Kaggle Challenge Report
Classification of lymphocytosis from white blood cells

Team: Diffused-Burgers

Florent Pollet[1] FLORENT.POLLET@ENS-PARIS-SACLAY.FR
Gauthier Multari[1] GAUTHIER.MULTARI@ENS-PARIS-SACLAY.FR
[1] *Ecole Normale Supérieure Paris-Saclay*

1. Introduction

Lymphocytosis represents a significant clinical challenge, presenting itself in a dual nature: it can be a benign response to factors such as an infection or it can indicate a more serious underlying condition such as a lymphoproliferative disorder, a cancer of the lymphocytes. The current diagnosis protocol involve both microscopic examination of the blood cells and the evaluation of various critical attributes, including the patient's age and lymphocyte count. These factors are not only crucial for determining the presence of lymphocytosis but also for classifying it into its respective subtype, which is needed to decide the subsequent treatment pathway.

Given these challenges, developing automated, reliable methods for the initial assessment of lymphocytosis could significantly streamline the diagnostic process, helping to more accurately identify patients who truly require further, more elaborate testing like flow cytometry. This approach could both enhance clinical efficiency and ensure that patients receive a timely and accurate diagnosis.

The dataset used for this challenge was built from the clinical records of 204 patients treated at the routine hematology laboratory of *Lyon Sud University Hospital*. The dataset comprises images of blood smears produced by a Sysmex automated tool and photographed using a DM-96 device, alongside the respective patient attributes. It is constituted by 142 subjects categorized into 44 reactive and 98 malignant cases for the training, and data from an additional 42 subjects for the testing.

The metric used to evaluate the performance of our model for this challenge is the **balanced accuracy**. This metric is particularly useful when, like in our case, there is an significant imbalance between the classes of our set, which can lead to a bias in the model's performance towards the majority class. As our goal is to provide a model that could be used in a clinical setting, we want to avoid a high false positive rate.

This report explains the models built to classify patients between reactive cases and cancerous cases. The main idea developed through the challenge was to leverage both information contained of the clinical data of the patients and nucleated cells images. These two different modalities were evaluated on their own in the first place before merging them in an efficient way. The GitHub repository of the code developed for this challenge can be found here: `https://github.com/florian6973/dlmi_challenge`.

2. Data overview and methodological components

In this section, we are going to present how we tackled the issues regarding data and the strategies we adopted. We will first discuss the intuitions behind the different choices we made before reporting the impact of the combinations tested.

As discussed in the previous section, the dataset of this challenge yields two different kinds of features. The first one comes from the clinical data of the patients, namely their date of birth, gender and lymphocyte count. The second one is the set of 224x224 RGB images of the blood smears.

2.1. Clinical features

We first looked at the clinical features to evaluate whether or not they could help discriminate the patients. After converting the date of birth variable to the age of the patient, which is a single number easier to handle for a model, we obtain the Figure 1. The patients with cancer are colored in red whereas the other ones in blue. We can clearly see that all the patients that have a high number of lymphocytes have cancer, though a low amount of lymphocytes does not mean that the patient does not have cancer. Also, older people are much more likely to have a cancer for the same lymphocyte count. The gender does not seem to discriminate the patients. From this observation, we decided to try and use this information in our final model. To do so, we tried various architectures of Multi-Layer-Perceptron in order to try and classify the patients based on these features, because it is a very common and flexible supervised Machine-Learning method for simple tabular data.

2.2. Image features

The image features are more complicated. Each patient has a set of various sizes (between 16 and 198) of unlabeled images. We have no other information regarding these images than the patients' labels. One could assume that each image inherits the class of its patient but this would lead to many issues. Indeed, a sick patient could have both normal and abnormal lymphocytes. It is therefore necessary to classify a patient using all its images together instead of considering images individually.

To tackle this problem, we decided to try the **Multi-Instance Learning** (MIL) approach. In this approach, objects/bags are described by multiple observations (instances) with labels being provided only for the bags as a whole. This method does lead to other difficulties, for instance we can imagine that some of the images contain misleading information about their patient, as we cannot expect all of the images to be abnormal for a patient with tumor. Furthermore, it is necessary to heavily adapt the training process, as the usual Computer Vision models are trained for single instances tasks.

Then, there is the question of how to aggregate the results of the different images of a bag to make a prediction. There are many different methods to do so (Dietterich et al., 1997; Ilse et al., 2018; Zhou and Xu, 2007; Feng and Zhou, 2017). The most simple and widely used is Max Pooling. The basic idea is to take the maximum value over a specific feature across all instances in the bag as the representative feature of the bag. Max pooling works well in scenarios where the presence of at least one positive instance makes the whole bag positive. Similar to Max Pooling, Mean Pooling aggregates the features of instances by computing their mean. This approach assumes that every instance contributes equally to the label of the bag and is useful when the relationship between instances and the bag label is linear. We can also cite Attention-based aggregation as well as Convolutional aggregation, which can be relevant for this problem.

Besides, the MIL approach is very memory hungry, as we have to load all the images for a patient at once for a batch. Given the capabilities of the hardware at our disposition (a NVIDIA RTX 4080 12GB laptop GPU and a NVIDIA V100S from OVH Cloud), the first idea that came to mind was to use a relatively small model in order to extract the features from the images without running out of VRAM. Our first choice was a ResNet18 given the known performances of this model.

Another problem of this challenge is the very small number of patients to train large model like ResNet18. To tackle this issue, we chose two approaches. The first one is to use **transfer learning**. By starting with a pre-trained model, i.e re-using the weights of the ResNet18 trained on ImageNet, we are able to use the pattern recognition capabilities of the model without having to retrain it completely, by simply replacing the last layers with custom blocks.

The second approach is to use **data augmentation**. There are many different way of augmenting the dataset, but we chose to remain simple. Indeed, augmentations like color jitter can be unnecessary or even counterproductive given that all images have a similar luminosity, contrast and color distribution. It is to be noted that, if we intended to use the model for images from other types of microscopes, such transformations would help to have a more robust model that generalizes better. Our final choice is to use random horizontal and vertical flip of the images, as well as random rotations at an angle up to 180 degrees. We also tried different normalizations so that images have the same distribution as the one of the images used to train the model, for the cases where we used a model with pre-trained weights.

3. Model architectures, tuning and comparison

3.1. Development environment

Various choices were made concerning the model tuning and comparison. On the technical side, we decided to use **Pytorch Lightning** (Lightning, 2020) as base framework for our developpement as it allows for a faster prototyping, better modularity and easier setup for reproducibility than vanilla Pytorch. We used the open-source **Mlflow** tool (MLflow, 2023) to visualize and track our different experiments and Hydra (Hydra, 2023) to have a clean configuration tool for our different models. Finally, we used the **Ray** framework (Ray, 2023) to automatize the research of the best hyperparameters for our different architectures. The seed of the pseudo-random generators can be set from the **Hydra** configuration files, allowing for complete reproducibility of each experiment. Configuration files are provided with the code.

3.2. Architectures

First architecture First, as stated in the previous section, the chosen type of model to classify the patients solely based on their clinical features is a Multi-Layer-Perceptron. We tested various types of architectures, with different amounts of hidden layers and different layer sizes. We found that the bigger models were not suited for this problem, as they would overfit very easily. The final model has a fully connected layer with 3 inputs (age, lymphocyte count and gender) with a ReLU activation function, followed by another fully connected layer with ten hidden inputs and two outputs, one for each class, followed by a LogSoftmax layer, which guarantees a better numerical stability than Softmax. To train this model, we used a Cross-Entropy loss and an Adam optimizer with a learning rate of 10^-2, a batch size equal to 1 for 20 epochs.

Second architecture Second, to extract the image features, we used a modified pre-trained ResNet18. It consists of the regular ResNet18 layers, with the weights trained on the ImageNet dataset. We replaced the final fully connected layer by two fully connected layers. The first one is of size 512 with a ReLU activation function followed by the final classification layer. We adapted the training step in order to aggregate the different instances of each bag. We performed experiments using Max Pooling as well as Mean Pooling. Max Pooling performed extremely poorly, and we achieved our best results using Mean Pooling. We tried to fine-tune the model with all the weights unfrozen, but it did not manage to learn properly, as expected given the low amount of data available. We tried two approaches for the freezing of the parameters. The first was to leave only the Layer 4 as well as the classification head unfrozen, and the second to only unfreeze the classification head. Leaving the Layer 4 unfrozen led to lower performances than only training the final head. To train this model, we used a Cross-Entropy loss and an Adam optimizer with a learning rate of 10^{-3}, a batch size equal to 5 for 50 epochs.

Mixture of Expert architectures The main idea behind these architectures is to use the information extracted from the images and the clinical data together. To do so, we created a third model that takes both of the outputs of the other models. We chose to use another simple MLP, with 4 input features (2 for each model), a hidden layer of size 10 and a classifier layer. All models have separate optimizers, and this one uses a learning rate of 10^{-2} like the other MLP. We experimented different combinations of MLP/image feature extractor, the main results can be seen in Table 1. Moveover, building separate models and assembling them together can give us the same insights as an ablation study.

3.3. Training strategy and internal validation procedure

As data preprocessing, we performed some feature normalization, to rescale all clinical attributes to $[0, 1]$. Moreover, given the low amount of data with regard to the difficulty of the task, the models have a very high risk of over-fitting. To properly evaluate the generalization capabilities of our models, we used a shuffled **stratified k-fold cross-validation** with 5 folds. It allows to keep the same class proportion in each random subset. Had we gone further, we would have used Bayesian Deep Learning techniques like Monte-Carlo Dropout to predict the uncertainties of each patient and clearly help the practitioners to assess the reliability of the prediction. Furthermore, during a training, we make sure to keep the best checkpoint of the models by taking the one with the best validation balanced accuracy. Finally, we used custom seeds to make sure all our experiments are reproducible, and we tested some different seeds to check the variability of our results as well.

3.4. Results

In this section, we will discuss the results of the main experiments performed, and reported in Table 1.

When mixing the MLP with the ResNet18 without data augmentation, we obtain an average of 87% balanced accuracy on the validation dataset folds with a 6% standard deviation. This translates to an 83% accuracy on the (half) test dataset, a 4% improvement over the single MLP. When using data augmentation, the average balanced accuracy further improves to reach 91% and the standard deviation decreases by 1%. This seems to indicate that when trained with data augmentation, the model's ability to generalize is improved.

From our testings, increasing the batch size improved the results significantly, but we were limited to a size of 5 due to our hardware limitations, so we were not able to test it further.

We did a final experiment with a much larger image feature extractor, the base version of the Vision Transformer 16. The idea was that, given the model is much larger, it may capture better the features in the images and have a better ability to generalize. We had to reduce the batch size to 3, in order for it to fit in our GPUs. We got an average of 89% balanced accuracy on the validation dataset, but this time with a 2% standard deviation, a very promising result that indicates that this model does generalize better.

Please note that despite our standard deviation estimate, we obtained some significantly lower results on the public leaderboard, especially for the ViT transformer, which was more prone to over-fitting during the training. This may be understood due to its even smaller size than the validation set. Estimating generalization on different dataset is therefore a very difficult task.

4. Conclusion

With a very specific clinical task of lymphocytosis classification, this challenge was not only a concrete example but also a hands-on opportunity to understand better some of the challenges related to medical imaging: limited datasets, partial labeling, and multi-modal data. More than using the State-Of-The-Art architectures, what allowed us to improve our score is to precisely understand the data we are dealing with, to avoid overfitting or accurately predict the test balanced accuracy. And our score can still be improved by more hyperparameter tuning or other MoE strategies for example.

In any case, this is only the beginning of the journey: deploying such a model in a clinical setting would require much more work, to assess the uncertainty of the prediction on different datasets, help physicians understand which image features are useful for the predictions (via Grad-Cam for instance)... Hopefully this model can ultimately save lives, and no one could dream of a better social impact for research in AI.

5. Appendix

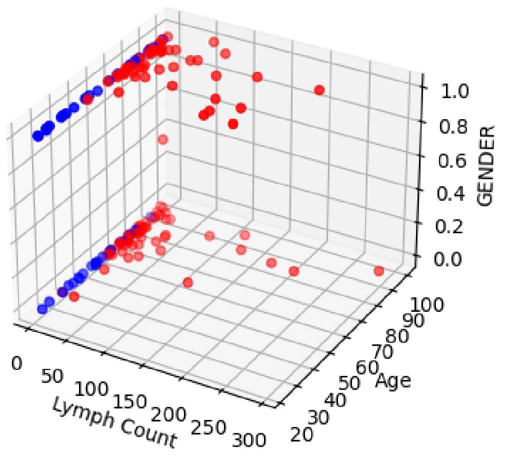

Figure 1: Clinical features of the training dataset. The lymphocyte count is represented on the x axis, the age on the y axis and the gender on the z axis. The patients with cancer are colored in red and the other ones in blue.

#	Attributes	Images	DA	Fusion	BS	Acc. Mean	Acc. Std	Lb.
1	MLP	None	N/A	N/A	1	0.81	0.06	0.79
2	MLP	ResNet18	No	Mean+MLP	5	0.87	0.06	0.83
3	MLP	ResNet18	Yes	Mean+MLP	5	0.91	0.05	0.83
4	MLP	ViT_b_16	Yes	Mean+MLP	3	0.89	0.02	0.79

Table 1: Our results. DA stands for Data Augmentation, BS for batch size, Lb. for public leaderboard. Accuracy relates to the balanced accuracy of the validation sets of each fold, for which we compute mean and standard deviation. The selected models for the submission are in bold.

References

Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

Jing Feng and Zhi-Hua Zhou. Deep miml network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

Hydra. Hydra - A framework for elegantly configuring complex applications. https://github.com/facebookresearch/hydra, 2023.

Maximilian Ilse, Jakub M Tomczak, and Max Welling. Attention-based deep multiple instance learning. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2127–2136, 2018.

PyTorch Lightning. Pytorch lightning. https://github.com/PyTorchLightning/pytorch-lightning, 2020.

MLflow. MLflow: An Open Source Platform for the Machine Learning Lifecycle. https://github.com/mlflow/mlflow, 2023.

Ray. Ray: A fast and simple framework for building and running distributed applications. https://github.com/ray-project/ray, 2023.

Zhi-Hua Zhou and Jiayuan Xu. On the relation between multi-instance learning and semi-supervised learning. In *Proceedings of the 24th International Conference on Machine Learning*, pages 1167–1174. ACM, 2007.