Cell-type classification from microscope imaging

Bérangère Colbois, Niko Guirlinger and Paul Juillard

***Abstract*—This paper reviews the use of Machine Learning tools applied to microscope images to investigate the interactions between tumor cells and their micro-environments which is a step towards personalized therapy for cancer. We compared traditional ML methods, both supervised and unsupervised, with deep ML. On labeled images, our models performed satisfyingly, but all generalized poorly to the mixed cell types images suggesting a lack of domain transferability.**

I. Introduction

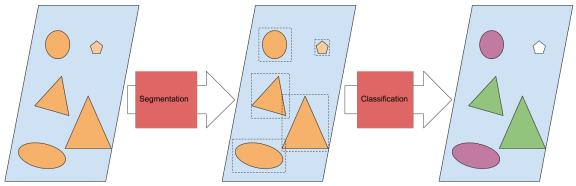

Figure 1. Full task diagram

The goal of our project is to segment and classify cells from microscopy images containing mixed cell types, namely tumoroids, fibroblasts, and macrophages. Image segmentation is the task of identifying single cells in microscope images, while image classification is the task of labeling cells by type. Both are common applications of machine learning to life sciences. These allow extracting novel biological insights from data, which is of great interest to many actors of the sector [1]. The LSCB (Laboratory of Stem Cell Biology), at EPFL, is looking to develop robust algorithms that can discriminate cells based on their type, to ultimately monitor the cell cultures in real-time. It would allow, for instance, to better understand the dynamics of the tumor microenvironment in cancer patients, which includes tumors cells as well as cancer-associated fibroblasts and macrophages.

State of the art methods for cell segmentation use Unet [2]. They generate pixel-level predictions of cell interiors and edges versus background. Cell type classification makes use of deep learning models trained on a large dataset like Imagenet [3], replacing the final layer to suit the new classification task, and retrain the model on a smaller set of annotated data [4]. This can be seen as a indicator of the difficulty to access labeled data in this domain.

II. Approach

For our task, we first need to process the input images into useful data. In our global task graph, this corresponds to the segmentation block.

We were given access to a series of gray-scale time-lapses of 3D microscopy captures. These images thus have dimensions $(time, Z, X, Y)$. We refer to a (X, Y) plane as a frame. Our input images contained either a single, or combinations of, the following three cell types: tumoroids, fibroblasts, macrophages. Each cell type has different morphological characteristics like size, convexity, or elongation, which make them distinguishable to an untrained eye. However, the quality of images and some edge cases make it hard to attain great accuracy even for experts. To achieve our goal we must first go through a step of segmentation of single cells from the original image, which can contain up to a few hundred per frame. We chose to use the Cellpose algorithm and API, based on accessibility, transparency, and previous experience [5]. Cellpose is a generalist, deep learning-based segmentation algorithm. It's Unet [2] architecture was adapted with residual blocks (skips, global skips and style indication for upsampling) [5]. The API allows to specify (and finetune!) a set of hyperparameters, namely i) the expected mean *diameter*, which can also be estimated by Cellpose; ii) the *flow threshold*, i.e., the maximum allowed error of the flows for each mask; iii) the cell probability threshold, used to run dynamics and determinate masks.

Cellpose outputs a pixel by pixel mask overlay where 0 is background and i is membership to the detected cell i. We used Cellpose as a segmentation tool for our data preparation. Segmentation is a key pillar of our project, hence it should be as good as it can be. Cellpose is not tailored to segment multiple cell types. Also, it seems to have been trained mainly on round-shaped cells and did not generalize well to fibroblasts which tend to be very elongated.

This motivated us to try and train the model further on a few images annotated by hand. This is, starting from the available pre-trained model, rerunning a few training epochs on our annotated data. This is a common approach[4]. This proved to yield relatively better segmentations, especially for images containing multiple cell types, refer to [Appendix A] for a comparative table. In addition to this segmentation (*Retrained*), we used two other segmentations as input data to provide comparison. One only uses the default parameters of Cellpose (*Baseline*), the other (*Fine Tuned*) is given cell-type-specific parameters, selected using ad hoc grid-search.

The hardness of our task lies in the inaccessibility of ground truth. Labeling this raw data is extremely time-consuming. Moreover, what is considered a 'correct' cell segmentation is not clearly defined, resulting in a lack of metrics for a quantitative assessment of segmentation performance. Nonetheless, it is possible to compare segmentations and argue on which is better than the other. The main criteria are to avoid segmentation false positives (detecting a cell where there is none), over-segmentation (a single cell detected as multiple cells), and unbalanced true positives ratio (segmentation biased by the cell-type).

In the absence of ground truth, unsupervised approaches seem like the go-to solution. However, supervised methods are known to perform much better. Willing to try both, we looked to generate labeled data to be able to train models. Using images containing a single cell type, we generated single cell cuts for which we knew the labels, as seen in the *Figure 2*. Importantly, we *trained and tested* our ML methods on these single cell images, due to the need for labelled data. The final performance is assessed qualitatively in section IV, using a small hand-labelled sample generated from multi cell-type images.

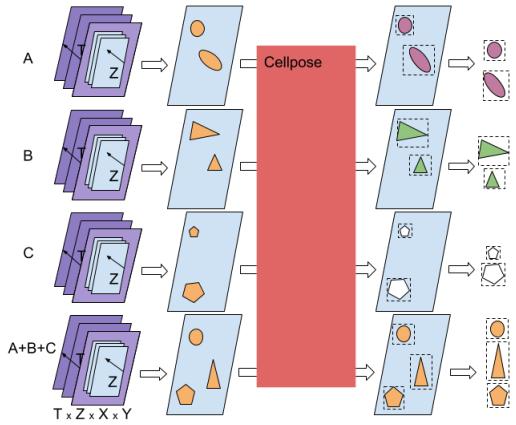

Figure 2. Raw input to training and test data diagram

III. Methods, Models

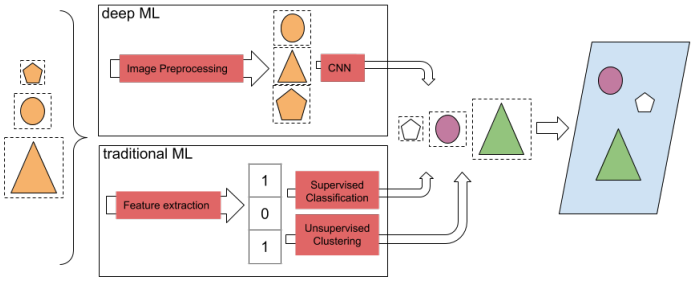

Figure 3. Single cell classification methods

One approach we take in this paper is the traditional ML approach, a.k.a. the ML before publication of [6] on ImageNet and CNN in 2012 . The pipeline encompasses three steps: data collection, feature selection, and ML method. The final performance is often only as good as the feature set, which should be chosen by a domain-expert.

We extracted 5 features for each cell, from the *mask* output by Cellpose after segmentation. The most basic yet powerful are the *area* and the *perimeter* of the cell – macrophages being significantly smaller than tumoroids. We also included 3 shape-related features, namely the *convexity*, the *solidity*, and the *compactness* to distinguish the elongated fibroblasts from the round tumoroids/macrophages. We decided not to include color-related parameters. In practice, the environment and the wide diversity in the microscope settings heavily influence coloration. The effect being cell-specific, simply removing background color is not sufficient.

We opted for a 70/30 train/test split, the large test ratio being justified by our large dataset (more than 130,000 datapoints). The features having different scales, normalization is needed. Standardization is an appropriate choice as the data is approximately normal by the central limit theorem. Having only 5 features, we are not concerned about the curse of dimensionality. However, as an optional pre-processing step, principal component analysis (PCA) was run to remove potential noise by reducing the dimensionality to 4. Moreover, we used t-SNE for 2D visualization only.

K-nearest neighbors (KNN) is a simple yet powerful machine learning method, extensively used for classification [7], with many applications in the life sciences [8]. The prediction rule on which KNN is based is straightforward:

$$f_{S_{train};k}(x) = majority\ element\{y_n : x_n \in nbh_{S_{train};k}(x)\}$$

KNN seems to be a wise choice for our classification problem, as we are working in low dimensions, with complex decision boundaries, but where an underlying spatial correlation exists. We determined the optimal number of neighbors k using 10-fold cross-validation on 9 values ranging from 2 to 512, on a logarithmic scale [9]. We considered two distance metrics, namely the Euclidean and the Manhattan distance [10].

Unsupervised ML does not require labeled data. This scenario is desirable in practice, as biologists would have to set up fewer microscopy experiments, thus saving time. Our approach here is to investigate and visually assess how unsupervised methods are behaving on our labeled data – images containing a single cell type – so that we can estimate the prospective efficiency of such methods on unlabeled, mixed cell-type images.

K-means aims at finding the cluster assignments z minimizing the distance of each datapoint x_n to the center of its cluster μ_n and is widely used in practice [11].

$$min_{z;\mu} L(z,\mu) = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\|x_n - \mu_k\|_2^2$$

We tested different numbers of clusters k on a right-skewed log-like scale (2,3,4,5,7,10), using the k-means implementation provided by the library sklearn with the default parameters [12]. One approach to select k is to visually find a significant "knee" on the Sum of Squared Errors (SSE) plot. Another approach is to maximize the silhouette score, to try to maximize inter-cluster distance while minimizing intra-cluster distance [13][14]. K-means has hard limitations. First, k needs to be given. Second, having a large number of datapoints leads to a high computational cost. Last but not least, k-means clusters are constrained to be spherical while real cell clusters are often elliptical.

In contrast to k-means, the network-based approach DBSCAN determines the number of clusters and can identify non-convex shapes. Concretely, a cluster is made of datapoints that have at least *MinPts* neighbors less than *Eps* away from them [15]. We used the default value of 5 for MinPts and decided to analyze 20 values for Eps, on a $\log_{10}$ scale, ranging from 10^{-1} to $10^{1.1}$.

A more modern approach is to use Convolutional Neural Networks (CNNs), a kind of deep neural network [16]. First, we used Cellpose to generate images for which we know the cells' labels as described in *Figure 2*. It yields single cell images of very different dimensions. We removed single cell images smaller than 20x20 pixels, which are unlikely to contain any of the types of cells we want to classify. We redimensioned the others to the standard 224x224 size [16]) through a combination of blurring, resizing, and padding, using the cv2 [17] python library. With these generated labeled single cell images, we can train and test our CNN models.

To avoid model overfitting, we performed data augmentation [18]. We used horizontal and vertical flips, zoom, as well as width and height shifts, using keras [19]. Then, we separated the

data into 3 sets: a train set (64%) to train the models, a validation set (16%) to compare methods while limiting overfitting, and a test set (20%) to obtain the final accuracy of the best model on unseen data [20].
We designed two different architectures for the CNNs: a rather simple one, inspired by the research we did on CNN for image classification, and a more complex and promising architecture inspired from the recently published Nature paper [16] [Appendix G]. We fixed the number of epochs to 50, for time and computation resources reasons. A larger number could yield better performances, but may also promote overfitting. Finally, we used keras' [19] categorical cross-entropy loss function, which performs well with unbalanced data [21], which is our case. The precise allotment can be found in [Appendix F].

IV. Results

Using traditional ML, each cell is represented by 5 features and corresponds to one row of the feature matrix. Statistics about this matrix demonstrate the potential of the selected features to predict cell type [Appendix H]. In all cases, the variance explained by the recombined 4 features exceeded 96%. It implies that the model performance will not change much by reducing the dimension from 5 to 4. It was expected as our original dimensional space was already small.
Unsupervised techniques did not succeed at distinguishing cells by cell-type. The best results we obtained with K-means were using the finetuned segmentation model, without PCA. The "knee" point is located at k=4 on the SSE plot; however, the silhouette plot suggests that k=3 is the optimal choice for the number of clusters [Appendix D]. As there are three cell types to distinguish, this result is expected.

The visual assessment using t-SNE revealed that k-means learns broad clusters but the performance is not satisfactory [Appendix D]. Indeed, the method falsely classifies numerous tumoroids as macrophages, resulting in a low recall for tumoroids. In all other datasets, the silhouette plots suggested picking k=2 as the number of clusters, which is aberrant. The computational cost associated with the DBSCAN network-based approach is too high to yield results on large datasets like ours. Nonetheless, we visually investigated the performance of DBSCAN on a sample of 10,000 datapoints, which revealed a dramatic failure [Appendix E]. The results obtained for KNN are summarized in *Figure IV*. On the training set, using cross validation, we obtained the best accuracy of 92.14% when using the Manhattan distance ($p = 1$), with $k = 16$, on the data that was segmented with the fine-tuned model, without PCA as pre-processing ($\star$). As expected, the segmentation model heavily influences the performance of the algorithm, with the best performances being achieved with the fine-tuned model. The mean accuracy of the baseline model (81.9%) is roughly 3 points larger than one of the retrained model (78.7%). It does not mean that the retrained model is less performant. The confusion arises as the retrained model identifies around 30% more fibroblasts than the baseline model, which are more complex to classify. In contrast, the baseline model over-identifies tumoroids, while the retrained model counts half of these easily-classifiable cells [Appendix H]. Interestingly, as the segmentation model gets more general (fine-tuned being the most specific and baseline being the most general), the value of k yielding the best accuracy increases.

Fine-tuned segmentation				Retrained segmentation				Baseline segmentation			
PCA	Distance	Accuracy	k	PCA	Distance	Accuracy	k	PCA	Distance	Accuracy	k
False	1	0.9214	16	False	1	0.7886	64	False	1	0.8188	128
	2	0.9209	16		2	0.7875	64		2	0.8186	128
True	1	0.9192	32	True	1	0.7864	64	True	1	0.8186	128
	2	0.9199	32		2	0.7869	64		2	0.8184	256

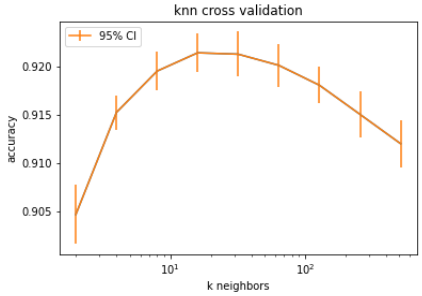

Figure 4. KNN comparative result table and cross validation for $\star$ model

This theoretically makes sense, as smoother decision boundaries are favored by less specific labels. As expected, PCA does not affect scoring much. Moreover, as our datasets perfectly fulfill the requirements to apply KNN, the method is expected to work well for most distance functions. Indeed, the distance metrics' impact is near-negligible. From the above results, we can conclude that ($\star$) is the best traditional ML model. *Figure IV* shows the cross-validation plot for this model. The error bars indicating the bootstrapped 95% confidence interval for the accuracy demonstrate that $k = 16$ truly is the right choice for the number of neighbors. If k is selected to be too small, the model will tend to overfit the training set, leading to unreasonably spiky and very training-set specific decision boundaries. On the other hand, if k is too large, the training set will most probably be underfitted and the model will not learn the prediction rules well. The accuracy for the ($\star$) model on the test set is 92.3%. One can visually appreciate how well KNN learns the clusters, corresponding to the three cell types, on the 2D t-SNE plot, in *Figure 5*.

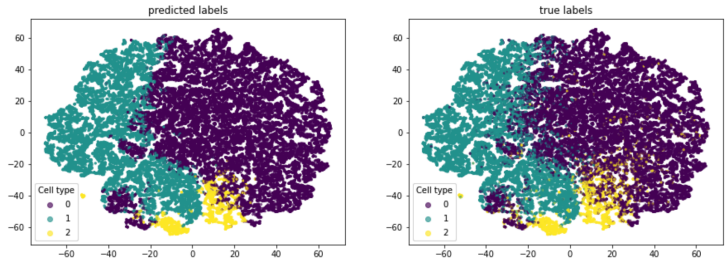

Figure 5. T-SNE KNN test visualization

Finally, we computed the contingency matrix [Appendix C] and in turn, deducted the precision, recall and f1-score for each class [Table I].

	precision	recall	f1-score
macrophages	0.933	0.916	0.925
tumoroids	0.920	0.961	0.940
fibroblasts	0.884	0.600	0.715

Table I
Precisions, recalls and f1-scores for the 3 cell types - KNN

For CNN models, the complete table of results can be found in VIII. The best model is the CNN inspired by [16] referenced as "complex", using the retrained Cellpose segmentation, with data augmentation. The best model performed well on unseen data, with a loss of 1.33 an accuracy of 77.8% We also computed its precision, recall and f1-score [Table II].

	precision	recall	f1-score
macrophages	0.65	0.92	0.76
tumoroids	0.88	0.68	0.77
fibroblasts	0.96	0.74	0.83

Table II
PRECISIONS, RECALLS AND F1-SCORES FOR THE 3 CELL TYPES - CNN

V. Results Discussion

From the above results, it is clear that the unsupervised approach we took is not appropriate for distinguishing cells based on their type. K-means yields biologically irrelevant results, and DBSCAN cannot be properly applied to our large datasets.
Overall, the ($\star$) model is doing well at classifying the cells, using images containing only one cell type, according to the f-score in [Table I]. However, the performance is cell-type dependent. Indeed, from all cells that actually are fibroblasts, the best KNN method merely identified 60%. This was to be expected as these cells are difficult to segment and are under-represented in the dataset. Importantly, fibroblasts are not the cancer-promoting cells nor the main fighter in our body. Thus, false negatives on fibroblast classification is not too impactful. What is extremely biologically relevant is the recall value for tumoroids (96.1%), as we do not want to under-estimate the presence of tumors.
The complex CNN model performs overall better than the basic one, in the same conditions - namely data, epoch number, and loss function. As the model has more parameters, it is more expressive, and thus performs better, if we can avoid overfits. In the absence of data augmentation, the accuracy on the train set is significantly larger than the one on the validation set, suggesting overfitting. This undesirable effect disappeared with data augmentation [18]. The impact of the segmentation method seems not to be significant or at least not powerful enough to overweight the other parameters. We can still notice that the baseline segmentation slightly performs worse overall. The best accuracy is higher than 77.81% on unseen images from this dataset.
Now that we have tested our two best models on unseen data coming from the same kind of images as the train and validation set, we need to see how well they perform on the images containing the three cell types, our project's goal.
Using retrained Cellpose segmentation yielded the best results, with the following accuracies: 0.596 for the KNN model, and 0.481 for the CNN model. The visualization of these classifications can be found in the [Appendix B]. Those results are by far not as good as the one on the test set, suggesting that our models do not perform well at domain adaptation.
For the CNN, this suggests that the model also learnt irrelevant domain features, such as the color of the background or the size of the borders. There are several ways to address this problem: either by eliminating domain features, but that seems difficult, or by training the model on more diverse domains, so that it is forced to only learn the cell characteristics. Another solution could be to use the model only on the same domain we trained the model on, but that would be really difficult to put into practice in the case of microbiology images, and it would cause the use of the model to be very limited.
The main limitation of the KNN method is its large memory requirement and computation complexity. Moreover, noisy or unrelated features are known to heavily impact the model performance [22]. However, we previously took these limitations into account, and they fail to explain the difficulty of KNN to generalize to multi-cell type images.
As a result, we suggest that the origin of the problem may lie in the quality of input data; above all, the segmentation may well be the crux of the issue.

VI. Project Discussion

Facing new file formats, in particular the images ones, such that .nd2, made the generation of suitable input data difficult. Training our models on data that are from different domains than the one we were trying to achieves segmenting and achieving made the process really difficult. Another challenge is caused by the large biases because life science raw data is tailored for easy comparisons for an expert-eye. This hampers the generalization capacity of the learning process. The main challenge we encountered was the nonexistence of real metric to assess our final performances, as there is no ground-truth for the segmentation and classification of the three types of cells. This caused the assessment to be qualitative instead of quantitative. Our approach is made more robust by the wide range or different solutions attempted, that all have strengths and weaknesses.
As said before, our models do not generalize well, because of domains adaptability problems. This causes an important lack of robustness and generalization capacity. Also, cutting the original image and classifying on single cells independently implies that we lose positional and relative positioning information by going through the pipeline, which might be useful for finer analysis.

VII. Summary

Starting with microscopy images as input, we divided the data in a labeled dataset - images containing cells of a single cell type – and an unlabeled dataset – images containing cells of different cell types. Our approach was to use the labeled dataset to estimate the performance and quantitatively compare traditional ML methods (KNN, K-means, DBSCAN) with a more modern, deep ML approach (CNN). The best traditional model (a KNN) and the best CNN achieved a test accuracy of 92% and 77%, respectively. Then, the final performance was assessed qualitatively, using a small hand-labeled sample of the unlabeled dataset, i.e. images containing cells of different cell-types. The obtained accuracies – 59.6% and 48.1% for the KNN and the CNN model, respectively – suggest that the models need to be improved to become domain transferable.

VIII. Acknowledgments

We would like to thanks Prof. Matthias Lütolf, head of the LSCB, as well as Hwanseok Jang and Nicolas Broguiere, also from LSCB, for the project, the data and their supervision. Thank you to Mohamed Ridha Chahed for his precious help and advice as a Machine Learning student assistant.

APPENDIX

A : RETRAINED CELLPOSE MODEL PERFORMANCE COMPARISON

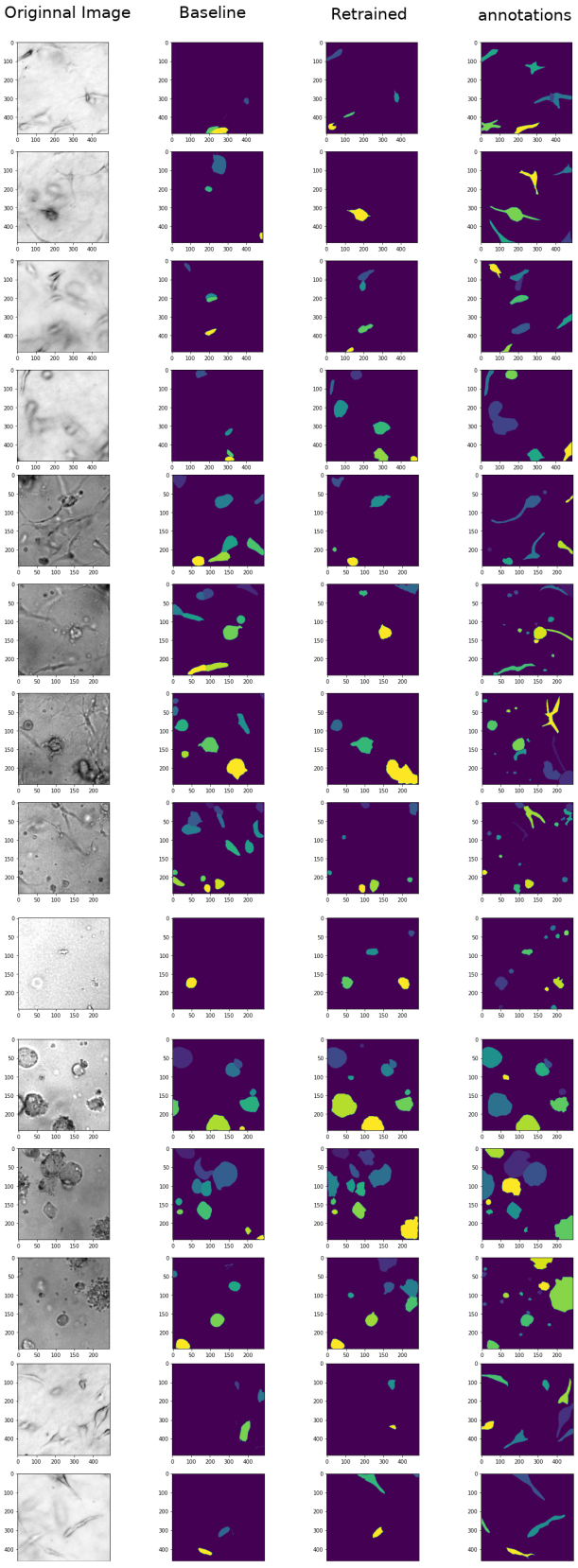

Figure 6. Visualization of the Cellpose model Retraining

B : PREDICTION PERFORMANCE VISUALIZATION

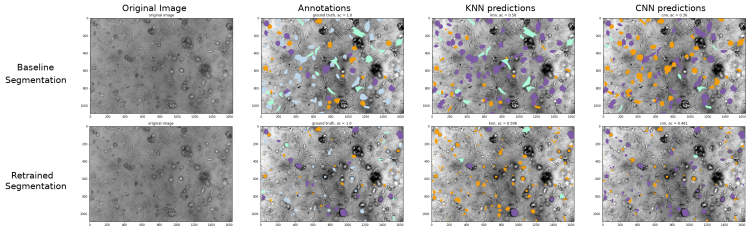

Figure 7. Visualization of the final results

C : CONTINGENCY MATRIX FOR THE BEST TRADITIONAL ML METHOD

The best traditional ML method which was selected is KNN using the manhattan distance (p=1) and k=16, without PCA, and using the fine-tuned model for segmentation. On Figure 8, you can find the contingency matrix for this model, on the test set.

real	tum	mac	fibro
pred			
tum	28989	1380	1128
mac	981	15547	138
fibro	210	39	1896

Figure 8. Contingency matrix for the method ($\star$)

D : K-MEANS: SSE PLOT - SILHOUETTE PLOT - T-SNE VISUALIZATION

The best results we obtained with K-means were using the fine-tuned segmentation model, without PCA. The "knee" point is located at k=4 on the SSE plot for this model (Figure 9). However, the silhouette plot suggests that k=3 is the optimal choice for the number of clusters (Figure 10).

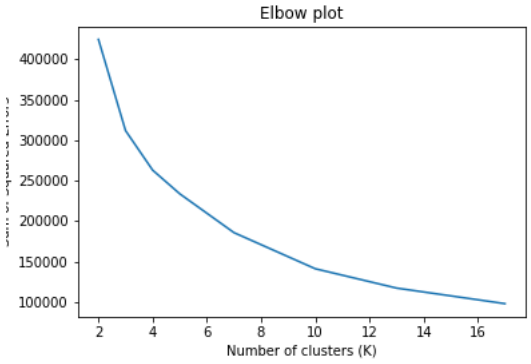

Figure 9. Knee plot for k-means

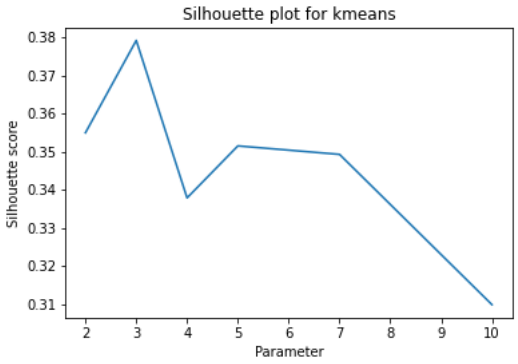

Figure 10. Silhouette plot for k-means

On Figure 11, one can appreciate the qualitative performance of k-means. We use t-SNE to represent the feature matrix in 2D. The left panel corresponds to the prediction made by the model (k-means). The right panel depicts the true labels of every cell. Tumoroids, macrophages, and fibroblasts are depicted by purple (0), green (1) , and yellow (2) dots respectively. The model learns broad clusters but the performance clearly is not satisfactory. Indeed, the method falsely classifies numerous tumoroids as macrophages, resulting in a low recall for tumoroids. That is particularly dramatic biologically speaking, as tumoroids promote cancer progression, while macrophages fight tumors.

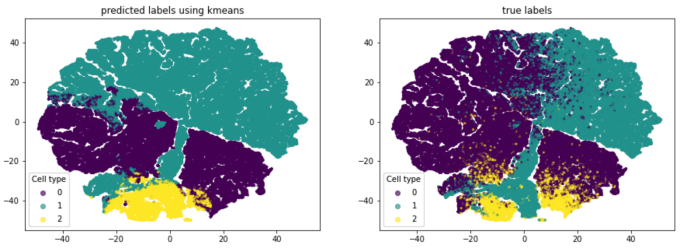

Figure 11. Comparison of true labels and clusters learnt by k-means

E : DBSCAN: SILHOUETTE PLOT - T-SNE VISUALIZATION

We visually investigated the performance of DBSCAN on a sample. The results were obtained by sampling 10,000 points from the dataset using the best segmentation (fine-tuned) and PCA to boost the computation. Figure 12 shows that the silhouette score is maximized for eps values between 2 and 10.

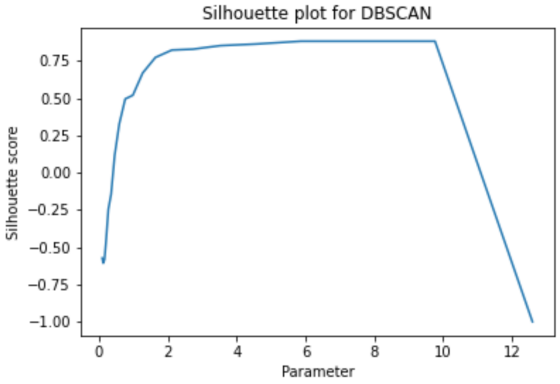

Figure 12. Silhouette plot for DBSCAN

However, the clustering failed and grouped nearly all cells in one cluster. We visualized this in 2D on Figure 13, using tSNE.

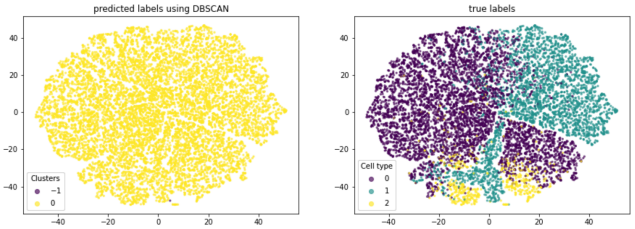

Figure 13. Comparison of true labels and clusters learnt by DBSCAN

F : CNN RESULT TABLES

Data distribution for the CNN training set	baseline	finetuned	retrained
macrophages	296	500	500
tumoroids	500	500	500
fibroplasts	208	500	329

Table III
DATA DISTRIBUTION - CNN TRAINING SET

CNN	**Segmentation**	**Aug.**	**Loss**	**Acc.**	**Val loss**	**Val acc**
Basic	Baseline	No	0.061	0.9844	3.0236	0.5466
Basic	Baseline	Yes	0.8603	0.5639	0.9134	0.5031
Basic	Finetuned	No	0.0111	0.9979	3.0373	0.5583
Basic	Finetuned	Yes	0.8077	0.6010	0.7883	0.6333
Basic	Retrained	No	0.0057	1.0000	2.9323	0.6620
Basic	Retrained	Yes	0.8992	0.5918	0.8768	0.6009
Complex	Baseline	No	0.1014	0.9611	1.1324	0.7081
Complex	Baseline	Yes	0.6758	0.6620	1.4235	0.5155
Complex	Finetuned	No	0.0646	0.9771	1.2641	0.7500
Complex	Finetuned	Yes	0.4756	0.7854	0.7396	0.7125
Complex	Retrained	No	0.0304	0.9882	3.1403	0.5728
Complex	**Retrained**	**Yes**	**0.0330**	**0.9906**	**1.5704**	**0.8075**

Table IV
TABLE OF CNN MODELS PERFORMANCE, WITH SPECIFIED SEGMENTATION TYPE AND AUGMENTATION (AUG.). IT SHOWS LOSS, ACCURACY (ACC.), VALIDATION LOSS (VAL LOSS) AND VALIDATION ACCURACY (VAL ACC.)

G : CNN ARCHITECTURES

We used two different architectures for the CNN, one rather simple,

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 242, 242, 32)      320
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 240, 240, 64)      18496
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 120, 120, 64)      0
_________________________________________________________________
dropout (Dropout)            (None, 120, 120, 64)      0
_________________________________________________________________
flatten (Flatten)            (None, 921600)            0
_________________________________________________________________
dense (Dense)                (None, 128)               117964928
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 387
=================================================================
Total params: 117,984,131
Trainable params: 117,984,131
Non-trainable params: 0
_________________________________________________________________
```

Figure 14. Simple Model Architecture

and one more complex, inspired by this paper [16] :

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 242, 242, 32)      320
_________________________________________________________________
batch_normalization_5 (Batch (None, 242, 242, 32)      128
_________________________________________________________________
average_pooling2d_5 (Average (None, 121, 121, 32)      0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 119, 119, 64)      18496
_________________________________________________________________
batch_normalization_6 (Batch (None, 119, 119, 64)      256
_________________________________________________________________
average_pooling2d_6 (Average (None, 59, 59, 64)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 57, 57, 64)        36928
_________________________________________________________________
batch_normalization_7 (Batch (None, 57, 57, 64)        256
_________________________________________________________________
average_pooling2d_7 (Average (None, 28, 28, 64)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 26, 26, 64)        36928
_________________________________________________________________
batch_normalization_8 (Batch (None, 26, 26, 64)        256
_________________________________________________________________
average_pooling2d_8 (Average (None, 13, 13, 64)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 11, 11, 64)        36928
_________________________________________________________________
batch_normalization_9 (Batch (None, 11, 11, 64)        256
_________________________________________________________________
average_pooling2d_9 (Average (None, 5, 5, 64)          0
_________________________________________________________________
dropout_4 (Dropout)          (None, 5, 5, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               204928
_________________________________________________________________
dropout_5 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 3)                 387
=================================================================
Total params: 336,067
Trainable params: 335,491
Non-trainable params: 576
_________________________________________________________________
```

Figure 15. Complex Model Architecture

H : STATS BY CELL TYPES FOR EACH SEGMENTATION TYPE

To investigate the potential of the features we selected to predict the cell type, we compute some statistics from the feature matrix. The figures 16, 17, and 18 show, for each segmentation type (baseline, finetuned, retrained, respectively), the mean, standard deviation, first and third quartiles of each feature, grouped by cell type. For instance, it allows us to notice i) the striking difference in the area distribution between tumoroids and macrophages – the mean size of tumoroids being at least twice as large as the mean size of macrophages, for each segmentation type ii) the moderate gap between fibroblasts and the other cell types in terms of solidity. The number of cell each segmentation model found is also included in the table under to column *count*. It enables us to determinate bias of the segmentation models for different cell types.

	count	area				convexity				solidity				compactness				perimeter			
		mean	std	25%	75%	mean	std	25%	75%	mean	std	25%	75%	mean	std	25%	75%	mean	std	25%	75%
cell_type																					
fibro	3369.0	1511.904	1325.638	984.0	1722.0	0.933	0.096	0.933	0.953	0.871	0.211	0.905	0.954	0.837	0.203	0.812	0.956	162.640	72.555	130.468	188.024
mac	29294.0	614.297	370.213	344.0	812.0	0.949	0.028	0.946	0.963	0.948	0.031	0.943	0.965	0.981	0.032	0.980	0.996	90.095	28.915	68.527	110.083
tum	139808.0	1261.665	872.277	607.0	1721.0	0.949	0.027	0.944	0.959	0.952	0.046	0.946	0.971	0.977	0.045	0.975	0.996	129.450	47.968	93.355	160.811

Figure 16. Feature matrix statistics using the baseline segmentation model

	count	area				convexity				solidity				compactness				perimeter			
		mean	std	25%	75%	mean	std	25%	75%	mean	std	25%	75%	mean	std	25%	75%	mean	std	25%	75%
cell_type																					
fibro	10730.0	5767.865	5616.569	2780.25	6929.75	0.829	0.050	0.810	0.855	0.840	0.114	0.809	0.907	0.854	0.126	0.796	0.945	381.214	163.343	278.452	449.227
mac	56987.0	1077.698	670.328	546.00	1538.00	0.898	0.045	0.892	0.921	0.915	0.069	0.906	0.953	0.968	0.049	0.963	0.995	128.290	42.265	96.527	160.083
tum	99975.0	3989.529	2227.234	2358.00	5110.00	0.864	0.021	0.850	0.876	0.938	0.022	0.929	0.952	0.974	0.036	0.972	0.994	264.617	78.451	208.770	311.983

Figure 17. Feature matrix statistics using the fine-tuned segmentation model

	count	area				convexity				solidity				compactness				perimeter			
		mean	std	25%	75%	mean	std	25%	75%	mean	std	25%	75%	mean	std	25%	75%	mean	std	25%	75%
cell_type																					
fibro	4347.0	1263.735	782.390	738.5	1602.5	0.930	0.034	0.921	0.946	0.898	0.090	0.880	0.949	0.887	0.107	0.830	0.969	156.216	64.577	112.083	192.024
mac	60493.0	435.506	238.121	292.0	529.0	0.947	0.021	0.939	0.959	0.930	0.035	0.920	0.950	0.965	0.038	0.955	0.990	78.735	23.436	64.627	90.426
tum	71089.0	1411.721	1284.691	469.0	1985.0	0.938	0.034	0.931	0.954	0.938	0.051	0.928	0.963	0.976	0.046	0.973	0.994	134.414	70.252	82.284	177.983

Figure 18. Feature matrix statistics using the retrained segmentation model

REFERENCES

[1] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[2] T. Falk, D. Mai, R. Bensch, Ö. Çiçek, A. Abdulkadir, Y. Marrakchi, A. Böhm, J. Deubner, Z. Jäckel, K. Seiwald *et al.*, "U-net: deep learning for cell counting, detection, and morphometry," *Nature methods*, vol. 16, no. 1, pp. 67–70, 2019.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[4] E. Moen, D. Bannon, T. Kudo, W. Graf, M. Covert, and D. Van Valen, "Deep learning for cellular image analysis," *Nature methods*, pp. 1–14, 2019.

[5] C. Stringer, T. Wang, M. Michaeloas, and M. Pachitariu, "Cellpose: a generalist algorithm for cellular segmentation," *bioRxiv preprint server*.

[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] D. J. Hand and N. M. Adams, "Data mining," *Wiley StatsRef: Statistics Reference Online*, pp. 1–7, 2014.

[8] S. Begum, D. Chakraborty, and R. Sarkar, "Data classification using feature selection and knn machine learning approach," in *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*. IEEE, 2015, pp. 811–814.

[9] V. Barral, C. J. Escudero, J. A. García-Naya, and P. Suárez-Casal, "Environmental cross-validation of nlos machine learning classification/mitigation with low-cost uwb positioning systems," *Sensors*, vol. 19, no. 24, p. 5438, 2019.

[10] L. Yang and R. Jin, "Distance metric learning: A comprehensive survey," *Michigan State Universiy*, vol. 2, no. 2, p. 4, 2006.

[11] M. Mahajan, P. Nimbhorkar, and K. Varadarajan, "The planar k-means problem is np-hard," in *International Workshop on Algorithms and Computation*. Springer, 2009, pp. 274–285.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[13] T. Thinsungnoena, N. Kaoungkub, P. Durongdumronchaib, K. Kerdprasopb, and N. Kerdprasopb, "The clustering validity with silhouette and sum of squared errors," *learning*, vol. 3, no. 7, 2015.

[14] T. M. Kodinariya and P. R. Makwana, "Review on determining number of cluster in k-means clustering," *International Journal*, vol. 1, no. 6, pp. 90–95, 2013.

[15] K. Khan, S. U. Rehman, K. Aziz, S. Fong, and S. Sarasvady, "Dbscan: Past, present and future," in *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*. IEEE, 2014, pp. 232–238.

[16] K. Yao, N. D. Rochman, and S. X. Sun, "Cell type classification and unsupervised morphological phenotyping from low-resolution images using deep learning," *Scientific reports*, vol. 9, no. 1, pp. 1–13, 2019.

[17] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[18] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019.

[19] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[20] Y. Xu and R. Goodacre, "On splitting training and validation set: A comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning," *Journal of Analysis and Testing*, vol. 2, no. 3, pp. 249–262, 2018.

[21] A. . M. L. E. George Seif. (2018) Handling imbalanced datasets in deep learning. [Online]. Available: https://www.kdnuggets.com/2018/12/handling-imbalanced-datasets-deep-learning.html

[22] K. Machhale, H. B. Nandpuru, V. Kapur, and L. Kosta, "Mri brain cancer classification using hybrid classifier (svm-knn)," in *2015 International Conference on Industrial Instrumentation and Control (ICIC)*. IEEE, 2015, pp. 60–65.