LOW RESSOURCE DEEP LEARNING BASED APPROACH FOR DIAGNOSIS OF LYMPHOCYTOSIS IN MULTI-INSTANCE LEARNING SETUP.

A PREPRINT

Omar Souaidi*
Department of Computer Science
ENS Paris-Saclay - CentraleSupelec
Gif-Sur-Yvette 91190
omar.souaidi@student.ecp.fr

Antoine Benady
Department of Computer Science
ENS Paris-Saclay
Gif-Sur-Yvette 91190
antoine.benady@ens-rennes.fr

August 6, 2021

ABSTRACT

In this project, we investigate the use of the recent advances in deep learning, specially in computer vision and propose an approach that combines information from two types of data (images and clinical attributes) with low computational ressources for the diagnosis of lymphocytosis in a multi-instance learning setup. The first step is to extract features from the images in a self-supervised learning way, so that we can embed each image into a low dimensional space and store this embedding in order to use it in the second step where we will combine information from these images embedding as well as clinical attributes for diagnosis of lymphocytosis. We propose also a different approach to ensemble model so that it can generalize better and deal with hard labels. Our code can be found at **https://github.com/omarsou/MIL-lymphocytosis**.

Keywords Multi-Instance Learning · Computer Vision · Lymphocytosis · Self-Supervised · Low Ressources

1 Introduction

Lymphocytosis is a common finding, which can be either a reaction to infection, acute stress, and so on (termed reactive), or the manifestation of a lymphoproliferative disorder –a type of cancer of the lymphocytes (termed tumoral). In existing clinical practice, diagnosis as either reactive or tumoral relies on visual microscopic examination of the blood cells together with the integration of clinical attributes such as age and lymphocyte count. Taking into consideration the visual assessment based on clinical attributes together with texture and size of the lymphocytes in the blood smear, a diagnosis of the subtype of lymphoid malignancy is performed. On the positive side, such practice is fast and affordable. It suffers however from poor reproducibility. Additional clinical tests are required, with flow cytometry being the gold standard to definitively affirm the malignant nature of the lymphocytes. However, this analysis is relatively expensive and time-consuming, and therefore cannot be performed for every patient in practice. Therefore, the development of automatic and accurate processes could lead to a better way to determine which patient should be referred for flow cytometry analysis, augmenting and assisting the assessment of the clinicians.

The images used for this problem are acquired from blood smears, which are made by placing a drop of of blood between two slides in order to create a thin, uniform layer of blood so that individual blood cells are non-overlapping and can be observed under a microscope. These images are then captured using a DM-96 device while focussing on individual lymphocytes. Blood smears and patient attributes were collected from 204 patients from the routine hematology laboratory of the Lyon SudUniversity Hostpital. All the samples were anonymized as required by the General Data Protection Regulation. We have access to **142 subjects with 44 reactive and 98 malignant cases for training and 42 subjects for testing**. The link to the kaggle competition is here.

*https://github.com/omarsou

In this project, we present a novel approach for the challenging task of diagnosis of lymphocytosis. Our proposed method is able to predict the nature of symptoms combined with clinical attributes such as age and lymphocytes concentration.
First we propose a Convolutional AutoEncoder Network for extracting features from multiple microscopy images and embed each image in a low dimensional space. Second we introduce a simple Multi Mixture Perceptron Layer (MMLP) that combines both images embeddings and the patient's clinical attributes to render a reliable diagnosis. Finally, we show how we propose a method to ensemble models and combine this with simple Test-Time Augmentation so that it can generalize better and deal with hard labels. [1]

2 Deep Multi Instance Learning

We will first present our approach for the task of predicting the presence of a lymphoproliferative disorder.

Let's first introduce some notation to describe our approach. We represent the data of a patient i as :

$$D_i = ([X_i^1, ...X_i^{N_i}], a_i, c_i, y_i) \tag{1}$$

Where $[X_i^1, ...X_i^{N_i}]$ represents the N_i images obtained from the i-th patient, and a_i, c_i, y_i represent their age in years, lymphocyte count in number of cells per litre of blood and their target class, respectively. As it is a binary classification, $y_i \in (0, 1)$.

2.1 Features Extractor

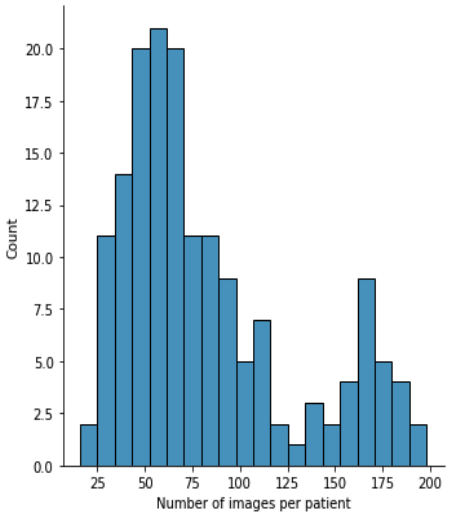

Figure 1: Distribution of the number of images per patient.

Our first approach is bag-images based approach where each batch represent a bag of images where we classify each patient from his bag of images directly. If we denote by f the model, we can write :
$f(D_i) = y_i$
We tried different way in order to fit the model and images in memory (Note that we used Colab to run all our experiments and the GPU's memory was 15Go).
One way was to use a batch size equal to one (one patient), nevertheless as you can see in the figure 1, there are some patients that had more than 100 images and it was impossible to transfer and do the forward propagation with a GPU even with a small ResNet34 model [2] . A Walk around was to measure the maximum number of image M_{image} that we can feed to the model, and then we split each bag D_i into $|D_i|/M_{image}$ if $|D_i| > M_{image}$ bags else $|D_i|$, and make $max(|D_i|/M_{image}, 1)$ forward pass and do the backward propagation until we process all the images of the patient. Unfortunately, it did not result in good performance.

The other way was to embed first each images into a low dimensional space and use them later for the classification task. The problem is that if we use a pretrained model (typically one that was trained on ImageNet [3]), we will not get a good embedding of the image since the domain gap between images from ImageNet and the one that we have (blood

cells) is large. In order to ensure the quality of the embeddings, we based our approach on the self-supervised learning. The term self-supervised learning (SSL) has been used (sometimes differently) in different contexts and fields, such as representation learning [4], neural networks, robotics [5], natural language processing, and reinforcement learning. In all cases, the basic idea is to automatically generate some kind of supervisory signal to solve some task (typically, to learn representations of the data or to automatically label a dataset). In self-supervised learning the task that we use for pretraining is known as the “pretext task”.

2.1.1 Image Reconstruction

Our "pretext task" was simply to reconstruct the image from the original one using a simple AutoEncoder Model. The overall architecture of the model is shown in the figure 2.

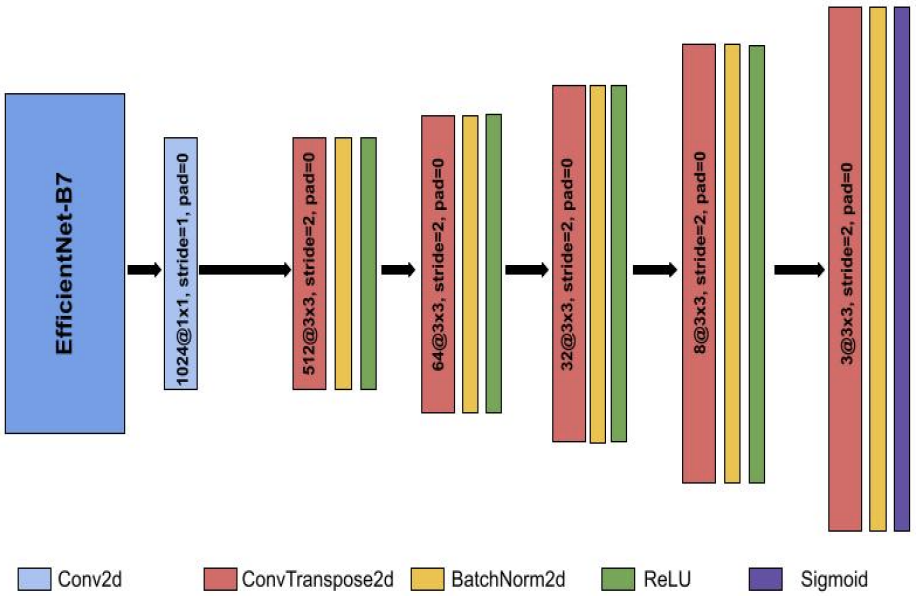

Figure 2: Our autoencoder architecture.

The encoder part is composed mainly of the EfficientNet-B7 architecture [6] followed by a 2D Convolution. This latter is also made to reduce the dimension of the embedding given by the EfficientNet (2560). The decoder part is composed of five block, each block is a sequence of Transposed Convolution 2D followed by a 2D Batch Normalization and a ReLU activation (except for the last one where the activation function is the sigmoid function). On figure 3 and 4, you can see the evolution of the quality of the reconstructed image between the second epoch and the last epoch (90).

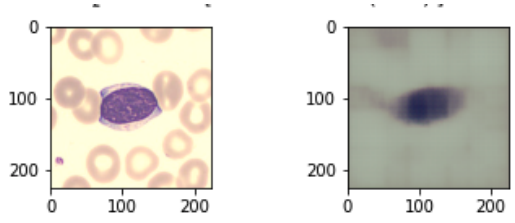

Figure 3: Epoch 2 - On the left, the original image and on the right the reconstructed image.

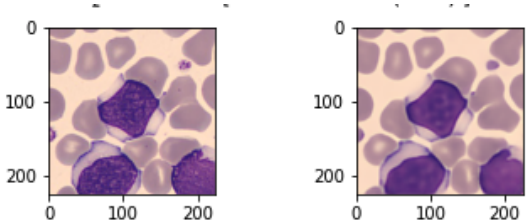

Figure 4: Epoch 90 - On the left, the original image and on the right the reconstructed image.

2.1.2 Embeddings

For each image, we extracted after applying an adaptive average pooling one embedding after the EfficientNet-B7 (dimension = 2560) and another one after the last 2d convolution layer of the encoder (dimension = 1024). We will

denote the first extractor by f_{large} and the second one by f_{small}
Furthermore, we applied for each image two transformations : Horizontal and vertical flip. Hence for each image of the patient, we have three embedding, the one obtain with the original image, one with the original image vertically flipped and the one with the original image horizontally flipped. Hence for each patient i we have :

$$D_i^{large} = (f_{large}([X_i^1, ..., X_i^{N_i}]), a_i, c_i, y_i) \tag{2}$$

$$D_i^{small} = (f_{small}([X_i^1, ..., X_i^{N_i}]), a_i, c_i, y_i) \tag{3}$$

$$D_{i_{vertical}}^{large} = (f_{large}([X_{i_{vertical}}^1, ..., X_{i_{vertical}}^{N_i}]), a_i, c_i, y_i) \tag{4}$$

$$D_{i_{vertical}}^{small} = (f_{small}([X_{i_{vertical}}^1, ..., X_{i_{vertical}}^{N_i}]), a_i, c_i, y_i) \tag{5}$$

$$D_{i_{horizontal}}^{large} = (f_{large}([X_{i_{horizontal}}^1, ..., X_{i_{horizontal}}^{N_i}]), a_i, c_i, y_i) \tag{6}$$

$$D_{i_{horizontal}}^{small} = (f_{small}([X_{i_{horizontal}}^1, ..., X_{i_{horizontal}}^{N_i}]), a_i, c_i, y_i) \tag{7}$$

For simplicity, we are going to use the following notation for any type of image/embedding :

$$D_i = (f([X_i^1, ..., X_i^{N_i}]), a_i, c_i, y_i) \tag{8}$$

2.2 Lymphocytosis Diagnosis Model

We denote $\{H_i^j\} = f([X_i^1, ..., X_i^{N_i}])$.
The proposed deep learning architecture for the final classification task (figure 5) consists of a special embedding layer that are going to embed the image features into a low dimensional space, this transformation is representend by a function $f_{special\_embedding}$, hence we have :

$$\{h_i^j\} = f_{special\_embedding}(\{H_i^j\}) \tag{9}$$

Then, we use a pooling function (the mean pooling) to obtain one pooled embeddings over all instances :

$$p_i = \frac{1}{N_i} \sum_j h_i^j \tag{10}$$

We concatenate the vector p_i with the clinical attribute

$$l_i = [p_i, a_i, c_i] \tag{11}$$

Which is followed by a Multi-Layer Perceptron architecture consisting of two hidden layer and one output layer to predict the probability of disease.

$$l_i^1 = ReLU(W_1 l_i) \tag{12}$$

Where W_1 represent the weight of the first hidden layer.

$$l_i^2 = ReLU(W_2 l_i^1) \tag{13}$$

Where W_2 represent the weight of the second hidden layer

$$\hat{y_i} = \sigma(\theta_{output}^T l_i^2 + \beta) \tag{14}$$

Where $\sigma(x) = \frac{1}{1+exp(-x)}$ is the logistic function, and θ_{output} and β are, respectively, the weight vector and the bias of the output layer.

3 Training

3.1 Features Extractor

For the self supervised learning task, we use all the images of the dataset (16711 images). Our autoencoder network is trained with the mean squared error loss. Training is performed on Colab (Python) with the PyTorch Library [7] and executed on a machine with a NVIDIA TESLA V100, a 4-core 3.5 GHz processor and 15 gigabytes of memory. The model was trained using the Adam Optimizer [8] with a learning rate = 0.002, a batch size of 16 and trained for 90 epochs. Standard data augmentation is introduced during training : Random Horizontal and vertical flips as well as random rotations from the set [0°, 90°, 180°, 270°].

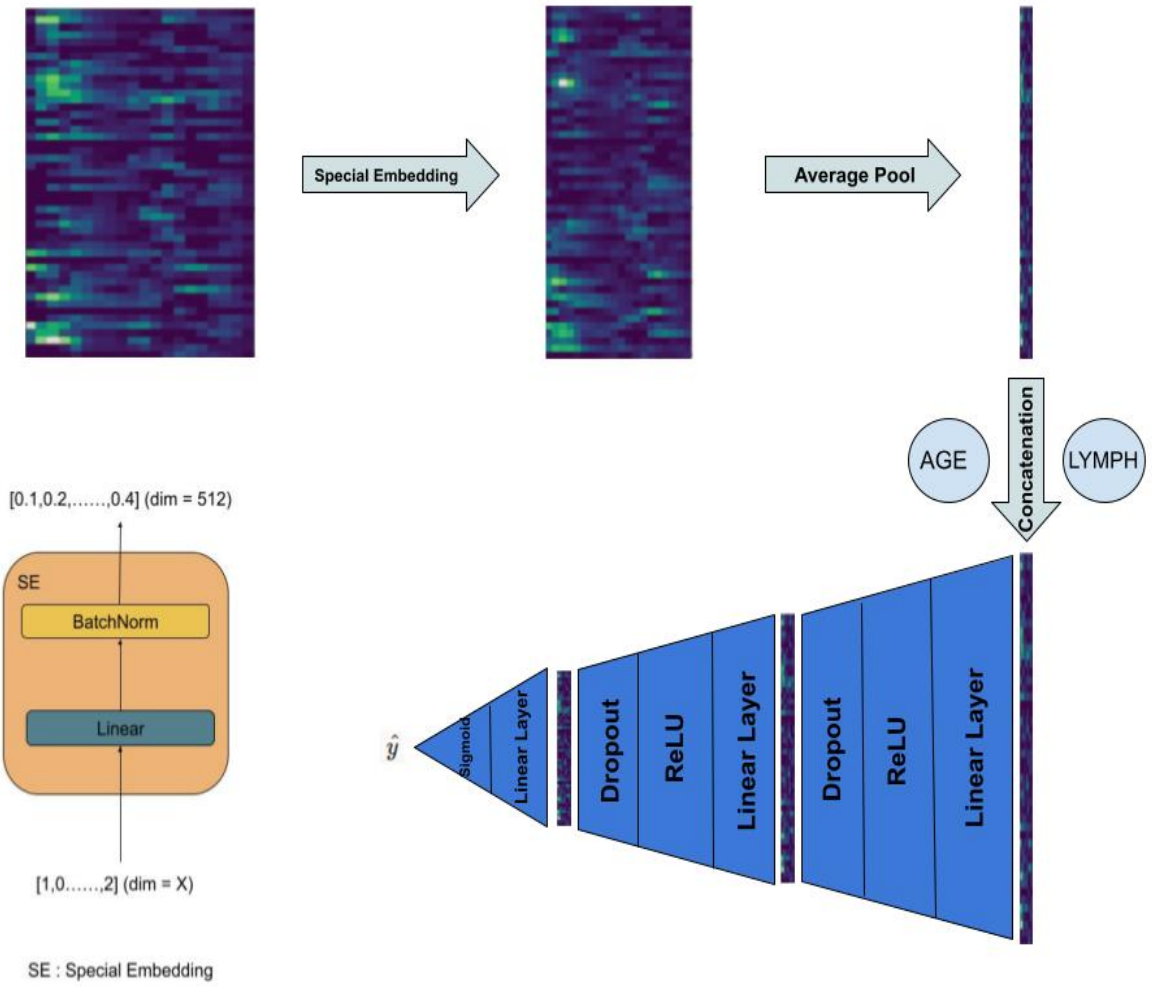

Figure 5: Our classifier architecture.

3.2 Lymphocytosis Diagnosis Model

For each of the embedding (large or small) we use all the images embedding variations (horizontal, verticial and the original). Our network is trained with Binary Cross Entropy Loss. Training is also performed on Colab (Python) with the Pytorch Library and executed on a machine with a NVIDIA TESLA V100, a 4-core 3.5 Ghz processor and 15 gigabytes of memory. The model was trained using the Adam Optimizer with a starting learning rate of 0.1, we reduce the learning rate by a factor 0.8 if the balanced accuracy loss on the validation set did not improve after 5 epochs (We used the module ReduceLROnPlateau from the pytorch library). We used a batch size of 32, and the training was done for 100 epochs. We evaluate after each epoch the model on the validation set, we saved the one that performs best on the validation set according to the metric of the challenge which is the balanced accuracy score.
For our case (binary classification), the balanced accuracy score BAS is :

$$BAS = \frac{1}{2}\left(\frac{|predicted\_positive|}{|actual\_positive|} + \frac{|predicted\_negative|}{|actual\_negative|}\right) \tag{15}$$

3.3 Ensemble Modeling and Test-Time Augmentation

One way to improve our score is to ensemble different models or to use Test-Time Augmentation. We decided to use both of them.
The goal of ensemble model is to combine the predictions of several base estimators built with a given learning

algorithm in order to improve generalizability/robustness over a single estimator.
To build several estimators, we use the same model but we propose five different training/validation split that was obtained with the train_test_split function of Scikit-Learn [9]. Of course, in order to take into account the unbalanced distribution of the classes, we used the stratify parameters, which means that each validation split had the same classes distribution. At the end we had 5 models for each of the embedding (large or small). Coupled with the test-time augmentation (using the original image, horizontal flipped image and the vertical flipped one), we have for each patient of the test set 15 output predictions (probabilities).

$$\hat{y_i} = \{\hat{y}_j^{type}\} \tag{16}$$

Where type ∈ ['original', 'horizontal', 'vertical'] and j ∈ [1, 2, 3, 4, 5]

3.3.1 Average prediction

One strategy is to average the prediction :

$$\hat{y}_i^{AVG} = \frac{1}{15} \sum_{j,type} \hat{y}_j^{type} \tag{17}$$

We will denote this method AVG.

3.3.2 Statistical Analysis

We notice that for some patient, we have one prediction of 0.001 and another one of 0.95, then we made some logical and intuitive rules to decide if we classify one patient as positive or as negative.
Each patient is now represented by some statistics about the fifteen predictions :
$P_i = [mean(\{\hat{y}_j^{type}\}), std(\{\hat{y}_j^{type}\}), max(\{\hat{y}_j^{type}\}), min(\{\hat{y}_j^{type}\})$
Mean is the average of the predictions (17), std is the standard deviation of the predictions, max is the largest prediction (in term of probability) and min is the smallest prediction (in term of probability).
We propose the next algorithm and we will denote this method SA.

Algorithm 1 compare_min_max

Require: max, min
  $prob\_positive\_max \leftarrow max$
  $prob\_negative\_max \leftarrow 1 - min$
  if $prob\_positive\_max > prob\_negative\_max$ **then**
    return 1
  else
    return 0
  end if

Algorithm 2 Final Prediction with P_i

Require: $mean$, std, max, min
  if $mean >= 0.5$ **then**
    if $mean - std > 0.5$ **then**
      return 1
    else
      return compare_min_max(max, min)
    end if
  else if $mean < 0.5$ **then**
    if $mean - std < 0.5$ **then**
      return 0
    else
      return compare_min_max(max, min)
    end if
  end if

4 Results

Name	Public Set	Private Set
Small AVG	0.823	0.843
Large AVG	0.857	0.921
Small SA	0.932	0.843
Large SA	0.932	0.843

Table 1: Small refers to the embedding images with dimension 1024, large to the ones with dimension 2560. AVG : Average Prediction Method / SA : Statistical Analysis Method

The results are shown in the Table 1.

5 Conclusion and Further Work

This project presents a deep MIL approach for diagnosis of lymphocytosis in case of limited computational resources. We first use a pretext task so that we can extract reliable features from the images, and then we combine these features with patient attributes. We ranked 14 (2th on the public leaderboard and 14th on the private one), we can see that we overfitted on the Public Leaderboard.
One way to improve the result can be to take inspiration from [10] and pretrain a sequence Transformer to auto-regressively predict pixels, without incorporating knowledge of the 2D input structure. That way, we can get more informative features.

References

[1] Divya Shanmugam, Davis Blalock, Guha Balakrishnan, and John Guttag. When and why test-time augmentation works, 2020.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[3] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015.

[4] Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning, 2017.

[5] Mirko Nava, Jerome Guzzi, R. Omar Chavez-Garcia, Luca M. Gambardella, and Alessandro Giusti. Learning long-range perception using self-supervision from short-range sensors and odometry, 2019.

[6] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[7] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[9] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[10] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1691–1703. PMLR, 13–18 Jul 2020.