RECOGNITION OF ANTINUCLEAR ANTIBODY PATTERNS USING MACHINE LEARNING

PROJET DE RECHERCHE ET D'INNOVATION MASTER (PRIM) - FINAL REPORT

Ramon G. B. Ribeiro
Data Science
Télécom Paris
ramon.ribeiro@telecom-paris.fr

Gabriel S. P. Medeiros
Artificial Intelligence
Télécom Paris
gasouza@telecom-paris.fr

France, January, 2022

Submitted in partial fulfilment of the requirements for an engineer's degree.
Supervised by:
Daniel Stockholm - daniel.stockholm@ephe.psl.eu
Elisabeth Brunet - elisabeth.brunet@telecom-sudparis.eu

ABSTRACT

Autoimmune diseases are an important field of medicine and sometimes it is necessary to identify cellular patterns to correctly diagnose an illness. This project aims to create a network that classifies specific antibody patterns for those kinds of diseases. Initially some types of image processing techniques were applied, which in hand with a cellular segmentation package called Cellpose aimed to automatically identify and segment every cell in a given image, using a U-net variant. Those cells were pre-processed and given to a neural network to train it. With it, a path for inputting and classifying external images was made.

1 Context

Multi-systemic autoimmune ailments are numerous but also very diverse. As such, it is essential to accurately diagnose the type of antibodies observed in any given case. Allied to the diagnostics reached by a medical professional after performing a clinical evaluation, this information is crucial to accurately reach a correct diagnosis of the pathology in order to put in place a patient specific therapeutic response. Thus, it is imperative to have a fast, but also robust automatic classification system in place to aid medical professionals achieve their conclusions as quickly as possible.

With the aid of Dr. Makoto Miyara from the “Centre d’Immunologie et des Maladies Infectieuses", this project’s objective is to develop a system capable of automatically classifying different antibody autoimmune patterns. These images should be collected automatically from an optical system integrated to a microscope.

2 Methodology

2.1 Database and Pre-existing work

Initially, we were given two folders: “patterns” and “HEP Annot”. Which contained all images originally given to us. The first of which consisted of fourteen different unlabelled patterns each containing from four to five images hand picked by medical student-intern Florian Jonard with technical criteria such as observability and relevance in mind. Accompanying each unlabelled pattern was a folder named “masks” which was supposed to contain a mask for every image, which could be used to isolate individual cells. Unfortunately, due to some sort of storage error, or faulty script these masks were empty.

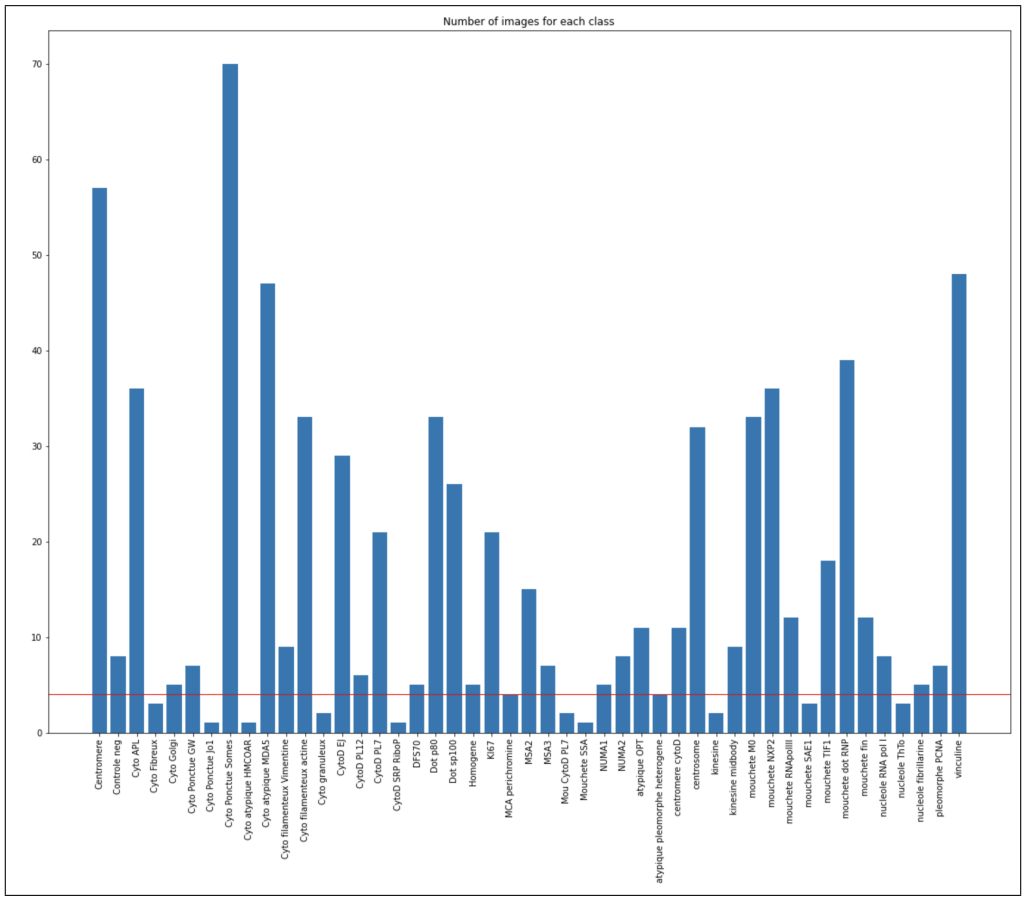


Figure 1: Number of images in HEP Annot.

The second folder, “HEP Annot”, contained ten sub-folders, each with their own subdivisions totalling 47 different subcategories. Each of which contained a largely varying number of images, ranging from just one to seventy. Clearly

illustrated in figure 1. Apart from images, we also received a rough draft from Professor Stockholm for a function that applies the individual masks to each image in order to generate individual cells.

Towards the end of the project, we received a hard drive from Mr. Jonard containing a new batch of images, these images are not only more numerous than our other two previous datasets, but also larger, each image being four times as large as our previous ones. However, unlike the previous datasets, these images weren't neatly separated into exclusionary folders for each type of pattern, but were sorted by collection date, with a folder for each date accompanied by a csv file containing the image information. The distribution of images through each one of its classes can be observed in figure 2. However, problem with this new batch of data is that some images were classified as pertaining to multiple classes which entailed a different classification problem than our other datasets.

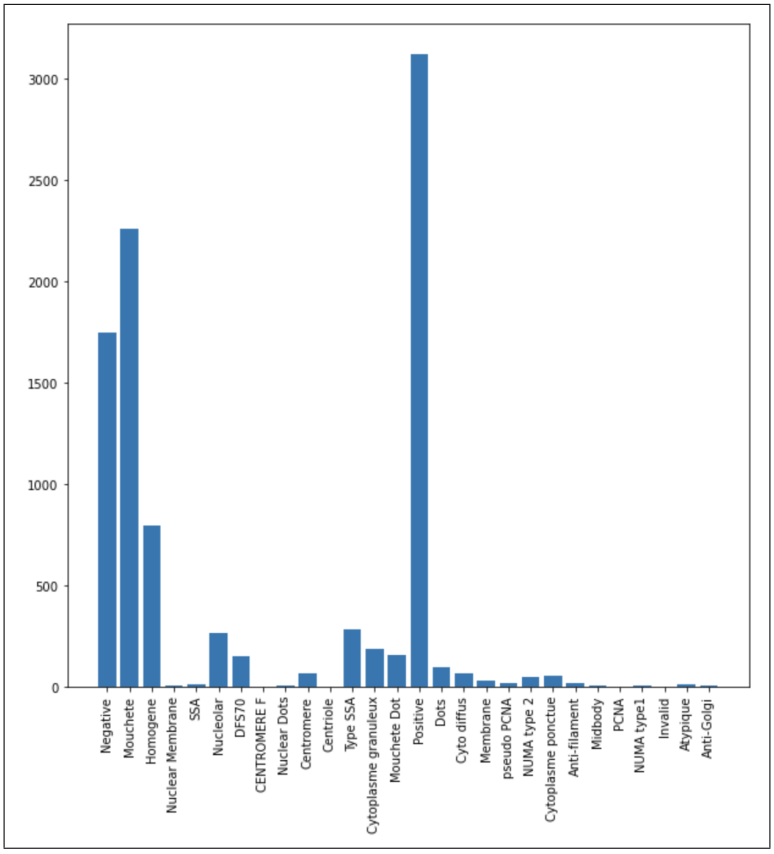


Figure 2: Distribution of images received with final hard drive

For the sake of clarity, in the first two datasets, "patterns" and "HEP Annot", since each image is preemptively divided into different folders, each pattern is necessarily exclusive. Which means that if an image is of one class, it must not be of another. In machine learning, the optimal approach for these problems is called multi-class classification, and it trains the model to choose the most appropriate class for a given object. However, for the final dataset, celular patterns were not exclusive. So a model should be able to classify if a given image belongs to each of the existing classes or not, irrespective of its conclusions for other classes. Which, in turn, implies the construction of a different model, one of multi-label classification.

2.2 Cellpose and Image Processing

Initially we were instructed to begin our project by finishing the implementation of Professor Stockholm's existing code that applied the masks to the existing images. However, it was quickly noted that obtaining the missing masks should be our actual starting point, this process proved to be far more challenging than expected. During the project's initial phases, Cellpose was inconsistent as it was going through internal updates, which would cause lasting bugs in our program. Once Cellpose finished updating itself, we were able to proceed more smoothly. After solving all existing bugs and finally exporting a list of cells, Professor Stockholm noted that our segmentation was insufficient as Cellpose was often segmenting a single cell into multiple different parts, as can be observed in figure 2. This occurred because

Cellpose would try to automatically estimate the cellular diameters for each image it was given, a function that does not work properly yet. Thus, we set out to manually find a "one size fits all" diameter. The value eventually settled upon was 75 px.

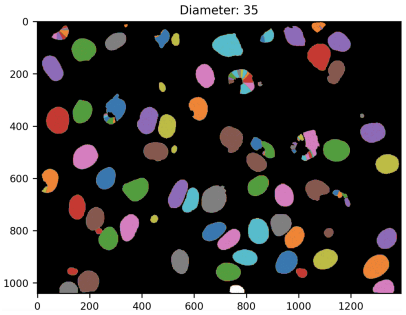


Figure 3: Cellpose output for diameter 0.35

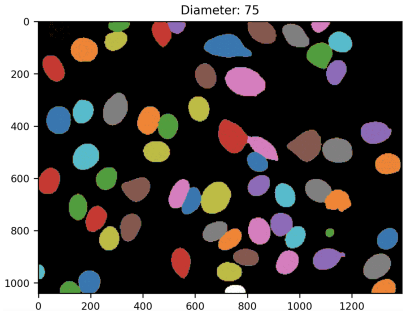


Figure 4: Cellpose output for diameter 0.75

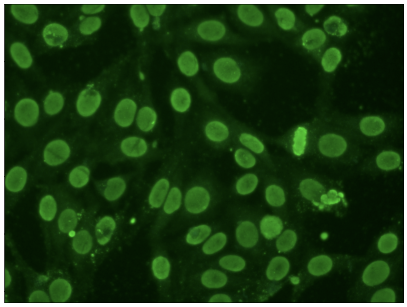


Figure 5: Original celular image

In parallel, as suggested by Professor Stockholm we investigated image processing techniques to aid Cellpose segmentation and the incoming convolutional neural network. For the segmentation, we first tried looking into histogram normalization techniques with the idea of enhancing contrast in order to make cell borders more defined and identifiable. The ones used were: contrast stretching, histogram equalization and adaptive histogram equalization. The foremost consists of a simple normalization technique. Histogram equalization is a technique that tries to "straighten out" an image's histogram, thus enhancing contrast. While the latter divides the image and apples equalization to its regions, enhancing local contrast. When compared to the original image, as can be observed in figure 5, all three methods generated many noticeable luminous dots, this is because enhancing contrast generally comes with the cost of noise amplification.

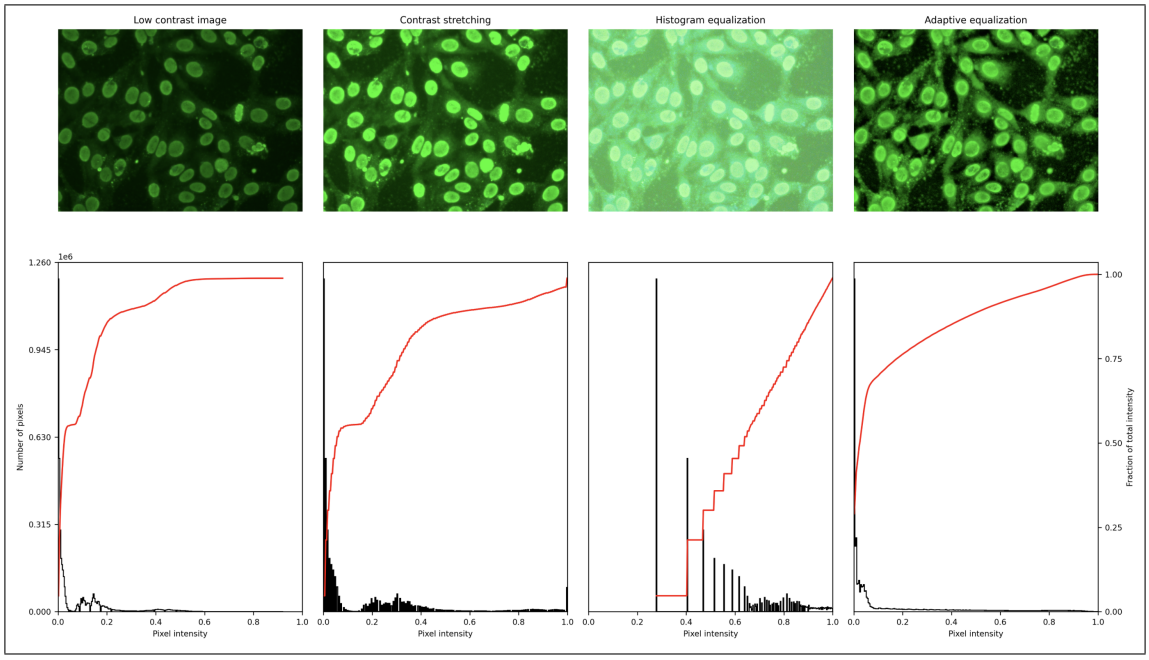


Figure 6: Contrast enhancing techniques

Subsequently, it was decided to use Otsu thresholds. Otsu's method is based on finding a threshold to split the image into two separate segments, based simply on their grey-level values. Since the maximum number of grey-levels is finite (and small: 256), we can use an exhaustive search.

The grey-level threshold is the one that minimises the following criterion :

$$argmin_\tau \ \omega_0(\tau)\sigma_0(\tau) + \omega_1(\tau)\sigma_1(t)$$

where $\sigma_0(\tau)$ and $\sigma_1(\tau)$ are the variances of the pixels in the first and second regions, $\omega_0(\tau)$ and $\omega_1(\tau)$ are weights of the first and second regions

and

$$\omega_0(\tau) = \sum_{p\in\Omega} \mathbb{1}_\tau(I(p)), \qquad \omega_1(\tau) = \sum_{p\in\Omega} 1 - \mathbb{1}_\tau(I(p))$$

and

$$\mathbb{1}_\tau(I(p)) = \begin{cases} 1 & \text{if } I(p) \leq \tau \\ 0 & \text{otherwise} \end{cases}$$

Then, in order to enhance its effects we also applied morphological erosion. The end result, as can be observed in figure 6, was that some of the noise was indeed erased, but most of the cells' cytoplasm was erased too.

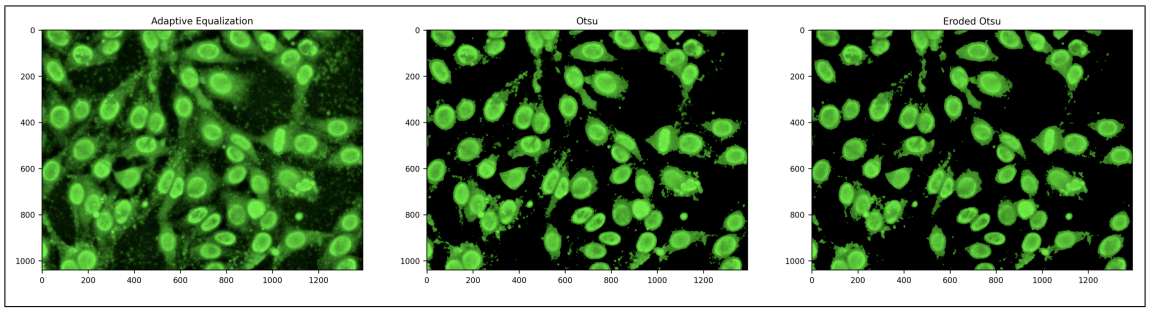


Figure 7: Otsu threshold results

In an attempt to capture entire cells, Cellpose responded best to the untreated image, rendering useless our image processing efforts, as seen in figure 6. Thus, we concluded that the cells must be normalized only after segmentation without trying to denoise and that the best diameter for individual cells is 75 pixels. After applying these segmentation parameters and manually observing each image contained in the "patterns" folder we also concluded that overall good segmentation was achieved, which despite not being perfect, seemed to yield reasonably good results. However, Cellpose presents difficulty in distinguishing the cell's nuclei from their cytoplasm, often segmenting only their nucleus despite being tasked with the segmentation of their cytoplasm. This is because Cellpose is built to expect different coloured dye for each cellular region, and the images collected at CIMI are entirely contained in the green channel.

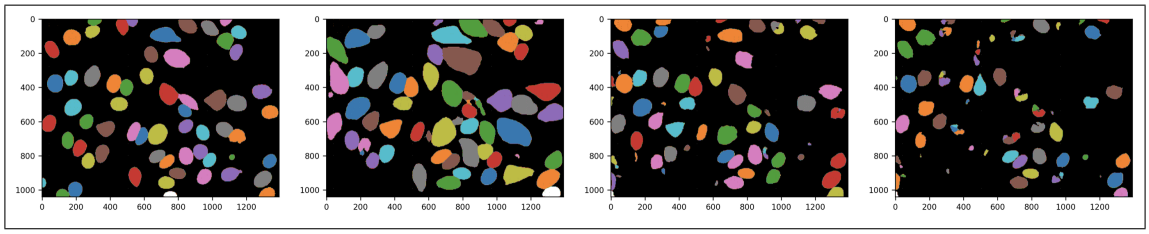


Figure 8: Cellpose segmentation from left to right: Untreated, Adaptive Equalization, Otsu, Eroded Otsu

Inspired by this realisation, we decided to investigate the possibility of automatically dyeing each cell's cytoplasm with a different colour in order to improve classification. Which would require a total reworking of all previous efforts.

Firstly, we aimed to create a mask containing only celular nuclei. After some manual testing with cellpose, we arrived at the conclusion that the ideal parameter for nucleic diameter was around 60 pixels. With this value, Cellpose would successfully isolate most of "HEP Annot"'s patterns' nuclei, struggling only with a few fibrous cytoplasmic patterns. In figures 9 and 10 below the differing quality of estimates can be observed, a good estimation is reached for the centromere pattern while a poor prediction was achieved for pattern containing fibrous cytoplasm.

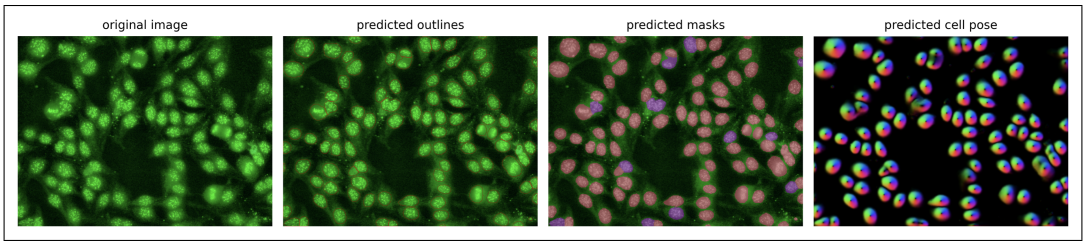


Figure 9: Good nucleic segmentation achieved for Centromere pattern

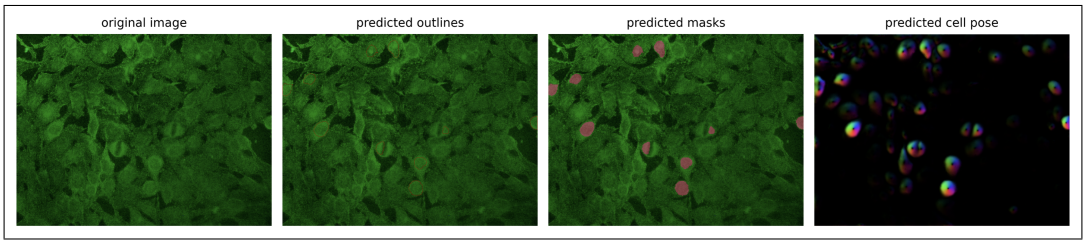


Figure 10: Poor nucleic segmentation for fibrous cytoplasm pattern

With the newly obtained nucleus mask, it is possible to isolate the cytoplasm from the rest of the image. Thus, for every image two other images were created, one that contained only the nuclei, and another that contained the cytoplasm and empty space. These images were then normalised and subjected to adaptive histogram equalisation because when compared to the images used for training Cellpose, our images exhibit very low contrast and low grayscale levels. We tried some other denoising techniques, such as the Otsu threshold again, but there were no noticeable advantages, so these efforts were abandoned. Lastly, they were reverted back to the same image, except this time occupying different channels. This new image was passed through Cellpose once more with a manually determined cellular diameter of 145 pixels for cells of nucleus size larger than 90 pixels. For cells with smaller nuclei, this value seemed inadequate, so we multiply the expected 145 pixels by that image's average nucleus size and divide by the total average nucleus size. The resulting segmentation can be observed in figures 11 and 12.

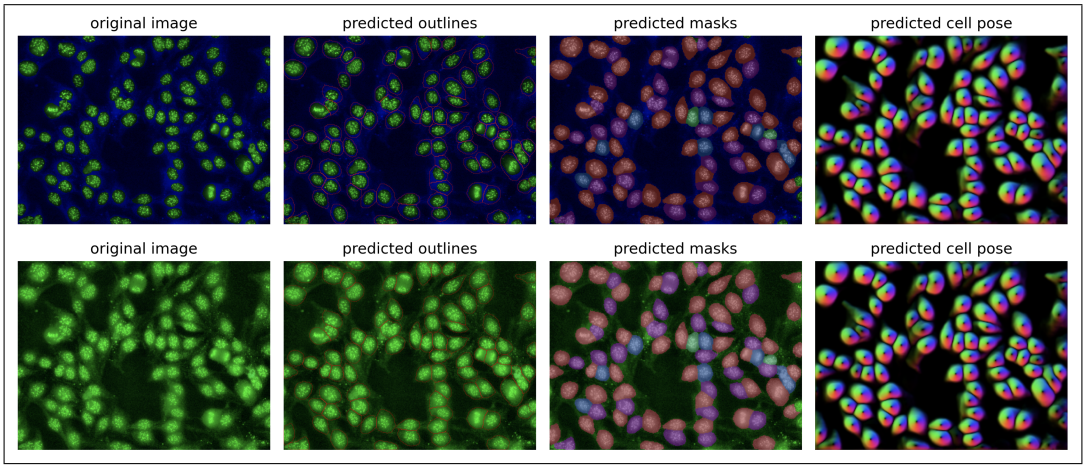


Figure 11: Final cellular segmentation for centromere pattern

The final segmentation obtained was overwhelmingly positive, and would very consistently capture the surrounding cytoplasm around the nucleus. Which would allow the following neural network to learn from the cytoplasm as well. A major improvement can also be observed in the patterns that were the most wrongly segmented, such as the one for fibrous cytoplasm. While not perfect, it managed to segment some cells, a great improvement when compared to the cell parts it would previously isolate. These smaller parts were a major contributing factor to miss classifications with earlier models, therefore, this cytoplasm detection advancement was useful not only because it provided the model with new consistent cytoplasmic information, but also because it was able to decrease instances where the model would learn

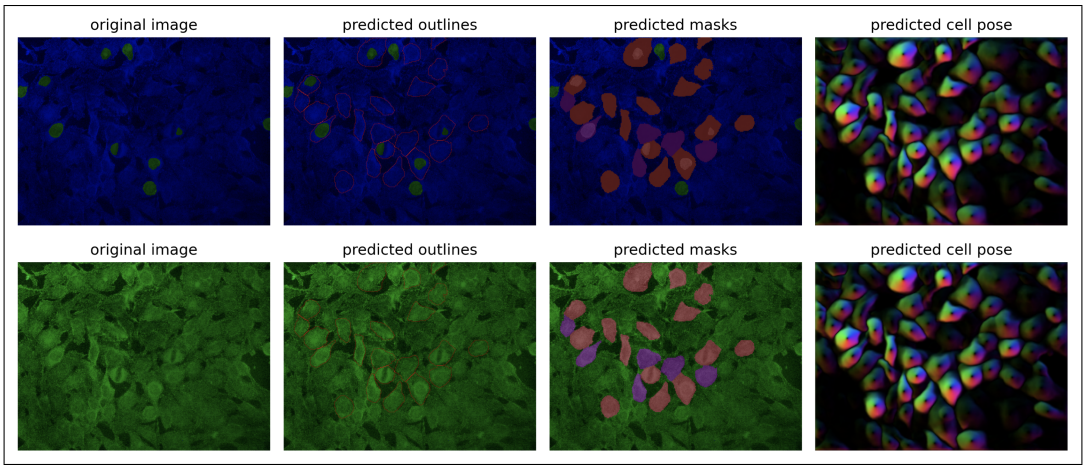


Figure 12: Final cellular segmentation for fibrous cytoplasm pattern

from unwanted sources, such as Cellpose miss segmentation. Finally, our final segmentation of some cells in a single image with different coloured cytoplasm can be observed in figure 13.

2.3 Convolutional Neural Network

Before applying neural networks it is first necessary to make the images uniform. At the end of the segmentation step, we output many different cells in different sizes, but a neural network only takes one fixed size. So we decided to make everything uniform. Also, despite being the best option available, Cellpose isn't entirely consistent, so there are still fragmented cells, empty spots and noise selected amongst our initial database. Firstly, we filtered the cells by size, excluding very small and very large cells. This was done by plotting each class' distribution and excluding the 5% outliers for each class. This way we would filter off every partially segmented cell and randomly segmented noise. Such distribution can be observed in Figure 14. Next we decided to filter out images that contained a large amount of black pixels to eliminate any possibility of Cellpose miss-identifying empty space as containing a cell. This was done by analysing the graph at figure 15, which plots each cell's percentage of black pixels by the number of pixels it contains. From this it is possible to arrive at a visual estimate of the characteristics possessed by outliers. As such, every cell not contained in the central rectangle wasn't used to train the neural network.

The remaining cells were encased by a square frame, by adding black bars to the sides of its smaller dimension. However, there was still a problem of uniformity, cells could vary from 10x10 to over 200x200 pixels. So we decided upon the fixed size of 96 pixels per cell dimension, a size that worked best with VGG's convolutional layers without exceeding our ram limits, and proceeded to up or downscale any cell square to that size.

In order to mitigate what is likely this project's primary constraint, the lack of data, we applied flipping and rotation to augment our input dataset. Flipping is a data augmentation technique that mirrors images horizontally, vertically or both in order to generate new images that, while similar, are still fairly different to be used as a new data point. Rotation is another data augmentation technique which rotates the original image to create a new one. Nevertheless, it wasn't always necessary to augment data, we only did so to classes that were lacking in images, represented in figure 1. Needless to say, these methods were applied only to the training dataset after the split and it showed an improvement of around 4% in classification accuracy. Finally, in order to use the neural network framework it was necessary for every image to have three channels, we attempted to use our cellular images with the nuclei in the green channel, the cytoplasm in the blue channel and image segmentation by use of the k-means clustering algorithm in the red channel, but this yielded worse results than just copying the green channel an additional two times.

In order to create our neural network, we decided to utilise existing technology through transfer learning from VGG16. Transfer learning is a machine learning method that reuses a model for a task that it was not necessarily created for. In this case we used VGG16, a deep convolutional neural network proposed by K. Simonyan and A. Zisserman in their paper "Very deep convolutional neural network for Large-Scale image recognition". Their neural network is trained to classify images according to the 1000 classes of a famous data-set called Imagenet which doesn't tackle the problem we are facing, so on top of the existing network we applied two Dense layers with ReLU activation functions and a dropout layer to avoid overfitting followed by a Dense layer with softmax activation for classification. Since

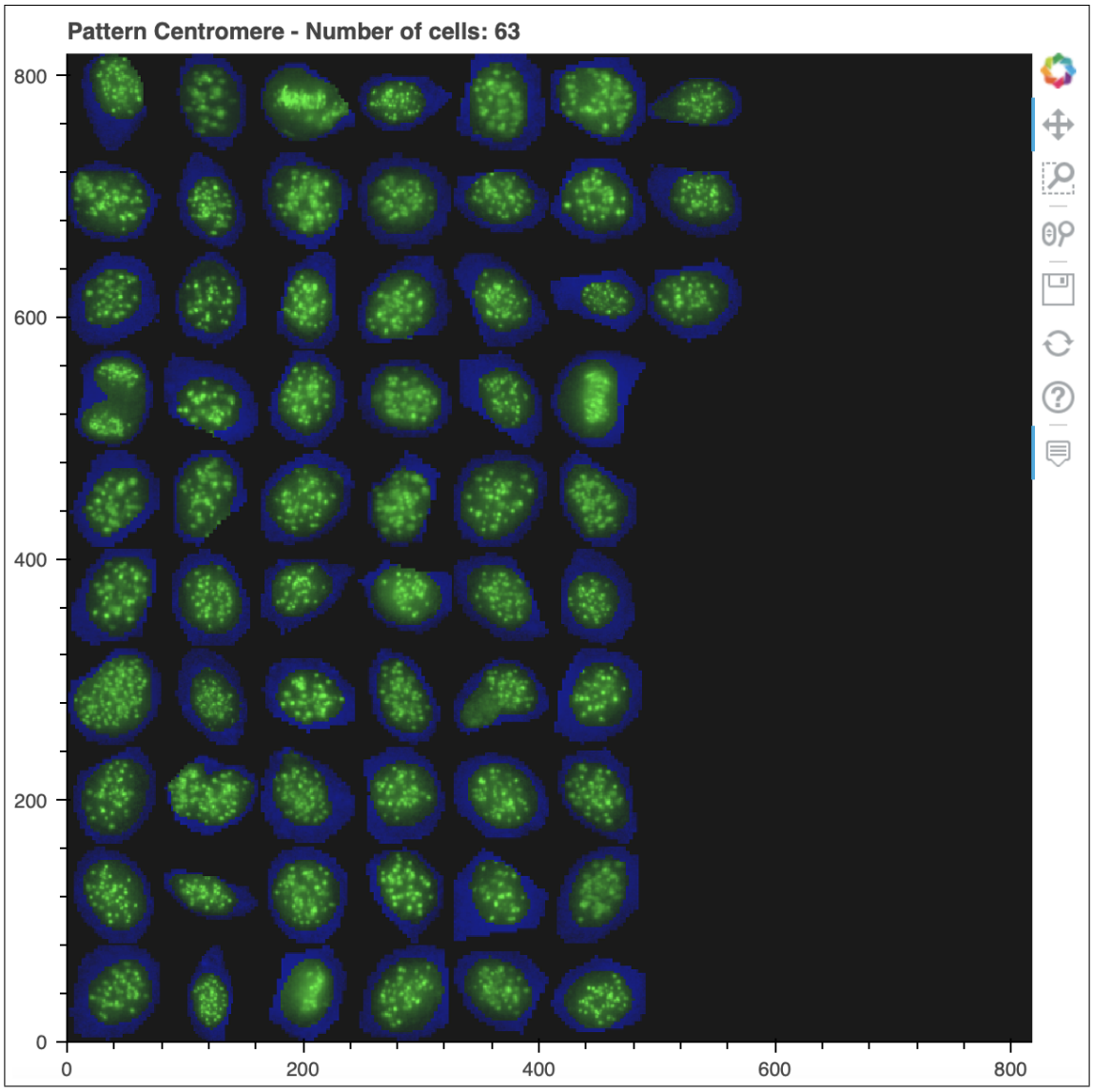


Figure 13: Final cellular segmentation for Centromere class

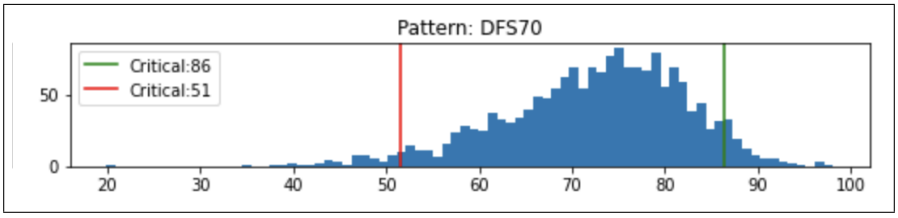


Figure 14: Pixel distribution for the DFS70 Class. Vertical lines represent the 5% percentiles

we were tackling a problem of multi-class classification the loss function chosen for minimisation was of categorical crossentropy, $L = -\sum_{i=1}^{n} y_i log(\hat{y}_i)$ where n is the number of output categories.

Upon testing and varying some of the parameters established above, i.e.: cell size threshold, percentage of black pixels, different sizes and numbers of dense layers at the end of the neural network, etc... A result of around 60% accuracy per cell was reached for the validation set for the 47 class dataset "HEP Annot", while a result of approximately

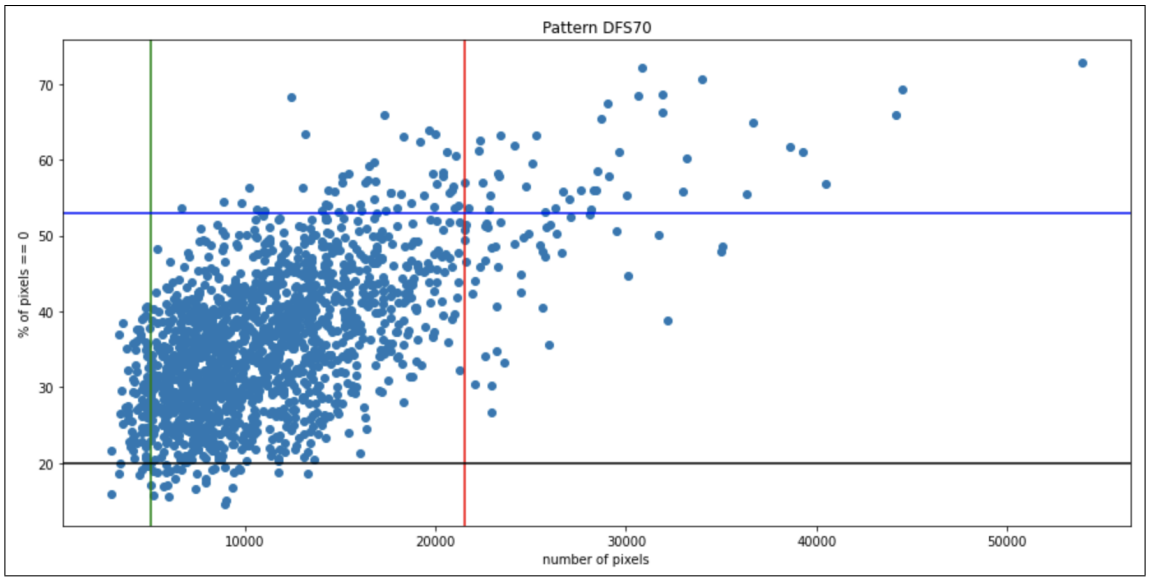


Figure 15: Percentage of black pixels by the total amount of pixels for each cell in the class DFS70.

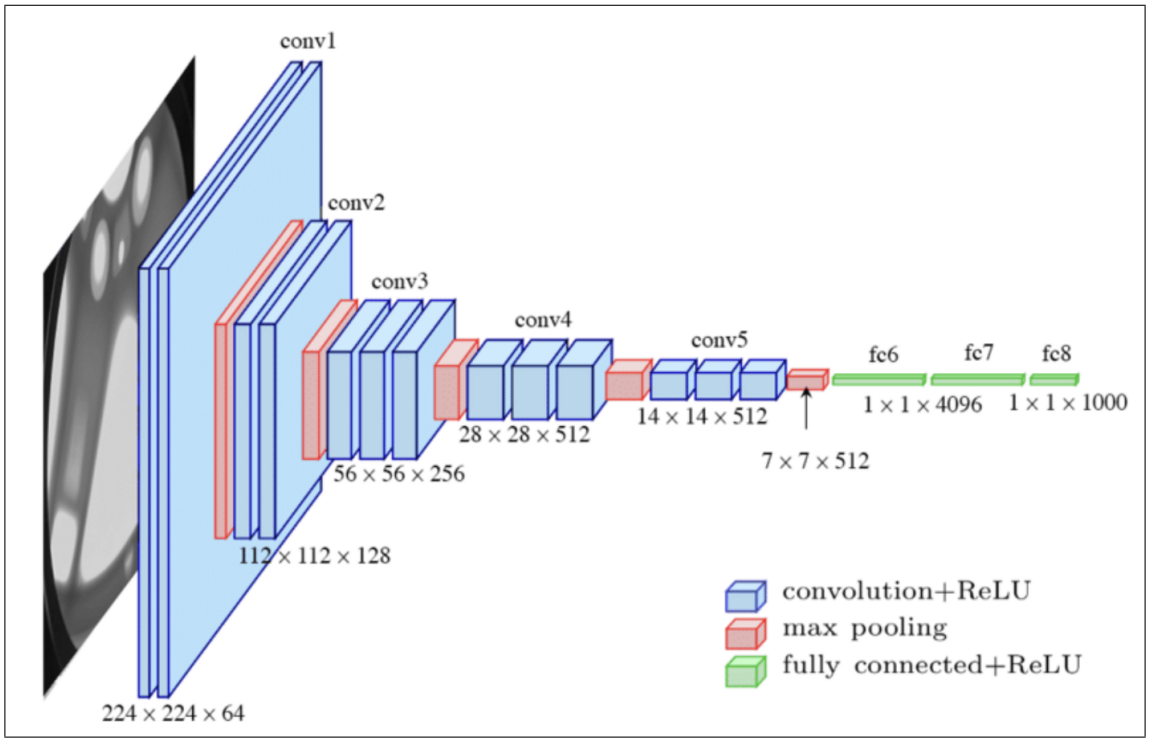


Figure 16: Architecture of VGG16

65% accuracy was achieved for the 14 class pattern dataset. This is an encouraging result because most images have over 50 cells so predicting an image correctly should be statistically easier just than a single cell.

Figures 17 through 20 show the neural network's performance over each dataset. While learning can be observed, it is clear that overfitting is present despite a dropout layer of 0.5. This could be due to several causes such as an inherent lack of non-augmented data for most patterns, or an inherent issue with the type of classification used. We believe that a more diverse training dataset would help the network's classification. To further analyse the classification results, we generated an error matrix in figure 21.

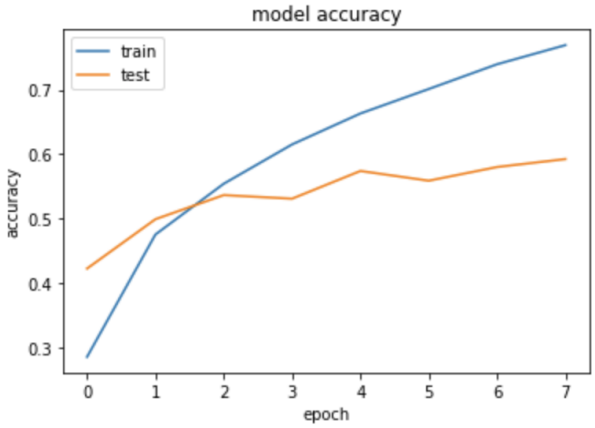


Figure 17: Neural network accuracy for HEP Annot dataset

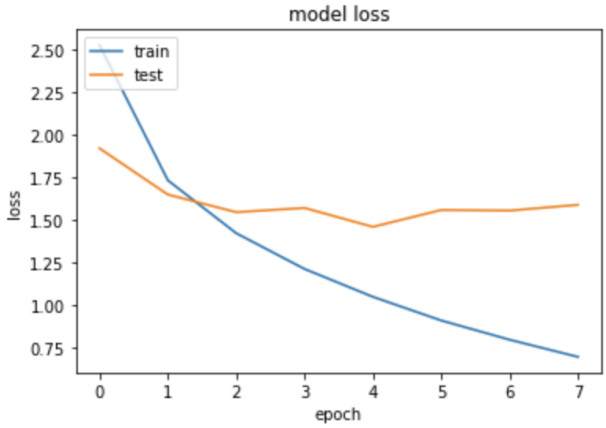


Figure 18: Neural network loss for HEP Annot dataset

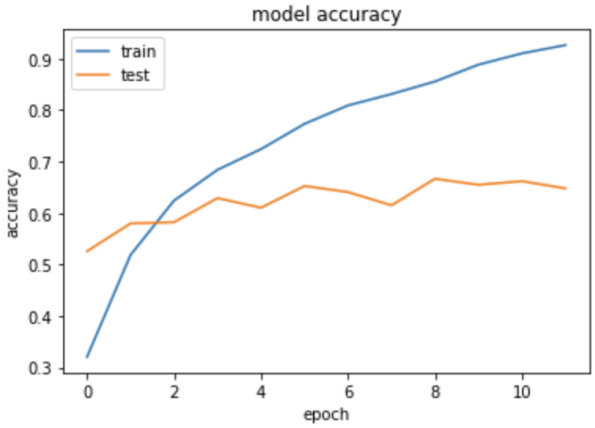


Figure 19: Neural network accuracy for HEP Annot dataset

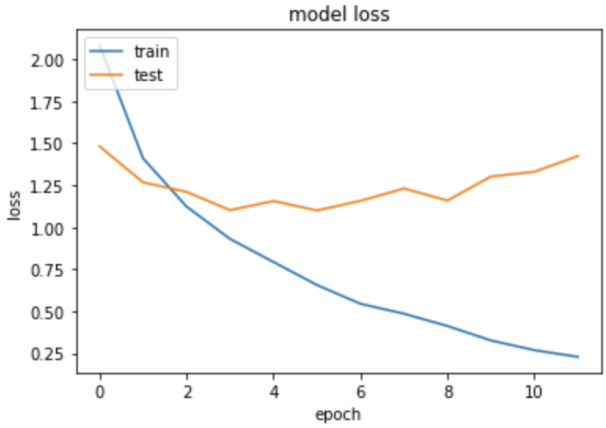


Figure 20: Neural network loss for HEP Annot dataset

As can be observed from the error matrix in figure 21, errors are concentrated in classes with the fewest images, and unfortunately, errors aren't randomly distributed amongst different classes. This concentration might trick the neural network into missclassifying classes with few images. Despite this, the confusion matrix points to optimistic results, as major classes were rarely confused with one another and less numerous classes that were confused tended to be similar. Such as the classes of "CytoD PL7" and "Mouchete CytoD PL7", ironically these errors are indicative that the neural network is learning from the actual cellular patterns and not from a random pattern that resulted from an error in Cellpose's behaviour or any error caused by the up/down-sampling of images.

After noticing the nature of these errors, we started to wonder if some of these classes weren't actually composite classes instead of exclusive ones. Perhaps, a cell classified as "Mouchete PL7" should instead be classified as positive for the categories "Mouchete" and "PL7". To investigate this possibility, we created a dendogram from our classification confuson matrix, in order to determine which classes were the closest to one another. This dendogram can be observed in figure 22.

The dendogram is a vivid demonstration of the nature of the classes themselves. It is a hierarchical/agglomerative clustering based on the classification similarities of the classes. Each side of a division by level shows the classes with the biggest similarities between them, ordering this clustering from most distinct class to the least (from left to right). Showing the high error in classification for "Cyto atypique HMCOAR" and "Cyto atypique MDA5", as well as pointing to a high correlation between them, thankfully.

The dendogram was created using a hierarchical linkage (similar to MATLAB's) using method='single', assigning:

$$d(u, v) = min(dist(u_i, v_i))$$

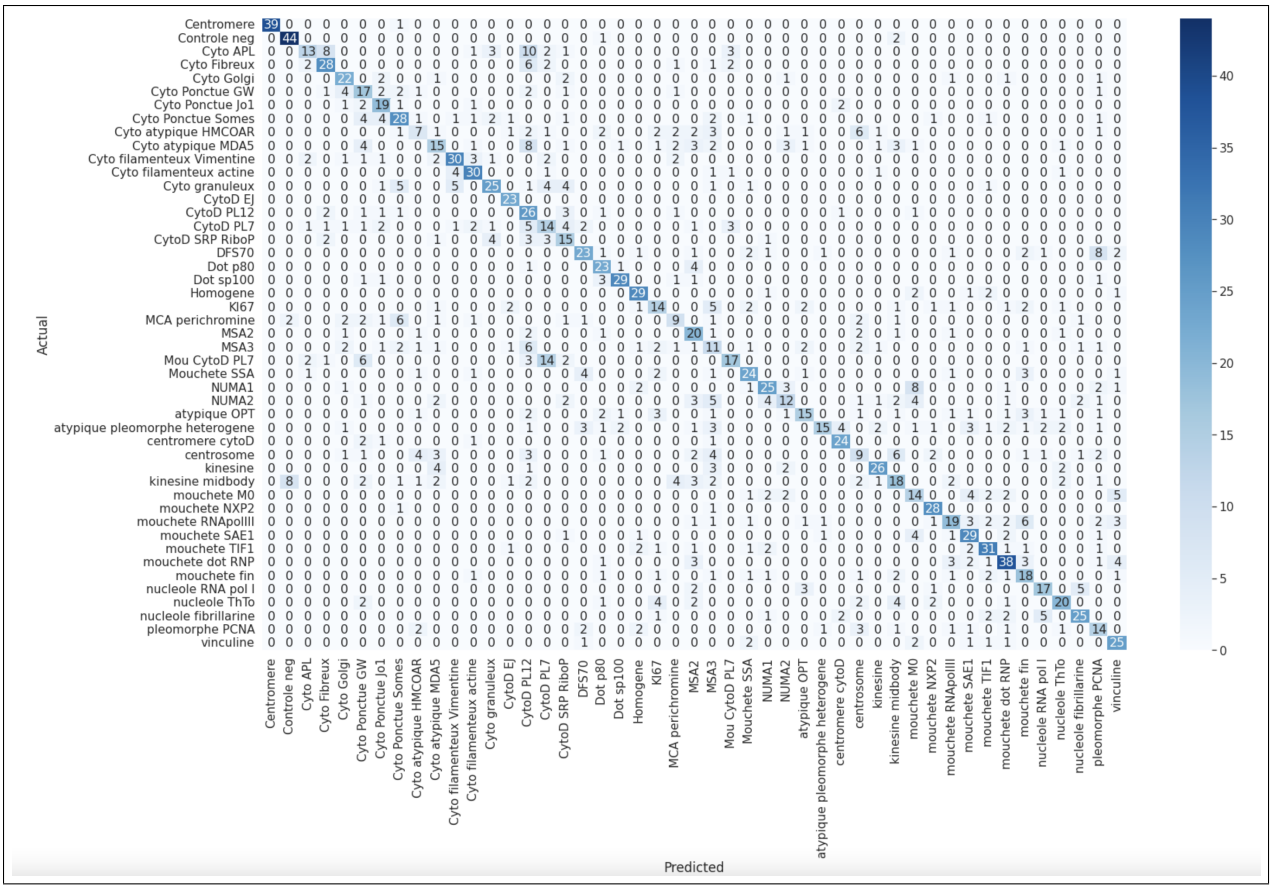


Figure 21: Confusion matrix for HEP Annot Classification

for all points i in cluster u and j in cluster v. This is also known as the Nearest Point Algorithm, we chose it for its ability to order them from left to right, as mentioned above.

This dendogram was intended to serve as a basis for class clustering. The idea was to group different classes into different clusters of similar classes and classify to which cluster a single image belongs to with higher accuracy. Then, for each cluster new neural networks would be trained to classify between its subclusters, or classes. In retrospect, this idea seems like a manifestation of the realisation that the problem tackled wasn't necessarily a multi-class classification problem but a multi-label one. Despite this newfound multi-label nature, this multi-class classification by levels should prove robust as a stand-alone classification for even multi-label problems; as an example of this, we have the classification of species of plants by google lens where first it classifies a plant as a plant, then proceeds to a secondary neural network pre-trained specifically for a family of plants, then one for species, and so on. This could circumvent the similarity between classes by learning to differentiate the label characteristics that set them apart indirectly.

2.4 Multi-label

Upon reaching this stage of the project, we received the hard drive dataset mentioned in section 2 with a far larger amount of images than any of the other previous datasets. But we quickly noticed that in this dataset images could be classified as pertaining to multiple classes. And after cleaning, organising the new dataset and arriving on 25 different classes it became clear that there was a miscommunication on the nature of the problem. While from the start, due to the format of our data, it seemed clear that this problem was one of multi-class classification the more recent dataset pointed to a multilabel problem. We consulted with Dr. Miyara to try to learn more about which format and classes to use, and he told us that some of the classes in the HEP Annot folder were actually composite, non-exclusive, classes. For this type of problem it is far more appropriate to approach it as a multi-label and leaving to the user (or not) the interpretation of a possible intersection.

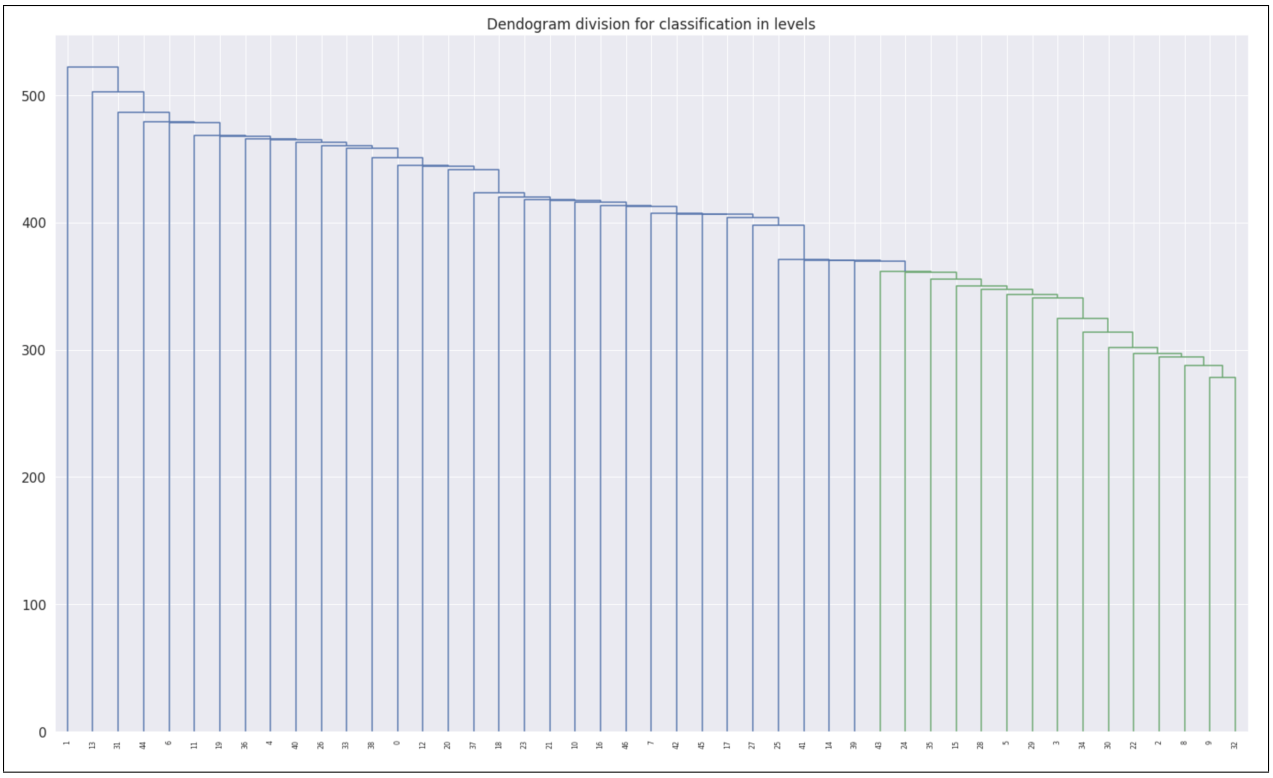


Figure 22: Dendogram with pattern numerical label on x axis

No.	Name	No.	Name	No.	Name
0	Centromere	16	CytoD SRP RiboP	32	centrosome
1	Controle Neg	17	DFS70	33	kinesine
2	Cyto APL	18	Dot p80	34	kinesine midbody
3	Cyto Fibreux	19	Dot sp100	35	mouchete M0
4	Cyto Golgi	20	Homogene	36	mouchete NXP2
5	Cyto Ponctue GW	21	KI67	37	mouchete RNApol
6	Cyto Ponctue Jo1	22	MCA perichromine	38	mouchete SAE1
7	Cyto Ponctue Somes	23	MSA2	39	mouchete TIF1
8	Cyto atypique HMCOAR	24	MSA3	40	mouchete dot RNP
9	Cyto atypique MDA5	25	Mou CytoD PL7	41	mouchete fin
10	Cyto filamenteux Vimentine	26	Mouchete SSA	42	nucleole RNA pol
11	Cyto filamenteux actine	27	NUMA1	43	nucleole ThTo
12	Cyto granuleux	28	NUMA2	44	nucleole fibrillarine
13	CytoD EJ	29	atypique OPT	45	pleomorphe PCNA
14	CytoD PL12	30	atypique pleomorphe heterogene	46	vinculine
15	CytoD PL7	31	centromere cytoD		

Figure 23: Dendogram pattern dictionary

While cleaning the new dataset, we decided to simultaneously investigate the HEP Annot dataset with a similar approach. Using semantic segmentation, we automatically transformed the classes in HEP Annot to labels that could intersect. This was done by parsing class names to create a new space of labels, and attributing to each of the previous classes a binary value for each label. So, for example, all classes containing: "Mouchete", "speckled" or "Mou" would receive label 1 for the label "Mouchete". As such, we were working with 57 labels instead of 47 classes.

In order to implement a multi-label classifying neural network we applied the binary relevance method, where we trained a binary classifying neural network for each label and applied them to the data independently. But since our data was already imbalanced, we couldn't just have it learn blindly on our cells. We decided to attribute weights to

each class, so that the underrepresented ones have a disproportionally larger influence. Also, a common percentual accuracy metric wouldn't suffice for an imbalanced dataset, we decided to use an AUC ROC accuracy. The model used is overall the same one used previously, with two dense and a dropout layer stacked on top of VGG16. As for the loss to be minimized we applied binary cross-entropy: $L = -\frac{1}{n}\sum_{i=1}^{n} y_i log(\hat{y}_i) + (1 - y_i) \cdot (1 - \hat{y}_i)$ with n being the number of scalar values in the model output.

The following paragraph will consist on a brief explanation of the AUC score. It was chosen because it is an effective way to measure binary classifiers for imbalanced data, more so than a simple accuracy metric. The AUC score is calculated from the ROC curve, which measures a classifier's true positive rate across its false positive rate. Both criteria vary from 0 to 1. While the foremost indicates the amount of true values it gets right, the latter measures the amount of false values it gets wrong. Intuitively speaking, the more lenient one is with the false positive rate, the higher the true positive rate will go. Thus, this curve inevitably contains the origin and point (1, 1). The AUC score is merely the area beneath this curve, and the closer it is to 1, the better the classifier is. To illustrate, a random classifier would have an AUC score of 0.5. Intuitively, it can be interpreted as the probability that a positive sample is ranked ahead of a negative one.

As can be observed on figure 24, the results obtained for each label were excellent. With the lowest AUC score being around 0.8, a far more reliable number than what was previously obtained for multi-class classification. Another point of interest is the "Controle" label, which exhibits an AUC score of 1. This category represents control cells that do not exhibit any specific autoimmune pattern, a perfect AUC score means that the classifier can always tell if the patient exhibits a relevant pattern or not. Also, it is remarkable and expected that the baseline weightless model outperformed the weighted one for many different classes. The weights helped classify the images with low variance in cells but a high inter-compatibility with all cells due to its inherently common convolutional features.

We tested its functionalities on the patterns dataset, to see what kind of results it would produce. Figure 25 and 26 show the most relevant results for pattern 10. It identified not only that most of its cells are of label "Mouchete" but a also that a relevant minority are of class "Sae1". Incidentally, Mouchete Sae1 is one of the composite classes in the HEP Annot dataset. Unfortunately, since the patterns set is unlabelled, we don't know exactly which class it belongs to, but on a preliminary basis and from manual verification, the neural network seems to output consistent results.

3 Unsupervised Cell Clustering

3.1 Transfer-learning representation

The results from the multi-label classification showed us that certain classes were failing in being classified because of an incredible amount of varying cells. From a global view, some classes are homogeneous in variation, but for classes like "Cyto atypique MDA5" the class gains the label atypique from having atypical cells that have a more luminous intensity (with "cyto" form) than the rest of the cells (not necessarily "cyto" formed). The results of this section correspond to the class "Cyto atypique MDA5".

This lead us to create a unsupervised clustering method for each initial class from HEP annot, that could be used to differentiate the aspects of a type of cell within it. This differentiation could later be associated to the labelled classification seen in sub-section 2.4.

The method consists on creating a partial representation of each cell using transfer learning using the output of the FC-layer as a transferred vector. This is done similarly to the previous classifications with the removal of the top layer of the VGG-net. Sub-sequentially, we use the output with a plethora of clustering methods that helps us differentiate them.

3.2 Kmeans++

The main clustering algorithms we use are K-means++ and Gaussian Mixture, both yielded good results, but, from a performance point of view, the complexity of the G.M. method is too large and is unusable for the large dimensional data we have to deal with. Furthermore, to deal with the non-linearity in which the G.M. outperforms K-means, vector space transformations can be used to mimic this as well as lowering complexity. So in this report we will focus on the K-means approach.

The vector space transformations we implemented uses the Kernel trick onto the transfer-representation data to be able to put it through a KPCA using a pre-computed kernel using the Nystrom approximation. Later, to this modified data we applyed the K-means, seeing that for the "Cyto atypique MDA5" example, the atypical cells represented 4% of all the cells and proves to us that the reason the multi-label classification didn't have great results for this class was due to under-representation and cross-over within the labels of this class.

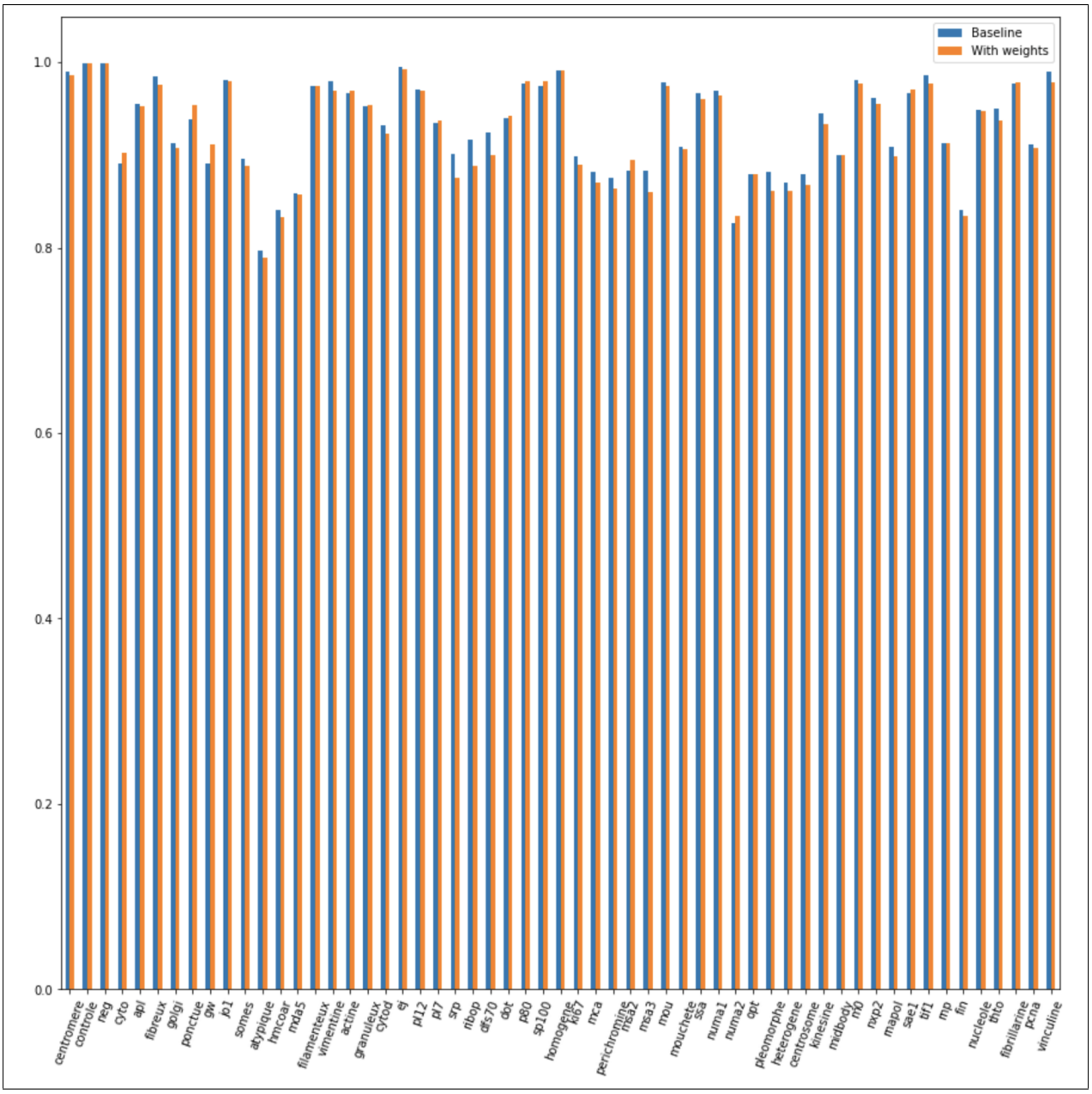


Figure 24: AUC scores for each label with weighted after un-weighted neural network training

K-means++:

It is an algorithm that chooses initial values of centers, followed by k-means clustering. Given a set of observations $(x_1, x_2, ..., x_n)$, where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k ($\leq n$) sets $S = \{S_1, S_2, ..., S_k\}$ so as to minimise the within-cluster sum of squares (WCSS) (i.e. variance). Formally, the objective is to find:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

where μ_i is the mean of points in Si. This is equivalent to minimizing the pairwise squared deviations of points in the same cluster:

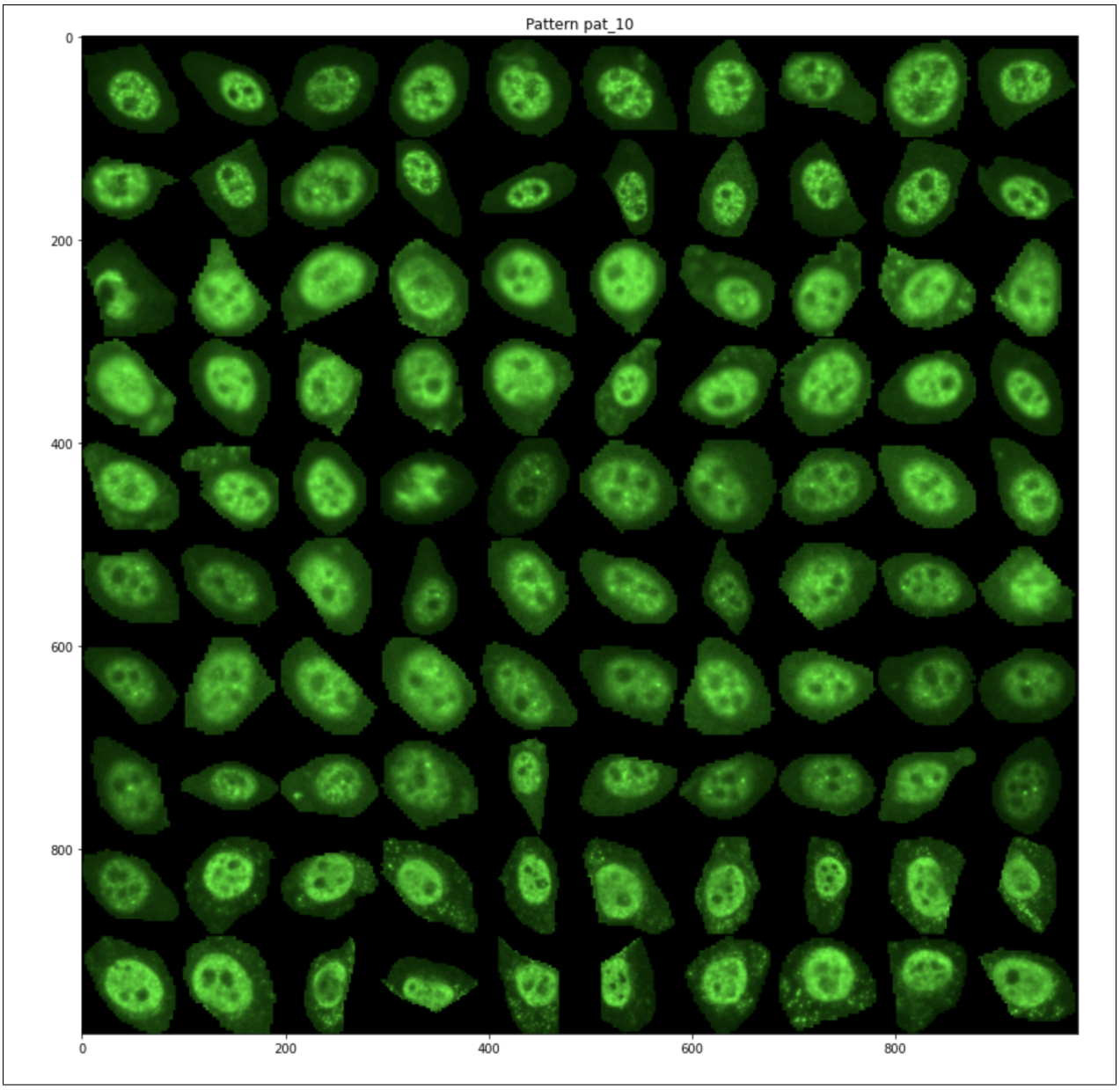


Figure 25: Visualisation of all cells in pattern 10 predicted as Mouchete

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \frac{1}{2|S_i|} \sum_{\mathbf{x},\mathbf{y}\in S_i} \|\mathbf{x}-\mathbf{y}\|^2$$

The equivalence can be deduced from identity:

$$\sum_{\mathbf{x}\in S_i} \|\mathbf{x}-\boldsymbol{\mu}_i\|^2 = \sum_{\mathbf{x}\neq\mathbf{y}\in S_i} (\mathbf{x}-\boldsymbol{\mu}_i)^T(\boldsymbol{\mu}_i-\mathbf{y}).$$

Because the total variance is constant, this is equivalent to maximising the sum of squared deviations between points in different clusters (between-cluster sum of squares, BCSS), which follows from the law of total variance.

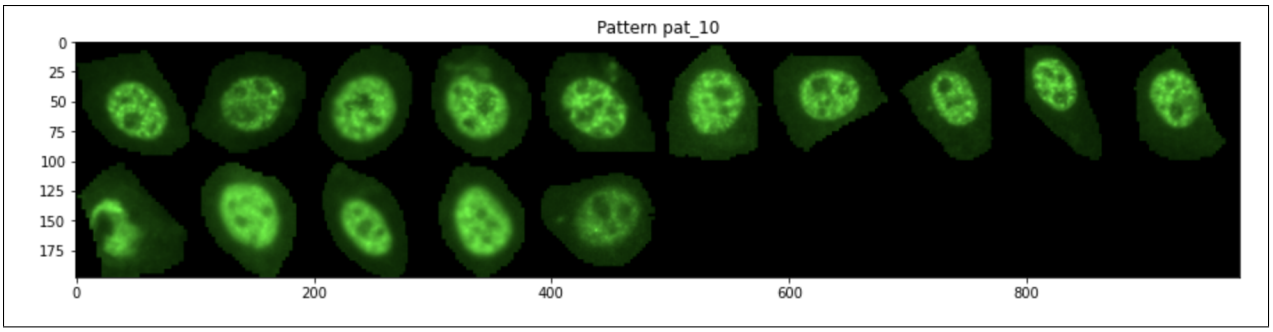


Figure 26: Visualisation of all cells in pattern 10 predicted as SAE1

3.3 KPCA

Notations.

Let $S = (x_1, ..., x_n)$ be a set of training vectors $x_i \in R^d$. Let $k : R^d x R^d \to R$ be a positive definite kernel. Let $[K]_{ij} = k(x_i, x_j)$ be the Gram matrix of S. Let be $\mathbb{1} \in R^{nxn}$ the matrix with every element equal to one. We note $\phi : X \to H$ the canonical feature map associated to the positive definite kernel $k(x, x')$ s.t. $\forall x, x' \in X k(x, x') = (\phi(x), \phi(x'))_H$.

Centring the kernel.

With $K_{ij} =< \phi(x_i), \phi(x_j) >$. Let $\Phi = [\phi(x_i)]_n^i$, $K = \Phi\Phi^T$ and b the mean of Φ

$$K^c = (\Phi - b)(\Phi - b)^T$$
$$b = \frac{\Phi\mathbb{1}}{n}$$

$$K^c = (\Phi - \frac{\Phi\mathbb{1}}{n})(\Phi - \frac{\Phi\mathbb{1}}{n})^T$$
$$K^c = (I - \frac{\mathbb{1}}{n})\Phi\Phi^T(I - \frac{\mathbb{1}}{n})$$
$$K^c = (I - \frac{\mathbb{1}}{n})K(I - \frac{\mathbb{1}}{n})$$

From here we suppose that the training point are centered. For any direction $f \in H_k$ s.t. $||F||_{H_K} = 1$, the associated captured empirical variance is:

$$Var`(S, f) := \frac{1}{n}\sum_{i=1}^{n}\langle(x_i), f\rangle_H^2 = \frac{1}{n}\sum_{i=1}^{n} f(x_i)^2$$

Hence, the i-th principal components is defined by

$$f_i = argmax_{||f||=1}\frac{1}{n}\sum_{i=1}^{n} f(x_i)^2$$

Decomposition.

We then have to map each feature space to a new value so that $f(x_i)^2 = \langle\phi(x_i), f\rangle^2$ for the centered data.

To evaluate the projection from a point in the feature space $\Phi(\mathbf{x})$ onto the kth principal component V^k (where superscript k means the component k, not powers of k)

$$\mathbf{V}^{k^T}\Phi(\mathbf{x}) = \left(\sum_{i=1}^{N}\mathbf{a_i}^k\Phi(\mathbf{x_i})\right)^T\Phi(\mathbf{x})$$

We note that $\Phi(\mathbf{x_i})^T\Phi(\mathbf{x})$ denotes dot product, which is simply the elements of the kernel K. It seems all that's left is to calculate and normalize the $\mathbf{a_i}^k$, which can be done by solving the eigenvector equation

$$\lambda\mathbf{a} = K\mathbf{a}$$

where λ and $\mathbf{a}$ are the eigenvalues and eigenvectors of K. Then to normalize the eigenvectors $\mathbf{a}^k$, we require that

$$1 = (\mathbf{V}^k)^T\mathbf{V}^k$$

Therefore:

$$f_i = \alpha\phi(x)$$

$$\alpha_i = \underset{\alpha}{argmax}\alpha^T K^2\alpha$$
$$\alpha_i^T K\alpha_i = 1$$

Using SVD decomposition,

Theorem:

If square G matrix is a real and symmetric matrix ($G = G^T$) then $G = VDV^T$, where V are the eigenvectors of G and Diag(D) are the corresponding eigenvalues.

$$K = \phi(x)^T\phi(x) = VDV^T$$

Where V is the eigenvector and D the diagonal matrix with the eigenvalues.

The eigen-decomposition of $G = \alpha^T K^2\alpha$, also real and symmetric, becomes:

$$G = \alpha^T VDDV^T\alpha = SFS^T$$
$$\alpha^T V = V^T\alpha$$

Using $\alpha_i^T K\alpha_i = 1$, we have that $\alpha^T VD^{1/2}D^{1/2}V^T\alpha = 1$

Projection.

Finally,

$$\alpha = VD^{1/2}$$
$$\alpha_i = \frac{1}{\sqrt{\mu_i}}e_i$$

Seeing that we can represent the gram matrix as:

$$\underset{\in R_{np}}{X_{kpca}} = f(x) = [\langle f, \phi(x_i)\rangle_{H_k}]_i^n = [\langle\alpha\phi(x), \phi(x_i)\rangle_{H_k}]_i^n$$

$$X_{kpca} = \alpha\phi(x)^T\phi(x) = \alpha K$$

3.4 Kernel Nystrom approximation

We approximate the full kernel matrix K for n samples, where:

$$K = [\kappa(x_i, x_j)]_{nxn}$$

First we sample d points from the original dataset X, which we denote by $\hat{X} = [\hat{x}_1, ..., \hat{x}_d]$.

Construct a low rank matrix of rank r by:

$$\hat{K}_r = K_d\hat{K}^{inv}K_d^T$$

Where $\hat{K}^{inv}$ is the pseudo inverse of $\hat{K}$, the full kernel matrix of the sampled points, using SVD decomposition.

$$\hat{K} = [\kappa(\hat{x}_i, \hat{x}_j)]_{dd}$$

And K_d is the partial kernel matrix for the d sampled points in relation to the original n samples.

$$K_d = [\kappa(x_i, \hat{x}_j)]_{nd}$$

From the low rank matrix we create a vector space $z_n(x)$.

$$z_n(x) = \hat{D}_r^{-1/2} \hat{V}_r^T (k(x, \hat{x}_1), ..., k(x, \hat{x}_d)) >$$

Where $\hat{D}_r = diag(\hat{\lambda}_1, ..., \hat{\lambda}_r)$ and $\hat{V}_r = (\hat{v}_1, ..., \hat{v}_r)$, from:

$z_n(x_i)^T z_n(x_j) = [\hat{K}_r]_{ij}$

This pre-computation of the kernel is linear and has no information loss in comparison to regular KPCA for a dimension d bigger than 35.

The results of this can be studied from the eigenvalue plotting of the KPCA method for each variation fro 150 components, as seen in figure 27. They point to a better representation using the transfer learned representation of data, as well as no information loss from using the pre-computation with linear complexity with more than 60 components. An improvement from the 4608 dimension of the FC-layer output.

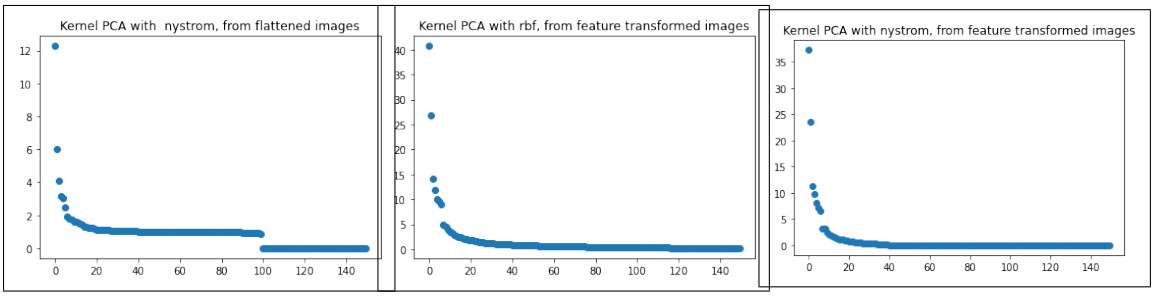


Figure 27: KPCA eigenvalue study for different variations

For the clustering algorithm itself the use of kernels can be great, as indicated by the paper [13], where the feature map transformation is applied directly to the K-means algorithm (aka Kernel K-means) and should have less information loss than KPCA, but would take too long to implement in this case.

3.5 Number of components

For choosing the best number of components(k) in the K-means++ method we use two methods, for which the results appears in Figure 28 for the clustering on the flattened FC-layer output directly and in Figure 29 with the application of KPCA after flattening:

- **The Elbow Method**
  Its approach consists in calculating the Within-Cluster-Sum of Squared Errors (WSS) for different values of k, and to choose the k for which WSS becomes first starts to diminish. In the plot of WSS-versus-k, this is visible as an elbow. Within-Cluster-Sum of Squared Errors is:
  - The Squared Error for each point is the square of the distance of the point from its representation i.e. its predicted cluster center.
  - The WSS score is the sum of these Squared Errors for all the points.
  - Any distance metric like the Euclidean Distance or the Manhattan Distance can be used, we used the euclidean.
- **The Silhouette Method**
  The range of the Silhouette value is between +1 and -1. A high value is desirable and indicates that the point is placed in the correct cluster. If many points have a negative Silhouette value, it may indicate that we have created too many or too few clusters. The Silhouette Value s(i) for each data point i is defined as follows:

$$s(i) = \begin{cases} \frac{b_i - a_i}{max(a_i, b_i)} & \text{,if } |C_i| > 1 \\ 0 & \text{,if } |C_i| = 1 \end{cases}$$

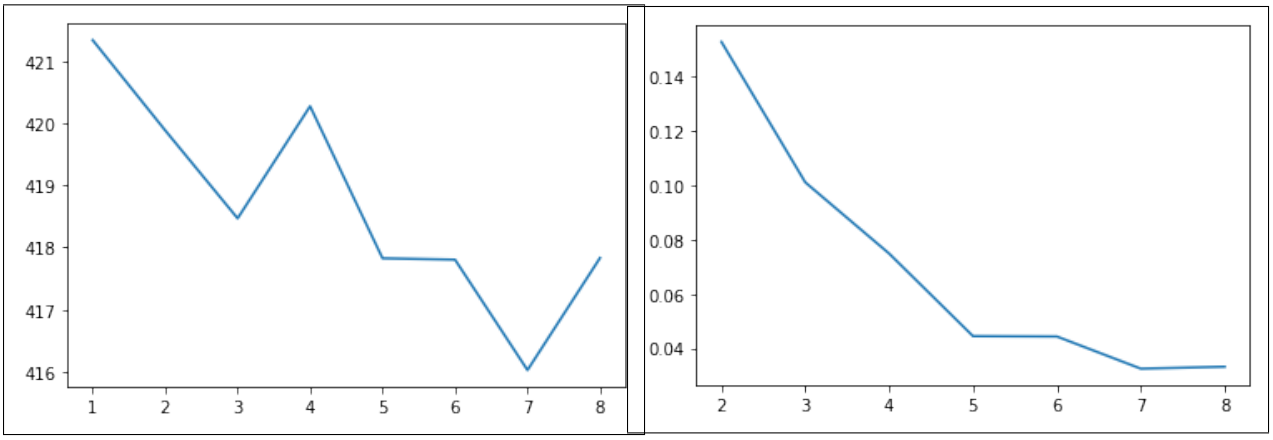
Note: s(i) is defined to be equal to zero if i is the only point in the cluster. This is to prevent the number of clusters from increasing significantly with many single-point clusters. Here, a(i) is the measure of similarity of the point i to its own cluster. It is measured as the average distance of i from other points in the cluster.

$$a_i = \frac{1}{|C_i| - 1} \sum_{i,j \in C_i, i \neq j} d(i,j)$$

Similarly, b(i) is the measure of dissimilarity of i from points in other clusters.

$$b_I = min_{i \neq j} \frac{1}{|C_j|} \sum_{j \notin C_i} d(i,j)$$

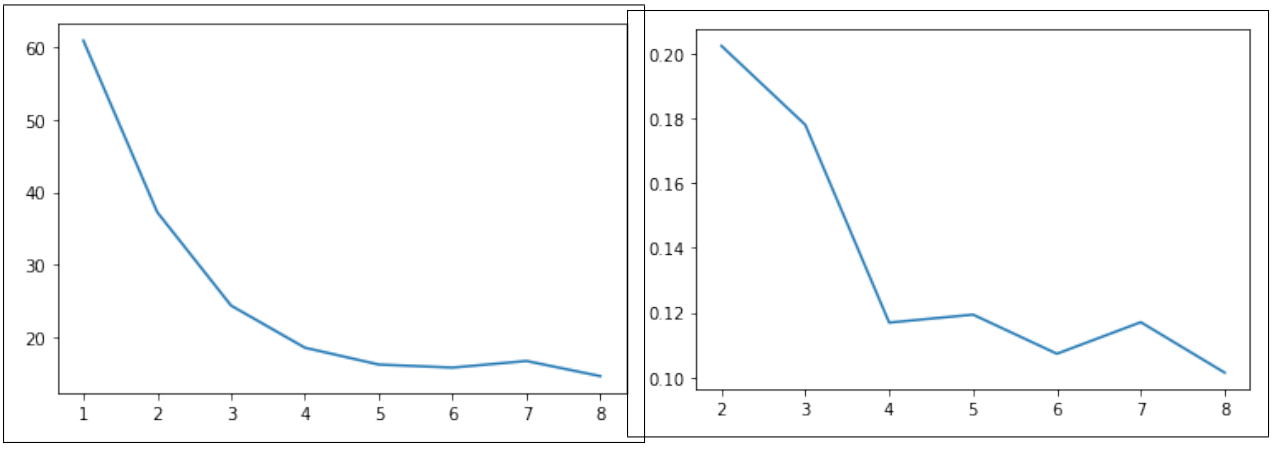
d(i, j) is the distance between points i and j. Generally, Euclidean Distance is used as the distance metric and it was the one we used.



(a) Elbow Method results

(b) Silhouette Method results

Figure 28: Clustering metrics vs. k, on flattened transformed data



(a) Elbow Method results

(b) Silhouette Method results

Figure 29: Clustering metrics vs. k, after KPCA using a Nystrom precomputed kernel on flattened transformed data

These results point to a bad representation of data for the non-kernel version of the K-means, due to its bad handling of non-linearity. This also shows that while the best number of components might be 5 from the elbow method, the

silhouette method shows that after 4 components there might be some over-clustering in play. For the kernel solution to be plausible, parameter fine tuning was necessary, with the gamma used for Nystrom approximation and KPCA being in the house of 10^{-5} for the transferred data from images with gray levels in $[0, 1]$.

4 Next Steps

The multi-label classifier is very promising, and we believe is yet to yield its best results, unfortunately, to the time of writing this report we didn't have the chance to fully explore its possibilities.

The first step towards obtaining a meaningful predictor would be to unify the labels with the aid of a medical specialist, since both of our datasets' differing tags were automatically generated and we don't have the expertise to judge them on the basis of validity. We already scheduled a meeting with Dr. Miyara to analyse our individual tags in order to reach a basal dataset from which we can unify both HEP Annot and our recently acquired hard drive's datasets. With the assurance of medically sensible results and a significantly larger dataset, we believe that there should be a visible increase in quality and decrease in AUC score variance.

Then we recommend using the clustering method in association with the medical feedback to generate perfect datasets for each label. From there, we would also recommend the investigation of more sophisticated multi-label classifiers.

Personally, we would recommend a very good approach by mixing auto-encoders and One-class SVMs with the use of Deep Learning and kernel pre-computation, such as Random Fourier Features and Nystrom Approximation, for outlier detection. Where the outliers would be the labels themselves. This method was show in [12] and we believe can be improved for this problem.

Another step could be implemented by using data from the classification of every cell in an image to indicate which class the entire image might fall into using an attention model, with the type of attention model that we think would perform the best being a Transformer model, but it should work as well for an RNN with non-forgetful LSTMs.

Finally, after the submission of this report, we intend to create a way for a user to interact with the project, either a notebook that streamlines all processes to obtain classification or ideally, a non-programmer friendly interface.

All our datasets and code are neatly stored in a google drive with instructions if someone decides to continue it. The code can also be found at the *GitHub repository for the code*.

We believe that if slightly polished it could serve as a discriminator for a GAN that generates these kinds of images.

5 Division of Labour

It isn't always easy to specify which person did exactly what because functions are generally re-written many times by different members, and the amount of hours spent might not directly translate into lines of code, as one might be looking for bugs. Also, the decision-making was largely done together. So the division of labour considering the amount of time spent is:

- Image Processing and Cellpose: 80% Ramon - 20% Gabriel
- Neural network: 50% Ramon - 50% Gabriel
- Unsupervised clustering: 100% Ramon
- Reports: 20% Ramon - 80% Gabriel

Both of us have spent over 225h on this project.

References

[1] Mehdi Habibzadeh and Mehdi Totonchi. Automatic white blood cell classification using pre-trained deep learning models: ResNet and Inception In *Conference: Tenth International Conference on Machine Vision (ICMV 2017)*

[2] Connor Shorten and Taghi M. Khoshgoftaar. A survey on Image Data Augmentation for Deep Learning In *Journal of Big Data - SpringerOpen*

[3] Xiaolei Huang and Gavriil Tsechpenakis. Medical Image Segmentation In *Information Discovery on Electronic Health Records*

[4] Dzung L. Pham, Chenyang Xu, Jerry L. Prince. Current Methods in Medical Image Segmentation In *Annual Review of Biomedical Engineering Vol. 2:315-337*

[5] Daniel J. Witheya and Zoltan J. Kolesa. A Review of Medical Image Segmentation: Methods and Available Software In *International Journal of Bioelectromagnetism* Vol. 10, No. 3, pp. 125 - 148, 2008

[6] Yuchi Hu, Michael D. Grossberg and Gig Mageras. Survey of Recent Volumetric Medical Image Segmentation Techniques In *Biomedical Engineering* DOI:10.5772/7865

[7] Humayun Irshad, Antoine Veillard, Ludovic Roux and Daniel Racoceanu Methods for Nuclei Detection, Segmentation, and Classification in Digital Histopathology: A Review—Current Status and Future Potential In *IEEE Reviews in Biomedical Engineering* VOL. 7, 2014

[8] Erik Smistad, Thomas L. Falch, Mohammadmehdi Bozorgi, Anne C. Elster and Frank Lindseth a,b Medical image segmentation on GPUs – A comprehensive review In *Medical Image Analysis* Volume 20, Issue 1, February 2015, Pages 1-18

[9] Elnakib A., Gimel'farb G., Suri J.S., El-Baz A. Medical Image Segmentation: A Brief Survey. In *El-Baz A., Acharya U R., Laine A., Suri J. (eds) Multi Modality State-of-the-Art Medical Image Segmentation and Registration Methodologies.* Springer, New York, NY. https://doi.org/10.1007/978-1-4419-8204-9_1

[10] Karen Simonyan, Andrew Zisserman Very Deep Convolutional Networks for Large-Scale Image Recognition In *arXiv:1409.1556*

[11] Khuyen Le An overview of VGG16 and NiN models In *Medium.com*

[12] Minh-Nghia Nguyen and Ngo Anh Vien Scalable and Interpretable One-class SVMs with Deep Learning and Random Fourier Features In *https://arxiv.org/abs/1804.04888*

[13] *Debolina Paul, Saptarshi Chakraborty, Swagatam Das, Jason Xu*
Kernel k-Means, By All Means: Algorithms and Strong Consistency In *https://arxiv.org/abs/2011.06461*