Challenge: Lymphocytosis classification

Kelthoum KERBOUA[*1] KELTHOUM.KERBOUA@TELECOM-PARIS.FR
[1] *Telecom Paris - ENS Paris Saclay*
Fatima BALDE [*2] FATIMA.BALDE@TELECOM-PARIS.FR
[2] *Telecom Paris - ENS Paris Saclay*

Abstract

This report explains in detail our work on the "Lymphocytosis classification" challenge as part of the "Deep Learning for medical imaging" course in the MVA master's program at ENS Paris Saclay.

Keywords: Lymphocytosis, Classification, Medical Imaging, Deep Learningn, Multi-Instance Learning, Multi-Modal Data

1. Introduction

The main objective of this challenge is the diagnosis of lymphocytosis. In clinical practice, this diagnosis relies on visual microscopic examination of blood cells, as well as consideration of clinical attributes such as age and blood lymphocyte count. Among the characteristics observed are the texture and size of the lymphocytes. There's every reason to believe that the use of deep learning and computer vision techniques will improve diagnostic reproducibility.
For this challenge, we have at our disposal data from 142 subjects with 44 reactive cases and 98 malignant cases. The model is evaluated on 42 subjects on Kaggle. For each subject in the training set, we have an indeterminate number of blood smears (which complied with certain inclusion criteria: lymphocyte count greater than $4 * 10^9/L$, and absence of opposition to research) and the patient's clinical attributes (date of birth, lymphocyte count, diagnostic label).

2. Architectures and methodological components

The first model presented was used for the final prediction with a score of 0.83 on Kaggle. The second model also performed very well, scoring 0.75 on Kaggle.

2.1. First Model

2.1.1. Architecture of the model

Our model is strongly inspired by the paper "Deep Multi-Instance Learning Using Multi-Modal Data for Diagnosis of Lymphocytosis" ((Sahasrabudhe et al., 2021)). The architecture of our model addresses a main problem. We don't have the same number of images for each patient, so our model must not treat images as being of a fixed number, and must

[*] Contributed equally

be invariant to the number of blood smears it is supplied with. In fact, in our code, we go one step further: the model first processes the images, without knowing their provenance (which patient they belong to), and then sorts them at the aggregation stage.

First of all, the blood smears (images) are sorted in such a way that their origins cannot be determined (the initial indexes are kept so that this information can be retrieved later). We create batches of images and run them through a pre-trained model of ResNet18 (without the last fully connected layers). We choose not to supply all the images at once: for a set of images corresponding to 16 patients, we end up with 1000-2000 images; for a set of 16 patients, it is therefore necessary to create several image batches and process them successively. At the end of this step and at the output of the pre-trained ResNet, we have a feature vector f_i of size 512, for each image. We choose to aggregate the features of these blood smears for each patient, using an averaging process, giving us an embedding vector for each patient $e_{patient} = \sum_{i=1}^{N_{patient}} f_i$. These embedding vectors pass through a fully connected layer, which returns an output of size 1.

We apply the sigmoid function to the output and threshold at 0.5 to obtain the final label.

2.1.2. Alternative model

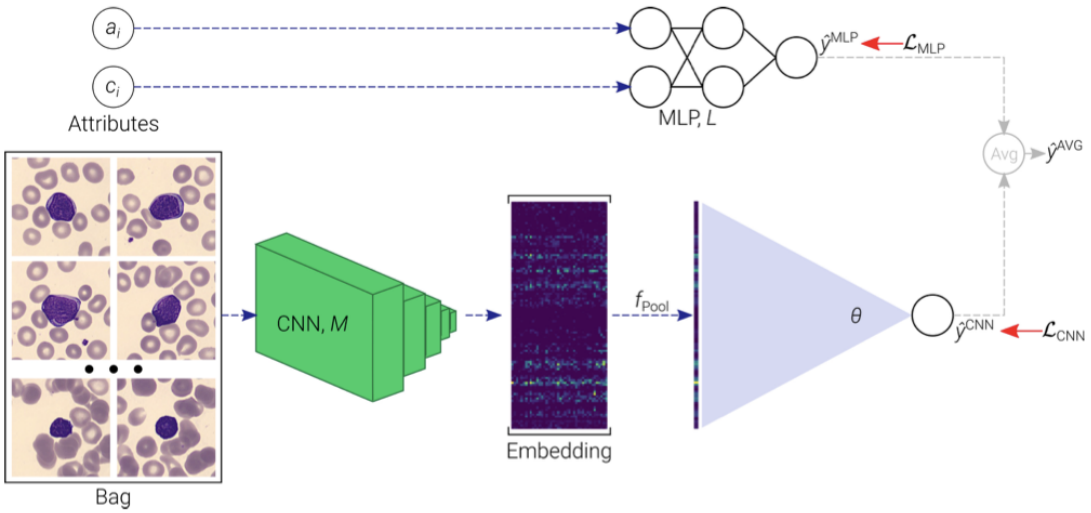

Figure 1: Architecture of the alternative model

We tried to solve an other problem in an alternative model: we aimed to extract information from both sets of medical images, blood smears, and clinical attributes. The two types of data are different, so we tried to introduce a model capable of handling multi-modal data.
For clinical attributes, these are first processed independently of blood smears. A vector is

created containing the month and year of birth, the age of the person and the number of lymphocytes in the blood. This vector is supplied to an MLP consisting of a fully connected layer of size 4x2, followed by a ReLU activation and then a fully connected layer of size 2x1. At the end of the two independent processes (image and clinical attributes processing), we have two outputs y_{MLP} and y_{CNN}. Our final loss function is calculated as the sum of a loss function on output y_{MLP} and a loss function on output y_{CNN}. This forces both outputs to provide the right information. The two loss terms imply two independent parts of the architecture. However, the final output, used for prediction, is the average of the two outputs y_{MLP} and y_{CNN}.
We realize that the model is more difficult to train and doesn't give better performance for all that. This can be explained by the fact that it's difficult to predict the label using clinical attributes alone. However, those have as much weight as blood smears in this architecture.

2.2. Second Model

The second model is also inspired by the paper (Sahasrabudhe et al., 2021). Like the first, it includes a resnet model and an MLP model for the classification of clinical data (age and lymphcount). The resnet model does not perform average pooling and is not pre-trained, but performs averaging of the embeddings of the images of each patient and makes the prediction using a final linear layer. The MLP model consists of 3 linear layers (of respective sizes 2*150,150*200,200*1), the first two being followed by the ReLU activation function. Unlike the first model presented, the data is organised in batches of size 1 and each batch consists of all the concatenated images of a single patient. These two parts are also trained separately with a lr=1e-5

2.3. Influence of our choices

2.3.1. First Model

For the first architecture, we choose to finetune a pretrained ResNet18 model. Since we have little data in our training set, it may be more efficient and easier to start with a pre-trained model on the ImageNet dataset (accuracy of 89.078 % on the dataset which contains millions of images classified in over 20,000 distinct categories). The entire model is finetuned, not just the last layer, as the ImageNet pre-training dataset is not composed of medical images, so it is necessary to modify the features it extracts to achieve better performance.

We also tested other model sizes (ResNet34, ResNet50), which did not improve the classifier's performance. Indeed, we have little data and therefore not enough to train a very large model, which risks suffering from overfitting.
When features are extracted by ResNet18, the blood smears are mixed together, and provenance information is lost. Each image is processed independently. In this way, no bias is introduced, and each image alone is forced to provide good features for prediction.
We also modified the reference model's architecture so that it can take into account clinical attributes, in an attempt to strengthen the final prediction. The model is trained with the sum of the losses on each of the outputs. As each predictor can be trained separately, there is no common training. The result isn't more stable performance, and training takes

longer. We therefore decided to not use that modification of the architecture for the final predictions.

2.3.2. Second Model

Since the images from Imagenet are not medical images , we train resnet18 from scratch for the second architecture. We obtain rather good performance by controlling the learning rate with a scheduler which lower it by a factor of 0.1 every 10 epochs to avoid overfitting. However the first model with pretained weights performs better.
We also separate the two training process unlike in the first model. This choice is motivated by the fact that the resnet and MLP process different datas therefore they may not converge the same way. The only loss used here is the binary cross entropy and the final prediction is the average of the two. We normalize the clinical attribute to avoid vaninshing/exploding gradient during the training process.

3. Model tuning and comparison

3.1. Preprocessing

Since we have little data, we use data augmentation techniques to avoid overfitting and improve model generalization and performance. This is achieved by performing random rotations as well as vertical and horizontal flips on the image. In addition, we notice that each blood smear has a lymphocyte at its center, so we crop the image to remove the edges that divert the model's attention to the lymphocyte in question. The image is cropped to output size 112x112, the standard input for the ResNet18 pre-trained model. In addition, we also perform normalization like the pre-trained inputs of the ResNet18 model.

3.2. Training

The aim is to increase the balanced accuracy of the validation and testing set. We therefore looked for the best loss function to use in order to improve our model's performance. First, we used binary cross entropy with logits. We noticed that we had an imbalance in our dataset: we had more malignant cases than reactive ones. As a result, the accuracy deviates significantly from the balanced accuracy. When a dataset is unbalanced and one class is much more represented than the others, machine learning models may tend to predict this majority class predominantly, even though this may lead to significant classification errors when other classes are present. So we wondered whether there might be a loss function that would take this imbalance into account. We then tested Dice Loss. In the Dice Loss formula, loss calculates the similarity between predictions and ground truth, taking class size into account. This means that the loss function is just as influenced by good or bad prediction for samples belonging to a minority class, as it is for samples belonging to a majority class.
In the case of binary classification, Dice Loss can be adapted to correspond to a measure of similarity between prediction and ground truth. However, Dice Loss is mainly used in the context of image segmentation, where predictions and ground truths are binary masks representing regions of interest.
However, the use of Dice Loss did not improve the performance of our model. We therefore

decided to stick with the BCE with logits loss.

Formulas for the different loss functions mentioned:

$$\text{BCE with logits} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \cdot \log(\sigma(\hat{y}_i)) + (1-y_i)\cdot \log(1-\sigma(\hat{y}_i))\right]$$

$$\text{Dice Loss} = 1 - \frac{2 \times \text{intersection}(y_{\text{true}}, y_{\text{pred}})}{\text{total pixels}(y_{\text{true}}) + \text{total pixels}(y_{\text{pred}})}$$

We also had to test several learning rates. A learning rate a little too high would prevent convergence, and a little too low would make training too slow. We set this one at 10^{-4}. We also tried adding an exponential scheduler to decrease the learning rate during training, but the model overfits, so even if the training set loss decreases, the performance on the validation set is not improved.

The training process of the second architecture differs a little from the process described here. We use the binary cross entropy for the resnet and the MLP which are trained separately.

3.3. Ablation Study and other models

We studied the performance of our first model by removing or modifying certain parts of the architecture.

- Removing the ResNet18, and testing a scratch-trained CNN model, resulted in a drop in performance. The ResNet18 has 18 layers. It's harder to train a model of this size without overfitting due to the small amount of data. Furthermore, smaller models perform poorly even on the training set.
- By removing the part of the architecture dedicated to clinical attributes, final convergence is more stable and performance is slightly better. Convergence is also faster.
- We also tried adding more layers at the output of ResNet 18 or in the MLP that processes clinical attributes. No performance improvement was obtained, as we are already in a situation of overfitting, and increasing the model size does not improve the situation.

3.4. Validation Process and Scores on Kaggle

As regards the model validation process, we divided the dataset 50/50, into a training and validation set. We are conscious of losing a lot of data for training by dividing the dataset in this way, but otherwise we can't assess which is the best model. We were also careful to keep the same proportion of patients with the malignant label as with the reactive label; otherwise, we risk ending up with either a validation set that gives us a performance that is neither representative nor reliable, or a training set that is not representative of the population and will result in poor model training.
We then choose to evaluate our model on the validation set at each epoch. The loss and

balanced accuracy are calculated and displayed. We store the best validation balance accuracy. If, at the end of the epoch, the newly calculated validation balanced accuracy is better than the one stored in memory, we save the model in an external ".pth" file. At the end of training, we reload the model judged to be the best according to its performance on the validation set and calculate the predictions on the testing set for submission to Kaggle. We obtain a test score (balanced accuracy) of 0.83116 for the first architecture and 0.75584 on the second one.

References

Mihir Sahasrabudhe, Pierre Sujobert, Evangelia I. Zacharaki, Eugénie Maurin, Béatrice Grange, Laurent Jallades, Nikos Paragios, and Maria Vakalopoulou. Deep multi-instance learning using multi-modal data for diagnosis of lymphocytosis. *IEEE Journal of Biomedical and Health Informatics*, 25(6):2125–2136, 2021. doi: 10.1109/JBHI.2020.3038889.