ribeiro, T. S. Nazare, T. Bui, and 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)*, pages 17–41, Oct 2017.

[13] Fábio Perez, Cristina Vasconcelos, Sandra Avila, and Eduardo Valle. Data augmentation for skin lesion analysis. In *OR 2.0 Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis*, pages 303–311. Springer, 2018.

[14] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[16] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[17] Manuel Gonzalez-Hidalgo, FA Guerrero-Pena, S Herold-Garcia, Antoni Jaume-i Capó, and Pedro D Marrero-Fernández. Red blood cell cluster separation from digital images for use in sickle cell disease. *IEEE journal of biomedical and health informatics*, 19(4):1514–1525, 2014.

[18] Mengjia Xu, Dimitrios P Papageorgiou, Sabia Z Abidi, Ming Dao, Hong Zhao, and George Em Karniadakis. A deep convolutional neural network for classification of red blood cells in sickle cell anemia. *PLoS computational biology*, 13(10), 2017.

[19] Wei Qiu, Jiaming Guo, Xiang Li, Mengjia Xu, Mo Zhang, Ning Guo, and Quanzheng Li. Multi-label detection and classification of red blood cells in microscopic images. *arXiv preprint arXiv:1910.02672*, 2019.

[20] Laith Alzubaidi, Omran Al-Shamma, Mohammed A. Fadhel, Laith Farhan, and Jinglan Zhang. Classification of red blood cells in sickle cell anemia using deep convolutional neural network. In Ajith Abraham, Aswani Kumar Cherukuri, Patricia Melin, and Niketa Gandhi, editors, *Intelligent Systems Design and Applications*, pages 550–559, Cham, 2020. Springer International Publishing.

[21] Laith Alzubaidi, Mohammed A. Fadhel, Omran Al-Shamma, Jinglan Zhang, and Ye Duan. Deep learning models for classification of red blood cells in microscopy images to aid in sickle cell anemia diagnosis. *Electronics*, 9(3):427, Mar 2020.

[22] Manuel Gonzalez-Hidalgo, FA Guerrero-Pena, S Herold-Garcia, Antoni Jaume-i Capó, and Pedro D Marrero-Fernández. Red blood cell cluster separation from digital images for use in sickle cell disease. *IEEE journal of biomedical and health informatics*, 19(4):1514–1525, 2015.

[23] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123, 2019.

[24] Huan Liu, Farhad Hussain, Chew Lim Tan, and Manoranjan Dash. Discretization: An enabling technique. *Data mining and knowledge discovery*, 6(4):393–423, 2002.

[25] Maithra Raghu, Chiyuan Zhang, Jon Kleinberg, and Samy Bengio. Transfusion: Understanding transfer learning for medical imaging. In *Advances in Neural Information Processing Systems*, pages 3342–3352, 2019.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[27] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[28] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks*, pages 92–101. Springer, 2010.

[29] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[30] Yann LeCun, D Touresky, G Hinton, and T Sejnowski. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, volume 1, pages 21–28. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988.

[31] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[32] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012.

[33] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[34] Ian Dewancker, Michael McCourt, and Scott Clark. Bayesian optimization for machine learning: A practical guidebook. *arXiv preprint arXiv:1612.04858*, 2016.

[35] Stephen V Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote sensing of Environment*, 62(1):77–89, 1997.

[36] Michael W Browne. Cross-validation methods. *Journal of mathematical psychology*, 44(1):108–132, 2000.

[37] James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2546–2554. Curran Associates, Inc., 2011.