Challenge Report: Lymphocytosis Classification

Quentin Bourbon QUENTIN.BOURBON@TELECOM-PARIS.FR
Vincent Bardusco VINCENT.BARDUSCO@TELECOM-PARIS.FR
Télécom Paris - MVA
Kaggle Team: Les fous du wan

1. Introduction

This report outlines our strategy for the Kaggle challenge which treats the problem of Lymphocytosis classification. The problem was the following: given some clinical attributes of a patient (age, sex, and the number of lymphocytes per L) and RGB images of lymphocytes of the patient, predict if the patient has a reactive or a malignant case. Apparently, this problem is a classical classification problem, but it combines several particularities that makes it untypical. First of all, as often in Medical Imaging, the dataset is quite small: we have access to 163 samples, and the test set is composed of 42 samples. Secondly, the dataset is highly unbalanced: 50 reactive and 113 malignant cases. Thirdly, the number of images is not the same for all the patient, which makes harder the treatment of such data. Finally, we dispose of 2 types of data that we have to take into account. Hence we have dealt with aggregation of features as well as multi-modal learning during this challenge.

2. Architecture and Methodological Components

2.1. General Framework

The model has to take as input two different types of data: images and scalar metadata (age and number of lymphocytes). Thus, we have to use multi-modal learning to solve this problem. Another complexity was that the number of images depends on the patient. Therefore, we also have to develop a method to aggregate the images so that the model can be used regardless of the patient. We were very inspired by the model presented by Sahasrabudhe et al. (2021). It addresses the same problem and proposes an end-to-end trainable multi-instance neural network that implements a mixture of experts framework. The idea there is to separate images and metadata, and to perform a classification using only images on the one hand, and only scalar values from metadata on the other. A gate then combines these two classifiers using adaptive weights. More precisely, their model consists of 3 main blocks:

- An image features extractor that encodes the bag of images in a latent space. Then the features of all the images are aggregated by an aggregation function. The aggregated features are then passed to a classifier to get a prediction $\hat{y}_{images}$.
- A simple MLP followed by a linear classifier that takes as input the metadata. It predicts $\hat{y}_{metadata}$.

- A gate G which is a linear model that takes as input the metadata and a single scalar representation of the images aggregated features and outputs the weight π of how confident we should be about the prediction of the images so that the final prediction is $\hat{y} = \pi\hat{y}_{images} + (1-\pi)\hat{y}_{metadata}$

In our setting, we were able to obtain other homemade scalar features for each image (see 2.2). Therefore, we added a fourth block consisting of a simple MLP that embeds these homemade features in a latent space where they are aggregated by an aggregation function. They are then concatenated with the metadata to give $\hat{y}_{scalar}$. In addition, a single scalar representation of the aggregated homemade features is taken as input by the gate G, so that it also plays a role in the mixture of experts.

2.2. Handmade Features

To add relevant information as input to our model, we decide to manually compute features from the images using machine learning. To do this, we follow the pipeline described by Tavakoli et al. (2021). It starts with segmentation of the nucleus, followed by segmentation of part of the cytoplasm. Shape and colour features of the region of interest are then extracted. The segmentation of the nucleus is done using the properties of different colour spaces with basic operations between them, and the final threshold is given by Otsu's algorithm. As a post-processing, a function that keeps the largest component and fills the holes is applied. Since segmentation of the cytoplasm is difficult and often inaccurate, only the part of the cytoplasm that lies within the convex envelope of the nucleus is considered, which is referred to as the Representative Of The Cytoplasm (ROC). The difference between a normal cell and a problem cell is that the normal cell is generally more circular in shape and therefore has a small ROC, as shown in Figure 1.

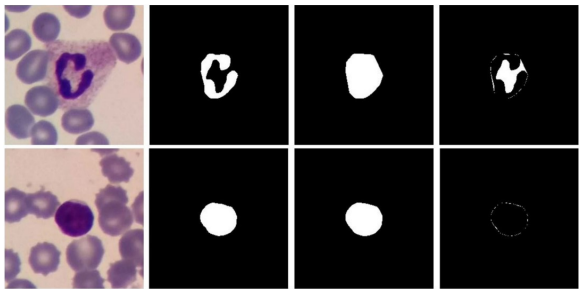

Figure 1: The first row shows an unusual cell and the second shows a healthy cell. From left to right: RGB image, nucleus segmentation, convex hull of the nucleus, ROC

Then we extract 3 shape features ($\mathcal{A}$ is the area while $\mathcal{P}$ is the perimeter):

$$\text{Solidity} = \frac{\mathcal{A}(\text{Nucleus})}{\mathcal{A}(\text{Convex hull})} \quad \text{Convexity} = \frac{\mathcal{P}(\text{Convex hull})}{\mathcal{P}(\text{Nucleus})} \quad \text{Circularity} = \frac{\mathcal{P}(\text{Nucleus})^2}{4 \times \pi \times \mathcal{A}(\text{Area of Nucleus})}$$

and 4 color features from each channel of RGB, HSV, LAB, and YCrCb color spaces:

$$\textbf{1.}\frac{\mu(\text{Nucleus})}{\mu(\text{Convex hull})} \quad \textbf{2.}\frac{\sigma(\text{Nucleus})}{\sigma(\text{Convex hull})} \quad \textbf{3.}\frac{\mu(\text{ROC})}{\mu(\text{Convex hull})} \quad \textbf{4.}\frac{\sigma(\text{ROC})}{\sigma(\text{Convex hull})}$$

2.3. Scalar Classifier

Once the hand-crafted features are computed for each image, we have scalar values for each patient, consisting of initial metadata and a certain number of features depending on the number of images associated with the patient. As explained in the general framework, we want to predict a first classification using only these features and not the images (i.e. not using the images directly in the deep learning model, as the computed features obviously depend on them).

Since the number of hand-crafted features is not the same for each patient, it is necessary to aggregate them so that the final number of features is the same and can be fed to the classifier. To do this, we decided to take the mean across all images for each feature, giving us a feature vector of size 8, which we concatenated with the number of lymphocytes and the age of the patient. We discarded the gender as it did not seem to be relevant for the predictions.

After this step, the scalar features are passed through a simple classifier composed of two linear layers with a ReLU activation and a sigmoid output, which gives us a first prediction $\hat{y}_{scalar}$ for the patient.

2.4. Image Classifier

The second classifier used for our mixture of experts is based on images only. We use a computer vision model to extract features for each image, and as before, since we have multiple images for each patient, we use mean aggregation to get a single image feature vector for each patient. We have tested different feature extraction models, which will be described later.

The features are then fed into a classifier head consisting of a linear layer followed by a sigmoid to give the second prediction $\hat{y}_{images}$.

2.5. Gate

With the two predictions from the two 'experts' given by the classifier, the final step is to combine them using a gate. As explained earlier, a gate simply combines the predictions by giving them an adaptive weight. The weights are computed by a linear layer taking as input control features, which in our case are the concatenation of the two metadata scalars (age and lymphocyte count) and two new scalar values computed by two linear layers taking as input the image features and the handmade features respectively. After a sigmoid, the final output is π, which gives the final prediction $\hat{y} = \pi\hat{y}_{images} + (1-\pi)\hat{y}_{scalar}$. This framework allows the weight of each classifier to be modified depending on the images and features, so that the model automatically selects the 'best expert' depending on the patient.

3. Model Tuning and Comparison

3.1. Validation

In this section we explain our validation strategy to select the best model. We immediately noticed that the distribution of the dataset is very unbalanced. Moreover, the dataset is quite small and then shows a high variability in the results when we use a classical function

to split the dataset into a training set and a validation set. Therefore, it seems necessary to construct a function that splits the dataset while preserving the distribution of positive and negative samples. This means that we assume that the distribution in the dataset is close to the global distribution. However, without this, we could have a validation set with only samples of the most frequent class, which would make it not relevant at all. Furthermore, to get a more robust validation score to choose the components of our model, we use k-fold cross-validation as described by Bradshaw et al. (2023) with $k = 5$ and such that each fold follows the initial class distribution. For each fold, we train with the same procedure, which includes an early stopping criterion and a decrease of the learning rate, as described in 3.4. After obtaining a relevant validation score that allows us to select the best parameters for our model with cross-validation, we divide the dataset in two (always keeping the same distribution) and train it. A larger validation set (than the one used in the cross-validation procedure) helps to avoid overfitting, since all the stopping criteria and the learning rate decrease are based on this set. The results presented in this report were obtained using this last validation set.

Seeing that the validation results sometimes differed from the public test results, one idea we had was to take a validation set with a balanced distribution. However, the results we obtained were not satisfactory either, and we also had the intuition that this would only lead us to overfit on the public test set.

3.2. MultiTask Strategy

To help the model generalise and to ensure that the extracted features are relevant, we implemented a multitask architecture. As we found that the number of lymphocytes was highly relevant for prediction (based on tests and opinions of medical experts), we added a regression head that takes the image features as input to predict this value. While this prediction is discarded during inference, it is used during training by adding a regression term in the loss computing the MSE between the predicted and real number of lymphocytes. When compared with the simple OneTask model without regression head, we obtained a 10% balance accuracy increase on the validation set.

3.3. Image Features Extractor

For the image feature extractor model, we tested different architectures: ResNet18 ((He et al., 2015)), ConvNext ((Liu et al., 2022)) and ViT-small ((Dosovitskiy et al., 2021)). In all experiments, we initialised the models with pre-trained weights from the ImageNet classification and removed the classification head. ConvNext and Vit-small were always trained with the MultiTask framework.

ResNet18 was fully tuned. ConvNext was either fully tuned or frozen. In the frozen case, to ensure that the multitask model was still useful, we added an intermediate linear layer between the ConvNext encoding and the classification and regression heads (otherwise there was no common gradient between the two heads, rendering the regression part useless). Vit-small was fine-tuned using the LoRA framework. The results obtained with the different models on our validation set are gathered in Table 3.3.

ResNet OneTask	ResNet MultiTask	ConvNext tuned	ConvNext frozen	Vit LoRA
0.84	0.94	0.95	**1.0**	0.89

Table 1: Balanced accuracy results on validation set

3.4. Additional Implementation Details

In this section, we describe the hyperparameters we tested and some implementation tricks we used. First, we applied simple transformations and data enhancement to the images, such as normalisation (to match the pre-trained feature extractors) and random vertical and horizontal flips to make the model more robust. The global loss consists of the classification loss and the regression loss. Since the data are unbalanced, for the classification loss we choose the Binary Cross Entropy (BCE) with weights that give more importance to the less present class, which is consistent with the balanced accuracy metric. Similarly, to take into account the difference in range between BCE and MSE, we use the Normalised Mean Squared Error (NMSE) for the regression loss so that we can properly define its weight in the global loss. Another problem we faced was the impossibility of creating a PyTorch dataloader with a batch size greater than 1, since the number of images depends on the patient. However, one method to simulate a batch of samples is to accumulate the loss during bs samples and then (after dividing by bs to maintain coherence with the learning rate) perform the backward operation on the loss and the optimiser step. This method allows the gradient to be composed of samples with greater diversity at each update step, making the model more robust. We use AdamW with default hyperparameters with a learning rate starting from 1e-3 and decreasing by a factor of 10 if the validation loss does not improve during 5 epochs. Similarly, we stop training if the validation loss does not improve during 10 epochs. Finally, we take the model with the best validation loss.

4. Conclusion

In this report, we have detailed our strategy for the Kaggle challenge, in which we managed to achieve a balanced accuracy of 86.49% on the public test set with the frozen ConvNext feature extractor using the multitask architecture. We have explored multimodal learning as well as aggregation methods, keeping in mind the importance of validation by ensuring the robustness of the model.

If we had more time and resources, we would have liked to test different ways of aggregating which could have improved the results, such as aggregating after prediction with top-k aggregation or other methods.

References

Tyler J Bradshaw et al. A guide to cross-validation for artificial intelligence in medical imaging. *Radiology. Artificial intelligence*, 5(4):e220232, May 2023. doi: 10.1148/ryai.220232.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

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

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s, 2022.

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.

Sajad Tavakoli et al. New segmentation and feature extraction algorithm for classification of white blood cells in peripheral smear images. *Scientific reports*, 11(1):19428, Sep 2021. doi: 10.1038/s41598-021-98599-0.