Multiple Instance Learning Techniques for Lymphocytosis Classification

Florencia Boccarato FLORENCIA.BOCCARATO@ENS-PARIS-SACLAY.FR

Mohammad Ali Jauhar MOHAMMAD.JAUHAR@ENS-PARIS-SACLAY.FR

Abstract

Lymphocytosis, defined as an absolute lymphocyte count increase, is a common hematologic abnormality. It can be either *reactive* or *tumoral*. As such, doctors refer to a variety of clinical information and images for diagnosis. In this work, we use these information to formulate a machine learning solution of lymphocytosis classification and compete in the associated *kaggle* challenge. We achieve a balanced accuracy of **0.86** on the test set. We present our methods and corresponding results in the main text and parts of the ablation study, things that did not work, and other related information in the appendix.

Keywords: Lymphocytosis, Multiple Instance Classification, Autoencoders.

1. Introduction

Lymphocytosis is an abnormal increase in the number of lymphocytes in the blood, and it can have a variety of causes. In particular, it is classified as "reactive" when it is a reaction to factors such as infections, stress, and so on; and it is classified as "tumoral" when it is the manifestation of a lymphoproliferative disorder. This diagnosis is usually done by visually examining the blood cells, as well as integrating other clinical attributes. This project aims to assist the clinicians on the diagnosis of the subtype of lymphoid malignancy, in order to reduce the demand for further studies such as flow cytometry.

To achieve it, labeled information for 205 patients is available, divided into 163 patients for training and 42 for testing. Each patient's information includes a variable number of lymphocyte images, other personal and clinical data, and the diagnosis made by an expert on whether the patient has "tumoral" lymphocytosis or not. This problem is, therefore, well-suited on the frame of Multiple Instance Learning (MIL), where there are not available labels for each instance (the images, in this case), but only on the patient level. Even if a patient has been diagnosed with tumoral lymphocytosis, it is not certain that all of the lymphocytes will be affected.

2. Architecture and Methodological Components

The problem at hand is a classic MIL problem, as we only have labels per bag instead of labels per instance. This is a common situation in medical imaging since, in many clinical situations, the images obtained from the pathological slides are very high-resolution and computational and memory constraints make them unusable as such. Often times, large images are divided into smaller patches to create bags for each patient, which leads to MIL problems (Qu et al., 2023; Butke et al., 2021).

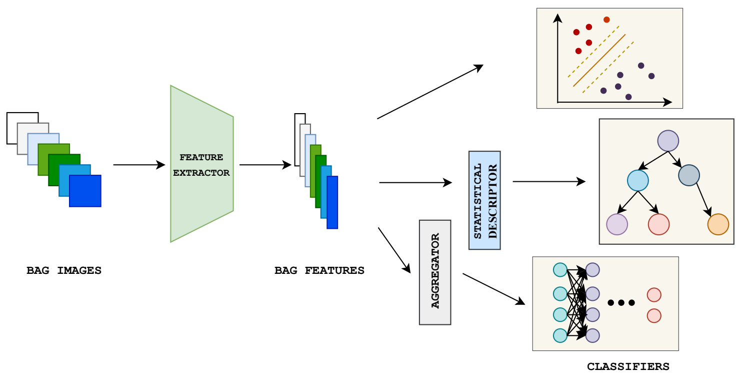

Figure 1: Overview of our methods for MIL. The pipeline consists on performing feature extraction with an Encoder, and then aggregating the resulting features per bag either by a neural aggregator, statistical descriptor or mi-SVM.

Our intuition is to divide the problem into three steps: (a) feature extraction for each image in a bag, (b) aggregation of individual images to create a combined feature map for a given bag, and (c) using the aggregated features to train a classifier. In our experience, the first two steps are particularly challenging.

Developing a good feature extractor is not trivial. A good feature extractor should produce features that are a good representative of the image or they are a good representative of the problem at hand. Literature usually employ techniques such a contrastive learning to push the images into separate regions corresponding to their labels in the feature space. However, we found that a good enough representation of an image could be obtained by training an *Encoder-Decoder* network for reconstruction, and then taking the Encoder to generate feature maps.

Aggregating the feature maps of instances in a bag is also not straightforward. Few images in the bag are actually the true discriminator for the given bag for the positive cases. Hence, getting a good aggregator that correctly emphasizes them is essential. We think a learnable aggregator is the most generalized path to be taken.

These two challenges, coupled with the desire to integrate the numerical clinical information to the MIL pipeline, shape our explorations and solutions into three big paths, as shown in 1. In the following sections, we present our methodologies, their advantages, performance, and limitations.

2.1. Clinical Information Only Approach

On the first hand, we chose to train simple Machine Learning (ML) algorithms on the classification task using the clinical information provided only: *Date of Birth*, *Gender* and *Lymphocyte count*. To this aim, we had to pre-process the data to: (1) encode the categorical Gender feature into a numerical one, where the classes are represented as 0 or 1, and (2) transform the *Date of Birth* into a simpler representation such as ‘Time since Birth’. This value corresponds to the age at the moment of acquisition, plus an additive bias as we do not know when the acquisitions were made.

We fitted three different classifiers: **Random Forests**, **XGradient Boosting** and **Logistic Regression**. In all cases, the hyperparameters were chosen by performing a non-exhaustive manual grid search. 1 compares them on both the train set and the validation set, and the best model was evaluated on the test set on the Kaggle platform.

Table 1: Balanced Accuracy Scores for *Clinical-information only methods*.

Algorithm	**Train Score**	**Val Score**	**Test Score**
Random Forest	0.1	0.8516	0.8649
XG Boosting	0.9522	0.8775	-
Logistic Regression	0.8663	0.8775	-

2.2. Feature Extraction and Classical ML Approach

To handle the visual information, an **Autoencoder** was trained on the reconstruction task to get a representative feature vector for each image in a bag. This can be done in an unsupervised manner using *Mean Squared Error* (MSE) as the objective. If the decoder is capable of reconstructing the original image from the latent space representations, we can safely assume that they are representative enough for our task too.

Once we had the trained AutoEncoder, we obtain features for every image using the Encoder only. Afterwards, we took two different "classical" aggregation approaches:

Statistical Description of Patients The first aggregation method was just taking statistical features per bag, being this features: *Arithmetic Mean* (AM) and *Minimum* and *Maximum* (MM). Moreover, we used a *Discriminative Mapping* (DM) for each bag, taken from the mil library[1]. All of these features were added to the Random Forest presented above. After manual tuning of the parameters, we got the results presented in Table 2

Table 2: Balanced Accuracy Scores for *Statistical Descriptor* methods.

Features	**Train Score**	**Val Score**	**Test Score**
DM	1.0000	0.9028	0.7481
DM + AM + MM	1.0000	0.8593	-
AM + MM	1.0000	0.7000	-

Multi Instance SVM Lastly, we used the misvm[2] library to perform classification directly on the bags. This library implements multiple classification techniques, adapted to the multi-instance setting, from where we chose to experiment with mi-SVM (Andrews et al., 2002), NSK (Gärtner et al., 2002) and sMIL (Bunescu and Mooney, 2007). A small description of these methods is presented in the Appendix A.

1. `https://github.com/rosasalberto/mil`
2. `https://github.com/garydoranjr/misvm`

We trained 7 total classifiers on two sets of features for each image, obtained by using different reconstruction losses in the AutoEncoder training phase (MSE and LPIPS). Table 4 in the Appendix A summarizes the information of the trained classifiers and their performances on the train and validation sets. The output of all these 7 models is then averaged to achieve better robustness capabilities in our model. This last model achieved 0.8843 and 0.8212 scores on train and validation, and was evaluated on the Kaggle test set, obtaining a score of 0.7558.

One last model consisted on appending the predictions of all the different classifiers to the numerical data and training a one layer (800 neurons) MLP on the full data. This achieved 0.9283, 0.8240 and 0.7896 on train, validation and test.

2.3. End-to-End Deep Learning Approach (DeepAGG)

Lastly, we designed our own deep learning network, with inspirations from current literature at various steps. We used a ResNet18 encoder which we trained ourselves on the challenge dataset in an AutoEncoder setting, following the same method as previously mentioned. The encoder is used as feature extractor in our network, and we aggregate the features of each patient's images using the method proposed by DSMIL (Li et al., 2021). The aggregation takes place using a maxpooling block and another masked non-local block. Further details about the aggregator are given in the appendix. We further used weighted losses to counter the class imbalance in the training set (**30:70**).

We obtained a balanced accuracy of **0.85**, **0.76**, and **0.59** on the train, validation, and kaggle test, respectively. However, we note that the results are not optimum, as we think the network needs further hyperparameter tuning which we have been unable to carry out given our limited computational resources. We justify our claims through the following observations. The network loss does not converge even after 150 epochs of training. This shows that the network has a lot of learning capacity left. Due to computational limits, we are only able to train the network one-bag a time, which translates to an irregular loss curve and slow learning. We believe that with a higher batch size, the model should converge faster. Moreover, we had to train our feature extractor with a reduced input size of (64×64). We believe we would obtain a better feature map when using the full images, improving classification as well.

3. Model Tuning and Comparisons

3.1. Data Exploration and Validation Methodology

To perform all validations we split the whole available data with a **4:1** ratio, having **130** patients left in the training set and **33** in the validation set. The splitting was done by stratifying on the labels, so that we have the same proportion of positive cases on each set. This splitting was saved into different '.csv' files in order to use the same splitting to train and evaluate all the different proposed models. The distribution of the variables is shown in the Appendix C.

The metric for all tests was Balanced Accuracy, defined by the average of sensitivity and specificity.

3.2. Feature Extraction Tuning

For the extraction of features from the images we trained an *Autoencoder* on the reconstruction task. The following aspects were considered for the design of the network: (1) Size of the embedding space, (2) Size of the input images, (3) Architecture of the Autoencoder, (4) Loss function. The decisions were finally taken over the visual results of the reconstruction, presented in Figure 2. In particular, we valued specially the ability to reconstruct basic shapes and sizes, as these are known to be relevant for classification.

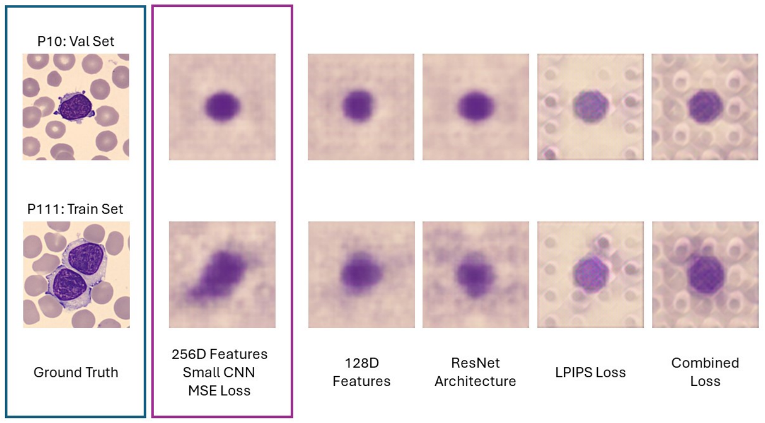

Figure 2: From left to right: (1) GT image; (2) Best reconstruction achieved with a 256D embedding space, a small CNN as encoder and MSE loss; (3) Baseline with 128 latent features; (4) Baseline with a ResNet18 as encoder; (5) Baseline with LPIPS loss and (6) Baseline with a combination of MSE and LPIPS as loss.

4. Conclusion

In this work, we tried several approaches with varying degrees of complexity to solve the *Multiple Instance Classification* problem. Table 3 compares the selected models that were evaluated on the test set. Here, we see that we can learn to classify with a reasonable score ($\approx$ **0.75**) both using only numerical data and only visual data; but we do not see any increase on the performance when combining the two sources of information. Moreover, we obtained the best result with a very simple approach that does not even take into account the images, which we assume that is just chance. We carried an extended ablation study with varying components at the feature extraction, aggregator, and classifier levels.

Table 3: Balanced Accuracy Scores of main models on train, validation and test sets.

Algorithm	**Train Score**	**Val Score**	**Test Score**
Numerical-Only Random Forest	1.0000	0.8516	0.8649
Statistical Descriptor	1.0000	0.9028	0.7481
Voting SVMS	0.8843	0.8212	0.7558
MLP w/features	0.9283	0.8240	0.7896
DeepAGG	0.8559	0.7609	0.5714

References

Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support vector machines for multiple-instance learning. *Advances in neural information processing systems*, 15, 2002.

Razvan C Bunescu and Raymond J Mooney. Multiple instance learning for sparse positive bags. In *Proceedings of the 24th international conference on Machine learning*, pages 105–112, 2007.

Joshua Butke, Tatjana Frick, Florian Roghmann, Samir F El-Mashtoly, Klaus Gerwert, and Axel Mosig. End-to-end multiple instance learning for whole-slide cytopathology of urothelial carcinoma. In *MICCAI Workshop on Computational Pathology*, pages 57–68. PMLR, 2021.

Thomas Gärtner, Peter A Flach, Adam Kowalczyk, and Alexander J Smola. Multi-instance kernels. In *ICML*, volume 2, page 7, 2002.

Bin Li, Yin Li, and Kevin W Eliceiri. Dual-stream multiple instance learning network for whole slide image classification with self-supervised contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14318–14328, 2021.

Linhao Qu, Yingfan Ma, Xiaoyuan Luo, Manning Wang, and Zhijian Song. Rethinking multiple instance learning for whole slide image classification: A good instance classifier is all you need. *arXiv preprint arXiv:2307.02249*, 2023.

Appendix A. Multi-Instance SVM

The algorithms chosen to be used for the Multi-Instance SVM approach are the following:

- mi-SVM : This approach modifies the standard SVM formulation so that the constraints on the instance labels correspond to the MI assumption that at least one instance in each bag is positive.
- NSK : The Normalized Set Kernel approach uses a kernel to map entire bags into features, and then uses the standard SVM formulation to find bag classifiers. The advantage of using this formulation is that it has a complexity of $\mathcal{O}(n\_bags^2)$, and does not depend on the amount of instances on each bag.
- sMIL : This algorithm bias the SVM formulations to handle the assumption that there are very few positive instances in each positive bag. Even though we do not have information of positive instances, we could assume that this is our case.

The detailed description of the evaluated models is as follows:

Table 4: Balanced Accuracy Scores for *Multi-Instance SVM methods.*

Name	**Algorithm**	**Kernel**	**Loss**	**Train Score**	**Val Score**
clf$_1$	NSK	Quadratic	MSE	0.9278	0.7848
clf$_2$	NSK	Linear	MSE	0.8114	0.8212
clf$_3$	NSK	Linear	LPIPS	0.7653	0.7500
clf$_4$	mi-SVM	Linear	MSE	0.7069	0.7174
clf$_5$	mi-SVM	Quadratic	MSE	0.9374	0.8516
clf$_6$	mi-SVM	Linear	LPIPS	0.8188	0.7167
clf$_7$	sMIL	Linear	MSE	0.6501	0.8512
Average	-	-	-	0.8843	0.8212

Appendix B. DSMIL Feature Aggregator and Classifier

Let $B = \{x_1, x_2, ..., x_n\}$ denote a bag of patient images. For a feature extractor f, we get feature maps for each image as $h_i = f(x_i)$.

In the **first block**, the max-pooling one, we multiply a weight vector W_m with all the feature maps. And then take the maximum as our max-pooling score. Furthermore, we call the feature vector corresponding to the $score_m$ below as critical vector, h_m.

$$score_m = max\{W_m h_1, W_m h_2, ..., W_m h_n\} \tag{1}$$

In the **second block**, we have two different weight vector, W_q and W_v, that generates the query and value vectors. We get $q_i = W_q h_i$ and $v_i = W_v h_i$, respectively. We then define a similarity score between each query vector and the query vector for the critical vector (h_m) as,

$$U(h_i, h_m) = \frac{exp(< q_i, q_m >)}{\sum_k exp(< q_k, q_m >)} \tag{2}$$

where $<,>$ denotes the dot product.

We then take another learnable weight vector W_b and get our second score as,

$$score_b = W_b \sum_i U(h_i, h_m) v_i \tag{3}$$

Our final score for the bag is the average of both the scores and this score is used for the classification objective.

Appendix C. Data distribution

We observe a significant imbalance in the data distribution with respect to different features. Around a third of the patients carry tumorous lymphocytes, and the positive cases are more prevalent in the patients with higher age and they are consequently represented more in the dataset. Furthermore, the lymphocyte count is also skewed towards a lower number for most patients. In short, creating a balanced dataset for training and validation that represents all these imbalances is particularly challenging and carries a significant weight in improving the overall classification performance in the test dataset.

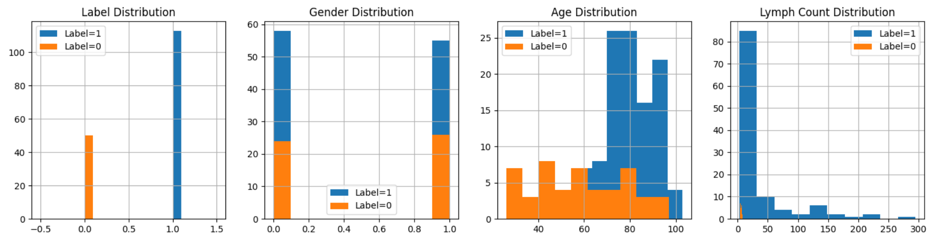

Figure 3: Distribution for different features in the dataset

Appendix D. Things that did not work

Our first attempt on the problem was just performing a naive propagation of labels from the patient to all the images of the bag. However, we did not find good results on this pathway. Afterwords, we attempted to formulate a semi-supervised approach for the problem. Our hypothesis was that most of the images are cancer-free, even for diagnosed patients. If we start with pseudo-labels on a few positive images, we could iteratively propagate the labels to other positive images, and when we have labeled a significant amount of images, we could train the images in a supervised non-MIL fashion. We tested our initial hypothesis by projecting all the images into a lower dimension (2D and 3D) using UMAP dimensionality reduction method 4, and we observed that indeed most of the images cluster into a single blob with relatively few images in small cluster all around. This approach looked promising. However, as is the case with semi-supervised methods, if the initial pseudo-labels aren't sufficiently good, the error starts propagating. In the end, we couldn't come up with a sufficiently reasonable pipeline and we chose to explore other methods as discussed already.

In our DeepAGG framework, we also tried an end-to-end learning approach where we tried to make our network learn to create feature maps without the need to plug in previously trained encoder using a ResNet18 feature extractor. However, it did not work, and we observed some very strange phenomena. We observed that our training loss was going down significantly and for hundreds of epochs, however, neither the training nor the validation performance would improve. In our investigation [3], we found that this is a common phenomena with binary classification tasks and the use of binary cross entropy objective. However, the usual remedy would be to make the network more complex. We tried ResNet50 as feature extractor and increases the layers in the classification block. However, none of it worked.

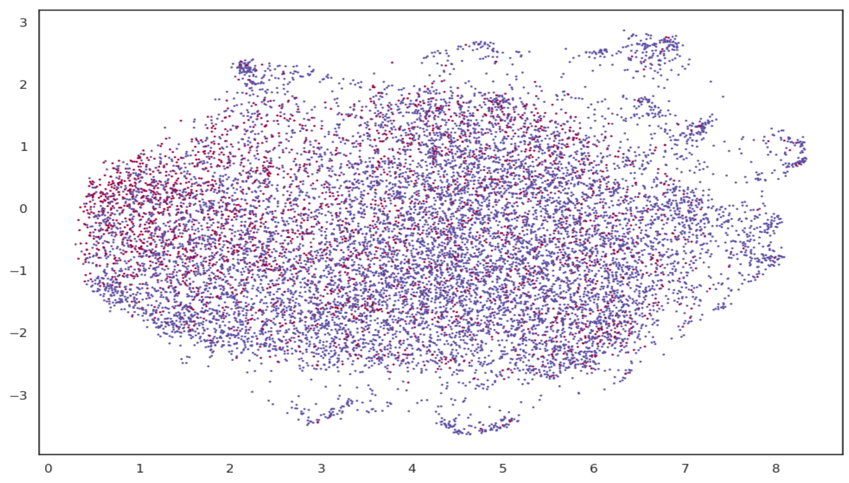

Figure 4: UMAP projection of all the images in the dataset. Red represents the cancer-free patients (true label 0), and Blue represents the images from the cancerous patients (pseudo-label 1)

3. https://stackoverflow.com/questions/43499199/tensorflow-loss-decreasing-but-accuracy-stable