Reaching high accuracy in lymphoma images classification

Giulia Rizzoli †, Anna Vettoruzzo ‡

***Abstract*—The increase in the usage of medical imagery and the development of acquisition devices and computing technologies allow developing efficient techniques for image analysis and classification. The latter plays an important role in diagnostic and teaching purposes in medicine. Traditionally, many different features were extracted from the image and a classifier was used for label assignment. Since 2014, deep neural networks started to become mainstream, yielding substantial gains in image classification tasks, although the increase in the model size and computational cost. In this paper Deep Learning is used to classify lymphoma images into three different types: Chronic Lymphocytic Leukemia (CLL), Follicular Lymphoma (FL) and Mantle Cell Lymphoma (MCL). In the proposed scenario, convolutional neural networks (CNN) and recurrent neural networks (RNN) are used for classification purposes. Different color spaces and different network architectures have been taken into account to reach the highest accuracy on a test set of images. The results reveal that the cascade of CNN and RNN layers with RGB images is able to extract the relevant features from each image and perform the final classification with good fidelity, despite the increase in the computation cost and overfitting. This model outperforms the traditional Machine Learning approaches reaching an accuracy higher than 97%, despite the higher computational burden.**

***Index Terms*—Image Classification, Lymphoma Images, Neural Networks, Convolutional Neural Networks, Recurrent Neural Networks.**

I. Introduction

Lymphomas are neoplasms that originate in the immune system from either T-cells or B-cells and represent one of the most common types of cancer in the World population [1]. According to published statistics, more than 12,700 new cases were estimated in Brazil between 2018 and 2019 and more than 77,240 in the United States in 2020 [2]. These statistical data demonstrate the importance of new researches that contribute to a more accurate diagnosis of lymphomas.
There are more than 38 subtypes of lymphoma but the most common ones are Chronic Lymphocytic Leukemia (CLL), Follicular Lymphoma (FL) and Mantle Cell Lymphoma (MCL). They can be distinguished based on the pattern of growth and cytologic features of the tumour: CLL shows pale areas at low magnification and small round nuclei with condensed chromatin at high magnification; FL exhibits a follicular pattern with regular-shaped nuclei at high magnification; MCL type has the highest variability and it shows at high resolution irregularly shaped nuclei, while at a lower one it reveals several patterns such as mantle zone, nodular, diffuse and blastic [3]. However the variability in the histopathological features and in the clinical presentation and preparation have made diagnosis a complex task for specialists. These problems have led to the development of different techniques that can assist the specialist during the diagnosis of cancer. In the literature many attempts to solve the lymphoma image classification problem have been done adopting Machine Learning techniques. However, these techniques have some drawbacks because they are not able to generalize to new unseen data and they cannot deal with large datasets [4]. Therefore the main purpose of this study is the development of a Deep Learning architecture which is able to discriminate between the three different classes and identify the corresponding lymphoma sub-type. We start with a quite simple CNN architecture with images from different color spaces (RGB, grayscale, Lab, H&E). Then we try to use a more sophisticated architecture considering the combination of CNN and RNN in order to exploit the correlation in the patches extracted from the images to perform the final classification. The code developed for this study allows to choose different configuration and combination of parameters in order to select the best model for the purpose of the user: each model architecture and each color space has its own advantages and disadvantages that they will be described in section V.

This report is structured as follows. In Sections II we resume previously exploited approaches for lymphoma classification while underlying their critical issues. In Sections III we detail the methodologies used in the various stages of the proposed approach. Section IV presents the learning strategy, applied to solve the problem, and section V the results of the different model architectures and color spaces. Concluding remarks and future improvements are provided in Section VI.

II. Related Work

As already mentioned, much of the previous works in lymphoma classification and medical images classification in general, involve a two-step approach where features are first extracted from the images and then fed into a classifier for labels assignment. This approach is described in the paper of Orlov et. al. [4] where computer vision techniques are used to characterize the content of the images. Simple and compound transforms have been applied to each image in order to obtain a feature vector for automated classification with different classifiers (e.g Naive Bayes Networking (NBN), Weighted Neighbour Distance (WND) and Radial Basis Func-

†Department of Information Engineering, University of Padova, email: {giulia.rizzoli.1}@dei.unipd.it
‡Author two affiliation, email: {vettoruzzo}@dei.unipd.it
Special thanks / Michele Rossi and Francesca Meneghello.

tion (RBN)). The performance of this method is computed on the same dataset of images considering different color spaces (Gray, RGB, Lab, H&E [5]). It was concluded that the H&E color space is the best one for this purpose, giving 98% of accuracy. However with this approach is not possible to diagnose new cases on unseen images without the use of standardization in sample preparation. To address the issue of standardization, in 2016, Janowczyk et. al. [6] used a dataset where samples have been prepared from different pathologists at different sites, which we also utilized in our work. Janowczyk et. al. introduce the concept of patching the images, splitting the dataset into patches of size 36x36 with a stride of 32 to augment the training set and reduce the overfitting. These patches are used to train AlexNet architecture using a 5-fold cross-validation. For testing, patches are extracted with the same methodology and a majority voting scheme is applied in order to assess the predicted label for each image, achieving an accuracy of 96.6%. However, in some cases, there is not a voting victory for sure, so the slide should be reviewed manually. Furthermore, at test time, the K-fold method further reduce the variance of the output, computing the mean of many rotations, and also increase the computational burden. In our work, we started from the procedure described in [6] trying to implement a data augmentation approach with random cropping to increase the variability of the training data. Then we try to change the network architecture in order to reach higher accuracy and reduce as much as possible the overfitting, taking also into account the suggestions in [7].
In the last years new approaches, based on a combination of CNN and RNN, are emerging for image classification purposes. In particular a CNN is used to generate discriminative image features that become the input of a recurrent network to model the sequential relationship of the features and so to assign the correct label to each image. In the paper of Qiwei Yin et. al. [8] a CNN-RNN model is used to classify images on the CIFAR-10 dataset obtaining a test accuracy of 80%, that is higher with respect to use only the CNN. However this architecture increases the complexity of the model and the risk of overfitting. The rationale behind Yin's paper has become the starting point for the final part of our work: we design a cascade of convolutional and recurrent layers, followed by dense layers in order to increase the performance obtained only with a CNN and to keep the complexity of the model as low as possible.

III. Processing Pipeline

Our work is organized in different parts that are linked together in order to obtain the final classification. A diagram of the various blocks is shown in Fig. 1.

Dataset description. The dataset used in our work consists of 374 images of size 1388x1040. These images are further divided into three sub-classes representing the three most common forms of lymphoma: 113 for the chronic lymphocytic leukemia (CLL), 139 for the follicular lymphoma (FL) and 122 for the mantle cell lymphoma (MCL). This dataset represents a collection of samples prepared by different pathologists at different sites with a large degree of staining variation to reflect real-worlds situation.

Color space selection. By default the dataset is loaded in the RGB color space, but the user can select different color spaces among RGB,grayscale, Lab, H&E, H, E. Even if from the pathologist's point of view malignancy of the tissue is reflected in morphology and not in the colors, the performance of deep learning models are highly dependent on the color space selected. A widely used technique is to treat the samples with stains that have selective affinities for different biological substances, like Hematoxylin-Eosin (H&E) color staining [5]. Also Lab color space is often used with medical images due to its ability to represent color in a device-independent and perceptually uniform way. Instead grayscale allows to perform the classification in an easier and faster way with respect to other color spaces, without considering the information associated with the colors.

Processing of images and patches generation. Once images are loaded in the code some pre-processing operations are carried out in order to normalize the dataset and to adapt the dimensions of the vectors when only one channel is used to represent the images. The dataset is split into training and test set with respectively 80% and 20% of images. Overlapping patches with size 36x36 and with a stride of 32 are extracted from the images to augment the size of the dataset, ensuring a sufficiently large set of samples to efficiently train the network and reduce overfitting.

Data Augmentation. Further augmentation is performed in real-time with the main purpose of increasing data variability rather than enlarging the dataset size. Several spatial transformations are taken into account: height and width shift, rotation, shear intensity change, zoom and flip. The last image manipulation is random cropping: the extracted patches are randomly cut into 32x32 pixels, which are fed to the model.

Model selection. Three models are proposed for this study: standard convolutional neural network for image classification; recurrent neural network to exploit the correlation in the images and the combination of the two in which CNN is used as a feature extractor and then RNN accounts for correlation and discriminates between classes. Advanced details about network structures and functions are given in the next section.

Majority voting strategy. During testing time a majority voting strategy is used to classify the images in the test set based on the classification of the corresponding patches. This strategy considers the predictions of all the patches extracted from a single image and assigns with the overall image the class associated to the majority among the patches classification. If the predicted label is the same as the real one the final accuracy is incremented by one unit. Finally, the code returns a warning in case the image is correctly classified but less than $2/3$ of the associated patches belong to that class and an error message if the image is wrongly classified.

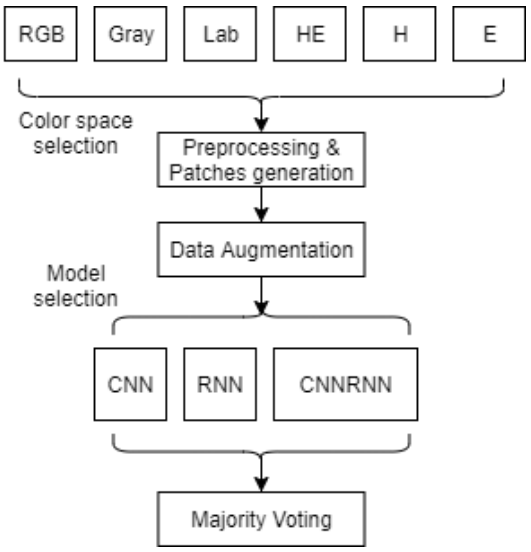

Fig. 1: Processing pipeline

IV. Learning Framework

In this section, we describe more in details the model architectures that we are taking into account in this project.

CNN Model. CNN is a deep neural network that takes in input an image and feeds it to a network that mimics the connectivity and the learning mechanism of the neurons in the human brain. The architecture of the CNN is designed to take advantage of the structure of the input image exploiting local connectivity of the neurons and shared weights. Furthermore, it employs multiple filters to detect different features in an image and usually a pooling operation to reduce the resolution of the feature map decreasing the computational burden and adding some deformation invariance.
The structure of the network used in this project consists of a sequence of basic units and pooling layers followed by dense layers for image classification purposes. Each basic unit is composed as follow:

- a convolutional layer extracts features from the input by applying multiple filters (kernels). The image is first padded, adding some zeros at the boundaries, then each filter is slid across the image and convolved with it at multiple locations in order to generate a feature map;
- a batch normalization layer normalizes the output of the previous layer and smoothes the objective function to allow a faster training convergence;
- an activation function layer applies a ReLU function to remove negative values and to introduce non-linearity in the data, that is very common in the real world.

Each feature map is then sub-sampled with a max-pooling operation to reduce the dimensionality of the data before entering in the next layer. After four basic units and three pooling layers, the three fully connected layers allow to create a hierarchy among the features to reduce the under-fitting of the network and to obtain a more robust classification. On the other hand it may happen that the model obtained fit the training data too well, but fails to generalize to new examples. This problem is called over-fitting and it is solved adding three dropout layers, among the fully connected ones, to randomly drop some neurons with a probability $p = 0.4$. For the final classification a softmax activation function is used at the output layer to convert each output into the probability that the input patch belongs to a specific class. In Tab. 1 a summary of the model is provided, where each *Conv* layer represents a basic unit.

Type	No. Kernels	Kernel size	Stride	Activation
Conv 1	32	5x5	1	ReLu
Max Pooling 1	-	2x2	2	-
Conv 2	48	5x5	1	ReLu
Max Pooling 2	-	2x2	2	-
Conv 3	64	5x5	1	ReLu
Max Pooling 3	-	2x2	2	-
Conv 4	128	2x2	1	ReLu+ Dropout
Dense 1	128	-	-	ReLu+ Dropout
Dense 2	64	-	-	ReLu+ Dropout
Dense 3	3	-	-	softmax

TABLE 1: CNN Architecture.

RNN Model. Recurrent neural network is a generalization of feedforward neural network to make use of sequential information. RNNs are called recurrent because they perform the same function for every input data with the output being dependent on the previous computations. In order to do so each neuron has a ”memory” which captures information about what has been calculated so far. Thanks to this ability RNNs are mainly used with sequential data. However they could be used also for image processing tasks with some modifications in the way of thinking, since the training data are not sequences, but single images. We adopted two different strategies depending on the number of channels of the selected color space. Either way, we treated a single patch as a sequence of data. In multi-channel images we considered each pixel as an entry and we exploited the correlation between pixels of a patch. The cropped patches have a size of 32 (rows) and 32 (columns). Therefore, to implement this approach, the patches are flattened over the two first dimensions, producing a new vector of size (1024, no. of channels). In the opposite case - for instance with grayscale images - flattening is not needed:

the row pixels are used as time steps and the column pixels as features. In this way for each row pixel, the correlation between all the column ones is given, without losing spatial information.

The model adopted in this project is described in Tab. 2. It consists of two GRU layers, each one followed by a dropout layer, and two dense layers for the final classification. The Gated Recurrent Units(GRUs) act as memory cells and are able to solve the vanishing gradient problem during backpropagation. Gradients are values used to update neural networks weights, but as the gradient has to back propagate through time could happen that the gradient shrinks, blocking the RNN to fulfil their promise. A GRU has internal mechanisms called gates that can regulate the flow of information, learning which data in a sequence is important to keep or throw away. It is also possible to develop a bi-directional RNN in such a way that it learns from both past and future samples. In our work we have also tried to use this approach, substituting the standard GRU layers with bi-directional ones, but the improvements in the performance were not enough to balance the huge increase in the computational cost.

Type	No. Kernels	Activation
GRU 1	256	ReLU+ Dropout
GRU 2	128	ReLU+ Dropout
Dense 1	32	ReLU+ Dropout
Dense 2	3	softmax

TABLE 2: RNN Architecture.

CNN and RNN mixed Model. Although deep CNNs have shown good results in the classification of lymphoma images, they are unable to fully exploit the long-term dependence relationship between the features extracted from the images. To resolve this problem we have introduced RNN. However the latter has a weaker ability in exploiting the local structure of the images, so the combination of CNN and RNN becomes necessary to improve the image classification task: CNN allows to extract relevant features from the input image and RNN exploits the output of the CNN to generate the label and optimize the classification [9]. In particular, in our work, we combine the two architectures described before with slight modifications in order to improve the performance of the classifier and reduce as much as possible the overfitting. For this purpose we consider three basic units, like the one described above in "CNN Model", and two max-pooling operations to reduce the data dimensionality and increase the generalization. In order to compare this model with the results obtained with the CNN, the pool size and the stride of max-pooling layers are selected in such a way that the output of the last convolutional layer has the same dimensions than in the CNN. Then two reshapes and a flatten operation are needed to adjust the dimensions of the vectors that will enter in the RNN in order to exploit the correlation among pixels in each image. In particular, we want to keep the dimension associated with the feature extracted from the last convolutional layer unchanged, performing a flattening on the other two dimensions. In this way we finally obtain a matrix with the flattened vector considered as time dimension and the values produced by the filters of the last CNN as entries [10]. This matrix enters in a RNN with two GRU layers and dropout layers to prevent the overfitting of the network. When several GRU layers are concatenated, they adopt a "return-sequences" mode that produces a sequence of vectors forming the entry point to the next GRU layer [10]. In this way the temporal dimension of data input remains always the same. After that, two fully connected layers perform the final classification with a softmax function. The summary of the architecture is shown in Tab. 3.

Type	No. Kernels	Kernel size	Stride	Activation
Conv 1	32	5x5	1	ReLu
Max Pooling 1	-	2x2	2	-
Conv 2	48	5x5	1	ReLu
Max Pooling 2	-	4x4	4	-
Conv 3	128	2x2	1	ReLu+ Dropout
GRU 1	256	-	-	ReLu+ Dropout
GRU 2	128	-	-	ReLu+ Dropout
Dense 1	32	-	-	ReLu+ Dropout
Dense 2	3	-	-	softmax

TABLE 3: CNN+RNN Architecture.

Performance evaluation. For classification problems, the performance of the classifier is typically defined according to the accuracy and the confusion matrix. Based on the entries of the matrix it is possible to compute other measures like precision, recall and F1-score, that are more suited in the case the dataset is unbalanced. Precision is the fraction of true positive over the total number of positives; recall quantifies the number of correct positive predictions made out of all positive predictions; F-measure is the harmonic mean of precision and recall. The formulas for these metrics are described below where FP and FN indicate the number

of false positive and false negative, respectively, that reflects the wrong predictions, and TP and TN the true positives and true negatives when the detection or no-detection is the same of the groundtruth.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2\frac{Precision \cdot Recall}{Precision + Recall}$$

In our work we take into consideration the training and test accuracy, the accuracy returned using majority-voting strategy and the metrics described above. Furthermore, we plot two confusion matrices: the first one considering the patches extracted from the images in the test set and the other only with the images. However, for the sake of clarity, we'll report all the measures only for the optimal results and for all the others we show only some of these metrics.

V. Results

In this section we describe the main results of our work, starting with a short description of the initial proves that we did and going on with the following improvements. In particular the section is divided into three paragraphs, each one related to a specific model architecture, and the numerical results are gathered in Tab. 5.

In all the simulations some parameters related to the training options are maintained fixed. The network is compiled using Adam optimizer and sparse categorical cross-entropy as loss function. Adam optimization is a stochastic gradient descent method that takes advantage of an adaptive method to find individual learning rates for each parameter. Categorical cross entropy function is used in multi-class classification tasks to estimate the errors of the predicted values with respect to the real labels, as shown in the next equations:

$$Loss = -\sum_{i=1}^{\#classes} y_i \cdot log(\hat{y}_i)$$

where y_i is the true label and $\hat{y}_i$ is the prediction. The overall loss is then computed by considering the average over samples in a training batch. The training of the network is performed using 20 epochs and mini-batches with 32 observations at each iteration. A validation set with 20% of training data is initially used to test the performance of the network and to assess the overfitting but the final model is trained without considering a validation set.

CNN results. As described in section II we tried to design an architecture similar to the one described by Janowczyk [6]. However we realized that both the AlexNet architecture and the k-fold techniques were not suited for our purposes. The first one because it allowed to achieve an accuracy of only 87% on test data in RGB color space and the latter for the high computational cost due to the sheer amount of models that had to be trained. Therefore we changed our mind considering the model described in the previous section. In this way, both the training time and the test accuracy improves a lot reaching a value of 92.54% on test data in RGB color space. The same model is trained with the same dataset but in different color spaces and the numerical results are summarized in Tab. 5. In particular we can notice that the best performance is obtained in Lab color space with a test accuracy of about 93.8%, slightly better than RGB. This is also confirmed by the majority voting strategy that provides an accuracy higher than 98% with only one wrongly classified image among the 75 in the test set (see Fig.2). This is probably related to the fact that the Lab color space was designed as an orthogonal space that preserves all the information content that we have in a RGB image.

As expected, instead, considering an image in grayscale significantly worsens the performance of the classifier, due to the loss of information during the conversion.

As regards the H&E color space it is not very clear how it behaves. The model with H&E images reaches a quite high accuracy on the training set, but the performance are only slightly better than in grayscale. Furthermore the loss computed on training and test data suggests that the overfitting is quite high maybe because the stains are not able to highlight in an efficient way the different parts of the tissue or this differentiation is not sufficient for our classifier.

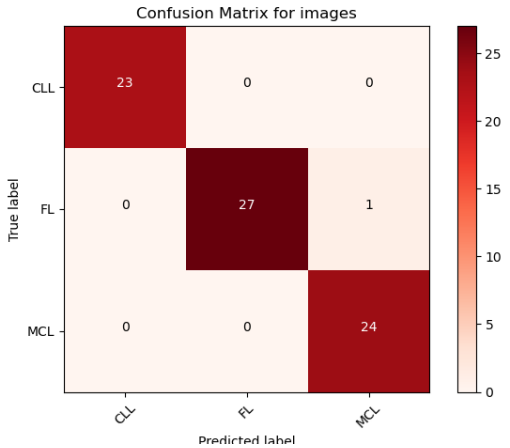

Fig. 2: Confusion matrix for images in Lab color space obtained with CNN classifier.

RNN results. In literature recurrent architectures are used on streaming of data such as audio and text signal, videos etc. Since we are dealing with images that are not time-related, we would have expected lower performance with respect to the CNN architecture. This is evident in

the case of single-channel data in which the test accuracy is very low: it seems that losing color information deeply affects the learning process of this model. However it is also true that for 3-channel spaces the training process was not concluded despite the long time allotted to the run of the code. Nevertheless, the best results are obtained in RGB color space, with an accuracy of 91.65% on the test set and 98.67% on majority voting (1 misclassified image over 75), as it is shown in Tab. 5. Similar perfomance is obtained in Lab color space with test accuracy of $87.45\%*$ and majority voting of 96%, confirming the discriminative ability of this color space. The more relevant issue with RNN model is its higher complexity in terms of time due to the image dimensions. This problem could be addressed by applying a dimensionality reduction technique, like PCA, but this is out of our scope. We tried, instead, to combine the CNN and RNN architectures in order to have a better trade-off between time and accuracy performance.

CNN+RNN results. Even if the performance obtained with CNN is very good we tried to implement a different architecture in order to increase the accuracy and try to adapt the RNN model to our dataset. At this scope we tried different architectures combining convolutional layers with recurrent ones, hoping to improve the learning process exploiting the correlation in the images. All the following attempts have been performed considering RGB color space. We started combining the optimal CNN architecture with the RNN described in section IV. As we can see in Tab.4 the overfitting is quite high because the CNN had already good performance and adding the recurrent layers we only increase the number of parameters. Therefore we have tried to exploit an easier architecture with only 2 basic units followed by two GRU layers and two fully connected layers. However, in this case, the accuracy on the test set was around 74% and the overfitting was still quite high. So we design an architecture in between the previous two reaching good performance both on training and test set, with only 2 images wrongly classified in RGB (see Fig.3). However, even if this architecture is the best one for our purpose with RGB images, it performs worse than the CNN architecture in the other color spaces. In particular, looking at Lab or H&E color space, we can notice that the model reaches a very high accuracy on the training set, but it fails to generalize to test data.

	Training Accuracy	*Test Accuracy*	*Majority Voting*
Optimal CNN + RNN	94.11%	85.97%	93.3%
2 Conv + RNN	93.61%	74.45%	77.33%
3 Conv + RNN	94.88%	90.25%	97.33%

TABLE 4: Performance of three different CNN+RNN architectures in RGB color space.

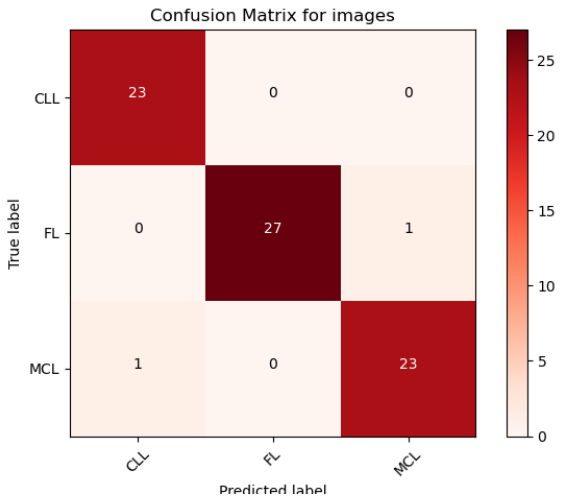

Fig. 3: Confusion matrix for images in RGB color space obtained with CNN+RNN classifier.

VI. Concluding Remarks

The lymphoma image classification problem is developing a great interest nowadays due to the large number of cases. This report aims at describing different strategies used to solve the problem exploiting deep neural network. We started considering a quite simple CNN applied on images belonging to different color spaces. Even if we reached very good results, in particular with Lab color space, we tried to further improve the performance of the classifier considering a RNN architecture. RNN performance was good in some selected color spaces, however due to high dimensionality of images it is not suitable for real-time adjustment of the model since the training has a very high time complexity. Therefore our final attempt was to develop a more complex architecture considering the cascade of CNN and RNN layers in order to exploit the correlation among the extracted features to optimize the final classification. To be honest, we were not been able to fully reach our goal because the final classifier achieves good results in RGB color space, but its performance is worse with other color spaces. Therefore we can conclude that the best architecture in terms of time, performance and color space generalization is still the CNN model that we have tried at the beginning.

Despite the quite high performance of the model, there is definitely scope for improvements. It could be interesting to train the models for more epochs because in some cases the accuracy continued to improve considering only 20 epochs. Furthermore the number of images in the dataset is too low and some operations, like the split of the dataset and the patches extraction, are performed randomly, influencing significantly the performance of the network. Finally, the proposed model is limited to identifying only three sub-types of lymphoma even if there are more than 30 different types. So it would be interesting to extend the work to all the lymphoma types and also to similar problems in the medical field.

During the development of this project we encountered

several difficulties in particular due to the lack of information and examples regarding the usage of RNN and the combination of CNN and RNN for image classification tasks. Indeed, most of the references cited for these two tasks, deal with other types of problems that we have tried to adapt to our specific purpose. The main difficulties are found with 3-channels images because we didn't know how to reshape the data in order to input them to the networks. But also the development of an efficient neural network architecture caused us some problems. However, thanks to the fact that we have always worked together, sharing our ideas and trying to implement different codes, we succeeded in reaching our goal. Furthermore, all the difficulties encountered during this work allow us to better understand neural networks and in particular their usage in image classification tasks. So we can conclude that with this project we improved our knowledge, but above all, we learned how to work together in order to reach good results.

REFERENCES

[1] F. Bray, J. Ferlay, I. Soerjomataram, R. L. Siegel, L. A. Torre, and A. Jemal, "Global cancer statistics 2018: Globocan estimates of incidence and mortality worldwide for 36 cancers in 185 countries," *CA: a cancer journal for clinicians*, vol. 68, no. 6, pp. 394–424, 2018.

[2] A. C. Society, "Non-hodgkin lymphoma cancer statistics." https://cancerstatisticscenter.cancer.org/cancer-site/Non-Hodgkin20lymphoma, 2020. [Online; accessed 01-Sep-2020].

[3] D. Gupta, M. S. Lim, L. J. Medeiros, and K. S. Elenitoba-Johnson, "Small lymphocytic lymphoma with perifollicular, marginal zone, or interfollicular distribution," *Modern Pathology*, vol. 13, no. 11, pp. 1161–1166, 2000.

[4] N. V. Orlov, W. Chen, D. M. Eckley, T. Macura, L. Shamir, E. S. Jaffe, and I. G. Goldberg, "Automatic classification of lymphoma images with transform-based global features," *IEEE Trans Inf Technol Biomed*, vol. 14(4), 2010.

[5] M. Macenko, M. Niethammer, J. S. Marron, D. Borland, J. T. Woosley, X. Guan, C. Schmitt, and N. E. Thomas, "A method for normalizing histology slides for quantitative analysis," in *Proceedings of the 2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, (Boston, MA, USA), June 2009.

[6] 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, 2016.

[7] G. E. H. Alex Krizhevsky, Ilya Sutskever, "ImageNet classification with deep convolutional neural networks," *Foundations and Trends in Machine Learning*, vol. 60, pp. 84–90, June 2017.

[8] R. Z. Qiwei Yin and X. Shao, "CNN and RNN mixed model for image classification," *MATEC Web of Conferences*, vol. 277, pp. 1–7, June 2019.

[9] Y. Guo, Y. Liu, E. M. Bakker, Y. Guo, and M. S. Lew, "CNN-RNN: a large-scale hierarchical image classification framework," *Multimedia Tools and Applications*, vol. 277, pp. 10251–10271, Dec. 2017.

[10] M. Lopez-Martin, J. Lloret, and B. Carro, "Network traffic classifier with convolutional and recurrent neural networks for Internet of Things," *IEEE Access*, pp. 1–9, Sept. 2017.

	CNN			RNN			CNNRNN		
	Training Accuracy	*Test Accuracy*	*Majority Voting*	*Training Accuracy*	*Test Accuracy*	*Majority Voting*	*Training Accuracy*	*Test Accuracy*	*Majority Voting*
RGB	96.86%	92.54%	96%	94.65%	91.42%	98.67%	94.73%	90.25%	97.33%
GRAYSCALE	76.85%	70.27%	85.33%	62.23%	55.81%	54.67%	74.95%	38.40%	41.33%
Lab	97.19%	93.80%	98.67%	92.65%*	87.45%*	96%*	94.77%	65.18%	64%
H&E	95.17%	73.91%	80%	70.46%*	70.34%*	77.33%*	92.69%	30%	38.67%
H	37.12%	37.33%	37.33%	49.03%	45.83%	48%	65.90%	38.28%	38.67%
E	69.45%	63.51%	77.33%	53.25%	39.19%	37.33%	66.41%	54.52%	62.67%

TABLE 5: Summary of results. (* indicates that the training is not ended).