Deep Learning Medical Imaging - Report of the Challenge
Team: The WAY

Wassim BOUATAY - Amrou CHOUCHENE - Yassine NAJI

Abstract

The goal of this challenge is to predict whether a patient's lymphocytes are reactive or tumoral. Every patient is represented by a bag of blood smear images and few clinical attributes. The main idea behind our approaches is to use Multi-Instance-Learning (MIL). In this report, we will detail the different methods which we will assess by various metrics.

1. Related works

The most natural method for this kind of problem is Multiple-Instance-Learning (MIL), which is a generalization of binary supervised classification in which training class labels are associated to a set of bags instead of individual instances. Several approaches had been proposed to adapt the task of medical imaging.

Let us first introduce the general methodology of MIL. A bag is considered negative (label 0) if all its instances are negative. A bag is considered positive if at least one (or small ratio) instance is positive. There are two types of MIL approaches:

1. Instance-Level approach: A classifier gives instance-level predictions that will be later aggregated to determine the bag's score.
2. Embedding-Level approach: A function is used to map each instance to a low-dimensional embedding that will be later aggregated to give a bag representation. Then, a classifier is applied to the bag's representation.

Many deep learning studies using CNN have been adapted for MIL, for instance in the field of medical imaging [1]. [2] also uses deep-learning-based approach for the same problem we have. There are also other works with attention-based approaches such as [3]. We will detail our approach in section 4.2.

The approach that has been proposed in [4] is using the MIL SVM algorithm on top of hand-crafted features (using image processing techniques) in order to determine regions of interest. These features are fed to the MIL-SVM. The authors were able to get relatively good results on the task of detecting Melanoma skin cancer : (Testing correctness : 83.75 % , Testing sensitivity : 81.33 % , Testing specificity : 90.83 %). We will detail in section 4.1 the implementation of MIL SVM and how we tried to improve this approach and adapt it to our task.

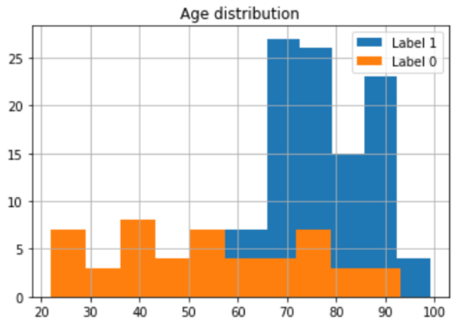

Figure 1: Age Distribution in both classes

2. Data

For each patient, we are given a set of blood smear images (the number of images varies from one patient to another), and clinical attributes (the lymphocyte count, the date of birth and the gender of the patient). For each patient in the training set, we are given the label of the patient without any instance-level labels. In fact, it is challenging for biologists to annotate each individual image.

2.1. Data Exploration

First, we analyzed the distribution of each clinical attribute. We first notice that the 'Gender' is equally distributed between the classes 'reactive' and 'malignant'. But on the other hand, we notice that the younger a patient is the more probable to have reactive cells as we can see in the figure 1. The Lymphocyte count is also an important attribute where patients with relatively high count (higher than 8) are all malignant 2.

Another important thing to verify was the number of patients in each class. We found out that the data is imbalanced. In the training set, we have 113 malignant cases and

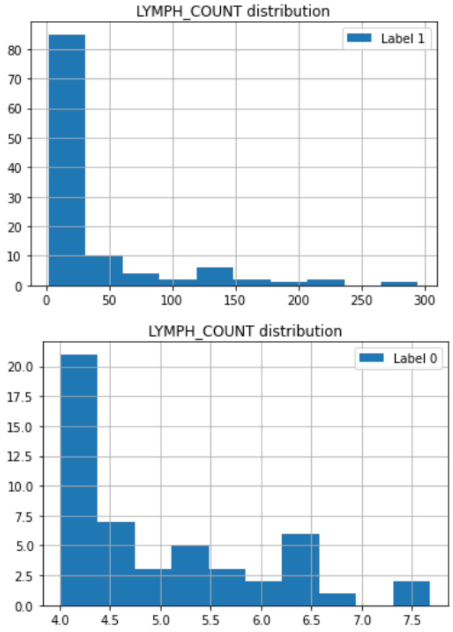

Figure 2: Lymphocyte Distribution in both classes

only 50 reactive cases. Thus, the need of being cautious and of using appropriate measures when we evaluate the model.

2.2. Stratification

In order to ensure a better generalization and to ensure that the data distribution of the validation is similar to the training set, we used Sklearn's `train_test_split` function to stratify patients with respect to the clinical attributes and the labels. But to do so, it is necessary to use quantized columns. For the 'gender' we changed it to binary (0 for male and 1 for female). We also changed the 'DOB' (date of birth) to an age and then quantized it into binary with balanced distribution (0 if the age ≤ 77 and 1 otherwise), similarly we quantized the Lymphocyte count to binary with balanced distribution (0 if count ≤ 7.81 and 1 otherwise).

We note that this quantification was only used to stratify the patients and we used the real values for classification.

2.3. Lymphocyte Segmentation

Learning directly from raw images seems to be a difficult task, especially that we dispose of only a few labels and many images. In order to help the network focus on the regions of interest, we performed an unsupervised segmentation technique that keeps only the lymphocytes on the image. In fact, the color of lymphocytes which is distinguishable from other cells makes their segmentation possible without any annotation. As a first prepossessing step, we transformed images from RBG basis to HSV, since we found out that the Saturation channel of the HSV color system highlights more the lymphocytes as we can see in the following figure 3

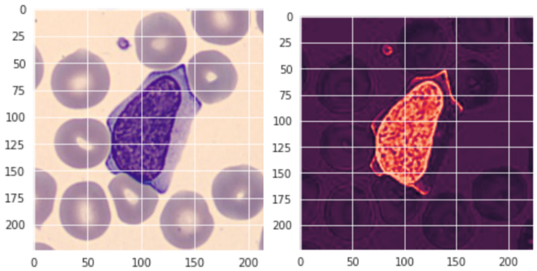

Figure 3: On the left the RGB system, on the right the Saturation channel of the HSV system

Once we did this prepossessing step, we apply K-means to the image with 2 classes in order to build a first mask, then we perform morphological operators, namely a closure operator to get a convex Lymphocyte and to recover the whole lymphocyte by adding few pixels from each side. We got good segmentation using this method as we can see in the figure 4

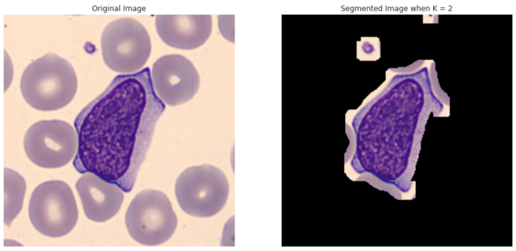

Figure 4: Segmentation output

We used the segmented images in the feature extraction task using the auto-encoder since we found out that encoding these images requires fewer dimensions (~ 4000) of the latent space compared to the original images (~ 8000) to be able to well reconstruct them.

2.4. Features Extraction

Since we don't have access to the label of each image, learning relevant features is difficult in a supervised way. Thus we managed to use sub-tasks to learn low-dimensional image representation, then to use these features as input to our models.

2.4.1 Via Auto-encoder:

A natural approach is to use an auto-encoder to learn a low-dimensional representation of images. Choosing the latent space is a sort of trade-off between the ability to reconstruct images (in high dimension) and having consistent features which will encode only relevant information of images (in low dimension) so that we prevent that the model which predicts labels from overfitting on those features. We tried various dimensions for latent spaces, and we found out that choosing 8192 as the size for the latent space in the case of non-segmented images and 4096 if they are segmented seems to be a good compromise.

The architecture we used for the auto-encoder is the following:

- ‘**Encoder :** Contains 4 convolutional blocks, each one is composed of a 3x3 convolution with ReLU activation, batch normalization and 2x2 average pooling. We used average-pooling instead of the classical max-pooling because we found out that it allows better reconstruction.
- ‘**Decoder :** Contains 7 blocks, each one is composed of a 3x3 transposed convolution with ReLU activation and batch normalization.

We trained the model on all the provided images (even the test set) for 5 epochs using the Adam optimizer with a learning rate of 10^{-3}. As we can see in figures 5, the auto encoder can reproduce well the main elements of the images.

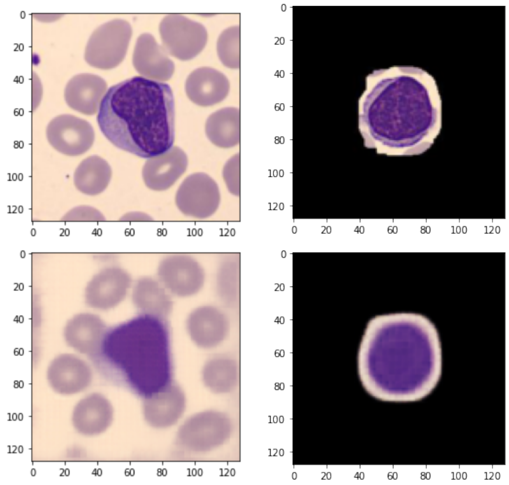

Figure 5: Reconstruction results: On the left the case of non-segmented images using 8192 features, on the right the case of segmented images using 4096 features

2.4.2 Via a model trained on a multi-classification task:

The task of classification of blood cells seems to be related to our task, since the model learns the characteristics of each type of blood cells in order to be able to distinguish between them. Thus, we trained a ResNet 50 model to distinguish between 8 types of blood cells : lymphocytes, neutrophils, eosinophils, basophils, monocytes, immature granulocytes, erythroblasts and platelets. We trained our model on the dataset available in [5], for 10 epochs using the Adam optimizer with a learning rate of 10^{-4}, we achieved an accuracy score of 98% on the validation set (20% of the whole dataset). We used the output of the last convolutional layer of ResNet as features for our task which consists of 2048 features. We didn't use these features since we found out that the auto encoder-decoder's features performed better in the different models.

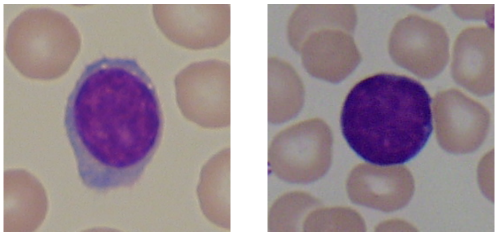

Figure 6: ALL-IDB2 dataset. From left to right: Healthy cell, malignant cell

3. Instance-level - Semi-Supervised Learning

3.1. Semi-Supervised Learning

Semi-supervised learning is a machine learning approach that combines a small amount of labeled data with a large amount of unlabeled data during training. These unlabeled data, when combined with labeled data, can produce high accuracy improvement as discussed in [6].

In our case, we have only negative labeled images since negative patients' images are all labeled as 0, but we don't have any information concerning the positive labeled images. For this reason, we will first train a model on ALL-IDB2 dataset [7], which is very similar to our dataset, and which contains instance-level labels. This way, we are able to obtain enough information about the positive labeled images and we can apply our Semi-Supervised Learning approach.

3.2. ALL-IDB2 dataset

ALL-IDB2 is an Acute Lymphoblastic Leukemia Image Database. It contains 260 images: 130 images labeled 0 and 130 images labeled 1. An image is labeled 0 if the cell placed in the center of the image is not a blast cell (healthy cell), and equal to 1 if the cell placed in the center of the image is a blast cell (malignant cell). The figure 6 shows one healthy cell sample, and one malignant cell sample from ALL-IDB2 dataset.

3.3. Training on ALL-IDB2 dataset

The next step is to build a strong Neural Network model capable of learning efficiently the features of the ALL-IDB2 dataset, because this model will be our starting model when training on our dataset in the Semi-Supervised approach.

For this purpose, we tried 4 pretrained models on ImageNet dataset: AlexNet, Inception_v3, Resnet18 and VGG19. For each model, we removed the last layer, we added a new Linear FC layer having output=2 and finally a LogSoftmax activation function. We trained the model on 200 balanced samples, and test the model on 60 balanced samples. We used the Negative Log-Likelihood loss and the SGD optimizer. The best validation accuracy was obtained using AlexNet and VGG19. We finally chose VGG19 because it's known to be an excellent feature extractor. This will be very helpful since ALL-IDB2 and our dataset are very similar.

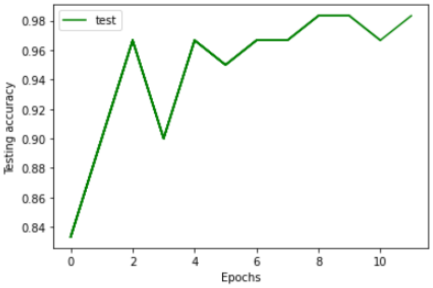

Figure 7: Validation accuracy on ALL-IDB2 using VGG19

3.4. Training the Semi-Supervised model

3.4.1 Preprocessing

Our dataset and ALL-IDB2 varies only in the mean and the standard deviation of each channel. Hence, we transformed our dataset images in order to make their distribution similar to ALL-IDB2 images' distribution. In other words, we changed the mean and the standard deviation (std) so that they match the mean and standard deviation of ALL-IDB2 samples.

To do so, we apply the following transformation: Mathematically speaking, if we apply the following formula on a vector X which has initially a mean m_0 and a standard deviation s_0:

$$X_{new} = m_1 + \frac{s_1(X - m_0)}{s_0}$$

then the new vector X_{new} will have a new mean m_1 and new standard deviation s_1.

We Applied this formula to the three channels of each image of our dataset. The results are shown in figure 8.

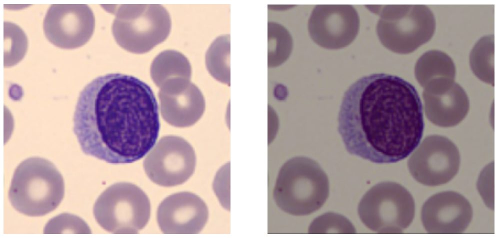

Figure 8: Preprocessing result. From left to right: original image, preprocessed image

3.4.2 Training: Semi-Supervised phase

Starting from the trained model on ALL-IDB2 , we make instance-level predictions on the bag of images of each patient labeled as 1. And at each iteration, we take the images predicted as 1 with the highest confidence score and we add them to the training set of ALL-IDB2. We add also images from healthy patients to the training set of ALL-IDB2 in order to keep the data balanced. Then we train again a new model on the new updated training set of ALL-IDB2. We repeat this process until we add 100 positive images and 100 negative images from our datat set. To avoid noisy labeling, during the first 5 iterations, we only add 2 images predicted as 1 having the highest accuracy among all 1-labeled patients and 2 images labeled as 0 from the healthy images. Then for the remaining iterations, we add the top 5 predicted images. We repeat the process 23 times in order to add exactly 100 images labeled as 1 to our training set.

3.4.3 Training: Final phase

Now that we prepared 100 predicted positive images from the positive patients, we combine these predicted images with 100 negative images from the healthy patients in order to form a sub-training set of labeled instances. We train the pretrained VGG19 over this new training set. Finally, for the prediction of the label of each patient, we construct a matrix of size (n_patients × 4), where for each patient we calculate the number of images predicted as 0, the number of images predicted as 1, the LYMPH_COUNT and DOB. Then we pass this matrix to a classifier. We used lightgbm and xgboost, but the latter gave better results. The pseudo-code 1 highlights the steps of the Semi-Supervised approach.

```
- initialize $X_{train}$ = samples from ALL-IDB2 ;
  $y_{train}$ = the corresponding labels ; i = 1 ;
while number of iterations i $\leq$ 23 do
    - model = train ( $X_{train}$ , $y_{train}$ ) ;
    - for each patient labeled as 1: make
      instance-level predictions of its images ;
    - if i $\leq$ 5 : select the best 2 1-predicted images
      and append them to $X_{train}$ , append 2 healthy
      images from healthy patients. ;
    - if i $\geq$ 6 : select the best 5 1-predicted images
      and append them to $X_{train}$ , append 5 healthy
      images from healthy patients. ;
end
- $X, y$ : eliminate ALL-IDB2 images from $X_{train}$
  and $y_{train}$ ( keep only 0-labeled images and
  1-predicted images from our dataset ) ;
- model = train (X,y) ;
- for each patient : calculate the number of
  0-predicted images, 1-predicted images using
  model. Combine these values with DOB and
  LYMPH_COUNT and construct a new training
  matrix $X_t$ ;
- Training xgboost classifier on $X_t$ ;
```

Algorithm 1: Pseudo code of Semi-Supervised approach

3.5. Results

Balanced accuracy	**0.963**
F1_score	**0.962**
Precision	**1.000**
Recall	**0.926**
Kaggle test score	**0.823**

4. Embedding-level Multi-Instance Learning approaches

4.1. MIL SVM

As we mentioned in the related works we tried to improve the approach of [4] by using learned features instead of hand-crafted ones. Indeed, we used the features learned via the auto-encoder as input to the MIL-SVM algorithm.

The SVM variant that we tried is called MI-SVM (discussed in [8] and implemented in [9]). Basically, we admit that for each positive bag there is at least one positive instance, and for negative bags all instances are negative. we consider for the positive bags only one data point which is the mean of all the points of the bag and we consider that this point has a positive label. However, we consider all the instances of the negative bags since we know that their labels are negative. At each iteration, we compute the decision hyper-plan and then the margin for all instances of the positive bag, and we consider the point which maximizes this margin as the representative instance of the bag. We repeat this steps until all the representatives of the positive bags don't change. We can summarize MI-SVM in the following pseudo-code :

```
Result: (w , b )
initialize $x_I = \sum_{i \in I} \frac{x_i}{|I|}$ for every positive bag $B_I$;
while Selector variables s(I) have changed do
    compute QP solution w , b for data set with
      positive exemples $\{x_I : Y = 1\}$;
    compute output $f_i = \langle w, x_i \rangle$ + b for all $x_i$ in
      positive bags set
      $x_I = x_{s(I)}, s(I) = argmax_{i \in I} f_i$ for every I,
      $Y_I = 1$;
end
```

Algorithm 2: Pseudo code MI-SVM

4.1.1 Training

We used the implementation of MI-SVM available in the repository [9], this implementation uses QP solver in order to compute the decision boundary at each iteration. We tuned therefore 2 parameters which are the regularization term C and the maximum number of iteration. We set the regularization term to C = 50 since we found out that it's a good compromise between overfitting ($C << 50$) and underfitting ($C >> 50$) and we set the number of maximum iterations allowed to 20 (usually the algorithm converges around only $\sim$ 10 iterations).

We used the features generated by the auto-encoder (trained on segmented images) as input of the MIL SVM. In order to prevent overfitting on the high number of features (4096), we performed PCA to reduce the dimensionality of the input to 100. We summarize our results in the following table, where we used 20% of data as a validation set stratified with respect to the clinical data and the labels:

Model	Balanced acc	F1-Score	Recall	Precision
PCA MIL SVM	**0.74**	**0.81**	**0.78**	**0.85**
Normalization without PCA	0.61	0.63	0.52	0.80
Without Preprocessing	0.54	0.71	0.69	0.72

4.1.2 MIL SVM & XGBoost

Since we can feed only the image features MIL SVM, we wanted to make use of the clinical data also since it give a good apriori on the state of the patient. Thus, we used the rate of positive samples in the bag (predicted by SVM) as a features for the XGBoost classifier which takes also into account also clinical data (Lymph count and age), we trained it using a learning rate of 0.1 and 100 random decision trees, we set the paramter `scale_pos_weight` to $\frac{98}{142}$ in order to avoid that our model predict many ones. We got the following results by considering 20% of the dataset as validation set stratified with respect to the clinical data and the labels:

Model	Balanced acc	F1-Score	Recall	Precision
MIL SVM & XGBoost	**0.82**	**0.91**	**0.95**	**0.88**
With only MIL SVM	0.74	0.81	0.78	0.85
With only clinical data	0.69	0.83	0.88	0.78

We scored a balanced accuracy of 82% on the public leader board using this method.

4.2. Deep Learning MIL

4.2.1 Architecture

For the deep learning approach, we feed our model a bag of features and also the clinical data.

The first thing to do is to aggregate the extracted features in the bag to create an embedding of the whole bag. We tried different aggregation functions, like the maximal value for each feature along all patients. We have also tried to take a concatenation of the mean and the standard deviation. We have finally chosen an attention based approach as described in [3].

$$h_{bag} = \sum_{i=1}^{I} a_i h_i$$

where I is the number of instances in the bag, a_i is the weight given to the features of the i-th instance h_i. The attention weights are given by:

$$a_i = \frac{\exp(w^T \tanh(V h_i^T))}{\sum_{i=1}^{I} \exp(w^T \tanh(V h_i^T))},$$

where $w \in \mathbb{R}^{L\times 1}$ and $V \in \mathbb{R}^{L\times M}$, are learnable parameters and M is the number of features per instance.

We create two separate Multi-layer-Perceptron (MLP). The first MLP consists of n fully connected layer and it is applied to the bag representation to create N features. The second MLP consists of f fully connected layer and it is applied to the clinical data (Age and Lymphocyte count) to give F features. After each fully connected layer, we apply as activation the function ReLU, a dropout layer of rate $p = 0.05$ to reduce the overfitting and a normalization layer to have better convergence.

The best parameters for the two MLPs found was to use only one hidden layer for both MLPs ($n = f = 2$) and a width equal to $N = F = 128$. For the attention we take $L = 64$.

Finally, we concatenate the features given by the two sub-networks, and we apply a final fully connected layer to output the prediction of the bag.

4.2.2 Training and Tuning

(a) **Optimizer**: For training, we used Adam as an optimizer with a very low learning rate equal to 10^{-6} with a scheduler that multiplies the learning rate by 0.8, if the validation loss doesn't decrease for 4 consecutive epochs. We chose a small learning because it was very hard to train the model, it was either unstable or diverges in most cases. We also tried other optimizers like RMSprop and SGD but they gave worse results.

(b) **Loss**: We used Binary Cross Entropy loss and we used a weighted loss. We penalize more the loss when the true label is 1, because it is more frequent in the dataset and should be easier to learn. Indeed, we multiply the loss by 98/142 (the ratio of patients with label 1 in the training set), if the label is 1 and by 44/142 if the label is 0.

(c) **Features**: The used features here are the 8192 features created by the auto-encoder over the non-segmented images. We apply a Standard Scaler on these features, then we apply PCA to take the 3000 first principle components. The PCA is applied in order to get the best possible information without feeding unnecessary features that will make the overfitting worse.

(d) **Including the auto-encoder**:

We had also thought to train the network end-to-end by including the auto-encoder in the network. We tried 2 approaches:

• The first approach consists simply on training the full network using the sum of the classification loss and the decoder's reconstruction loss with a factor λ (we consider BCE loss in both cases). This approach has several drawbacks. The first issue is that the encoder receives both gradients from the decoder and the classifier which makes the training unstable. The second one is that the two losses that we optimizes requires different learning rates (10^{-6} for the classifier vs 10^{-3} for the auto-encoder). This makes the training of the auto-encoder very slow, and thus the network don't seems to learn good features and simply overfits on the train.

• In order to solve the first issue we froze the classifier for the first 10 epochs in order to focus on the auto-encoder's reconstruction loss, then we back-propagate in 2 phases. In the first phase, we freeze the auto-encoder and we back-propagate through the classifier, and in the second phase we freeze the classifier and we back propagate through the auto-encoder. We also

Model	Balanced accuracy	F1-Score	Recall	Precision	Loss
D-MIL	**0.759** ± 0.036 (**0.834**)	0.845 ± **0.015** (0.888)	0.8322 ± **0.027** (0.869)	**0.860** ± 0.030 (**0.909**)	0.535 ± 0.048 (0.497)
Without PCA and StandardScaler	0.523 ± 0.231 (0.756)	0.594 ± 0.276 (0.875)	0.590 ± 0.276 (0.913)	0.600 ± 0.280 (0.857)	**0.396** ± **0.121** (**0.157**)
Without Clinical Data	0.745 ± 0.041 (0.806)	0.831 ± 0.030 (0.893)	0.813 ± 0.050 (0.913)	0.851 ± 0.028 (0.894)	0.621 ± 0.049 (0.564)
With segmented images	0.681 ± 0.046 (0.741)	0.818 ± 0.054 (0.884)	0.843 ± 0.115 (**1.000**)	0.803 ± 0.032 (0.857)	0.552 ± 0.038 (0.516)
Max as aggregation	0.657 ± **0.029** (0.719)	0.795 ± 0.019 (0.833)	0.801 ± 0.045 (0.869)	0.792 ± 0.024 (0.850)	0.564 ± 0.021 (0.532)
(Mean,Std) as aggregation	0.745 ± 0.033 (0.806)	**0.856** ± 0.030 (**0.893**)	**0.875** ± 0.063 (0.913)	0.840 ± **0.023** (0.875)	0.522 ± 0.016 (0.493)

Table 1: Ablation study of D-MIL model with different metrics averaged on 7 executions. We put the best value in each column in bold for each of the mean, the standard deviation and the maximum.

used two Adam optimizers with different learning rates (10^{-6} & 10^{-3}).

Since, we are feeding the network images instead of features, we tried to do a data augmentation. We used random horizontal and vertical flips to improve the model. But, it didn't help much.

In the end, we had been able to stabilize the training, however it was very slow comparing the the fixed features method, and it didn't improve our results, so we decided not to pursue this approach.

(e) **Training and Validation**: Training this model was very hard. The model overfits quickly even with weight decay and dropout.

The model also suffers from unstable convergence and fluctuates a lot due to the small number of samples. Adding dropout to handle the overfitting problem obstruct the good convergence of the model, hence the choice of a small rate $p = 0.05$.

Since the model suffers from instability, it gives different results from one execution to another. Thus, to better evaluate the model, we averaged several executions to tune the parameters. Moreover, we don't simply choose the best model given the balanced accuracy but also the loss to have a more significant and trustworthy evaluation.

One last remark about the validation set, we noticed a big difference in the results from a random split to another. We even got 98.5% as balanced validation accuracy but only 71.4% on the public test set. Thus, we concluded that the validation wasn't trustworthy and we decided to fix the split based on stratification to have a better validation. By doing stratification, we managed to overcome the problem of discrepancy between the validation and the test set.

(f) **Results**:

Using this approach, and using the best parameters, we get the following results:

Balanced Accuracy (In test)	F1-Score 1	Recall	Precision
83.48% (75.58%)	88.89%	86.96%	90.91%

As we can see, training the D-MIL model was very hard, and even with all the things that we have tried it fails to give better results than the Instance-Level Semi-Supervised approach.

In the following figure 9 and 10, we show the evolution of the balanced accuracy and the BCE loss as a function of time (epochs). These curves were averaged on 10 executions.

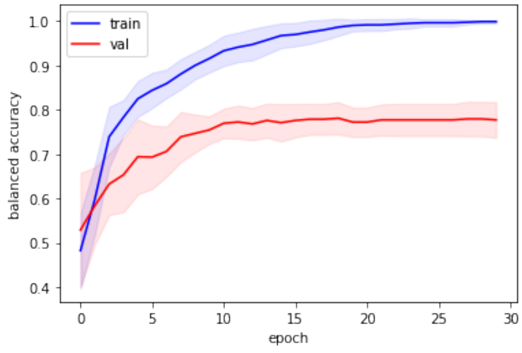

Figure 9: Balanced accuracy curve

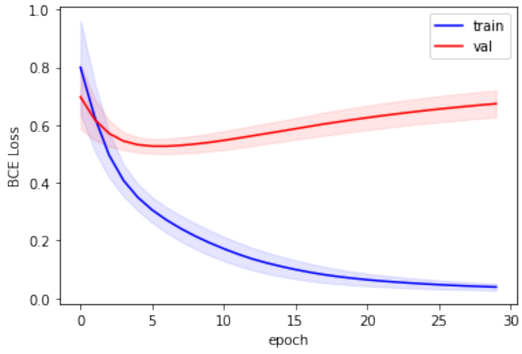

Figure 10: BCE loss curve

4.2.3 Ablation study

In this section, we make an ablation study on the model D-MIL to justify our work.

In the table 1, we have put the results of the different metrics used, averaged on 7 executions. We have also added the standard deviation and the maximal value reached between parenthesis. As we can see from the table our model D-MIL with the right component give the best overall result.

Changing the aggregation from attention to using a concatenation of mean and standard deviation also gives very close results, but using the maximum as an aggregation function decreases our results very much.

If we work directly on the 8192 features ie. if we don't use PCA and the Standard Scaler, we get very bad results, which varies a lot (standard deviation around 25% for each metric). On average, we also have a dramatic drop of more than 20% in all metrics.

By using the features of the segmented images the performance decreases too.

We had also tried to work without the clinical data, we get very similar results. We conclude that the model isn't overfitting on the simple task of using only the clinical data which are pretty informative but actually learning from the bags themselves. Moreover, using the clinical data helps our model by 1.15% on average in the balanced accuracy and by 2.8% on the best balanced accuracy.

5. Ensemble Method - Final result

As discussed in the previous sections, we detailed different approaches we have tried during this project. In order to make our solution more solid, we combined the best 5 solutions of the different approaches, then we used voting over all these solutions. This trick helped improving our score, as we obtained **85.71%** on kaggle's test set.

6. Conclusion

We discussed in this report our work for the classification of Lymphocytosis from blood-cell images. We developed embedding-level multi-Instance model (MIL-SVM and Deep-Learning based MIL) and an instance-level model (semi-supervised) which was our best approach.

With all the approaches that we have worked on, we expected scoring more that 85% on the public dataset, this shows how difficult can a be MIL-problem with few labels which is often the case for medical imaging tasks. Nevertheless, we are proud that our best model is able to outperform biologists.

References

[1] Oren Z. Kraus, Lei Jimmy Ba, Brendan Frey. *Classifying and Segmenting Microscopy Images Using Convolutional Multiple Instance Learning*. URL: https://arxiv.org/pdf/1511.05286.pdf.

[2] Mihir Sahasrabudhe, Pierre Sujobert, Evangelia Zacharaki, Eugénie Maurin, Béatrice Grange, Laurent Jallades, Nikos Paragios, Maria Vakalopoulou. *Deep Multi-Instance Learning Using Multi-Modal Data for Diagnosis of Lymphocytosis*. URL: https://hal.archives-ouvertes.fr/hal-03032875/document?fbclid=IwAR3WRNH-ZTVaboXk3CIJJP0qxoLPVm6UjGKwhI-JoaLvvg8KtQtEL_lnoAs.

[3] Maximilian Ilse, Jakub M. Tomczak, Max Welling 1. *Attention-based Deep Multiple Instance Learning*. URL: https://arxiv.org/pdf/1802.04712.pdf.

[4] Annabella Astorino , Antonio Fuduli , Manlio Gaudioso3 and Eugenio Vocaturo. *Multiple Instance Learning algorithm for medical image classification*. URL: http://ceur-ws.org/Vol-2400/paper-46.pdf.

[5] Andrea Acevedoab, Anna Merinoa, Santiago Alférezc, Ángel Molinaa, Laura Boldúa, José Rodellar. *A dataset of microscopic peripheral blood cell images for development of automatic recognition systems*. URL: https://www.sciencedirect.com/science/article/pii/S2352340920303681.

[6] *Semi-supervised learning*. URL: https://en.wikipedia.org/wiki/Semi-supervised_learning.

[7] *ALL-IDB2 dataset*. URL: https://www.kaggle.com/nikhilsharma00/leukemia-dataset?select=ALL_IDB2.

[8] Stuart Andrews, Ioannis Tsochantaridis and Thomas Hofmann. *Support Vector Machines for Multiple-Instance Learning*. URL: https://proceedings.neurips.cc/paper/2002/file/3e6260b81898beacda3d16db379ed329-Paper.pdf.

[9] *MISVM: Multiple-Instance Support Vector Machines*. URL: https://github.com/garydoranjr/misvm.

Appendix - Dropped approaches

One-Class-SVM

We know that a negative bag have only negative instances, but if it is positive there is at least one positive instance and we can't conclude anything else about the other instances.

Following the literature, we also know that the number of positive instances is very small and thus we consider them as outliers. Our idea was to use one-class-SVM to detect them. In fact, we know the labels of all instances of a negative patient and we can use them to train the one-class-SVM. The latter predicts the margin of an instance to a decision boundary (if the margin is negative then it is an outlier).

We inspected the histograms of different bags of label 1 and 0 and we didn't notice any pattern and there is no trustworthy threshold to decide whether the bag is reactive or not (we found some bags with many instances having very negative margins but corresponds to a negative patient and inversely we found some positive bags with few negative instances). Thus, we conclude that one-class-SVM also fails to distinguish the outliers and the reactive instances.

Naive Instance-Level deep MIL

This approach was our baseline, we tried a very naive instance-level deep MIL where we give as input the images to a Resnet18. We take the soft-predictions for each image individually. Then, we simply take as a prediction for the bag either the mean of the prediction or the maximum. We have also tried different architectures like SENet and a pretrained Resnet18 on ImageNet.

We have also applied Xgboost to the clinical data (Age and Lymphocyte count) with the predictions of each bag. This improved greatly the performance but only because the clinical data are pretty informative.

We did not pursue this approach because it was only overfitting on the clinical data.