Classification of Lymphocytosis from Blood Cells

Deep Learning for Medical Imaging

Clément Grisi
École des Ponts ParisTech
grisi.clement@gmail.com

Marius Schmidt-Mengin
École des Ponts ParisTech
marius.schmidt.mengin@gmail.com

Abstract

In this report, we emphasize our work for the data challenge organized as part of the Deep Learning for Medical Imaging class. The goal of the challenge is to classify lymphocytosis from blood cells. Through iterations in model definition and data representation, we managed to get 0.96623 *classification accuracy on the academic leaderboard (rank 1).*

1. Introduction

Lymphocytosis is an increase in the number or proportion of lymphocytes in the blood. It's a common finding, which can be either a reaction to infection or acute stress (reactive), or the manifestation of a lymphoproliferative disorder – a type of cancer of the lymphocytes (tumoral). In clinical practice, the diagnosis as either reactive or tumoral is performed by trained pathologists who visually inspect blood cells under a microscope. The final decision also takes into consideration clinical attributes such as age, gender, and lymphocyte count. In spite of being relatively fast and affordable, lymphocytosis diagnisis lacks reproducibility between experts. Additional clinical tests are often required to confirm the malignant nature of the lymphocytes. However, this analysis is relatively expensive and time-consuming, and therefore is not performed for every patient in practice. In this context, automatic classification has the potential to provide accurate and reproducible diagnosis, saving precious time and ressources by quickly identifying which patient should be referred for flow cytometry analysis.

2. Problem Definiton

2.1. Dataset

Blood smears and patient attributes were collected from 204 patients from the routine hematology laboratory of the Lyon SudUniversity Hospital. All included patients have a lymphocyte count above $4 \times 10^9/L$. The blood smears were automatically produced by a Sysmex automat tool, and the nucleated cells were automatically photographed with a DM-96 device. The training set consists of 142 patients (44 reactive and 98 malignant cases), and the testing set of 42 patients. For each patient, we have access to dozens of images of lymphocytes, as well as the following clinical attributes: gender, age, and lymphocyte count.

2.2. Evalutation Metric

This challenge is evaluated on balanced accuracy (BA), which normalizes true positive (TP) and true negative (TN) predictions by the number of positive and negative samples, respectively. In particular, if one denotes false positives as FP and false negatives as FN, we have:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

$$\text{BA} = \frac{\text{Sensitivity} + \text{Specificity}}{2}$$

2.3. Weak Supervision through Multiple Instance Learning

The idea of weakly supervised learning is to exploit coarse-grained annotations to automatically infer finer-grained information. Coarse-grained information is often readily available in the form of patient level labels, but finer grained annotations are more difficult to obtain. Without precise local annotations, classification models cannot be trained in a fully supervised manner. Therefore, various weakly supervised techniques have recently been developed to overcome this issue. One of these techniques relies on multiple instance learning (MIL), an existing framework largely used in classic computer vision that has recently showed state-of-the-art results in several medical imaging tasks [1].

Babenko [2] gives a good example to understand multiple instance learning. Imagine several people, each of them

having a key chain that contains a few keys. Some of these people are able to enter a certain room, and some aren't. The task is to predict whether or not a given key chain can open the door of that room. To solve this problem, one needs to find the exact key that is common to all the *positive* key chains. If one can correctly identify this key, one can also correctly classify an entire key chain - *positive* if it contains the required key, or *negative* if it doesn't.

Hence, the multiple instance learning framework allows the training of a classifier from weakly labeled data: instead of providing input-label pairs, labels ℓ_b are assigned to *sets* or *bags* of instances. In this setting, the true instance labels ℓ_i can be thought of as latent variables, as they are not known during training. In our case, we can assume that only a subset of the images available for a patient with tumoral lymphocytosis do carry the information necessary to correctly classify that patient. Identifying these special instances perfectly fits in the multiple instance learning framework.

3. Related Work

add 3-4 papers: Maria's, Paige's

4. Methodology

4.1. Standard MIL Assumption

Under the standard MIL assumption, *positive* patients must contain at least one instance classified as *positive*, while *negative* patients, instead, must have all their instances classified as *negative*.

The full pipeline is described on Figure **??**. Each forward + backward pass can be broken down into two consecutive stages:

- the *inference* phase: during *inference*, the models' weights are frozen. Each instance is run through the network and assigned a number between 0 and 1: the more likely an instance to indicate tumoral lymphocytosis, the closer the number assigned to 1. Once all the instances of a patient have been processed, they can be ranked according to their probability of being *positive*. For each patient, the top-1 instance is selected for the second stage: the *learning* phase.
- the *learning* phase: we now allow the models' weights to be updated. Given the standard MIL assumption detailed above, one can expect the top-1 instance of a *positive* patient to be *positive*, and the top-1 instance of a *negative* patient to be *negative*. Hence, as we run the top-1 instance through the network, we assign them the label of the patient they come from and update the weights accordingly: the network adjusts its weights in order to assign high probabilities to the top-1 instances coming from positive patients, while keeping low probabilities for the top-1 instances in negative patients.

At this stage, you might be wondering "*how this model converges* ?". It is true that it is not obvious as the model is initialized with ImageNet pre-trained weights, hence there's not reason to believe the model will assign high probabilities to the true positive instances, and low probabilities to true negative instances. And actually, the model does start by assigning random probabilities to each instances!

To understand how the model gradually learns to discriminate positive instances from negative ones, we need to focus on negative patients, who play a key role in the early stages of training. Under the standard MIL assumption , we know these patients must have all their instances classified as negative. At the end of the first *inference* phase, the model has assigned these instances random probabilities, which can be either high or low. Hence, the top-1 instance for negative patient is random. I claim that this doesn't matter. Let me explain you why. What's important is what happens during the first *learning* phase: for each negative patient, we have a (random) instance, which is negative. The model must adjust its weights to assign lower probabilities to each negative instances, such that at the following *inference* step, all instances that look similar to negative instances will be assigned lower probabilities. As a result, all negative instances start to be assigned low probabilities, and positive instances will srand out, with higher probabilities. At this point, the top-1 instance of a positive patient is more likely to be a true positive. And the model will adjust its weights to assign higher probabilities to this type of instance.

This simplified reasoning allows to understand how the model gently learns to discriminate suspicious instances from non-suspicious ones, and automatically infers fine-grained information from coarse-grained labels.

quick sentence to explain why using solely to top-1 instance might be, in some cases, problematic. this motivates the introduction of a hyperparameter k whcih controls how many positive instances there might be in a positive bag (different degrees of positivity: 4 moderate positive instances, top proba is 0.6)

4.2. Extended MIL Assumption

As explained above, the standard MIL assumption implies that a patient is classified *positive* as soon as the model identifies a suspicious instance in it. During training, it assumes there must be at least 1 *positive* instance for

a *positive* patient. During testing, if the top-1 instance probability is above the classification threshold τ_{class}, it classifies the entire patient as positive. During testing, instead of letting the top-1 instance decide for the whole patient, it makes more sense to derive the patient's prediction by looking at its top-k instances' probabilities.

The first experiments tended to confirm that MIL performances were sensitive to the value of the top-k parameter. Using a fixed top-k, however, does not generalize well, since different patients have different numbers of instances, hence different numbers of suspicious instances. In order to account for this, the interested reader can introduce a percentage-based top-k values by looking at $max(m, \text{ceil}(k\% \text{ of instances}))$ where m is a new hyper-parameter that controls the minimum number of instances to aggregate for the final prediction.

4.3. Taking More Tiles into Consideration

top-k & bottom-k

4.4. Taking All Tiles into Consideration

5. Architecture

We start from a simple baseline and we evaluate different modifications. For each configuration, we run 10 trainings. Each training has its own seed for initializing the model and splitting the dataset (50/50 split). The 10 seeds are kept the same for each configuration. Each model is trained for 40 epochs with a batch size of 16, resulting in 5 batches per epoch. Evaluation if performed at the end of every epoch. We always optimize with Adam and a learning rate of 10^{-4}. When we report a metric for any configuration, we average the 10 best values of each training, and again average this over all 10 trainings (the average is thus taken over 100 values). We always report 95% confidence intervals.

5.1. Baseline

We adopt a simple and explainable architecture composed of a ResNet [3] backbone followed by an aggregation module. For a given batch of patients, we take all their images and stack them into a batch. This batch is passed independently the backbone, without global pooling. A 1×1 convolution with one output channel is then applied to the resulting feature maps to obtain pixelwise prediction logits. These logits are rearranged into bags of scores, each bag corresponding to all prediction for one patient. Finally, the aggregation module transforms each bag of scores into a final patient prediction.

It is common that the features are aggregated before being linearly mapped to the final prediction score. In contrast, we apply the aggregation after reducing each feature to a single prediction logit. We did not find this to impact the results (and in the case where the aggregation function is linear, both methods are equivalent). Doing so makes the model more interpretable as each pixel (in the feature space) of each image can be given a score.

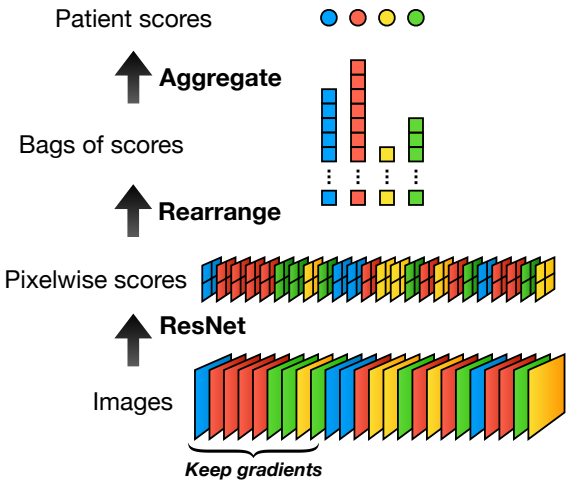

Figure 1: Baseline architecture.

As the number of images passed through the backbone is often large, we keep the gradients for only a random subset of these images. Pseudo-code for this is provided by Algorithm 1.

```
B, 3, H, W = image_batch.shape
max_forward_size = 512
max_backward_size = 512
features = []
i = 0
# forward images without keeping gradients
with torch.no_grad():
  while i < B - max_backward_size:
    j = min(i+max_forward_size,
            num_images-max_backward_size)
    features.append(backbone(images[i:j]))
    i = j
# forward image with gradients
features.append(backbone(images[i:]))
# Note: the images are supposed to be shuffled
# so that the subset for which the gradients
# are kept is random.
```

Pseudo-code 1: PyTorch-like pseudo-code for propagating the gradients through only a subset of the images, in order to not run out of GPU memory.

5.2. Augmentations

As the images are centered on a lymphocytes, we always apply a center crop of size 112 to all images (training, validation and testing). Initial experiments showed that this significantly improves the accuracy and reduces the computational resources. We use vertical and horizontal random flipping. We tried to use more agressive augmentations such

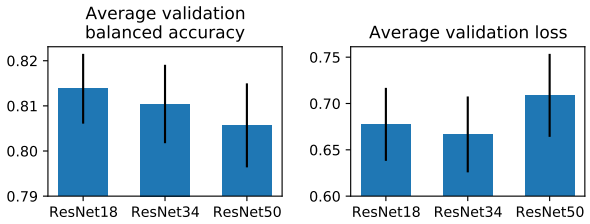

Figure 2: Impact of ResNet depth on validation loss and balanced accuracy. The black bars indicate 95% confidence intervals. Shallower seems to be better but the results are not very significative.

as affine transforms and color jittering but this heavily impaired the accuracy.

5.3. Backbone Choice

We compared three backbones: ResNet18, ResNet34 and ResNet50 [3]. We also tried a more recent backbone, EfficientNet [4], but did not obtain good results. The results are summarized in Figure 5. ResNet18 achieves the best validation balanced accuracy. We use it for all our subsequent experiments.

5.4. Aggregation Function

Let us denote all the predictions (logits) for patient j by $\ell_i^j,\ i = 1 \dots n_j$. To aggregate these predictions, we seek to define some weights $\alpha_i^j \geq 0$ such that we can compute the final prediction logit for patient j by

$$\ell^j = \sum_{i=1}^{n_j} \alpha_i^j \ell_i^j$$

(we assume that $\sum_i \alpha_i^j = 1$). The confidence score is then given by applying the sigmoid function: $y_j = \sigma(l^j)$. Two very common aggregation functions are the arithmetic mean and the maximum, where we would respectively set $\alpha_i^j = 1/n_j$ and $\alpha_i^j = \Big\{1 \text{ if } \ell_i^j = \max_k \ell_k^j \text{ and } 0 \text{ otherwise}\Big\}$.

We believe that these two aggregation function suffer from different problems. Let us express the gradient of the cross-entropy loss with respect to the logits ℓ_i^j:

$$\frac{\partial L_{\text{CE}}}{\partial \ell_i^j} = \alpha_i^j \left(y^j - \bar{y}^j\right)$$

The maximum function results in sparse gradients (only one α_i^j is nonzero), as only the image that effectively generated the maximum produces a nonzero gradient. On the other hand, the mean function generates equal gradients for all images (all α_i^j are equal to $1/n_j$), meaning that for a positive patients, the "pull" generated by the gradient on negative images (if there are any) is equal to the pull applied

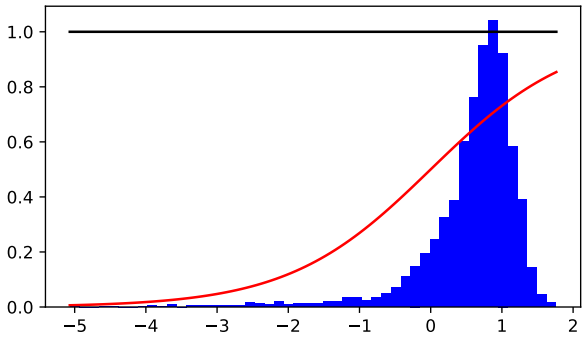

Figure 3: The distribution of the batch normalized logits over a bag is shown in blue. The red sigmoid shows the weight that is associated to each logit. The smaller the logit, the smaller the weight.

to positive images. We developped an aggregation function that intuitively overcomes this problem. The idea behind it is similar to focal loss [5], which scales the weight so as to put more gradient on predictions that are deemed important in the context of a heavy class imbalance. Our aggregation function defines the weights α_i^j by batch normalization [6] followed by the sigmoid function:

$$\alpha_i^j = \sigma\left(\beta + \gamma \frac{\widetilde{\ell_i^j} - \mu}{\sqrt{V + \epsilon}}\right)$$

where $\widetilde{\ }$ denotes the stop gradient operation (as in focal loss, no gradients are propagated through the weights), γ and β are trainable parameters, ϵ is a small constant and μ and V are respectively the mean and variance of the logits ℓ_i^j over a batch of patients:

$$\mu = \frac{1}{n_1 + \dots + n_B} \sum_{j=1}^{B} \sum_{i=1}^{n_j} \ell_i^j$$

$$V = \frac{1}{n_1 + \dots + n_B} \sum_{j=1}^{B} \sum_{i=1}^{n_j} \left(\ell_i^j - \mu\right)^2$$

where B is the batch size. Intuitively, α_i^j is small when ℓ_i^j is small compared to the rest of the batch, i.e, when it is likely to be a negative sample.

```
logits = conv(features) # 1 channel
weights = bn(logits.detach())
weighted_logits = weights * logits
y = sigmoid(weighted_logits.sum()/weights.sum())
```

Pseudo-code 2: PyTorch-like pseudo-code for our custom aggregation function

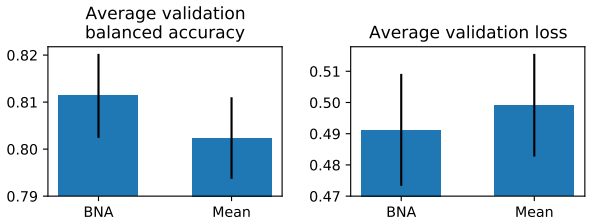

Figure 4: Impact of our BN aggregation function on validation loss and balanced accuracy. The black bars indicate 95% confidence intervals.

5.5. Number of images

In the previous sections, when doing one training step, we used all the images of each patient and aggregated their logits. In this experiment, we use only a small random subset of image per patient during each training step. We posit that selecting only a small subset of images provides some regularization to the training process. Indeed, assume that for a given positive patient, one image is easily identified by the network as positive. In this case, the network can ignore the other images. By selecting only a small random subset of images, we avoid this as the easy, positive images has a high probability of not being selected. Note that we implicitely make the assumption that in the case of a positive patient, a large number of images contains positive information.

6. Results

7. Explainability

8. Conclusion

Future work: dilated ResNet ?
Future work: aggregate embeddings using a recurrent neural network / self-attention encoder?

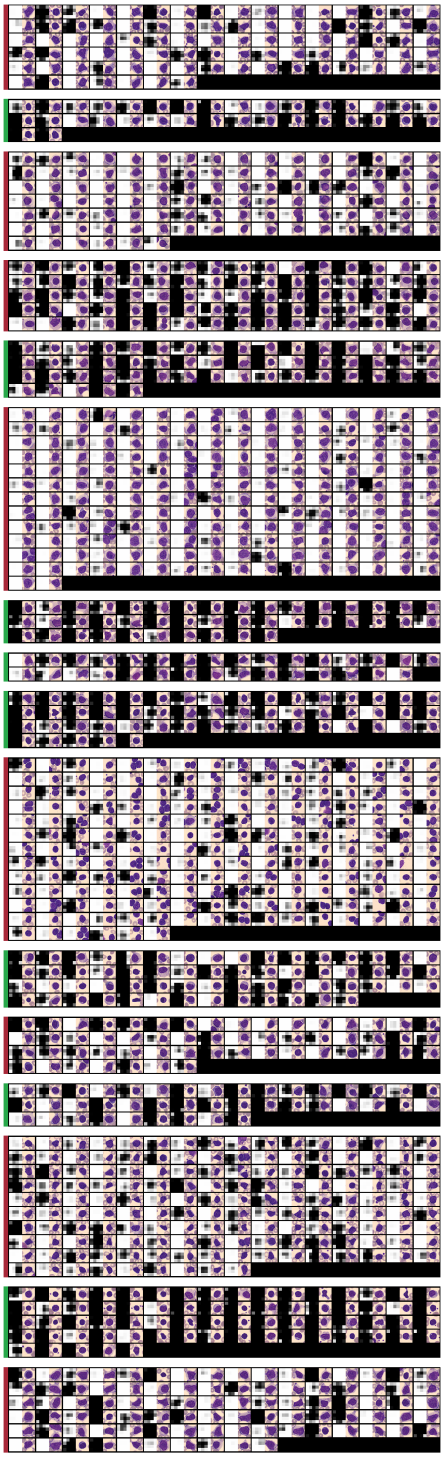

Figure 5: Predictions and images for 16 patients from the validation set. The patients where not cherry-picked. For each patient, a red left border indicates a positive ground truth label while green indicates a negative one. For each lymphocyte image, we provide on the left the 4×4 predictions of the network. White means a score of one and black a score of 0.

References

[1] Le Hou, Dimitris Samaras, Tahsin M. Kurç, Yi Gao, James E. Davis, and Joel H. Saltz. Efficient multiple instance convolutional neural networks for gigapixel resolution image classification. *CoRR*, abs/1504.07947, 2015. 1

[2] Boris Babenko. Multiple instance learning: algorithms and applications. 2008. 1

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 3, 4

[4] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 2019. 4

[5] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2999–3007. IEEE Computer Society, 2017. 4

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015. 4