Human Data Analytics

Final Project: Lymphoma Subtype Classification

Gianpietro Nicoletti[1] and Stefano Campagnola[2]

[1]Department of Information Engineering, ICT for Internet and Multimedia, University of Padua
[2]Department of Physics and Astronomy, Physics of Data, University of Padua

***Abstract*—Blood cancer, also known as lymphoma, has become widely studied in the digital pathology research field, with the objective of finding an automated process to perform quick and reliable diagnosis. For this purpose, machine learning applications on digitized image analysis have proven to give promising results.**

In this work, we aim to determine lymphoma subtypes identity on Haematoxylin and Eosin stained histological images. Proposed deep learning approach involves a self-supervised autoencoder architecture, which is compared with a simple CNN classifier. The goal is to exploit the image reconstruction pretext task to improve the classifier baseline performance. Our models are trained on squared patches of the original images, which are independently classified. Image labels are then retrieved from the patches' ones, in a voting mechanism that prevents ambiguity on the final classification. Moreover, we discuss and demonstrate how the choice of taking less information can help the model to be more robust.

The experimental results show how the best accuracy is reached with the self-supervised autoencoder model and is equal to 94.67% in the test dataset. Our framework outperforms by 10% our baseline showing how it is able to increase the performances of simple models reaching accuracies comparable with the state-of-the-art architectures.

***Index Terms*—Neural networks, Supervised Learning, Autoencoder, Classification, Lymphoma Subtypes Classification, Convolutional Neural Networks.**

I. Introduction

Lymphoma is the most common form of haematological malignancy in the developed world. Diagnosis for this type of illness is generally followed by subtype classification, in which H&E staining seems to be promisingly adopted by experts. Moreover, increasing use of scanning devices in Digital Pathology (DP) have made progressively possible to retrieve high resolution tissue images, which are suitable for machine learning classification techniques.

In this work, we present a Deep Learning (DL) approach based on an encoder-decoder architecture to afford reliable tissue classification. Previous papers tackled the problem of H&E image classification and succeed using pure convolutional classifiers, as AlexNet [1] and Inception nets [2]. The main drawback of these results are the size of the nets, which require heavy training computational power. On contrary, our model relies on much lighter amount of parameters, achieving nevertheless a consistent accuracy compared with the state of the art.

Our model follows the philosophy of self-supervised learning [3] in the exploitation of pretext tasks, such image reconstruction, to improve the performance of the downstream one (image classification). Autoencoders are in general pretrained in order to extract good features, and then a classifier is trained on the latent representation. Our pipeline is instead based on simultaneous training on both classification and reconstruction tasks. In particular, the dimensionality of the latent representation is chosen to be equal to the number of classes, in order to reflect the results of the network classification with no need of additive layers. All these tricks make our model, with respect to the state of the art, compact and minimal both from the architecture and training perspective.

This report is structured as follows. In Section II we describe previous works on DP and related topics, the designed pipeline and data preprocessing are respectively presented in Sections III and IV. The proposed autoencoder model is detailed in Section V and its performance evaluation is carried out in Section VI. Concluding remarks are provided in Section VII.

II. Related Work

A. Statistical classifiers

Earlier works on automated diagnosis of lymphoma subtypes [4] rely on the extraction of relevant features from the images at disposal. This is done through the combination of spectral transforms like Fourier, Chebyshev, and wavelets. Different colour schemes are considered to represent the dataset: RGB, Gray, Lab, H&E.

From the comparison of three statistical classifiers as Weighted Neighbour Distance (WND), Naive Bayes Networks (NBN) and Radial Bias Function (RBF) on the set of colour palettes, the best accuracy of 98-99% is reached with the H&E channels.

This study in particular reveals image processing techniques that facilitate the automatic recognition of the three classes of lymphoma. A limitation to this approach is the need for glass slide standardization among different laboratory groups, which is not always possible to impose. Neural networks are instead more suitable for datasets incorporating a variety of stainings, which motivate subsequent research.

B. AlexNet architecture

Deep learning contribution to DP goes beyond lymphoma subtype classification, as presented in [5]. In this article, sev-

eral subtasks concerning tissue segmentation or detection are tackled in a unified framework. The Existing CNN architecture called AlexNet is in fact adopted in all use cases, with the only need to modify image preprocessing.
Another key concept that is introduced in [5] is patching of images. Each slide coming from NIA dataset is indeed split into small squares of size 36×36, with a stride of 32 pixels chosen to augment the available data. Classification of a given image is related to the one of its patches in a "winner-takes-all" fashion. The same idea, with some custom expedients, is adopted also in our model.
Final accuracy of AlexNet on the lymphoma classification task reaches 96.58%.

C. Inception network

Another well-known CNN architecture used to tackle the same problem in [2] is the Inception V3 network [6]. This computer vision's milestone is designed to capture features at various scales and resolutions, making use of multiple parallel convolutional layers of different filter sizes within the same network. The amount of trainable parameters for this kind of model reaches the number of 22M, which is more than ten times the size of the presented model. Finally, measured accuracy for Inception V3 on test images stands at 97.33%, improving previous results in this field.

D. Self-supervised learning

Self-supervised learning [3] is a machine learning technique in which a model learns to extract meaningful representations or features from unlabelled data. This is in general achieved with pretext tasks, indeed learning objectives, that provide a surrogate supervision signal to guide the learning process. Knowledge is then transferred to a target task, called in this framework "downstream task", in order to enhance reliability of the latter. The main advantage of self-supervision techniques is on the fact that pretext tasks do not require extra information like new labels.
Self-supervised learning has gained significant attention and success in recent years, particularly in domains where labelled data is scarce or expensive to obtain.

III. Processing Pipeline

In this section the entire project pipeline is explained in a very summarized way, starting from the block diagram reported in Figure 1.

As shown, a group of patches is obtained from a given input sample. These patches are used both for the training, to update the weights of the model, and for the test in order to assign a label to the input image. While the model and the design choice are described in Section V, the important fact is that the classification of the full image is determined by the prediction of the single patches. In a majority vote rule, the assigned label for the image will be the most frequent one among its patches.

IV. Signals and Features

The NIA curated dataset is composed by a set of 374 RGB images of size 1388×1040 pixels. It contains samples from 3 different classes of lymphoma: 113 for the *chronic lymphocytic leukaemia* (**CLL**) class, 139 for the *follicular lymphoma* (**FL**) class and 122 for the *mantle cell lymphoma* (**MCL**) class. The dataset consists of samples prepared by various pathologists from different locations, which also display a range of colour variations in terms of staining. This is desirable, as it further enhances data variety used to train the model and makes it more resilient to location-specific artifacts.

The dataset was split into train set and test set with a number of sample equal to 80% and 20% respectively, the 12.5% of the train set was then reserved for the validation. At the end were obtained: 261 samples for the train, 38 samples for the validation and 75 samples for the test. According to Section III, the images were maintained as they are, but, instead of using the whole image, each of them was divided into $n_p = 40$ patches of size 128×128. The dimension of the latter represent a good balance to preserve biological information inside the patches and, at the same time, not increase too much the network complexity. Small squared shapes are also simpler to treat and generalize better our architecture usability. In fact, there is no dependence from the original image dimension in this case, and the dataset could have different resolution coming from different measuring equipments.
To explain better why only 40 patches were selected, one question can be asked: what happens if the images have less resolution? Considering the pixels in the original image they are $1388 \times 1040 = 1443520$, instead, considering the pixels in the patches they are $128 \times 128 \times 40 = 655360$ that leads to 45% of the original information (in the ideal case when there is not overlapping between patches). Concluding, the idea is to make sure that the model will be robust even if the information is less and in the section VI the effect of this choice is shown.

During the training phase, Gaussian noise with zero mean and standard deviation $\sigma = 2$ was added to the train samples, in order to prevent overfitting issues. In particular, to reduce the training time and to obtain a better comparison with the baseline (defined in Section V), the noise is added before the training, remaining fixed among the epochs. This in general can reduce the effect of such regularization technique, but was found to be necessary to achieve a fair comparison between baseline and autoencoder architecture.

The classes were mapped into integer numbers (Eq. 1) and the final labels were one-hot encoded in arrays with label smoothing parameter of value $\alpha = 0.1$. At the end the images are normalized, i.e. divide the channels' values by 255 in the RGB colour space in order to obtain values in range $[0, 1]$.

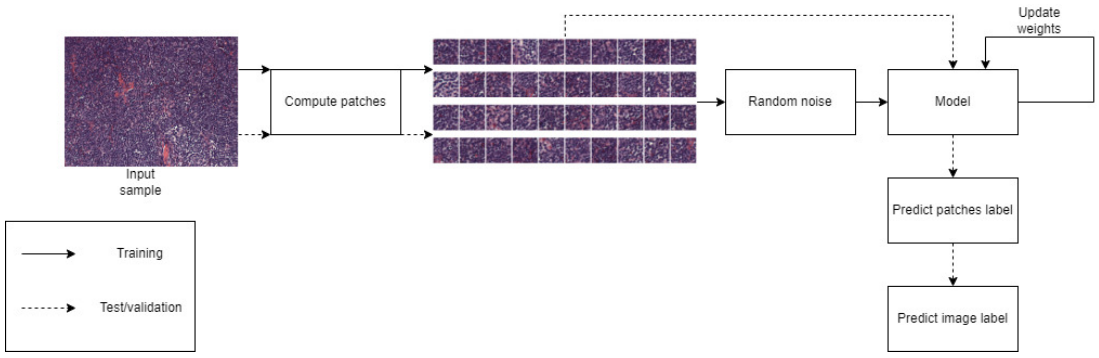
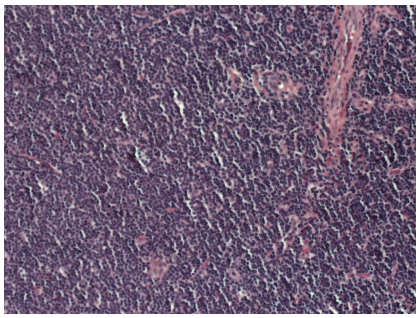
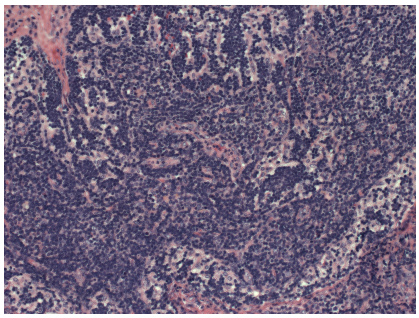


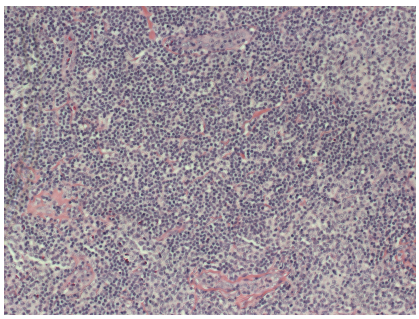
Fig. 1: Processing pipeline of the project.



(a) CCL



(b) FL



(c) MCL

Fig. 2: Example of samples from the three classes.

$$label = \begin{cases} 0, & \text{if class = CCL} \\ 1, & \text{if class = FL} \\ 2, & \text{if class = MCL} \end{cases} \quad (1)$$

V. Learning Framework

In this section, the model reported in Figure 3, the design choice and the prediction algorithm are presented.

A. Classifier

The structure of the classifier is a very standard structure: The network takes in input an image and produces its embedding using convolutional layers followed by linear ones. At the end, a prediction vector is produced, applying the softmax to the encoded representation.

B. Autoencoder

The autoencoder structure expects an encoder and a decoder part. Encoder coincides with the Classifier architecture except in the Softmax activation function. The decoder is designed to be a symmetric network with respect to the encoder. The output activation function is a Sigmoid, in order to obtain an output domain equal to the input one.

C. General details

In order to reduce the effect of the overfitting (see Figure 5 in Section VI), batch normalization is used at each layer, moreover the dropout is applied in the output of the first linear layer of the encoder. This is a strange choice, but this is a good compromise between good performances and overfitting. LeakyRelu is used as non-linear activation function.

Inspired by U-net architecture, long skip connections are considered between encoder and decoder layers. This prevents a common problem in deep networks, such as vanishing gradient, and favours image details recovery for the final reconstruction.

D. Training setup

Training was performed with RMSprop optimizer, learning rate was set to the constant value of $lr = 10^{-3}$ and the weight decay (similar to L2 regularization) was set to $\lambda = 10^{-2}$. Considered batch size was of 75 patches, manually tuned to

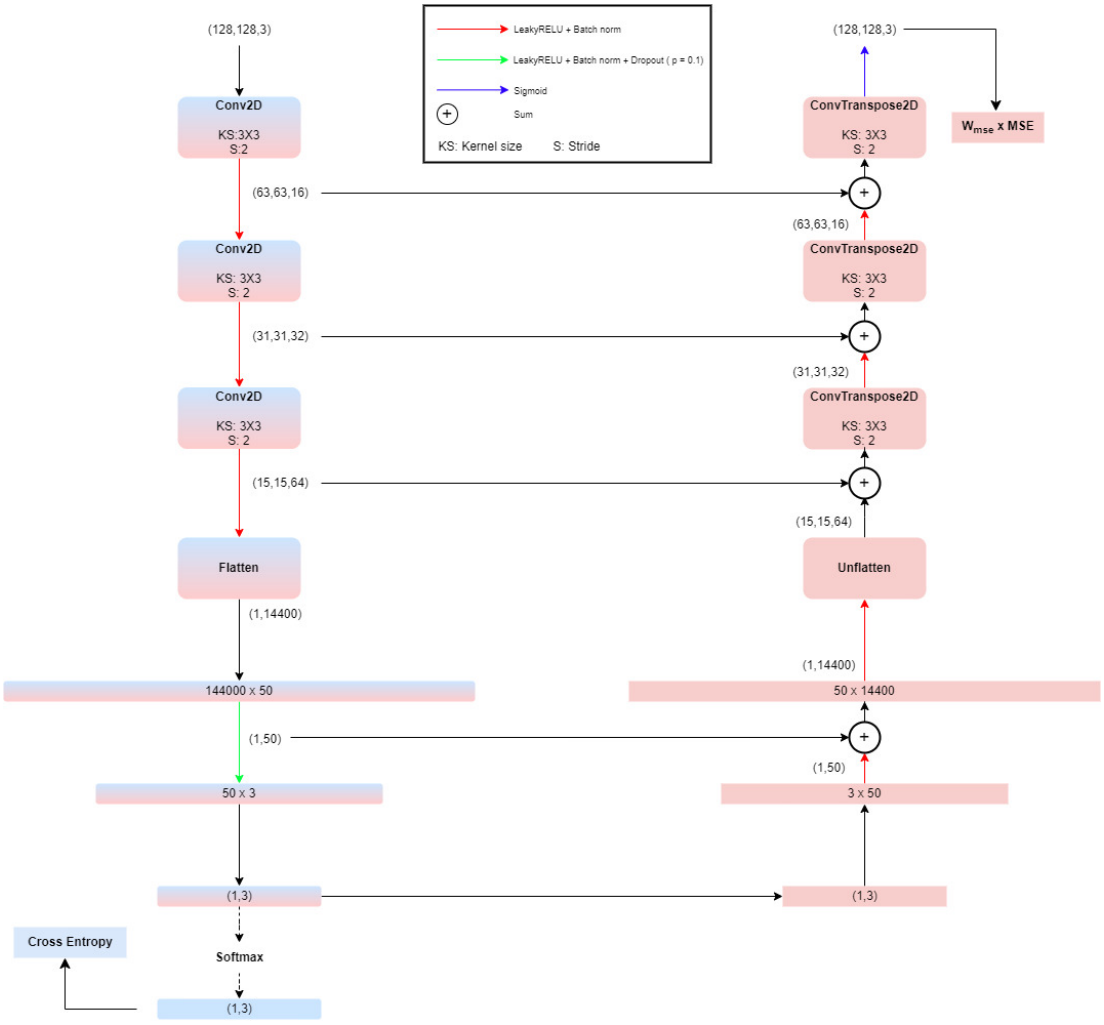


Fig. 3: Design of the model.

reach sensible training performance.
Among the epochs, models saved were two: the one with the best validation accuracy patches wise and the one with the best validation accuracy image wise. Our expectation is that the first one leads to better results with respect to the second one, due to the fact that:

- The model is trained considering the accuracy computed for each patch;
- Picking the model with the best accuracy image wise could favour predictions "on average" good (with slightly more than half of the patches classified correctly) but with lack of generality on new unseen images.

Since the main goal is combining the good compression technique of the autoencoder, in order to enforce the embeddings of the images with the classification task, the basic and intuitive idea is to use two losses: the MSE (L_{AE}) for the reconstruction task and the Cross Entropy (L_C) for the classification task. Notice at that point that, while the range of the Cross Entropy is $[0, +\infty]$, the range of the MSE, in case of the normalized samples, is in range $[0, 1]$. So, to better balance the two task, weights for the two losses W_C and W_{AE} are introduced in the Eq.2.

$$L_{total} = W_C \times L_C + W_{AE} \times L_{AE} \tag{2}$$

In practice W_C is considered only for notation purposes, but in the training it was set always equal to 1 and the only free parameter becomes W_{AE}, which controls the weight of the reconstruction loss during the training. In particular, for $W_{AE} = 0$ a simple classifier is obtained, which represents the baseline, while for $W_C = 0$ all comes down to a standard autoencoder.
What stated before is valid even if the not null weight (i.e. W_C in the case of the simple classifier or W_{AE} in the case of the autoencoder) remains $\neq 1$ (and > 0) since during the training every constant term that multiply the loss can be included in the learning rate of the optimizer.

E. Prediction algorithm

Vanilla approach:

The basic idea of the prediction algorithm is presented in Section III and the results will be presented in Section VI.

Formally, given an image img and the set of all its patches $\mathcal{S}_{img}$, and defining $o_{img,patch}$ to be the embedding of a patch belonging to img (i.e. the 1×3 vector obtained after the softmax activation function), the prediction label of that patch can then be seen as:

$$\hat{y}_{img,patch} = \arg\max(o_{img,patch}), \quad \hat{y}_{img,patch} \in \{0,1,2\} \tag{3}$$

Introducing $count_{img}(i)$ as the occurrences of the label i in the patches predictions $\hat{y}_{img,patch}$ belonging to $\mathcal{S}_{img}$, the predicted label for the entire image is:

$$\hat{y}_{img} = \arg\max_{i}(count_{img}(i)), \quad i \in \{0,1,2\} \tag{4}$$

The main problem with the vanilla approach is that there may be some cases of uncertainty in the final prediction, i.e. there exist at least two labels i and j such that $count_{img}(i) = count_{img}(j)$.

Solution to the "same count problem":

A different approach is proposed, merging the information inside $o_{img,patch}$ and the accuracy of the train set during the test phase, to solve the problem introduced by the vanilla approach. In particular, two quantities must be defined:

- The weighted sum of the prediction:

$$o_{img} = \sum_{patch \in \mathcal{S}_{img}} o_{img,patch} \tag{5}$$

  that is simply the sum of the embeddings for all the patches of the same image;
- Given the confusion matrix CM (See Figfure 9) of the train set computed during the test phase, we define $C_{i \to j}$ the number of times that the real label i was predicted as j:

$$C_{i \to j} = CM_{i,j} \tag{6}$$

Starting from these metrics, the following workflow can be adopted.

Prediction algorithm

for each image:

1) Compute the predictions i and j with the two highest $count_{img}(\cdot)$ (w.l.o.g. suppose $count_{img}(i) \geq count_{img}(j)$);
2) Compute o_{img} from all the $o_{img,patch}$;
3) if $|o^i_{img} - o^j_{img}| \leq threshold$, with $threshold \in (0, n_p]$, then:

$$\hat{y}_{img} = \arg\min_{k}(C_{h \to k}), \tag{7}$$
$$k = i,\ h = j\ (or)\ k = j,\ h = i$$

otherwise:

$$\hat{y}_{img} = \arg\max_{k}(o^k_{img}), \tag{8}$$
$$k \in \{i, j\}$$

In practice: if the weights of the two classes are close, the prediction that produced less error in the train set (using the vanilla approach) is taken, otherwise the class with the biggest weight is taken as prediction.
To be more formal "the weights of the two classes are close" means that the classes are considered equiprobable, in fact the condition at point 3) can be rewritten as follows considering the percentage w.r.t. the maximum value for sum^i_{weight} that is equal to n_p:

$$\frac{|o^i_{img} - o^j_{img}|}{n_p} \leq \frac{threshold}{n_p} \tag{9}$$

In more compact format:

$$\Delta_{i,j}(o_{img})_{\%} \leq threshold_{\%} \tag{10}$$

VI. Results

In this section, considered models are evaluated in terms of image and patch accuracy on all the three dataset without noise. In addition, train and validation losses trends during training are depicted in the same plots.
The network is trained and tested on a machine with 16 GB of RAM, a CPU Ryzen 5800H and a GPU Nvidia 3060. Some of the metrics related to time and space consumption are reported below. Computations are done with respect to the train dataset, consisting of 261 images made of 40 patches each.

- Number of parameters / Size of the network: 1531230 / 5.84 MB
- Time for an epoch (reading from the RAM): $26\frac{s}{epoch} = 0.1\frac{s}{image} = 0.002\frac{s}{patch}$.
- Time for the test (read from the local memory): $1min\,36s = 0.37\frac{s}{image} = 0.01\frac{s}{patch}$

A. Overfitting issue

In general, considered models tend to overfit on the training dataset, while validation accuracy oscillates a lot between good and bad performance. Due to this fact, the selection of the best model can be a non-trivial step of the processing. One way to ensure the onset of good weights is to run the training for enough epochs, which in our case were at least 80.

B. Baseline

As baseline, we consider the classifier network trained without the effect of the decoder, using $W_{AE} = 0$. As said previously, in this case the architecture is the one of a simple convolutional classifier.
Experiments on this network are useful in order to quantify the effect of the improved model in the final results.
Training stage trends plots of MSE loss, CE loss and accuracies are reported in Figure 4.
We choose to retain the model that reaches the best validation accuracy patch wise, and its evaluation is carried on using the Vanilla approach schema. Results in test stage are reported in Table 1.

Table 1: Baseline losses and accuracies during the test with the best accuracy patches wise.

Dataset	**MSE**	**CE**	**Accuracy** *(patches/images)*
Train	0.0291	0.8507	88.06% / 91.95%
Validation	0.0279	0.8561	88.03% / 94.73%
Test	0.0288	0.8936	81.20% / 84.00%

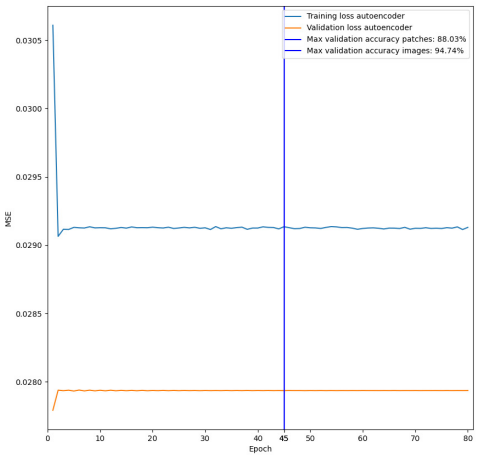
C. Autoencoder and classifier

For this model, the value of W_{AE} is set to 50 in order to add and balance the effect of the reconstruction during the learning phase. Also for this architecture, the trends of the metrics are reported in Figure 5 and the retained model performances during the test in Table 2.
An important thing to notice is that the model works correctly on the secondary task, reconstructing very well the input patches (Figure 6).

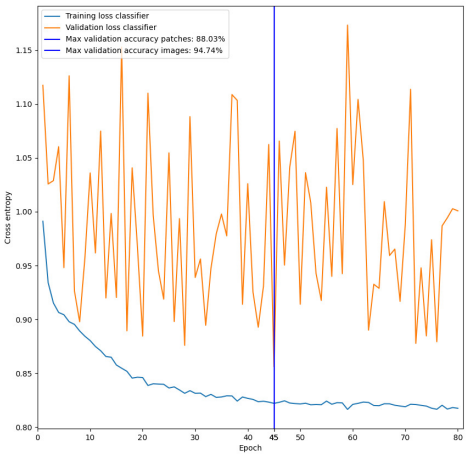
Table 2: Complete model losses and accuracies during the test with the best accuracy patches wise.

Dataset	**MSE**	**CE**	**Accuracy** *(patches/images)*
Train	0.0008	0.8397	89.60% / 93.10%
Validation	0.0009	0.8549	86.25% / 92.11%
Test	0.0008	0.8482	88.53% / 93.33%

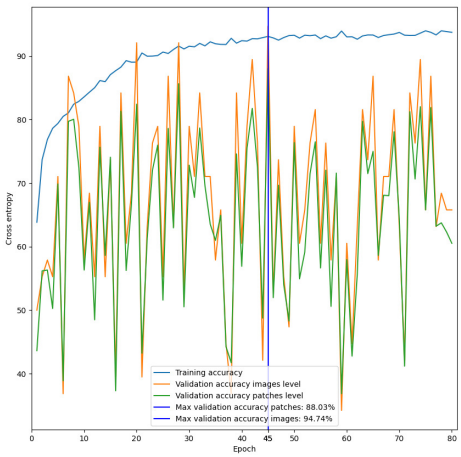
Comparison between the baseline and the complete model is straightforward: while in the complete model the value of the accuracies are quite similar for the three datasets, in the baseline the accuracy in the test set has a very low value. This is a sign of better generalization of the self-supervised model, since embeddings are enforced by the presence of the decoder. An interesting thing to notice is that, during complete model training, there is a quite high correlation between the value of the MSE and the accuracy. Even if the correlation is not clear, it is easy to see how, most of the time, small values of the MSE loss correspond to high values of the accuracy.



(a) MSE loss.

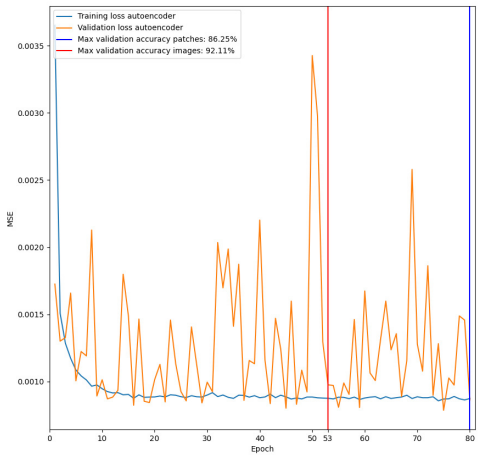


(b) Cross Entropy loss.

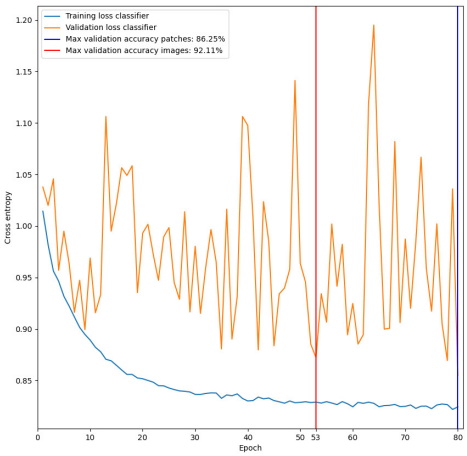


(c) Accuracies

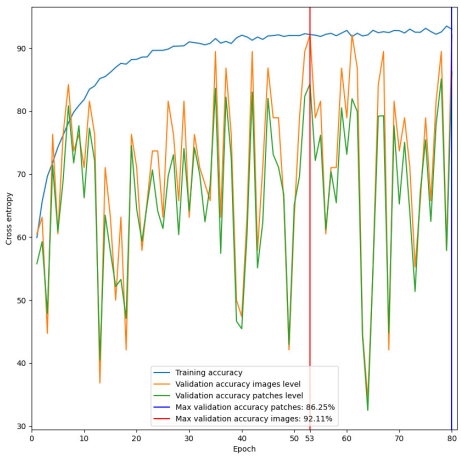
Fig. 4: Plots of the metrics for the baseline.



(a) MSE loss.



(b) Cross Entropy loss.



(c) Accuracies

Fig. 5: Plots of the metrics for the complete model.

A final observation is that the complete model reached the best value at the end of the training, while the baseline stopped to increase its accuracy after 45 epochs. This fact shows how, the presence of the secondary task, forces the network to continuously adapt the embeddings during the training to obtain a better latent representation.

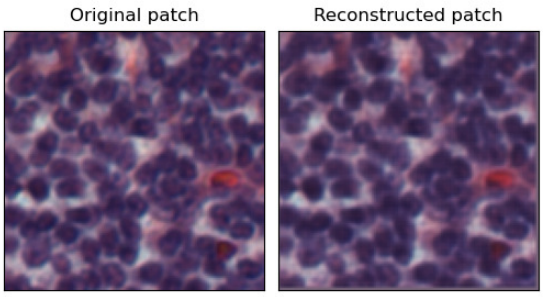


Fig. 6: Example of reconstruction of a patch.

Comparison between the best accuracy patches wise and images wise:
The purpose of this section is only to verify that the initial guess on the best model to retain was correct. To do so, again, the values are reported in Table 3.

Table 3: Complete model losses and accuracies during the test with the best accuracy images wise.

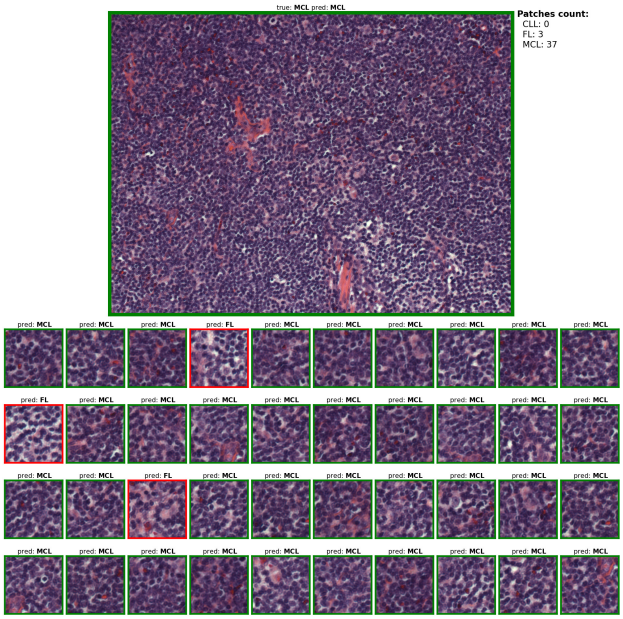
Dataset	**MSE**	**CE**	**Accuracy** *(patches/images)*
Train	0.0010	0.8669	86.29% / 90.03%
Validation	0.0010	0.8722	84.34% / 92.11%
Test	0.0010	0.8996	79.76% / 82.67%

D. Output example

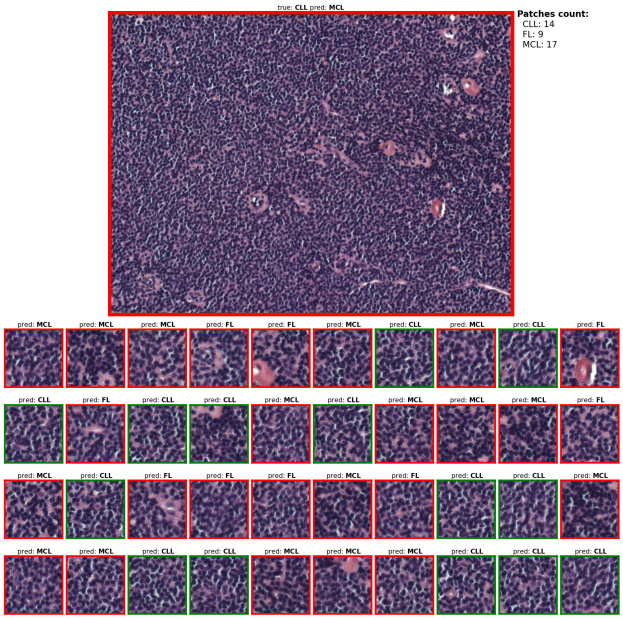
Some examples of image classification performed with the complete model are reported in Figure 7. Pictures show both the predictions for the patches and the final prediction for the entire image. Voting mechanism, in this case, is the Vanilla one.

E. Effect of the solution to the "same count problem"

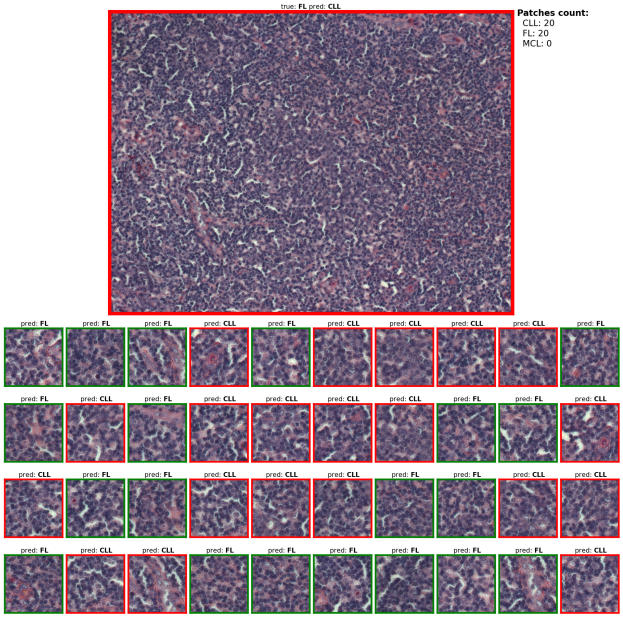
As stated in Section V, the proposed model must be merged with a different algorithm to assign the labels. The advanced approach was indeed applied using $threshold = 2$, which can be interpreted as setting $threshold_{\%} = 5\%$.
In this section the results are reported in Table 4 and the modified outcome of Figure 7c given by the proposed algorithm is reported in Figure 8a.
Moreover, the confusion matrix obtained from the training set with the use of the Vanilla approach is shown in Figure 9, since it is one of the information used in the new prediction algorithm. Confusion matrices in Figure 10 refer instead to



(a) Correctly classify sample



(b) Missclassify sample



(c) Same count example

Fig. 7: Example of outputs using the vanilla approach.

results with the new prediction rule.

While in Figure 8a the weighted count of the true label is the biggest, in Figure 8b this is not the case. So, for the latter, the usage of the confusion matrix of the train set was essential to produce the correct image prediction.

As in the Vanilla approach examples, both the predictions for the patches and the image are reported, with the addition of information about class weights for each patch and considered $C_{i \to j}$ values.

Table 4: Complete model losses and accuracies during the test with the best accuracy images wise using the proposed algorithm.

Dataset	**MSE**	**CE**	**Accuracy** *(patches/images)*
Train	0.0008	0.8397	89.60% / 93.10%
Validation	0.0009	0.8549	86.25% / 94.74%
Test	0.0008	0.8482	88.53% / 94.67%

F. Different number of patches

As the last experiment, a variation in the patch extraction method was introduced. Instead of randomly selecting 40 patches, a sliding window approach with a size of 128×128 and varying strides was applied to the images in the test dataset. It is worth noting that the performance of the prediction algorithm remains constant until a stride value of 1024. Specifically, when using a stride of 512, six patches are generated for each image (Figure 11).

VII. Concluding Remarks

In this work we proposed a self-supervised learning framework that combines a pretext task, i.e. a reconstruction task, to the classification task (the target task) in the context of the biological data. In particular, an autoencoder was designed to extend a CNN classifier model to increase the accuracy of the classification using a combined and balanced loss between the two tasks.

The images were split into 40 patches, which were used to train the model.

As baseline, only the classifier is taken into account, and it is used to evaluate the actual effect of the pretext task. We then compared the full autoencoder model results on two different prediction approaches: (1) using a simple count of class occurrences within the patch set for each image, and (2) employing a custom algorithm based on prediction weights to address situations of ambiguity. The second one turned out to be the best on exploiting patches information. The union of the autoencoder and the custom prediction algorithm is an extremely light model compared to state-of-the-art, while achieving similar accuracy and reducing time complexity. Final accuracy reached by this combo is of 94.67%, as shown in Table 5. In addition, the model has proven to give sensible results also on the secondary task, producing high-fidelity reconstructions.

In conclusion, the robustness of the model is tested by varying the number of patches, showing how, even with only 6 patches per image, the accuracy remains unaffected.

Based on these results, future work involves exploring the integration of our framework with a more complex network, to further improve accuracy and mitigate overfitting issues encountered during training. As mentioned in Section IV, incorporating a more effective data augmentation technique is another aspect that can be addressed, which was not fully explored in this study. Additionally, this framework should be tested on diverse datasets to evaluate its generalizability and performance across different domains.

Table 5: Summary of the tests.

	Accuracy *(patches/images)*
Baseline	81.20% / 84.00%
Proposed model (vanilla approach)	88.53% / 93.33%
Proposed model (weighted count)	88.53% / $\mathbf{94.67}\%$

References

[1] 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.

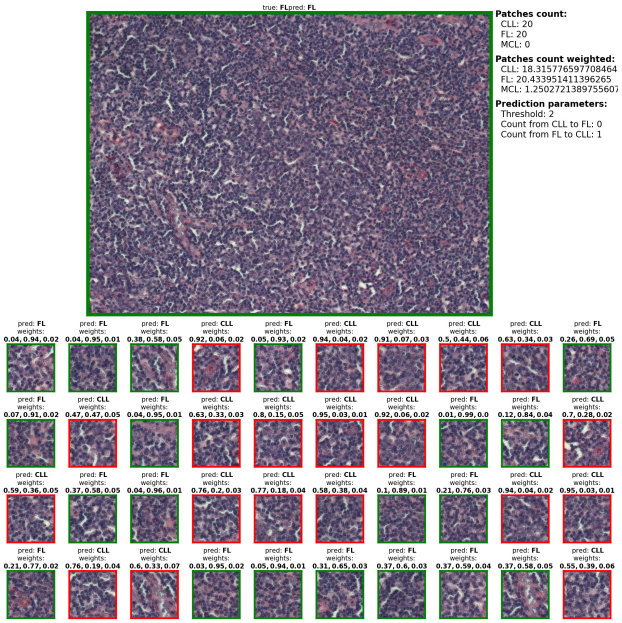
[2] R. Tambe, S. Mahajan, U. Shah, M. Agrawal, and B. Garware, "Towards designing an automated classification of lymphoma subtypes using deep neural networks," in *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, CODS-COMAD '19, pp. 143–149, Association for Computing Machinery, 2019.

[3] V. Rani, S. T. Nabi, M. Kumar, A. Mittal, and K. Kumar, "Self-supervised learning: A succinct review," *Archives of Computational Methods in Engineering*, pp. 1–15, 2023.

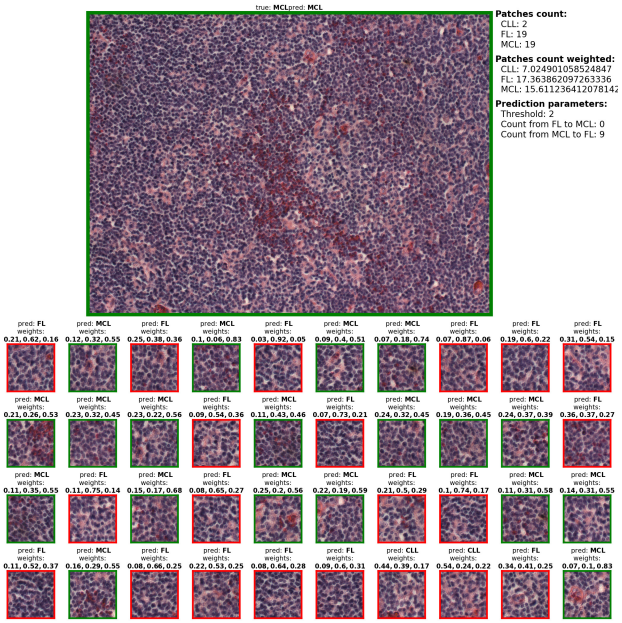
[4] N. V. Orlov, W. W. Chen, D. Mark Eckley, T. J. Macura, L. Shamir, E. S. Jaffe, and I. G. Goldberg, "Automatic classification of lymphoma images with transform-based global features," *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 4, pp. 1003–1013, 2010.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, pp. 84–90, may 2017.

[6] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.



(a) Correctly classify sample where the weighted count for the true label is the highest.



(b) Correctly classify sample where the weighted count for the true label is not the highest.

Fig. 8: Example of outputs using the proposed algorithm.

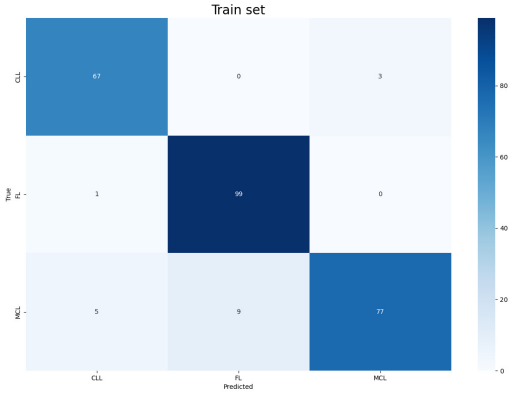
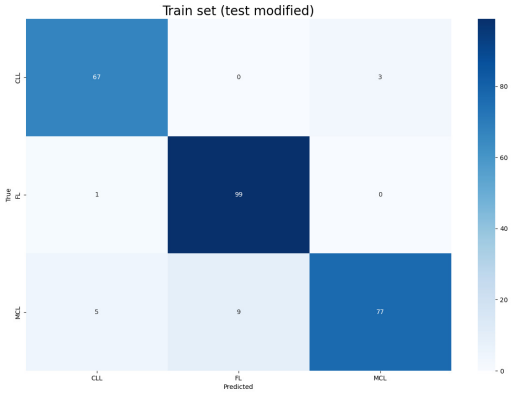
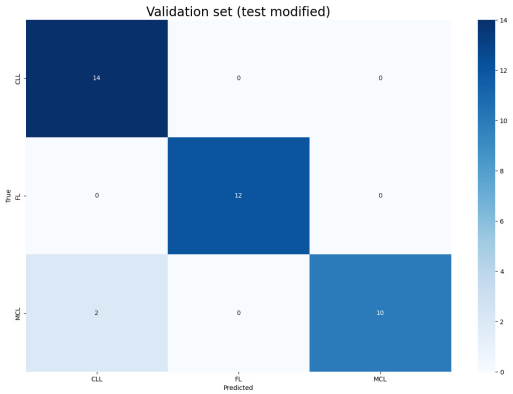


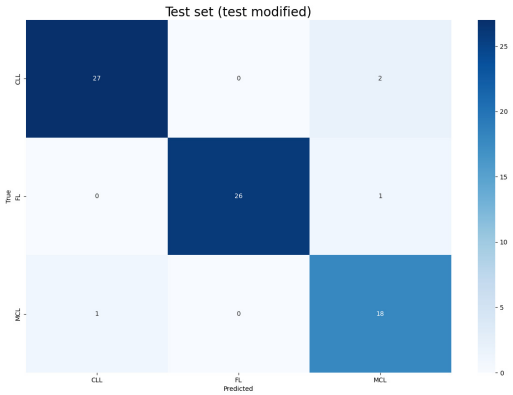
Fig. 9: Confusion matrix of the train dataset computed with the normal count.



(a) Train set.



(b) Validation set.



(c) Test set.

Fig. 10: Confusion matrixes computed with the proposed algorithm.

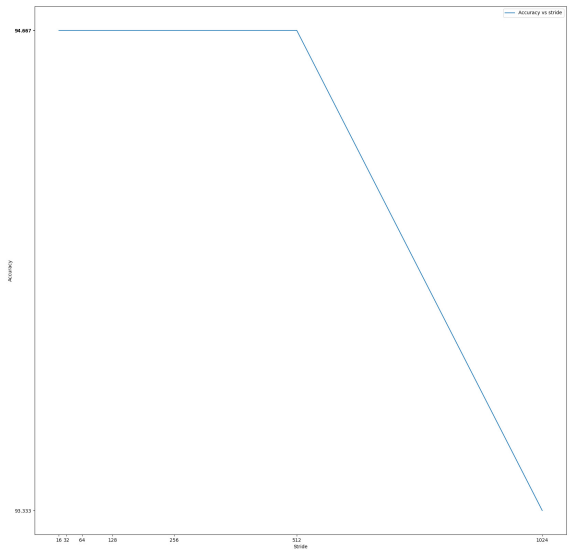


Fig. 11: Stride vs accuracy on the test dataset.