Lymphoma subtype classification, a performance-efficency trade-off approach

Nicola Bee[†], Lorenzo Saccaro[‡]

***Abstract*—Lymphoma is a type of cancer that originates from the cells of the immune system. In this study, we aim to classify the three most frequent types of lymphoma using a deep-learning approach. Our strategy is to leverage the ability of convolutional neural networks as feature extractors and then apply classical machine learning algorithms to the extracted features to perform the classification. We believe that the use of deep learning techniques in this area has the potential to significantly improve the accuracy and efficiency of lymphoma classification. To achieve this goal, we have employed a variety of eminent deep learning architectures, including Inception and ResNet. In terms of evaluation, we have utilized a variety of metrics, including accuracy, precision, recall, and ROC AUC, to assess the performance of our models. Our results show that our approach is capable of achieving high accuracy in the classification task, with a high degree of efficiency and low complexity. In conclusion, our study provides evidence that a mixture of classical machine learning and deep learning techniques can be applied effectively for the classification of lymphoma and that can lead to improved performance-efficiency trade-off. This research can serve as a foundation for future studies aimed at further improving the lymphoma classification task using combinations of computational and memory-efficient deep learning and machine learning techniques.**

I. Introduction

Lymphoma is a form of cancer that affects the lymphatic system, which is an integral part of the immune system. It is estimated that is responsible for half of the malignant blood diseases worldwide [1] The classification of lymphoma is vital for selecting the appropriate treatment and predicting the prognosis of the disease. Traditionally, morphological analysis has been used, but it is subjective and prone to variability. Deep learning techniques, particularly convolutional neural networks, have shown promise in accurately classifying lymphoma based on images and have the potential to revolutionize diagnosis and treatment.

According to the World Health Organization (WHO) [2], there are over 60 types of lymphoma. This paper will focus on the three most common malignant lymphoma types: mantle cell lymphoma (MCL), follicular lymphoma (FL) and chronic lymphocytic leukemia (CLL). The main differences between these three types of cancers are their location of origin, rate of growth, and progression. A series of histological images of these three diseases have been gathered by the National Institute of Health of the U.S. (we will refer to it as the *NIH dataset* [3]).

Department of Physics and Astronomy "Galileo Galilei", University of Padova

[†]nicola.bee.1@studenti.unipd.it

[‡]lorenzo.saccaro@studenti.unipd.it

In recent years, remarkable results have been achieved using increasingly complex machine learning tools so that most recent architectures [4] show almost perfect accuracy over the dataset. These models however do not consider the efficiency of the network using high computational resources. Our aim is to study a good trade-off between accuracy and complexity to reach good efficiency. This will allow us to save resources both for computational power, use of memory, and energy consumption. Efficiency demand using light yet valid models. In particular, the networks are exploited as feature extractors and then these features are passed to a smaller model, such as a support vector machine, for classification. In order to be more efficient we reduced the input size cropping the image into patches and applied a majority voting strategy.

Our study is focused on:

- Training different models, such as Inception or ResNet, in order to compare performances and evaluate which are the most reliable ones.
- Study dataset preprocessing differences: as previous study have stated, different performances are associated with different colorspaces.
- Combine and evaluate different machine learning tools, such as SVM or XGBoost with our networks in order to study accuracy improvements.
- Consider the memory consumption, the computational resources and the power demand of each architecture to better understand the trade-off between complexity and accuracy.

In this report, we will first present some related works. Then we will present how we are going to handle the problem by showing our pipeline. Later we will present the dataset preprocessing and the technical details of our architecture. Lastly, we will examine the results obtained from our study and discuss their possible implications.

II. Related Work

In recent years, computer vision and deep learning techniques have been applied to the classification of lymphoma based on images, offering the potential to improve the accuracy and reliability of diagnoses. Orlov et al. [3] present a classical computer vision method that uses image transforms, such as the Fourier or the wavelet transform, to extract global features from the lymphoma images. These global features capture high-level information about the images, such as texture, shape, and intensity distribution. The authors then use a support vector machine (SVM) with a radial basis function (RBF) kernel as a classifier to determine the type of lymphoma

present in the image based on the extracted features. They are also responsible for the gathering of the dataset, that is utilized in this paper and by the works that will be presented below. Janowczyk and Madabhushi [5] use instead AlexNet framework [6] to solve different digital pathology challenges, including lymphoma classifications. The aim of the paper is to show that a single architecture can be trained to solve various tasks with results comparable to or even better than state-of-the-art feature-based approaches. For lymphoma classification in particular it is reached an accuracy of 96.5% on NIH dataset (using 36×36 sub-patches of original images). Tambe et al. [7] achieve improved results with the implementation of Inception V3 architecture [8], obtaining an accuracy of 97.3% on the test set using full-sized images and employing data augmentation techniques. A more sophisticated analysis is introduced by Al-Mekhlafi et al. [4], which uses a decision fusion mechanism. In particular, two different neural networks, Resnet50 [9] and DenseNet [10], are exploited as feature extractors. The outputs of these networks are then compressed via principal component analysis (PCA) and classified with an SVM. A similar version of this architecture, tested in the same paper, involves extracting hand-crafted features from the images and concatenating them with features coming from the convolutional networks. The resulting feature vector is then classified by a feed-forward neural network. This work reached the perfect accuracy on the NIH dataset and an accuracy of 99.5% on a bigger dataset.
Although the latter method represents, to the best of our knowledge, the state-of-the-art for lymphoma classification, it incorporates a complex architecture utilizing three separate networks for feature extraction. Our objective instead is to achieve a balance between accuracy and model simplicity.

III. Processing Pipeline

Our architecture's task is to classify lymphoma images. To achieve this result each image is processed in the following way:

1) Firstly some preprocessing e.g. colorspace conversion and data augmentation is applied to the image, which is then split into patches
2) Each patch is fed to a convolutional network that acts as a feature extractor
3) Features are then passed on to a classifier
4) The label assigned to the full image is the one with the highest number of patches associated to it (majority vote)

This pipeline is visualized in Fig. 1.

As the very first thing, however, the convolutional network must be trained. In order to find the most efficient network several architectures were tested, specifically

- Inception V4 [11]
- ResNet-50 [9]
- A shallow CNN
- EfficientNet B0 [12]

These networks are trained end-to-end to classify different patches of the dataset images so that the loss function can be associated with single-patch misclassification.

This procedure is repeated for three different colorspaces[1]:

- *Grayscale:* a single channel color representation where each pixel is represented by a single value representing its intensity, with 0 being black and 255 being white
- *RGB:* that represents colors as a combination of three primary colors (red, green, and blue) with varying intensity values
- *L*a*b*:* that separate lightness and color information into three different channels, one for luminance, one for red-green, and the last one for yellow-blue

The first two colorspaces are selected for their wide diffusion in image representation. L*a*b* colorspace (CIELAB) is instead based on the human visual perception of color and is designed to be perceptually uniform, meaning that small changes in the values of the colors correspond to small changes in the perceived color difference.

The architectures are then used as feature extractors after training. To extract data from the model, the last dense layer (which performs classification) is removed, resulting in an output that is a flat version of the last convolutional layer. These features are then classified with machine learning algorithms that do not involve neural networks, in particular SVM and XGBoost.

A. Support Vector Machine

Support Vector Machine (SVM) is a popular machine learning algorithm used for classification and regression tasks. It is based on the concept of finding the optimal boundary between different classes, known as the maximum margin hyperplane. The support vectors are the data points closest to the hyperplane and have the greatest impact on determining its position. SVM employs a cost function that penalizes misclassifications, and the algorithm seeks the hyperplane with the greatest margin between classes while minimizing misclassifications. The algorithm can handle non-linearly separable data using the kernel trick, in which data is transformed into a higher dimensional space where a linear boundary can be found. In particular, this space can be obtained using the radial basis function (RBF) which is a function that decreases as the distance between two points increases.

B. XGBoost

XGBoost (eXtreme Gradient Boosting) is an advanced implementation of gradient boosting, a popular machine learning algorithm that combines multiple weak models to form a strong, accurate model by iteratively correcting the mistakes of the previous models and focusing on the samples that are difficult to predict. In particular, XGBoost uses decision trees as weak learners. These are trained to predict the negative gradient of the loss function, which guides the optimization

[1] for the EfficientNet we focused only on the RGB color space due to time constraints

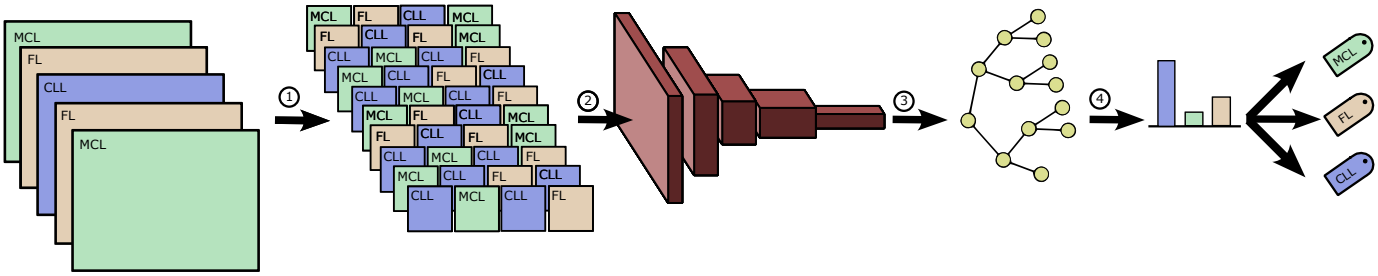

Fig. 1: Processing pipeline schema: ①: preprocessing and patches splitting; ②: feature extraction via CNN; ③: feature classification; ④: majority voting

process toward the minimum of the loss function. XGBoost also uses second-order gradient information to update the model, which allows it to capture complex interactions in the data. Decision trees are trained using a variety of techniques such as regularization, feature pruning, and weighting of instances, which helps to prevent overfitting and improve the model's generalization performance. Moreover, a wide range of hyperparameters is provided to control the model's behavior and avoid overfitting, such as the learning rate, number of trees, tree depth, and regularization parameters.

IV. Signals and Features

The dataset used in this work consists of 374 histopathological images obtained from H&E-stained biopsies by several pathologists from different sites. The dataset is subdivided into three types of lymphomas: 113 histology images of CLL, 139 histology images of FL, and 122 histology images of MCL. All images are stored as tiff with a resolution of 1388×1040.

Using the `StratifiedShuffleSplit` strategy [13], which maintains class balance, the dataset is divided into training, validation, and test splits with the respective fractions of 0.6, 0.2, 0.2.

After being loaded, images are converted to the selected color space (RGB, LAB, or grayscale), and the values are mapped to the $[0, 1]$ range. Patches are then extracted from each image using the `tf.image.extract_patches` [14] method. In this work, the patch size is set to 256 (299 for the inception model) and the stride to half that value: this means that there is a 50% overlap between patches. In Fig. 2 we show an example of extracted patches for the three color spaces.

During the feature extractors' training, data augmentation techniques such as image flipping, shifting, and rotation, are applied to increase the actual size of the dataset helping with avoiding overfitting and increasing robustness and performance.

V. Learning Framework

As stated in section III various networks were trained end-to-end to better extract the features of the patches. Aside from the shallow CNN, the selected networks were picked for their well-known performances or efficiency. The building blocks of each layer of the network are a series of a convolutional block, a batch normalization layer [15] and an activation function (usually a ReLU [16]).

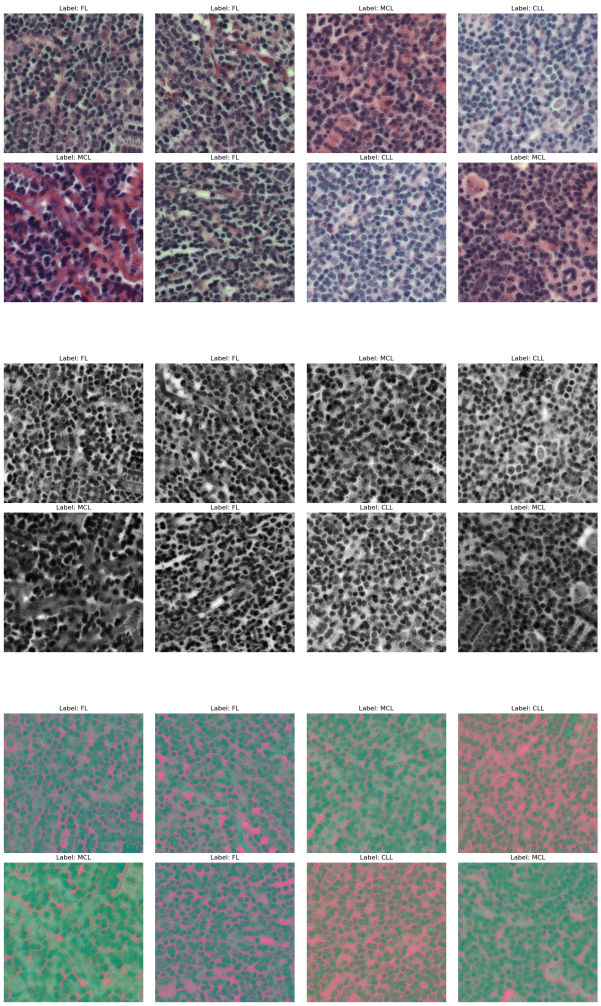

Fig. 2: Sample of extracted image patches in the RGB color space (top), grayscale (middle), LAB color space (bottom)

Inception V4 (2017)

Inception V4 [11] is a very popular deep-learning architecture for image classification. This network can capture both fine-grained and high-level information in the image using a combination of standard convolutional layers and inception modules. The latter are fundamental building blocks of the networks that perform multiple operations in parallel, allowing the network to learn more complex representations of the

input data. Every module consists of several parallel branches, each of which performs a different operation such as 1×1, $3\times$, or 5×5 convolution. The output of each branch is then concatenated and fed into the next layer of the network. By using multiple parallel branches, the inception module allows the network to learn multiple representation scales for each feature in the input data. In terms of computational efficiency, Inception V4 has relatively fewer parameters (41.2 million) compared to other architectures with similar accuracy, making it faster to train. However, the architecture can still be computationally expensive and memory-intensive for our purposes.

ResNet50 (2015)

ResNet50 [9] is another common choice for the image classification task. ResNet50 is a 50-layer deep variant of the ResNet architecture that counts 23.5 million parameter. It was the first architecture to surpass human-level accuracy on the ImageNet dataset. The key feature of the ResNet architecture is the use of residual connections, which allows the network to learn residual functions rather than direct functions. In a residual connection, the input is added to the output of a convolutional layer, allowing the network to learn the residual difference between input and output rather than the output itself. This allows the network to avoid the vanishing gradients problem and learn much deeper representations of the input data, as the residual functions can be stacked multiple times to create deeper networks. The architecture also includes a series of 1x1 convolutional layers, which are used to reduce the number of channels in the network, allowing the network to be more computationally efficient

Shallow CNN

Despite eminent architectures selected for our task, we tested a smaller and more naïve network. This architecture is made of four convolutional layers, each including batch norm and ReLU function. The kernel size is 4×4 and the stride is 2 so that the input size is halved at each step. The number of filters in each layer is respectively 64, 128, 256, and 512. Eventually, the last layer is flattened; if the training procedure is ongoing, a 20% dropout is applied to the output, which is classified with a dense layer that returns one of the three labels. The choice to delve into a smaller network, with only 2.8 million parameters, stem from the desire to balance complexity with efficiency. To accomplish this, we must investigate all options, from the most computationally expensive to the most basic.

EfficientNet (2019)

Since it was introduced, EfficientNet [12] has become one of the most popular and highly-regarded architectures for image classification due to its high accuracy and computational efficiency. EfficientNet is based on the idea of scaling up neural networks in a systematic and efficient manner: unlike other architectures that only increase the depth or width of the network, it uses a combination of depth, width, and resolution scaling to achieve the best accuracy-efficiency trade-off. This is achieved through the use of a compound scaling method that balances the different dimensions of the network to improve its accuracy and efficiency. The smaller version of this network, called "B0", uses only 4 million parameters while still achieving results comparable to models larger more than an order of magnitude. EfficientNet architecture takes inspiration from MobileNet [17], one of the most famous architectures to deepen the efficiency of neural networks. The model is based on Inversed Residual Blocks (IRB) with carefully selected dimensions. These blocks apply the following strategies:

- **Depth-wise separable convolutions:** In this technique the convolution operation is split into two separate parts: a depth-wise convolution that performs filtering on each channel of the input, and a point-wise convolution that combines these results. This reduces the computational cost of the network while maintaining high accuracy.
- **Squeeze-Excitation blocks**: These blocks use global information to re-calibrate the feature maps. This global information is generated by first "squeezing" the feature maps along the spatial dimensions with an average pooling to reduce their dimensionality, and then "exciting" the resulting representation using a fully-connected layer and a sigmoid activation function to produce a set of attention weights. These attention weights are then used to rescale the feature maps, effectively highlighting the most important features for the task at hand
- **Stochastic depth:** During training some layers are randomly skipped with a certain probability, effectively reducing the depth of the network and adding more diversity to the training process. It is similar to dropout but entire layers are removed rather than individual neurons. This results in a more robust model that can prevent overfitting, reduce the computational cost, and improve the generalization capabilities.

Training procedure

The training of the feature extractors is carried out for up to 100 epochs. An early stopping mechanism is in place to monitor the validation accuracy: if it does not improve for 10 consecutive epochs, the training is halted. In this work, we adopt the Adam optimizer [18] with an initial learning rate of 0.001. The learning rate is then reduced by a factor of 5 if the validation accuracy does not improve for 5 consecutive epochs. We keep in any case a minimum learning rate of 10^{-5}. The batch size is set to 32 throughout all experiments. The training on an NVIDIA GeForce RTX 2080 Ti takes around 2 minutes per epoch for the Inception model, $\approx 90\,\mathrm{s}$ for the ResNet, and $\approx 70\,\mathrm{s}$ for the CNN.

Once the training is completed, we remove the last dense layer and we leverage the model to extract the features from the image patches that are now used to train the two proposed classifiers: the SVM and the XGB. To further improve the performance of the model we optimize some of the most relevant hyperparameters of the classifiers. Using optuna [19], a bayesian optimization library, we perform a study with 100

trials, selecting the combination of hyperparameters with the highest validation accuracy.

For the SVM classifier, the following parameters are tuned:

- *C*: the regularization parameter. The inverse of C is the coefficient of the l2 penalty. Helps with overfitting
- *kernel*: which kernel to use. Possible choices are linear, polynomial, sigmoid, or radial basis functions (rbf)
- *degree*: the degree of the polynomial kernel
- *gamma*: kernel coefficient. When using the rbf kernel, this is the width of the Gaussian radial basis function

For the XGB classifier, the following parameters are tuned:

- *n_estimators*: number of decision trees in the model. A trade-off between performance and computational cost
- η: the learning rate
- *max_depth*: the maximum depth of each decision tree. Larger values could lead to overfitting
- α: coefficient of the l1 penalty
- β: coefficient of the l1 penalty
- γ: minimum loss reduction required to make a split in the tree. Regulates model complexity, higher values will result in fewer splits, and therefore a simpler model
- *min_child_weight*: determines the minimum number of samples required in a child node to be eligible for splitting. Larger values result in more conservative models.
- *subsample*: controls the subsampling ratio of the training data. A value of 1 means that all data is used. Can help with overfitting

With the best hyperparameters found a final training on the extracted features is performed and the model moves to the evaluation phase. For each image, we get the model prediction on each of the extracted patches and the class with the highest number of assigned patches (majority vote) becomes the final prediction for the full image.

VI. Results

In this section we present the results of our experiments: first we discuss the training outcomes of the feature extractors and the effect of the choice of the color space, then we showcase the overall performance of our model.

Feature extractors and color space

In Fig. 3 we report the training history for the feature extractors dealing with the different color spaces. It is easy to conclude immediately that converting images to a single grayscale channel leads to worse performances: this is true for all models. Training, with the exception of the ResNet architecture, is also cut off sooner, well before the 100 epochs limit, signaling that the models have reached capacity. Given this result, we decide to exclude the grayscale color space for the subsequent analysis.

Between RGB and LAB, the situation is not so clear: for the inception model, the RGB accuracy is higher while for ResNet the opposite is true. For the CNN, even if the training with the LAB images takes more epochs, the final validation accuracy reached is the same. For this reason, we keep both color spaces when studying the classifier performance.

In general, in the beginning, the training is not very stable due to the high initial learning rate, but then once it is automatically lowered the improvement becomes more stable and steady. It is also evident that the larger the model is the more epochs are required before it reaches capacity. The Inception is still marginally improving even at the 100 epoch mark.

While the Inception model is the one with the highest accuracy in classifying the patches, it is interesting to notice how the performances of the ResNet and our custom CNN are really close.

Classifiers performances

In Tab. 1 we collect all the metrics useful to evaluate the performances of our proposed models for the full image classification task. For each feature extractor, we report its standalone performance[2] and the combination with the SVM and the XGB for both the RGB and LAB color spaces.

As expected, the Inception model is the one with the highest accuracy and best performance among the models we are testing. The combination with the SVM or the XGB leads to no improvements and there is no benefit in introducing them in this case. For the Inception+XGB model, we show the confusion matrix for the train, validation and test split in Fig. 4 and the ROC curve in Fig. 5. The 97.3% accuracy that we achieve with this model translates in only 2 misclassified images: more in detail one sample of MCL is classified as CLL and one of FL as MCL, while all CLL images are classified correctly.

Things are different for the CNN and the ResNet: the addition of the SVM or the XGB delivers an increase in accuracy in the range of $2-3\%$ points w.r.t. the standalone model. The combination of EfficientNet and XGB is the only exception, showing a degradation in performance, while with the SVM there is a 1.3% increase (and almost a +7% for the validation set). Overall, for these three feature extractors, there is a combination of color space and classifier that allow them to all reach an accuracy of 93.3%.

This last result is very interesting if we put things into perspective: in Fig. 6 we position our models based on their accuracy and the number of floating-point operations that they require. Of course, the model with the highest number of parameters and the highest computational cost is the better performing. However, if we look towards optimizing not only the accuracy but also reducing the number of operations and the memory footprint, we can observe that our CNN and the EfficientNet have the same performance as the ResNet for a fraction of the number of parameters and the computational cost.

Finally, we can not from these results select a better-suited color space between RGB and LAB for this problem, since both have overall similar scores depending on the combination of feature extractor and classifier.

[2]the classifier, in this case, is the dense layer used during the first training phase

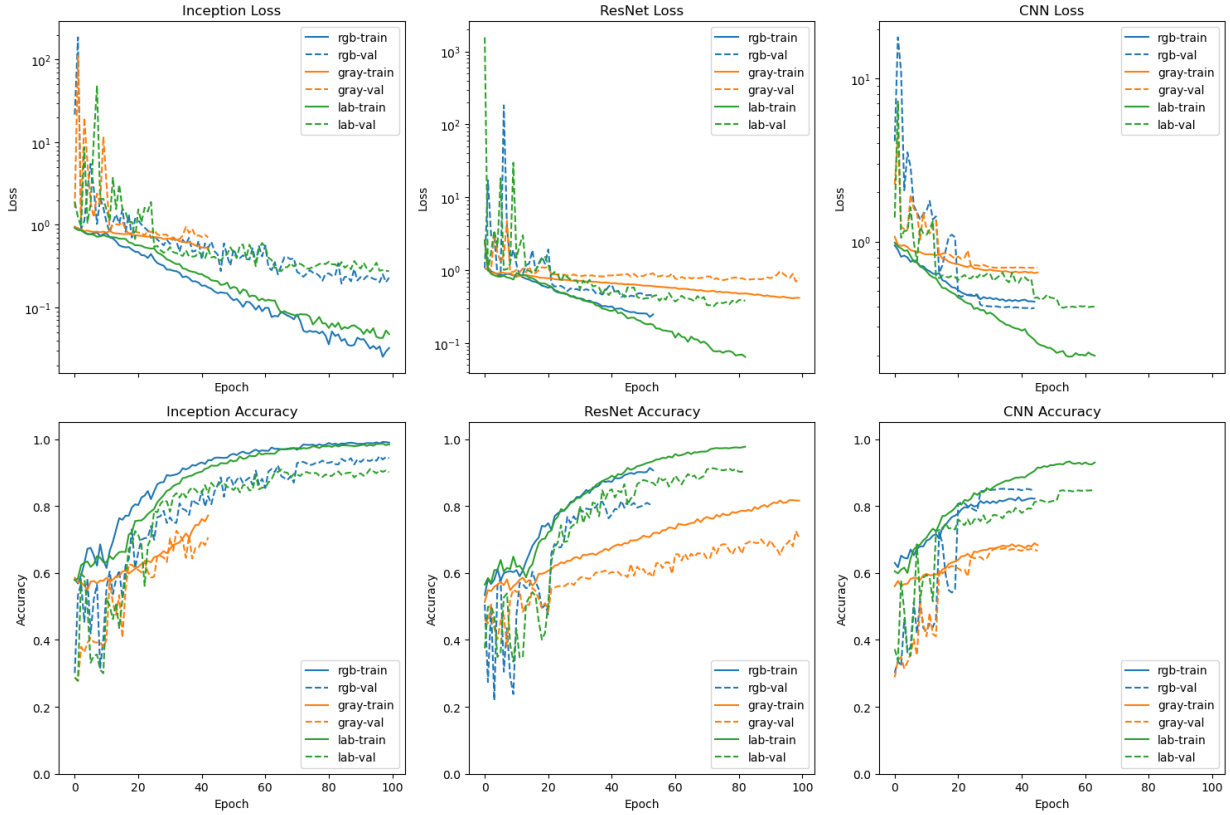

Fig. 3: Loss (top row) and accuracy (bottom row) curves during training of the (from left to right) Inception, ResNet and CNN. The dotted lines show the validation behavior. The RGB data is in blue, the grayscale in orange, and the LAB in green

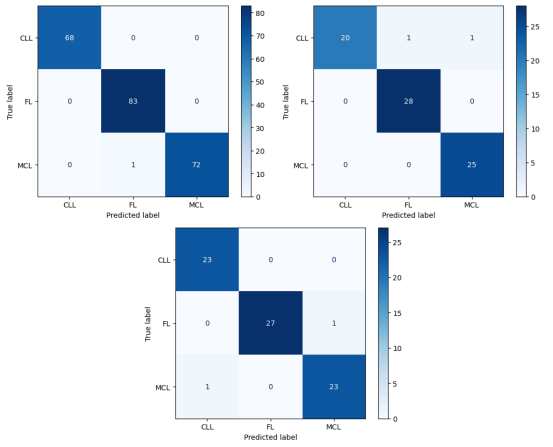

Fig. 4: Confusion matrix for our Inception+XGB model. From top to bottom the values for the train, validation and test set respectively

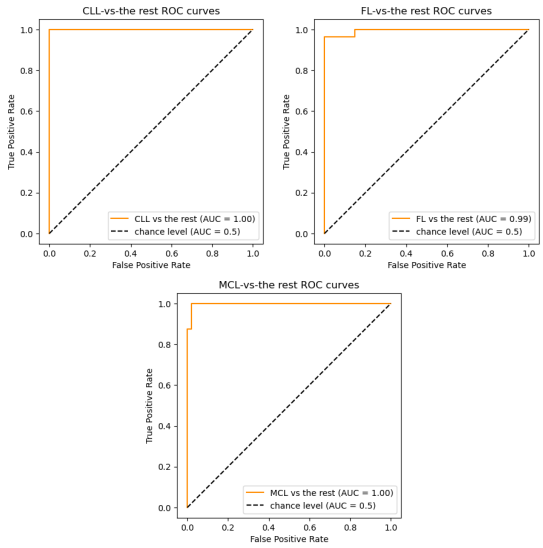

Fig. 5: ROC curve for the 3 classes of the test set for our Inception+XGB model

VII. Concluding Remarks

In this paper, we have proposed a solution for the lymphoma detection task exploiting different CNNs. Our best model, Inception, is close to the state-of-the-art accuracy for this task. One of the goals of this work was to study the efficiency of our model. Despite the fact that the best-performing architecture is the Inception one, we can observe that the other three models show similar accuracy to each other. This is reached by having nevertheless a different number of parameters and different numbers of operations: our CNN has five times fewer parameters than ResNet and less than half G-FLOPS. This fact suggests that models are perfectible and that further studies are

Model	Color space	Accuracy			Precision			Recall			F1-score			ROC AUC		
		Train	Val	Test	Train	Val	Test	Train	Val	Test	Train	Val	Test	Train	Val	Test
Inception	LAB	1	0.947	0.933	1	0.947	0.933	1	0.943	0.936	1	0.944	0.933	1	0.995	0.986
	RGB	0.996	1	**0.973**	0.996	1	0.972	0.995	1	**0.974**	0.996	1	**0.973**	1	1	**0.998**
Inception+SVM	LAB	1	0.947	0.947	1	0.947	0.946	1	0.943	0.948	1	0.944	0.946	1	0.994	0.985
	RGB	0.996	0.987	**0.973**	0.996	0.987	0.974	0.995	0.985	**0.974**	0.996	0.986	**0.973**	1	1	0.996
Inception+XGB	LAB	1	0.960	0.960	1	0.959	0.960	1	0.958	0.962	1	0.958	0.960	1	0.995	0.986
	RGB	0.996	0.973	**0.973**	0.996	0.976	0.972	0.995	0.970	**0.974**	0.996	0.972	**0.973**	1	1	0.997
ResNet	LAB	0.991	0.946	0.907	0.991	0.947	0.903	0.991	0.945	0.904	0.991	0.944	0.903	1	0.997	0.976
	RGB	0.893	0.867	0.893	0.888	0.864	0.892	0.888	0.866	0.889	0.888	0.864	0.889	0.979	0.977	0.977
ResNet+SVM	LAB	1	0.973	0.920	1	0.973	0.918	1	0.972	0.919	1	0.972	0.918	1	0.998	0.981
	RGB	0.915	0.867	0.88	0.912	0.872	0.878	0.912	0.870	0.877	0.912	0.866	0.877	0.988	0.981	0.979
ResNet+XGB	LAB	1	0.973	0.933	1	0.973	0.931	1	0.972	0.932	1	0.972	0.931	1	0.998	0.975
	RGB	0.888	0.893	0.893	0.884	0.893	0.892	0.882	0.892	0.889	0.882	0.891	0.889	0.973	0.980	0.970
CNN	LAB	0.982	0.867	0.907	0.982	0.867	0.907	0.982	0.867	0.908	0.982	0.865	0.906	0.999	0.978	0.973
	RGB	0.942	0.907	0.920	0.944	0.902	0.921	0.938	0.901	0.922	0.939	0.901	0.919	0.992	0.970	0.977
CNN+SVM	LAB	1	0.947	0.893	1	0.952	0.893	1	0.943	0.894	1	0.945	0.893	1	0.990	0.978
	RGB	1	0.920	0.933	1	0.917	0.933	1	0.913	0.936	1	0.914	0.933	1	0.981	0.975
CNN+XGB	LAB	1	0.907	0.933	1	0.905	0.936	1	0.903	0.937	1	0.903	0.933	1	0.988	0.971
	RGB	1	0.907	0.933	1	0.902	0.933	1	0.903	0.936	1	0.902	0.933	1	0.985	0.975
EfficientNet	RGB	0.964	0.813	0.920	0.963	0.964	0.964	0.816	0.821	0.814	0.922	0.924	0.920	0.996	0.965	0.967
EfficientNet + SVM	RGB	0.987	0.880	0.933	0.987	**0.986**	0.987	0.889	0.879	0.878	0.934	0.937	0.934	1.000	0.980	0.967
EfficientNet + XGB	RGB	0.955	0.867	0.907	0.956	0.955	0.955	0.871	0.869	0.867	0.907	0.910	0.907	0.998	0.970	0.965
Orlov et. al. [3]		0.990	0.990	0.990												
Janowczyk et. al. [5]		0.966	0.966	0.966												
Tambe et. al. [7]		0.975	1	0.973												
Zeyad et. al. [4]		1	1	1												

TABLE 1: Summary of the results obtained in our work and in the relevant literature. For the latter, we repeat the same value of the accuracy for the train, validation and test split when only the average accuracy is reported. The best value obtained for the test set by one of our models in each of the metrics is highlighted

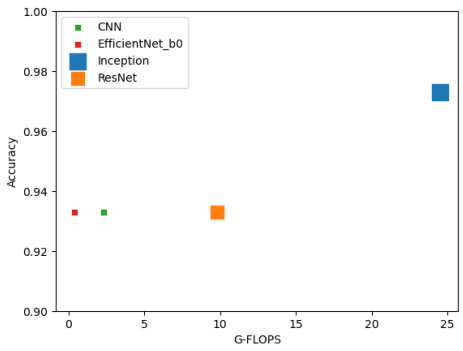

Fig. 6: Accuracy vs G-FLOPS plot. The marker size is proportional to the number of parameters of the feature extractor

necessary in this direction. Moreover, we speculate that with more experiments and some tweaks, one can easily increase the accuracy of these lighter models, also focusing on the preprocessing aspect

Furthermore, this work shows the limits of the dataset we have been working on: it suffices that a datum is misclassified to move our model away from the state-of-the-art model. It would be therefore necessary to expand the number of images inside the dataset to better evaluate the performances. As students it was really interesting to working on real data: usually deep learning laboratories uses artificial datasets that are specifically made for the classification task, whereas in our case it was taken from a real world scenario.

REFERENCES

[1] Y. Liu, J. Rao, J. Li, Q. Wen, S. Wang, S. Lou, T. Yang, B. Li, L. Gao, C. Zhang, *et al.*, "Tandem autologous hematopoietic stem cell transplantation for treatment of adult t-cell lymphoblastic lymphoma: a multiple center prospective study in china," *haematologica*, vol. 106, no. 1, p. 163, 2021.

[2] S. H. Swerdlow, E. Campo, N. L. Harris, E. S. Jaffe, S. A. Pileri, H. Stein, J. Thiele, J. W. Vardiman, *et al.*, *WHO classification of tumours of haematopoietic and lymphoid tissues*, vol. 2. International agency for research on cancer Lyon, 2008.

[3] N. V. Orlov, W. W. Chen, D. M. Eckley, T. J. Macura, L. Shamir, E. S. Jaffe, and I. G. Goldberg, "Automatic classification of lymphoma images with transform-based global features," *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 4, pp. 1003–1013, 2010.

[4] Z. G. Al-Mekhlafi, E. M. Senan, B. A. Mohammed, M. Alazmi, A. M. Alayba, A. Alreshidi, and M. Alshahrani, "Diagnosis of histopathological images to distinguish types of malignant lymphomas using hybrid techniques based on fusion features," *Electronics*, vol. 11, no. 18, p. 2865, 2022.

[5] A. Janowczyk and A. Madabhushi, "Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases," *Journal of pathology informatics*, vol. 7, no. 1, p. 29, 2016.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[7] R. Tambe, S. Mahajan, U. Shah, M. Agrawal, and B. Garware, "Towards designing an automated classification of lymphoma subtypes using deep neural networks," in *Proceedings of the ACM India joint international conference on data science and management of data*, pp. 143–149, 2019.

[8] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[10] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.
[11] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, 2017.
[12] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, pp. 6105–6114, PMLR, 2019.
[13] "Stratified shuffle split." https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html. [Online; accessed 10-February-2023].
[14] "Extract image patches." https://www.tensorflow.org/api_docs/python/tf/image/extract_patches. [Online; accessed 10-February-2023].
[15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, pmlr, 2015.
[16] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.
[17] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.
[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.
[19] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," 2019.