ODF semi-supervised learning, computer vision, and machine learning for cervical cancer detection and other applications in healthcare

Vincenzo Junior Striano

July 6, 2023

To all the beloved women of my life who made it charmed, brighter, and fuller.

1 Introduction

Abnormal cervical cells may become cancerous cells if not detected and treated. Cervical cancer is the fourth most common cancer among women globally and can be cured if diagnosed at an early stage and treated promptly.[21] Abnormalities in cervical cells can be detected thanks to cytology techniques like PAP smears, which consists of collecting cervical cells and examining them under the microscope.[3] Despite being widely used, Pap smears show low sensibility (approximately 60%) leading to a high number of false negatives.[40]

The goal of this thesis was to make new instruments available for medical study and measure their performance, with the aim of assisting doctors in the research and subsequent treatment of anomalies in women's cervixes. Currently, Pap tests are the preferred method for detecting such abnormalities, despite their poor sensitivity, which results in a fair number of false negatives, leaving women convinced they are healthy and so they never receive treatment, despite the risk that such abnormalities will degenerate into cancerous cells. Therefore, the use of computer-based tools makes it possible to greatly reduce the time and cost of analysis, allowing more patients to undergo it and reducing the occurrence and mortality of this cancer.

The proposed tools are based on machine learning techniques, computer vision, genetic algorithms, and an original, personally invented, machine learning algorithm category, which, due to its complexity, I have given a fancy name: *Optimized Derived Features semi-supervised learning* or *ODF-SSL*.

The main research foci are:

- **Establishing the predictive efficacy of cells pre-treated with well-established staining methods using convolutional neural networks.** A cytologist's typical samples are clusters of cells extracted from tissue using various instruments such as brushes, spatulas, or swabs. Later, cells are colored using Papanicolaou's approach, which has the advantage of keeping adequate

transparency of the cytoplasm while revealing exceptional detail of the nucleus structure. Finally, the cytologist examines the cells under the microscope, classifies them, and evaluates any abnormalities. **Eliminating the human component from the diagnosis could increase the effectiveness of the test as well as reduce costs and time.**

- **Whether there is at least one cell positive for structural abnormalities such as cancer, in a slide containing hundreds of cells, thanks to image segmentation with the convolutional neural networks.** This approach could increase the sensitivity since we believe that this is not ideal due to the problem of undersampling the sample. Image segmentation is the process of dividing an image into semantically relevant and aesthetically coherent areas, allowing for the investigation and comprehension of particular objects or regions within an image. **Using this approach, the machine can learn the characteristics of an abnormal cell and detect it alongside thousands of other cells. In contrast, slides traditionally contain only a dozen cells.**

- **Whether a lesser-used staining method could constitute the new state of the art in cytology, with CNN, K-means, t-SNE and a genetic algorithm, i.e., the AgNOR technique.** AgNOR staining is a commonly used technique in cytopathology for cancer cell detection but cervical ones, with Papanicolaou's being the most common. Indeed, **several studies concluded that the mean number of AgNORs of tumor cells is a significant prognostic factor in surgically treated lung cancer patients.**

Of particular technical significance is the invention of a new approach to machine learning, aimed at simplifying two-dimensional problems, typical of images, into vector problems. Convolutional neural networks are state-of-the-art tools for image classification using neural networks. As will be explained in detail in the machine learning section, such algorithms are able to learn which features are recurrent in a group of images belonging to the same category: if we submit a series of images of flags to a convolutional neural network, we expect that it will be able to classify as a flag any other unknown image in which a sequence of colored bands is represented. Convolutional neural networks, however, show at least three limitations:

- They require a large amount of data to be trained
- If the computer on which the training takes place is not performing well, the time devoted to this phase can take up to several hours
- The model's poor **interpretability**

The case of the repeated colored bands was only a didactic example: in reality, we are not able to say exactly what characteristics the machine actually associates with each class on which it is trained. The lack of interpretability of the models involved in artificial neural networks is a problem that scientists are still trying to solve. For example, a weather application is able to determine through a neural network whether it will rain the next day. The consumer is usually not interested in establishing which variables the machine considered to make the forecast; they only want to know the output. Meteorological

research, on the other hand, may be more interested in establishing which variables are decisive in forecasting weather conditions. The line of research conducted in this thesis on the positive correlation between AgNOR number in a cell's photograph and cancer positivity aims not only to establish whether cells treated with the method of the same name have predictive capacity but also what variables are involved in the prediction. The problem of interpretability is also raised here, which can hardly be solved by a convolutional neural network alone.

The new machine learning paradigm proposed (Optimized Derived Features semi-supervised learning) makes it possible to automatically derive features from an image, transform them into numerical indices and arrange them in a table. Here, then, the problem of analyzing and predicting from a two-dimensional variable, such as an image, is simplified to a vector problem. While the network is theoretically capable of modeling much of the variability contained in the pixels of each image, **this approach aims to simplify the problem by extracting only those features of an image that are assumed to be of greatest interest, such as cell size, expressed in a surface measure, or AgNOR count. If the data are arranged tabularly, simple and interpretable logistic regression algorithms can be used to establish any correlation between a numeric variable and a binary variable** After obtaining proof of the correlation between AgNOR number and cancer positivity, a working prototype of an algorithm based on the new approach was created. This consisted of deep image pre-processing, followed by a clustering algorithm and one based on graph analysis, the parameters of which were optimized by a genetic algorithm, together with reduced human effort.

The last chapter of the paper was devoted to testing the effectiveness of techniques similar to those just discussed in different diagnostic applications, demonstrating how machine learning can find further applications in improving human lives.

The thesis style will intentionally not be extremely technical so that hopefully even non-specialist medical personnel can understand what is being covered.

Contents

2 Medical overview

2.1 Cytology and Traditional Pap Smear Overview

Cytology is the branch of medicine aimed at evaluating the morphological and functional characteristics of a cell. If a tissue is morbid, its cells would show alterations, based on which a diagnosis can be made. A cytologist's usual samples are groups of cells taken from a tissue using various tools: brushes, spatulas, or swabs.[6] Cells are then colored using Papanicolaou's method (PAP), which has the merit of showing the nuclear structure in excellent detail and preserving sufficient transparency of the cytoplasm while giving it particular color tones.[31] At the end, the cytologist inspects the cells under the microscope, classifies them, and assesses abnormalities.

A Pap smear is a diagnostic test aimed at identifying any abnormalities in the cells of the cervix. The presence of abnormal cells does not indicate the presence of a carcinoma: only cells which are not detected and treated may become cancerous cells.[1]

The Pap test is considered an effective and cheap test that all women of reproductive age should undergo every 3 years or so[1]. However, the test shows an average sensitivity of about 60%: this means that out of 100 women who have abnormal cells in their cervix, 40 of them will be considered perfectly healthy.

Several sources associate the low sensitivity of the test with sampling errors and errors in sample interpretation. **We also believe this low sensitivity is related to a problem of under-sampling: if 60 thousand cells are present in 1mm^2 of skin and the slides to be analyzed are usually composed of a few dozen cells, is it really possible to determine the health status of the entire population of cells on the basis of such a small sample?**

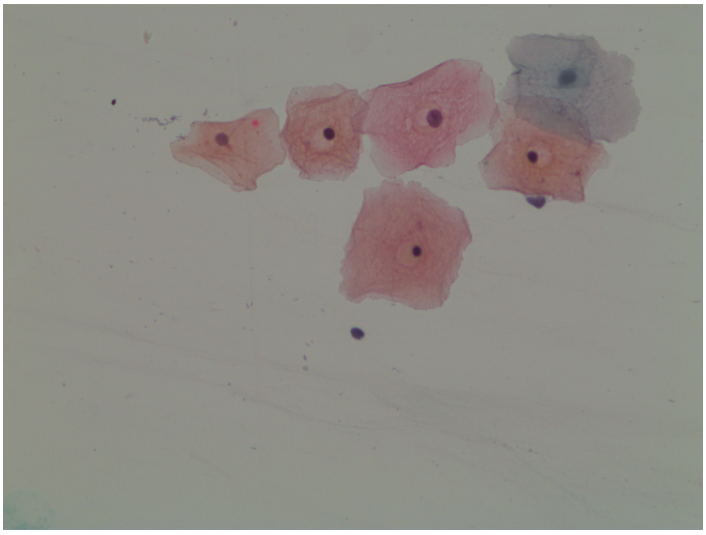

Figure 1: Slide showing 6 cells

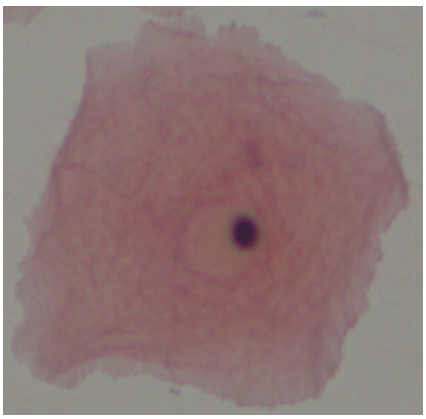

Figure 2: Cervical *superficial* physiological cell

2.2 Cervical cell abnormalities

The SIPakMeD data collection downloaded from Kaggle (or *dataset*) containing Pap-treated cells is labeled by cytologists according to five classes: *Dyskeratotic*, *Koilocytotic*, *Metaplastic*, *Parabasal*, and *Superficial*.[23] While Superficial, Metaplastic, and Parabasal cells are benign or normal, Dyskeratotic and Koilocytotic cells indicate abnormalities. Categorization - and so diagnosis - is done by cytologists who examine the slide under the microscope and assesses the morphological features of each cells. For example, koilocytotic cells show large nuclei, compared to the cytoplasm, and have a *halo* in the perinuclear area.[8]

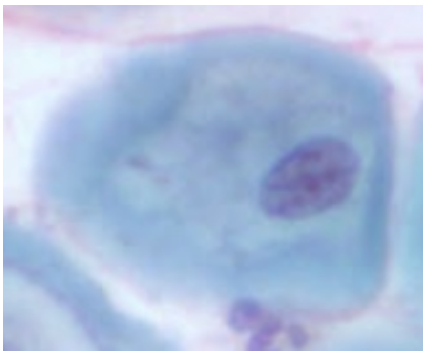

Figure 3: Koilocytotic cell

2.3 AgNOR staining technique

The *NOR*s (Nucleolar Organized Region) are DNA loops in the nucleus of a cell.[4] NORs are argyrophilic thanks to peculiar proteins in their structures, so when NORs are stained with silver they can be detected as black dots exclusively located in the nucleolar area. Once stained, NORs are called *AgNORs*.

AgNOR staining is a commonly used technique in cytopathology for cancer cell detection except in cervical ones, where Papanicolaou's is the most common. Indeed, several studies concluded that the mean number of AgNORs of tumor cells is a significant prognostic factor in surgically treated lung cancer patients.[27] One of the datasets we have used shows cervical cells treated with AgNOR

Figure 4: AgNOR stained cell showing multiple black NORs

both belonging to women given positive or negative diagnoses. **We will investigate the predictive performance of those slides using advanced machine learning techniques.**

3 Technical overview

3.1 Machine Learning

The term *machine learning* (*ML*) refers to a set of algorithms based mainly on linear algebra aimed at emulating human behavior that interprets and interacts with the unknown based on experience and comparison with what is already familiar. Although machine learning was conceived as early as the 1950s, it has only been possible in the last two decades to develop it. Increasingly high-performance computers have allowed algorithms to return results quickly. It has also been noted that certain algorithms perform much better in terms of predictive effectiveness as the amount of data they have increased. From the data - in the form of numbers, images, or sounds - the computer is able to extract patterns of recurring characteristics in these[53]; indeed, suppose we have a dataset containing information about the height and age of a group of individuals. We would likely expect that as age decreases height decreases too, while as it increases, height may have an asymptotic behavior. The algorithm is able to learn this relationship, so by submitting new height data to it, it will be able to determine what the age of the individual might be to a good approximation.

Regardless of the algorithm used, a common ML pipeline contains several steps:[49]

1. Data collection: The process of acquiring raw data from various sources, such as databases, files, and IoT devices. Data collection can be done manually or automatically and may include the use of scraping techniques to extract data from websites or social media.

2. Data analysis: The process of exploring and understanding the collected data in order to identifying patterns, relationships, or anomalies. This may include the use of data visualization techniques to graphically represent the data in order to facilitate understanding.

3. Feature engineering: The process of selecting, transforming, and creating new variables (features) from the raw data to improve the performance of a machine learning model. The goal is to extract useful information from the data and transform it into features that best represent the relationships among the variables.

4. Train-test splitting: A technique used to evaluate the performance of a machine learning model. The dataset is divided into two parts, one for training and one for testing, in which the model is trained on the training data and then tested on the testing data. This enables the model's ability to generalize to new data to be evaluated.

5. Hyperparameter tuning: The process of selecting the best values for the parameters of a machine learning model, known as hyperparameters. Hyperparameters are different from model parameters in that they are not learned during the training phase but must be set *a priori*. Hyperparameter tuning involves choosing the optimal hyperparameter values to improve model performance.

6. Training phase: The stage in which a machine learning model is trained on the training data to learn the relationships between variables and develop a prediction capability. At this stage, the model tries to minimize the error between its predictions and the training data.

7. Prediction phase: The phase in which the trained model is used to make predictions on new data, using the information learned during the training phase. At this stage, the model attempts to generalize relationships learned from the training data onto new data.

3.1.1 Supervised Learning

Supervised learning refers to regression and classification algorithms. In the case of regression, the algorithm finds the mathematical relationship between the numerical values in a certain instance: just as in the height and age case mentioned above. In the case of classification, on the other hand, the algorithm finds associations between the characteristics of an instance and the category associated.

Suppose we treat an instance with the following characteristics and the corresponding category: type: animal - area: desert - coat: striped - food: herbivore - category: zebra. The algorithm will be able to learn the relationships between the characteristics and the category so when an instance is identical to the previous one but with food: carnivore, it will be able to predict that we may be dealing with a tiger.

3.1.2 Unsupervised Learning

Unsupervised learning aims to group instances that have similar characteristics to each other (known as *clustering*). A real-world example might be a company that wants to improve the online experience of its users whose behavioral characteristics it knows. User A, of whom we know the geographic location, age, and time spent on the site often buys shower gel. User B, whose preferred item we do not know, has similar characteristics compared to User A. User C, however, is from a different city, is much younger than the other users and often buys motor oil. Users A and B will be aggregated into the same group, while User C will make his own.

3.1.3 Semi-Supervised Learning

Semi-supervised learning consists of a sequence of techniques designed to perform a classification or regression problem where it is costly to obtain labeled instances.[56]

1. The starting dataset of a supervised learning problem will consist of a collection of instances whose features are known, and whose categories are not (like unsupervised learning).

2. The developer manually categorizes a small number of instances.

3. Finally, a supervised learning algorithm will learn the association between the features of the labeled instances and their respective category.

A concrete example, referring to the case of the thesis, is that of a dataset of cells of which we know the shape, color, and area but not whether they are cancerous or not. A cytologist could label the first 10 cells and a classification algorithm can then learn the relationship between the characteristics of those cells and their category so as to predict it for all other instances.

ODF-SSL, the ML pipeline we propose, is based on semi-supervised learning with an additional level of complexity, which will be further investigated.

3.2 Deep Learning and Artificial Neural Networks

Deep learning refers to a set of ML algorithms where algebraic transformations aimed at learning and predicting occur sequentially, forming a "deep" or "layered" architecture.[52] Artificial neural networks constitute the most common example of a deep learning algorithm and are often used to perform classification and regression tasks.

An artificial neural network has as its input the features in the form of numerical indices of each instance present in the reference dataset. These values are subjected to a concatenation of parameterized linear transformations and are combined with each other through simple mathematical operations. At the end of the "chain", a predicted value is returned, which, in the learning phase, is compared with the already known category.[50] The difference between the predicted value and the category is measured and, by means of some algebraic operations, the parameters of the starting transformations are iteratively adjusted until the prediction matches the actual category.[51]

3.3 Computer Vision

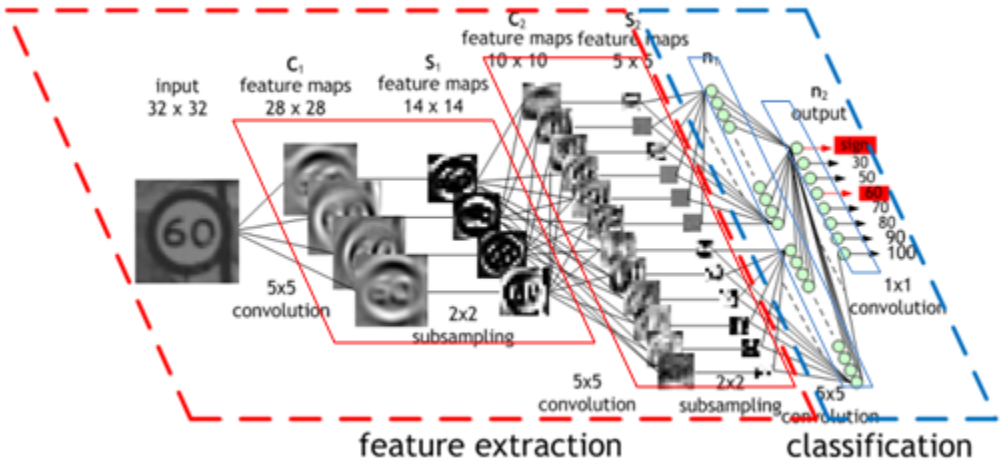

Figure 5: CNN, image by Maurice Peemen

The term *computer vision* refers to a branch of artificial intelligence that leverages several other disciplines to enable computers to learn features of the world around them from images (and not from numerical data as in the more specific case of artificial neural networks).[12] Artificial neural networks configured in a specific architecture or *Convolutional Neural Networks* (*CNN*s) represent one of the techniques through which visual learning can take place. This technique, in this thesis, will be widely used for different specific purposes.[30]

CNN considers images as matrices of numerical data, thus numbers arranged along two dimensions (and not one, in a vector, as in the case of tabular datasets). Suppose the image of interest consists of 5 pixels x 5 pixels. Each pixel takes a value, 0-1, depending on whether the pixel is white or black.

The CNN makes use of filters, i.e., smaller 0/1 matrices, which run along the image and return another matrix (*feature map*) populated by 1, where the filter and the image had the same value, and 0 if a different value or 0. The more similar the filter is to the image, the more 1s there will be in the feature map, the more "similar" the filter will be to the image. **If the resulting feature map had 1 in a specific block, the filter was been able to detect features in that specific area of the image.**

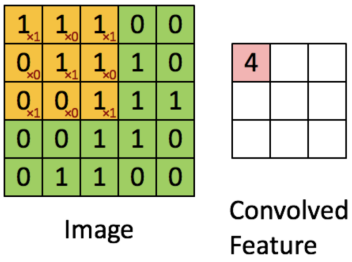

Figure 6: Feature map, size reduction. Image:: Sabyasachi Sahoo

Let us say that the filter is 3 pixels x 3 pixels. As shown above, the convolution will also result in a size reduction of the image. Several other transformations occur, which will not be further discussed. **Not only was the filter able to detect a feature in the image, but it also reduced its size.**

At the end of the process, a 2-pixel x 2-pixel feature map will be obtained, summarizing the characteristics of the original image. A *flatten* transformation will arrange the 2-pixel x 2-pixel matrix into a 4-pixel one-dimensional array.

The array will be passed into an artificial neural network as explained in the relative section and the prediction will be compared to the actual category. The difference will not only tweak the parameters of the ANN but also the values in the filters of the CNN. **The rationale is to identify filters that better capture and summarize the features of the original image**.

3.4 Optimization and Genetic Algorithms

An optimization problem consists of finding the points in the domain of a function that will minimize or maximize it. In this specific case, some algorithms will be optimized, changing their parameters, which will affect the *result* of the function. It will be necessary to find, therefore, those arguments of the function that minimize the result of that function.

Optimization problems can be divided into discrete or continuous optimization depending on whether the parameters can take integer values or real values. This distinction will be crucial in choosing the tools to be used in optimizing our case.[54]

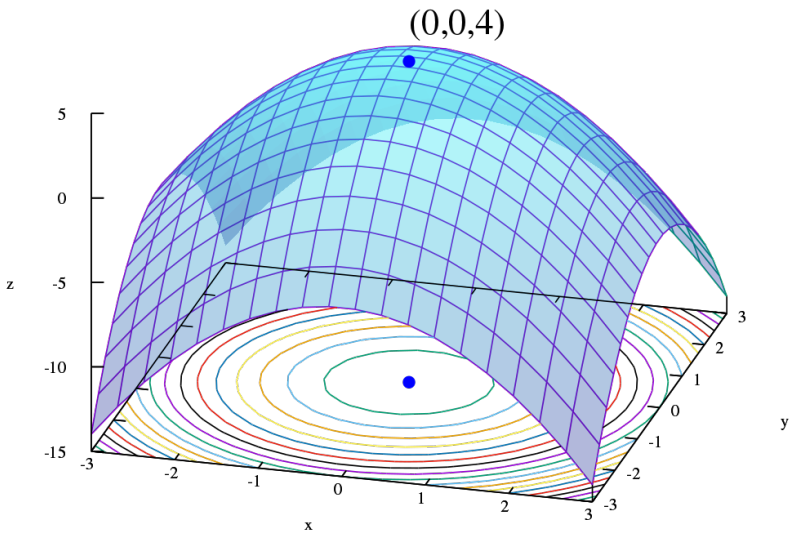

Figure 7: Graph of a function $z = f(x, y) = (x + y) + 4$. The global maximum at $(x, y, z) = (0, 0, 4)$. Image from Wikipedia

The simplest approach to optimization is *bruteforcing*[41]: suppose there are two parameters to be

optimized x, y and x can take three values: (1,2,3) while y: (A, B, C, D) Bruteforcing involves computing the value of the resulting function from all possible combinations of parameters: 1A, 1B, 1C, 2A, and so on (also called **search space**). This approach has the advantage of returning with absolute certainty the best solution to our problem. On the other hand, as the number of parameters increases, the time required to compute the results from all possible combinations of parameters increases considerably. Technically speaking, these algorithms are called **exact algorithms**, which mean it is always possible to find the best solution. However, with NP polynomial time problems, solutions may take a lot of time.

Moreover, **heuristic algorithms** use approximation to find good solutions to problems that are difficult or impossible to solve optimally using exact methods. Heuristic algorithms are a type of optimization algorithm that prioritizes speed and efficiency over guaranteed optimality.

A **genetic algorithm** is a type of optimization algorithm inspired by the process of natural selection that occurs in biological evolution. It is a search algorithm used to find approximate solutions to optimization and search problems by mimicking the process of natural selection, mutation, crossover and selection.[17]

- The initial population is composed of candidate solutions called chromosomes. A chromosome is a vector is composed of numbers, each of them called a gene. Each gene, at each generation, will evolve depending on the parameters of the algorithm.
- The fitness function determines the ability of an individual to survive, which is evaluated at each iteration of the problem, selecting the best individuals that will survive. It has to be minimized or maximized.
- Selection: Select the fittest individuals, passing their genes to the next generation. Individuals with high fitness have a better chance to be selected of being selected for reproduction, depending on the selection that is chosen in the algorithm.
  - Elitism selection: Copy the best chromosomes, whose fitness is larger than the threshold. Fitness function does not decrease in each generation
  - Route wheel selection: The probability of choosing a chromosome is proportional to its fitness
  - Rank selection: we do not mind the *absolute* fitness function, but we rank chromosomes in order. The probability of choosing the first-ranked chromosome is given to P_c, which can be chosen by the programmer. The probability of choosing the second is given by $1 - Pc * Pc$. The probability of choosing the n-ranked chromosome is given by $(1 - P_c)n - 1$.
- With mutation you change the genes within the same chromosomes.
- With crossover, given a set of possible solutions, you exchange genes within chromosomes to avoid collapsing within a local minimum/maximum.
- Search termination: converging or early callbacks.

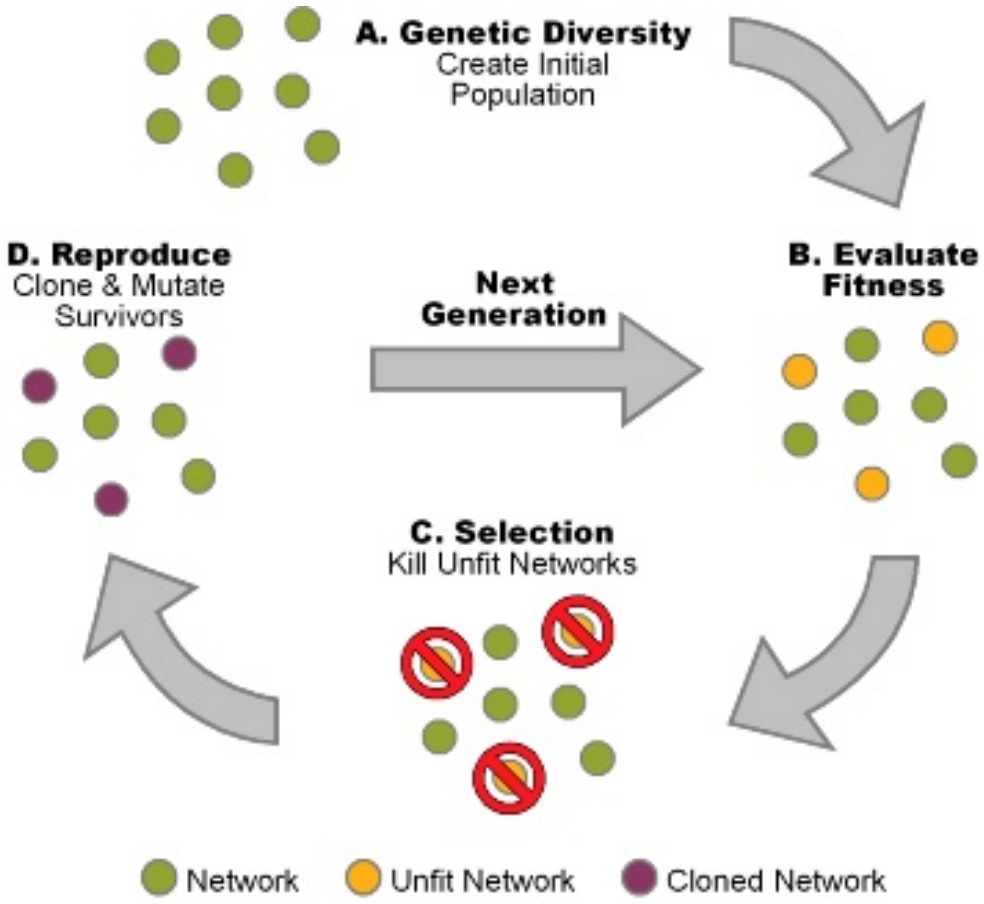

Figure 8: Genetic algorithm, image by Quantdare

4 Dataset used and Exploration

4.1 SIPakMeD dataset

The SIPaKMeD Database consists of 4049 images of isolated cells that have been manually cropped from 966 cluster cell images of Pap smear slides and treated with Pap staining. These images were acquired through a CCD camera adapted to an optical microscope. The cell images are divided into five categories containing normal, abnormal and benign cells. The dataset, downloaded from kaggle.com, shows the following distribution of images and cells.

Category	#Images	#Cells
Superficial/Intermediate	126	813
Parabasal	108	787
Koilocytotic	238	825
Metaplastic	271	793
Dyskeratotic	223	813
Total	966	4049

While the number of individual cells in each category appears balanced - about 800 individual cell images per category - the number of slides containing cells of different types appears unbalanced: slides containing parabasal cells amount to about a third of the number of slides containing metaplastic cells. This could create a class imbalance problem: in fact, the machine will be better able to learn the characteristics of metaplastic cells while leaving out those of parabasal cells. However, since the analysis is conducted on the individual cell and not on the slide as a whole this is not a concern. The effect of class imbalance could be further mitigated by the fact that slides in the categories that show a lower average number of cells per slide (the parabasal category) may devote more space on the slide to each cell and thus there is no crowding of cells on the surface that would make it challenging to spot the characteristics of each.

The dataset not only provides images of slides and cells but also accompanies each of these with several .csv-type text files in which the coordinates in pixels of the vertices of the polygon that contours the cell in the image are given. Such polygons will be called *masks*. Information related to subjects in an image can be also called *annotations*. Overlaying the reference image with the polygon will show that the cell can be perfectly contoured. Depending on the uses, we will transform the masks in different formats.

A sample polygonal mask file can be found below, where each row is a vertex, the first value is the x coordinate, and the second value is the y coordinate.

```
1164,873.63
1141,865.63
1118,862.63
1102,860.63
1088,864.63
1074,860.63
1071,856.63
1056,861.63
1045,872.63
1036,875.63
1019,868.63
1007,864.63
```

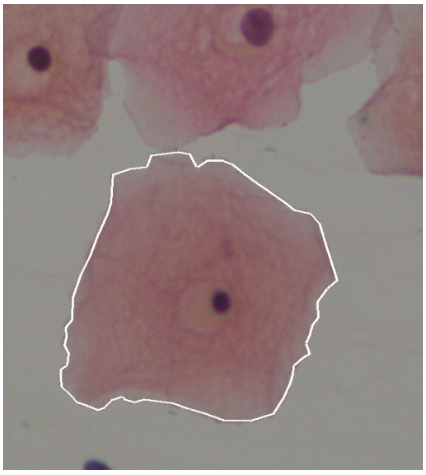

Figure 9: Koilocytotic cell

4.2 CCAgT: Images of Cervical Cells with AgNOR Stain Technique

The CCAgT dataset contains 9339 images with a 1600×1200 resolution where each pixel is 0.111µmX0.111µm from 15 different slides stained with AgNOR technique, with at least one label per image, having more than sixty-three thousand annotations. The images are from patients of Gynecology and Colonoscopy Outpatient Clinic of the Polydoro Ernani de São Thiago University Hospital of the Universidade Federal de Santa Catarina (HU-UFSC).[2]

Each image has a unique name with the format slide id_tile id_microscopy x position_microscopy y position. The slide id to differentiate the slides, and the patients respectively.

The read.me file provides some information about the numbering of images by patient and their diagnosis.

Patient	Diagnostics	Images	Annotations
A	CIN 3	1311	3164
B	SCC	561	911
C	AC	385	11420
D	CIN 3	2125	1258
E	CIN 3	506	11131
F	CIN 1	318	3365
G	CIN 2	249	2759
H	CIN 2	650	5216
I	No lesion	309	474
J	CIN 1	261	1786
K	No lesion	1503	13102
L	CIN 2	396	3289
M	CIN 2	254	1500
N	CIN 3	248	911
O	AC	262	2904

Where *CIN3*, *SCC*, *AC*, *CIN 3*, *CIN1*, *CIN2* are considered abnormal or malign diagnosis and *No lesion* is related to a physiological condition. The class imbalance is evident, which will be adequately addressed in the image elaboration phase. Annotations are stored into a .parquet file, a csv-like format, which stores information about polygon vertex coordinates and their labels like the Pap dataset.

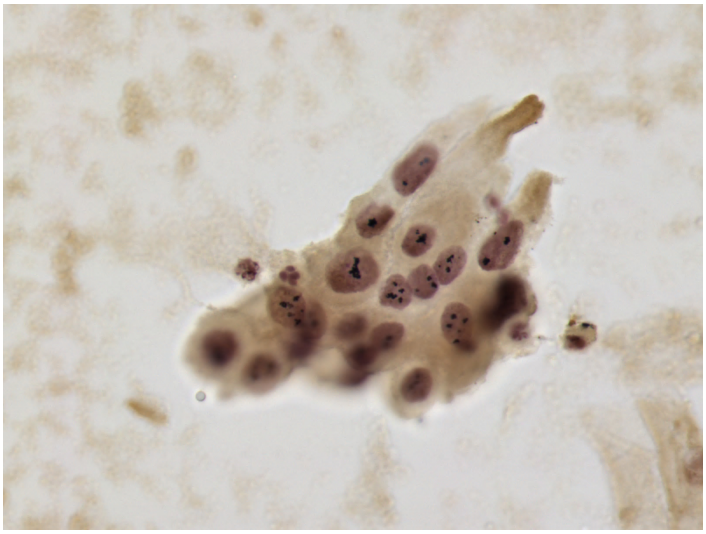

Figure 10: Agnor dataset sample

5 Yolov8n image detection to avoid undersampling

Although the Pap test is recognized by the scientific community as the elective test to recognize abnormalities in the cells of the cervix, the sensitivity of this test is only around 60%. This means that out of 100 women who certainly show abnormalities in the cervix, 40 of them will not receive any diagnosis. So, such cells will not undergo any treatment and are at risk of degenerating into cancerous cells. Some sources associate this low level of sensitivity with errors in sample collection and errors in sample interpretation. **We believe that another reason could be related to a statistical bias linked to the number of cells that are inspected to issue a diagnosis.** The datasets that were used in this thesis are actual slides used in diagnoses. A quick visual analysis shows that no more than a few dozen cells are shown on a slide, so it is on this number that the doctor issues a diagnosis. The number of cells on a square millimeter of skin can be estimated at about 60,000.[10] We believe that it is not possible to make effective generalizations at the population level with such a small sample size: **if the cervix consisted of only 60,000 cells, to make effective generalizations with a 99% confidence level and a 5% margin of error, about 659 cells would have to be analyzed. Thus, for an effective assessment, not only would it take time to analyze 659 cells, but we would have to be sure that the doctor would not make any mistakes in diagnosing each individual cell.**[5] The method proposed below aims to upload to the computer slides in which the number of cells displayed can theoretically be unlimited and the system can automatically highlight cells that need "attention" from the physician. Should the predictive performance of such an algorithm be close to 100% then it would be possible to obtain a precise estimate of the patient's health condition without any human intervention.

Therefore, an algorithm called Yolov8n is proposed, a complex open-source infrastructure of artificial and convolutional neural networks designed to recognize known subjects within different media. This infrastructure performs various computer vision tasks and presents some trained models with more disparate instances of images of cats, dogs, or humans.[48] We cannot expect the model to already know the features of cervical cells; therefore, it will be necessary to train it with ad hoc instances provided by the developer.

It is possible to use such infrastructure through any development interface that supports a recent version of Python or a command-line terminal on a device where the language is installed. Since the model will need a training phase that requires intensive use of computational power, it would be appropriate to use a machine with a quality CPU and GPU. The pro version of Google Colab was purchased for a month (about 12€) a web development interface based on a Jupyter Notebook giving the researchers access to about 150 GBs of secondary memory and GPU with 40GBs of VRAM. The use of such tools meant that algorithm development could take place much faster than not only the basic Google Colab infrastructure but also than could be achieved with an ordinary home computer.[9]

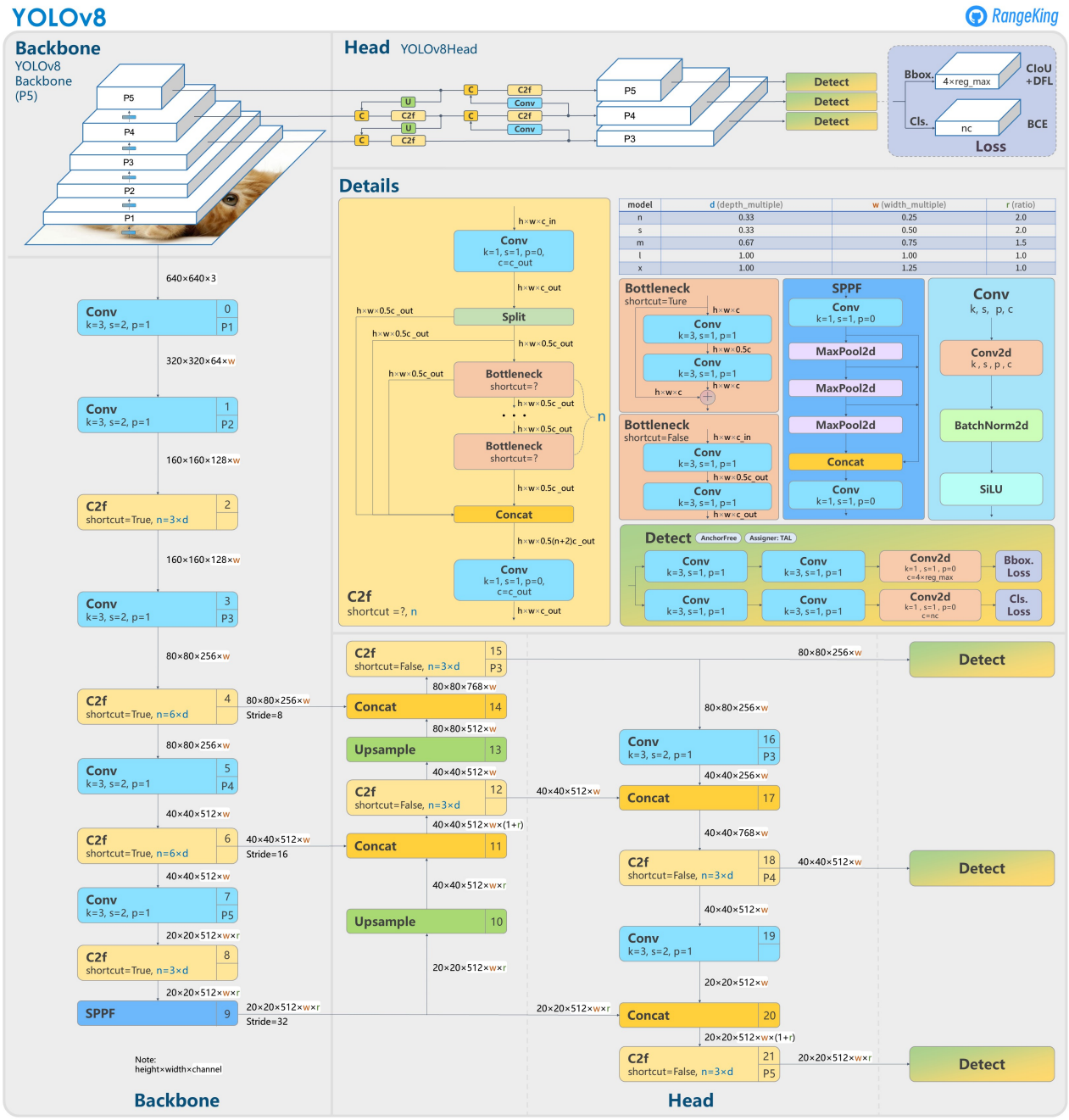

Figure 11: Yolov8n architecture by RangeKing

Yolo needs at least two types of data: the images from which to extract the subject of interest and the respective annotations, or textual documents that report the mask coordinates in a specific format. During the training phase, Yolo will apply a complex convolutional network over the image area contoured by the mask and learn its characteristics, associating the instance with the respective category.

Images and their annotations will necessarily need to have the same name, though obviously with different extensions. The arrangement of image and label files should also follow the tree structure shown in the image below.

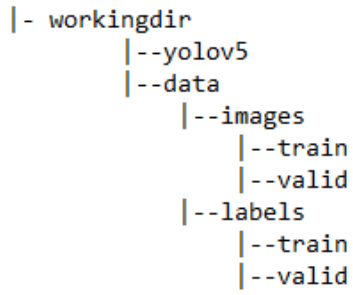

Figure 12: Yolo file structure by Joos Korstanje

While Yolo supports a large number of image and video formats, the annotations files must follow a specific format: each image must correspond to one and only one homonymous text file within which each line refers to a mask. Masks must be rectangular or *bounding boxes*. The first character per line must report the category to which the subject belongs in the image, then a space and four coordinates: the first two will be the b. box coordinates scaled by the size of the image, the last two the width and height of the b. box scaled by the size of the image.

```
0 0.438374 0.789473 0.018978 0.045673
```

The preparatory phase of the dataset consisted of renaming the files through terminal commands so that there would be agreement between image and annotation files. However, there were multiple annotation files for the same image: in fact, each original file referred to the mask of only one cell per image. In short, multiple text files were associated with one image.

Below is bash code used to quickly rename a large number of files according to the yolo requirements.

```
# To easily delete leading zeros
!for FILE in `ls`; do mv $FILE `echo $FILE | sed -e 's:^0*::'`; done

# To add $category$ at the end of the file
!for file in *; do mv "$file" "$(echo "$file" | sed 's/\(.*\)\.\(.*\)/\1
   _$category$.\2/')"; done

# To add .txt file
!for i in *; do mv "$i" "$i.txt"; done
```

The following pseudo code is used create a single text file per image from multiple files with the same reference. It also takes care of transforming the polygonal mask format, first to a Pascal VOC format and then to the Yolo format. The Pascal VOC format extracts the rectangle circumscribed by the polygon by returning four coordinates the minimum and maximum horizontal coordinates and the minimum and maximum vertical coordinates.

```
store the image width in img_width
store the image height in img_height

create a function to create appropriate annotations files whose parameter
   is the category, ranging from 0 to 4
    iterate over each image file and store its index
        create an empty list which will store coordinates from multiple
           files, called row_list
            iterate over original annotation files
                if the file name starts with the same name of the category
                    create a variable which stores the vertex of the
                       polygon, in the format x / y
                    get the minimum x
                    get the minimum y
                    get the maximum x
                    get the maximum y
                    write in the row_list the following row
                        category,  ((x_max + x_min)/(2*img_width)),
                                   ((y_max + y_min)/(2*img_height)),
                                   (x_max - x_min)/img_width,
                                   (y_max - y_min)/img_height])
        write the rows in a .txt file called with the same name of the
           image_index
```

Before the training procedure starts, a configuration file must be created in the working directory *config.yaml* listing the folders where train and test images and labels are located as well as the possible categories to be predicted. As indicated by one of the Yolo developers in one of the discussions on GitHub I initialized the train procedure with 300 epochs.

The training hyperparameter *epoch* is the group of training data that are processed together before the corrections are applied. If the hyperparameter was 3, the model would have adjusted its parameters 3 times after seeing the training dataset 3 times.

```
!pip install ultralytics
from google.colab import drive
drive.mount('/content/drive')
from ultralytics import YOLO
model = YOLO("yolov8n.yaml")
results = model.train(data="/content/config.yaml", epochs = 300)
```

A callback stopped the train procedure at 213 epochs, after a little more than 2 hours, having noticed that for 50 consecutive epochs the improvements in predictive performance were not significant.

Below are presented some interesting images displaying performances of the model.

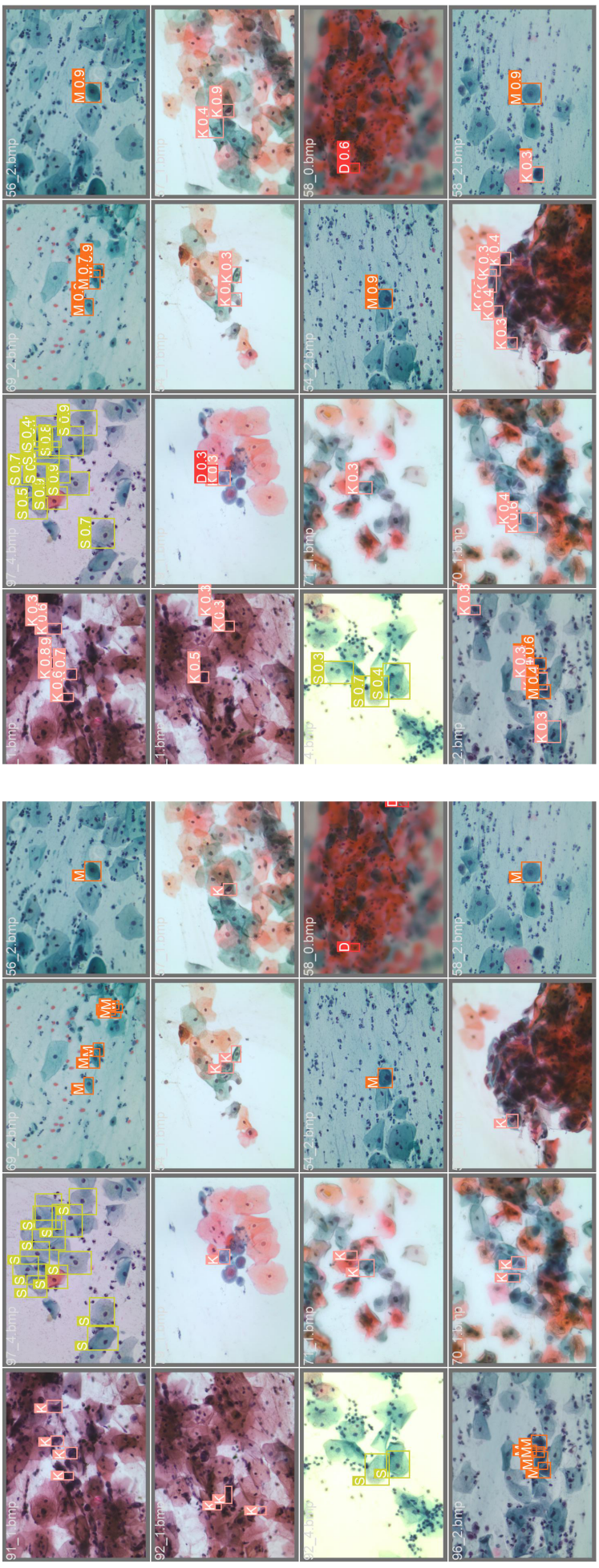

Figure 13: Batch of test images on the left with annotations and predicted instances on the right.

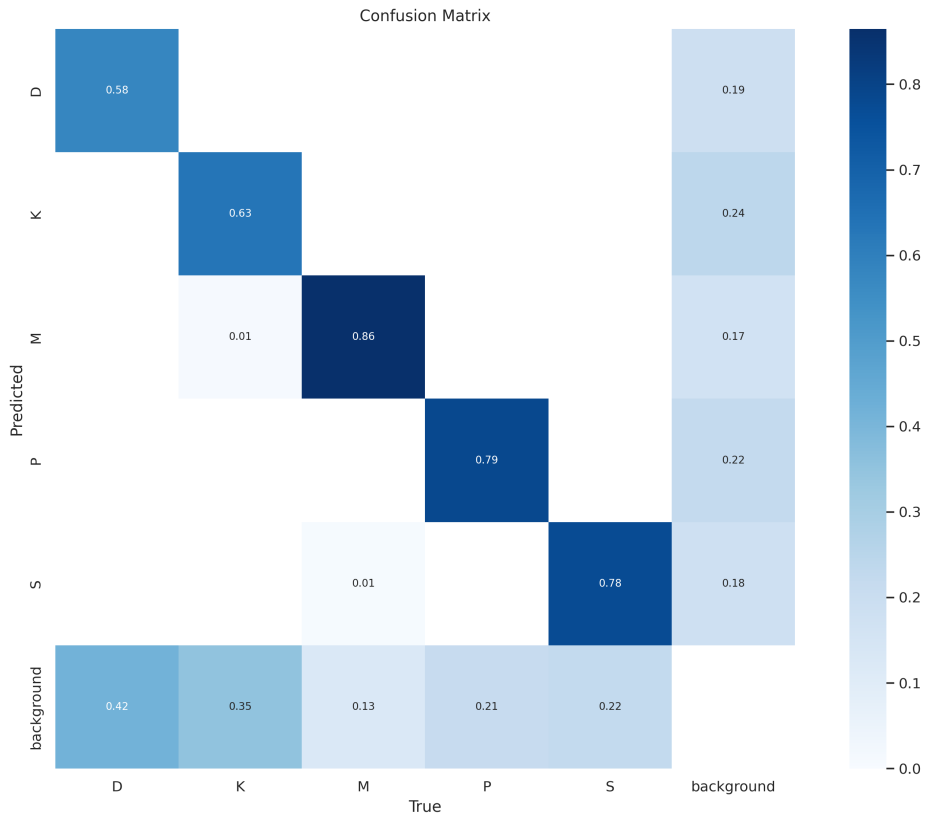

Figure 14: Confusion matrix

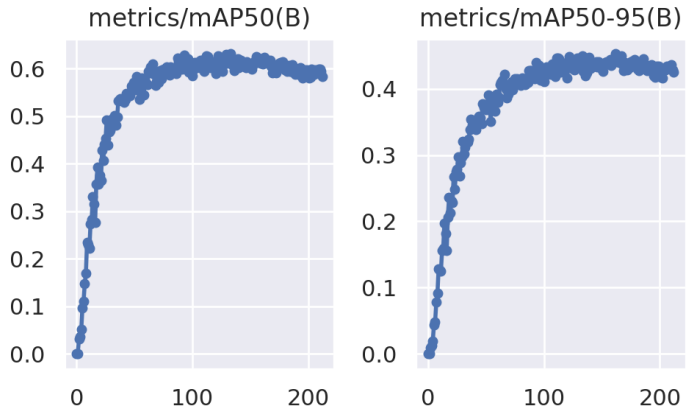

Figure 15: mAP50 and mAP50-95 plot vs. epochs

Figure 11, in the image block on the left, shows a set of images belonging to the test dataset on which the model has not learned what the areas of interest are (athough they are known to us and are shown) but predicted them. The block of images on the right shows the same images with the predicted areas of interest. Image 3.3 shows a Metaplastic cell. In the right section, the same cell is classified as Metaplastic with 90% confidence. In contrast, image 2.1 shows three Koilocytotic cells. The predicted areas of interest do not correspond to the actual ones.

Figure 12 shows the *confusion matrix* for the predictions. While it is not an exhaustive metric for performance, it is a table that shows the predicted and actual labels for each class in the problem. The rows of the matrix represent the true labels, while the columns represent the predicted labels.

The diagonal elements of the matrix correspond to the true positive relative count for each class. All the other numbers in the matrix correspond to misclassification. Metaplastic cells are classified quite efficiently, while Dyskeratotic ones are not.[36]

Figure 13 shows the value of *mAP50* and *mAP50-95* versus the number of epochs. Mean Average Precision (mAP) is a metric used to evaluate object detection models [11]. The mean of average precision (AP) values are calculated over recall values from 0 to 1. mAP formula is based on the following submetrics:

- Confusion matrix has already been explained.
- Intersection over Union(IoU): indicates the proportion of overlapping of the predicted bounding box over the actual bounding box.
- Recall: the proportion of true positives (TP) among all actual positives (TP+FN)
- Precision: the proportion of true positives (TP) over the number of positive predictions (TP+FP)

1. *50* in mAP50, means that the prediction is considered positive if the IoU is $\geq 50\%$.
2. Precision and recall are calculated considering the 50% threshold for all the classes available.
3. Precision is plotted against recall for all the classes available. An inverse relationship between these may be noted.
4. The area under the plots is calculated and averaged resulting in *mAP@50*

The model shows a mAP@50 of 64% which is a satisfying result compared to the scientific literature in the same field. The more restrictive mAP@50-95 results in of 42% which is average.

5.0.1 Conclusions and areas of improvement

A limitation in the approach used has been not to perform **image augmentation**[16]: a technique in which *synthetic* images - or those derived from the original ones - are created by making changes to the original ones e.g., blurring, zooming, rotating, changing colors and scaling. Synthetic images are then added to the original ones. The reason for this best practice in computer vision is that adding some *noise* into the training data could improve the generalization ability of the model. In fact, suppose that in all photos classified as dogs, those portrayed always have yellow fur. The machine will learn that dogs will only be those instances that have a yellow color. Should we introduce a picture of the same dog converted to black and white, the machine will not only learn that some dogs present a vivid yellow color but can also learn morphological features of its shape or edges, regardless of the predominant coat color. Nevertheless Yolo provides an initial layer of image augmentation thanks to the *Albumentations* library, adding blur, adding contrast and converting the images to black and white.

6 AgNOR staining techniques

The scientific literature attributes the restricted use of the AgNOR technique to the lack of standardization in staining process.[47] Yet, there is evidence that the number of AgNORs in cells may be a prognostic factor in some cancer patients. **The predictive ability of the AgNOR number in patients with and without abnormalities in cervical cells will be investigated.**

Part of our investigation will be carried out using a convolutional neural network that will not only aim to categorize images, as in the case of the Yolo approach, but also evaluate if a certain feature shows some correlation with the category the image belongs to.

The basic idea is to subject a set of images and their respective positive/negative labels, or the type of abnormality associated with them, to a convolutional neural network. We expect that given a new image the algorithm will be able to determine whether a cell is associated with a condition.

Part of our investigation will be based on an ad hoc invented machine learning approach, adapted to the specific case by mixing semi-supervised learning and computer vision, with the goal of eliminating human input in cellular diagnosis. Since, however, the positive correlation between AgNOR number and cancer positivity is not certain, whatever the outcome of our research, it will not be possible to ascertain its statistical significance: Even if thanks to the explored technique we were to find a positive correlation, we would not be allowed to conclude that such a correlation. The pipeline explaining this approach will be explained below, and it will be clearer what the reasons for this uncertainty will be.

6.0.1 Visualizing the dataset using t-SNE

T-SNE (t-Distributed Stochastic Neighbor Embedding) is a dimensionality reduction technique that is commonly used for visualizing high-dimensional data. It is particularly useful for visualizing datasets with many variables, such as image datasets.[34]

In the context of image clustering, t-SNE can be used to reduce the dimensionality of the image features extracted from a dataset, which can then be plotted in two or three dimensions. By reducing the dimensionality of the feature vectors, t-SNE can help to identify clusters of similar images in the dataset, which can then be visualized using a scatter plot, where each point represents an image and the position of the point is determined by its feature vector.

t-SNE first constructs a probability distribution that describes the similarity between each pair of high-dimensional data points. Then, it constructs a similar probability distribution for the low-dimensional data points. The algorithm then tries to minimize the difference between these two distributions using a technique called gradient descent.

We expect that, having assumed that images classified as positive have distinctive features compared with images classified as negative, they will be distributed in a two-dimensional scatterplot into two distinct clusters.

```
    import os
import numpy as np
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
from PIL import Image

X = []
filenames = []
for filename in os.listdir(image_folder_path):
    img = Image.open(os.path.join(image_folder_path, filename))
    X.append(np.array(img).flatten())
    filenames.append(filename[:1])
X = np.array(X)

tsne = TSNE(n_components=2)
X_tsne = tsne.fit_transform(X)

num_colors = len(list(set(filenames)))
color_map = dict(zip(unique_filenames, range(num_colors)))
colors = [color_map[filename] for filename in filenames]

plt.figure(dpi = 300)
plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=colors)
for i, filename in enumerate(filenames):
    plt.annotate(filename, (X_tsne[i, 0], X_tsne[i, 1]))
plt.show()
```

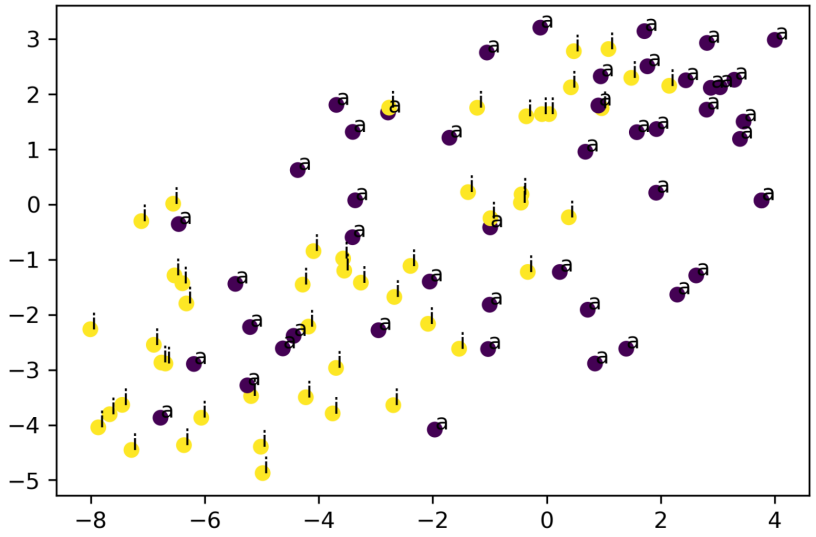

Figure 16: Images distribution after t-SNE

Although the division into distinct clusters did not occur, it would appear, however, that in the lower left corner of the plot, images classified as I (patients showing no lesions, accumulate, while in the upper right corner at the farthest, and therefore most different, point from the first cluster, images classified as A are concentrated. This evaluation, however, may be a consequence of confirmation bias.

Conclusions:

- **Two dense clusters of images A and I are arranged at the extremes of the scatterplot, suggesting that the t-SNE algorithm detects appreciable differences in their features. On the other hand, some of the images are mixed together.**

- 100 images is not an exhaustive sample, though they were randomly selected, to capture the variability of a dataset as large as the one available to us. Increasing the number of samples may lead the clusters to be marked more distinctly.

- At row 10 of the code provided above a *flatten* transformation allows one to arrange along a one-dimensional array an image that in its two-dimensional nature. such a transformation, some of the crucial information contained in an image, such as the arrangement of points in it, is lost. Methods for identifying features that preserve the two-dimensionality of the image will be used later.

- For the reason just given however, **it is not necessarily the case that if after a transformation t-SNE two distinct classes are not formed, the starting images will all be similar to each other**: different techniques will be used that enable the differences in each image's features to stand out. In our case, principal component analysis returned significantly worse results than t-SNE.

6.0.2 Assessing the predictive performance of silver-treated cell images using CNN

Suppose for a moment that the number of AgNORs in a cell has no correlation with the presence of lesions in the cervix. We remain interested, however, in investigating if the characteristics of a cell exist and which ones show correlation with the presence of abnormalities.

The architecture of a convolutional neural network, as already explained in the appropriate section, is able to create filters that best uncover and summarize the features of an image related to the category to which it belongs. This approach does not explain what the determining features are though it does detect some predictive ability of each image with respect to the label to which it belongs. **Such a limitation i.e., the well-known black box problem, the neural network architecture in fact is not able to provide high interpretability of the results: it is only possible to determine whether an image has predictive power, not which features give it this ability. While we believe it is of great interest to understand what the determining features are, we also believe on the other hand that it is much more useful to focus our efforts on establishing whether a cell is positive and not the reason behind that classification.** For this reason, a convolutional neural network will be implemented that can classify cell images as positive or negative, depending on whether the patient has been diagnosed with a lesion or not.

As per good practice, image augmentation will be implemented thanks to the *ImageDataGenerator* function in *from tensorflow.keras.preprocessing.image import ImageDataGenerator* library: a technique that consists of feeding the convolutional neural network not only the original images but also rotated, cropped, or zoomed versions in order to better generalize the characteristics of those images.

Tensorflow and keras were used to implement the convolutional neural network, and its hyperparameters and structures were adapted to the specific case. In fact, "traditional" artificial neural networks are not very robust with respect to changing input characteristics such as image size and the expected label. For this reason, all images were scaled to a 90 by 90 pixels.

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator

read the train labels stored in an excel files. Possible train labels are
   1 (lesions) and 0 (no lesions)
rename columns and transform them into string
repeat for test instances

train_img_dt_gen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)
```

```
test_datagen = ImageDataGenerator(
    rescale=1./255)

train_data_generator = train_datagen.flow_from_dataframe(
    dataframe=train_image_data_df,
    directory='/Users/Vincenzo/PycharmProjects/Tesi/agnor/
       agnor_x_train_resized_1',
    x_col='image',
    y_col='count',
    target_size=(90, 90),
    class_mode='categorical',
    batch_size=8)

test_data_generator = test_datagen.flow_from_dataframe(
    dataframe=test_image_data_df,
    directory='/Users/Vincenzo/PycharmProjects/Tesi/agnor/
       agnor_x_test_resized',
    x_col='image',
    y_col='count',
    target_size=(90, 90),
    batch_size=8,
    class_mode='categorical')

model = keras.Sequential([
    keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(90,
       90, 3)),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Conv2D(64, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Conv2D(128, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Flatten(),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')])

model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

```
model.fit(
    train_data_generator,
    epochs=50,
    validation_data=test_data_generator)
```

The dataset used for this model consisted of about one hundred images, equally distributed among instances classified as "A" if the patient showed cellular abnormalities, or "I" if the patient received a negative diagnosis. 80% percent of the images, randomly selected, were used to train the model, while the remaining 20% remained unstratified and were used to to test its performance. *Accuracy* on the test set was the metric used to evaluate model performance. Accuracy is the ratio of the number of correct predictions to the total number of predictions made by the model. For example, if a model made 100 predictions and guessed 80 correctly, its accuracy would be 80%[35].

While we recognize that neural networks can provide consistent results as the number of inputs increases, and therefore, 100 images is definitely a small number, it was possible to build a model that returned an accuracy on the test of about 71%.

While we are unable to determine here which are the characteristics of the images the machine uses to deduce their category of membership, such high accuracy in a model that has little data suggests that the images exhibit distinctive features.

6.0.3 Assessing the predictive performance PAP-treated cell images using CNN

To compare the predictive performance of the AgNORs dataset, we subjected 100 images from the SIPakMeD dataset to a convolutional network mostly identical to the previous one. Half of these represented cells were considered normal, and the other half were considered abnormal. Given, however, the substantial difference in resolution between the two datasets and that was necessary to scale the Pap-treated images to 400 by 400 pixels, for the same subject compared to the AgNORs image, we had 5 times the amount of data. **An ideal comparative analysis should have been conducted by comparing images of the same cells under two different treatments photographed at the same resolution.** The data augmentation performed, and the structure of the convolutional neural network was identical to that used in the dataset except for the input size of 400 by 400 by 3.

The accuracy on the test detected is 90%: an extremely satisfactory result and much higher than that of AgNOR.

6.0.4 Naively assessing AgNORs number predictive performance using the correlation coefficient

For the purpose of our analysis, about 100 images belonging to two patients with two different diagnoses were randomly selected Patient A had a CIN3-type lesion, while Patient I had no lesion. Approximately 50 cell nuclei were extracted from each image by an algorithm whose logic is superimposed on that used to crop Pap-treated cells by polygonal annotations.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import pyarrow
import cv2

data = pd.read_parquet("CCAgT.parquet.gzip")
coordinates = data[data['image_name'] == 'I_20321_-250560_158760']\
                                                    .geometry.tolist()
img = cv2.imread('I_20321_-250560_158760.jpg')

temp_coord = []

for coord in coordinates:
    temp_coord.append(coord[10:][:-2])

dictionary = {}
for coord in range(len(temp_coord)):
    dictionary[coord] = temp_coord[coord].split(',')

for i in dictionary:
    for j in range(len(dictionary[i])):
        x, y = dictionary[i][j].strip().split(' ')
        dictionary[i][j] = [x, y]

    dictionary[i] = pd.DataFrame(dictionary[i])
    dictionary[i] = dictionary[i].astype(float).astype(int)
    dictionary[i] = [int(float(dictionary[i][0].min())),
                     int(float(dictionary[i][1].min())),
                     int(float(dictionary[i][0].max())),
                     int(float(dictionary[i][1].max()))]

cropImg = RGB_img[dictionary[0][1]:dictionary[0][3],
                  dictionary[0][0]:dictionary[0][2]]
```

The extracted images, of different sizes, had a format of the type *a_1*, where "a" is the patient identifier and 1 meaning "first photo". Patient label A, by transitivity is associated with the CIN3 type lesion, while patient label I is associated with a lesion-free condition.

Figure 17: a_1 figure

Each image was manually displayed within an Excel file were and the AgNORs contained in each were numbered. An additional column with the type of diagnosis was added, taking value 1 in the case of CIN3 lesion and 0 in the case of no lesions.

The resulting dataset, therefore, had the following format:

Image	**AgNORs number**	**Diagnosis**
a_1	7	0
a_2	4	0
i_1	0	1

Table 1: Dataset example

The dataset was then imported into Python and was processed using the point biserial correlation function of the *scipy.stats* library. Point biserial correlation is used to measure the strength and direction of the association that exists between a continuous variable, in our case, the number of AgNORs, and a dichotomous or binary variable, lesion or not. It is a special case of Pearson product-moment correlation.[39]

The assumptions underlying this instrument are that there are no outliers in the continuous variable for each of the dichotomous variable categories. The continuous variable must be approximately normally distributed for each category of the binary variables and the the continuous variable must have an equivalent variance for each category of the binary variable. In the example dataset, it is assumed that these conditions are met.

The function or correlation returns not only the correlation coefficient between the two variables but also the p-value. The correlation coefficient is a measure of the degree of relationship between two variables.[55] The p-value measures with what confidence we can establish that the correlation coefficient was the result of chance. A p-value below the 0.05 threshold suggests that the null hypothesis must be rejected, or that the correlation between the two variables does not exist.[14]

The calculated correlation coefficient is 0.6, indicating a slight positive correlation, and the p-value is equal about $3.899061236279749e^{-11}$. **Based on our analysis, the number of AgNORs influences the diagnosis.**

```
import scipy.stats as stats
corr, p_value = stats.pointbiserialr(df[0], df[1])
print("Point-biserial correlation coefficient:", corr)
print("p-value:", p_value)

>>> Point-biserial correlation coefficient: 0.6082325426945231
>>> p-value: 3.899061236279749e-11
```

6.0.5 Optimized Derived Features Semi-Supervised Learning: the General Approach

Semi-supervised learning is a machine learning technique that aims to find statistical correlations between data in which there is no label. It will be up to the developer to manually assign the label to several instances, with the possibility of human error and the downside of being a time-consuming task. Only after a subset of data is classified will a classification algorithm learn the correlation between the characteristics of the labeled samples and their labels; therefore, the unlabelled subset will be associated with a predicted label.

In our specific case, it would be the doctor, at the time of each analysis, who would have to manually label a substantial portion of their samples from which to make inferences. In this specific case, the labeling would consist in finding, for each image, the AgNORs number.

The Optimized Features Extracted Semi-Supervised Learning is an approach used to extract features from an unlabelled data and based on these, compute its label using an unsupervised learning algorithm, whose parameters are optimized thanks to an optimization algorithm and no intervention from the final user.

This an approach still requires human intervention. However, its consistency is shifted from the final user, whose human input would be required for every analysis, to the developer alone, who is committed to supervision only once, on the occasion of training the labeling machine with benefits including time and cost savings and a decrease in human errors.

The phases within this technique are as follows:

1. **Semi-Supervision Phase:** A sample of 10% of the images is extracted from the dataset, which will be manually labeled by the developer. In our case, the developer will manually count the number of AgNORs in the nuclei, as noted in the "Naively assessing AgNORs number predictive performance" section.

2. **Feature Extraction Preprocessing:** The labeled images will be subjected to a series of transformations and subsequent unsupervised learning algorithms which will predict the label we have already derived before. **The order of transformations and their parameters will affect the prediction.** In our case, we subject the image to a series of graphic transformations: conversion to black and white, alteration of the threshold, blurring, and contrast enhancement. Such transformations are widely used to mark the differences between pixels having different colors in the same image. Indeed, recall that AgNOR-stained cells will show black AgNORs over a brown cytoplasm.

3. **Feature Extraction.1:** The transformed image will be subjected to a clustering algorithm such as k-means to achieve segmentation of the image or *the division of the image into portions that share the same characteristics*: usually the image of a cell treated with AgNOR will show three dominant colors: the white background, the brown of the cytoplasm, and the black of the nuclei. Ideally, the algorithm will divide the image into three parts according to the dominant colors.

4. **Feature Extraction.2:** The image returned, therefore, will be an array of three values of the same size as the source image: if a pixel was clustered in the brown cluster, we expect it to have the label "B".

5. **Feature Extraction.3:** The library *scikit-image* enables the number of *connected components* to be found: a connected component is defined as a group of pixels that share the same feature and are connected to each other either directly or indirectly. Directly connected pixels are those that are adjacent to each other, either horizontally or vertically, while indirectly connected pixels are those that can be reached from each other through a path of adjacent pixels. The connected components are just the number of AgNORs in the nucleus of the cell.[28]

6. Once the labels have been predicted, they are compared with the actual ones using an error metric, like MAE.[42]

7. This pipeline will be passed into a genetic algorithm that minimizes the error metric to finetune the parameters, finding the ones that are better and deriving the feature of the images.

8. **Classification problem:** Finally, we set up a regression between two values: the number of AgNORs and whether the cell is positive. In case the correlation is positive then as the number of AgNORs increases, the possibility that such a cell is abnormal also increases.

Suppose the transformations applied to the image and the clustering algorithm detects an incorrect AgNOR number. The correlation detected between the AgNOR number and positivity has no statistical significance.

As the labeling algorithm by definition is a set of fixed rules, it cannot chnge its parameters according to the specific case. In fact, it is possible that a certain parameter used for the transformation of a correctly labeled image will return a false number of AgNOR in another cell.

Therefore, an additional learning step was resorted to by implementing a labeling model based on a genetic algorithm to optimize the parameters of the transformations and the clustering to reduce the difference between the detected number of AgNORs and the actual number.

Having trained the labeling machine, several other subsequent classification models have been developed, which will be further investigated.

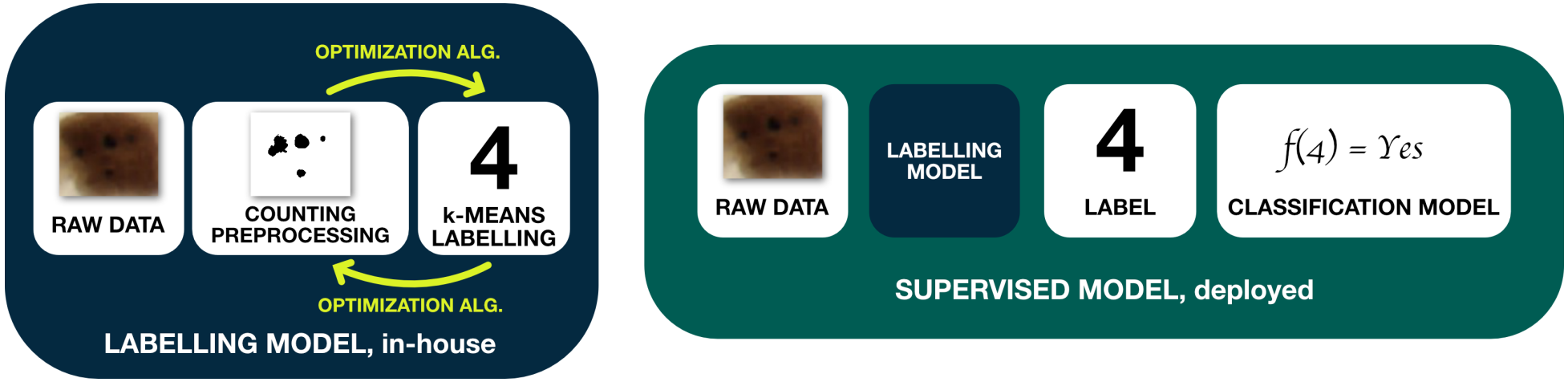

Figure 18: Optimized Derived Feature Semi Supervised Learning

6.0.6 Optimized Derived Features Semi-Supervised Learning: building the labeling machine using a clustering algorithm

The labeling machine aims to assign labels or extract features from images automatically without human intervention. What we are interested in extracting from the images is the AgNOR number. Thus, it consists of an unsupervised learning model optimized by a genetic algorithm to best generalize its parameters based on one-step supervision by the developer.

A good rule of thumb for working with images involves subjecting them to certain transformations to bring out the most important features: usually, these are applying Gaussian blurs to increase contrast or threshold transformations. Such transformations can be implemented thanks to the Python OpenCV library. Each operation can be and was performed thanks to a function whose arguments change the resulting image. The order of the transformations also affects the final image. Furthermore, it is mainly the source image and its characteristics such as contrast and color that can make the result deviate from the expected one.

The pipeline of the transformation and extraction of the number of AgNORs sequentially is enclosed in a function whose steps are as follows:

1. K-means clustering[26] algorithm aimed at extracting the main colors from the images. This algorithm involves as a parameter the number of segments or colors into which to group the image pixels. The reason behind using this algorithm is that AgNORs take on a distinctive black color compared to the beige and brown of the other elements in the picture.

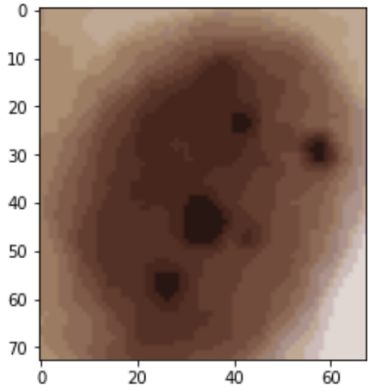

Figure 19: a_1.jpg after K-means transformation

2. The application of a threshold filter:[29] this filter converts in in high contrast black and white images. The threshold the parameter of such a transformation specifies what is the minimum intensity of each pixel's value to be converted to white, otherwise to black. The reason behind such a transformation is increasing the contrast between the black color of the needles and the brown background of the cytoplasm. In fact, we expect that at a certain threshold the black pixels will become white and the brown pixels will become black.

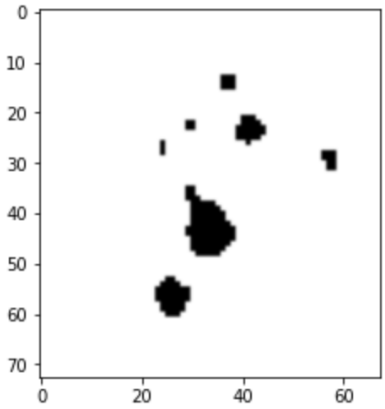

Figure 20: a_1.jpg after K-means transformation and threshold

3. Once a high-contrast, black-and-white image is obtained, it can be converted into a binary matrix of zeros and ones, zeros when the image is black and ones when is white.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

4. The *scipy.ndimage.label* function is used to identify and label separate features or objects in an array or image[38]. It does so by grouping together contiguous pixels or voxels with the same value and assigning each group a unique integer label. After such a transformation, therefore, we will obtain the number of connected components in an image.

$$\text{Number of components} = 7$$

5. The pipeline is repeated for each image in the dataset. The resulting data frame, therefore, will consist of a three-column list: one column dedicated to file identification and one column dedicated to the number of AgNORs.

Image	**AgNORs number**	**Derived AgNORs number**
a_1	7	4
a_2	4	4
i_1	0	1

Table 2: Resulting dataset

6. Thanks to the *mean_absolute_error* (MAE) function of the *scikit-learn* library[33], it is possible to obtain metrics of the difference in the predictions. MAE is calculated as the sum of absolute errors divided by the sample size:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

MAE can be interpreted as the average of the absolute differences between the true and predicted values for all samples.

7. The function containing the transformation pipeline will return the MAE.

```
def transformations(k, sigma):
    [...]
    return MAE

print("MAE is: ", transformations(14, 30))
>>> MAE is: 1.46
```

The code for the pipeline is available below:

```
from sklearn.metrics import mean_absolute_error as mae
from scipy.ndimage import label
import pandas as pd
import numpy as np
import cv2
import os

def function(k, sigma):
    lista = []

    for filename in os.listdir(directory_path):
        cropImg = cv2.imread('/Users/Vincenzo/PycharmProjects/Tesi/agnor/
            agnor_x_train/'+filename)

        pixel_vals = cropImg.reshape((-1,3))
        pixel_vals = np.float32(pixel_vals)

        criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER,
            100, 0.85)

        _, labels, centers = cv2.kmeans(pixel_vals, k, None, criteria, 10,
            cv2.KMEANS_RANDOM_CENTERS)
```

```
        centers = np.uint8(centers)
        segmented_data = centers[labels.flatten()]

        segmented_image = segmented_data.reshape((cropImg.shape))


        segmented_image = cv2.cvtColor(cropImg, cv2.COLOR_RGB2GRAY)
        _ ,thresh1 = cv2.threshold(segmented_image,sigma,255,cv2.
            THRESH_BINARY)
        new_arr = np.where(thresh1 == 255, 0, 1)

        structure =  [[1, 1, 1], [1, 1, 1], [1, 1, 1]]
        _, ncomponents = label(new_arr, structure)

        filename = filename[:-4]
        lista.append([filename, ncomponents])

    results_df = pd.DataFrame(lista)
    new_df = pd.merge(results_df, dataframe, on = [0])
    return mae(new_df['1_y'], new_df['1_x'])

mae = function(14, 30)
```

A genetic algorithm is implemented in order to find the parameters of the function which minimize the mae. The pseudocode of the GA is available below:

```
import random
population_size = 100
mutation_rate = 0.1
num_generations = 50

def generate_population(size):
    create a population of solutions, each composed by two parameters
       which will be randomly selected in a predefined range

def evaluate_population(population):
    compute the fitness of each individual in the populations

def select_parents(population, fitness_score):
    select two parents from the population using a weighted random
       selection method. The probability of an individual being selected
       as a parent is proportional to their fitness score

def mutate(individual, mutation rate):
    mutates an individual by randomly adding or subtracting 1 from each of
        its attributes x and y with a probability of mutation_rate

def crossover(parents):
    generates a new child individual by taking the average of the x values
        of its parents as its own x value, and taking the average of the y
        values of its parents as its own y value

generate the initial population
loops through each generation:
    evaluate the fitness of each individual in the population
    choose the best_individual
    loops through the rest of the population:
        select other individuals, mutate and crossover them and create the
            new population
```

The genetic algorithm converges to an optimal solution of (14,30), with a fitness of 1.34.

Once the predicted labels were obtained for each image, a convolutional network was constructed aimed at classifying the images and thus, predicting the number of AgNORs. The convolutional neural network will, therefore, have in its last dense layer a number of neurons equal to the number of labels to be predicted; in our case, we had images belonging to 8 categories as many as the number of diagnoses

present in each image. It is worth noticing, however, that the number of possible labels in the training set must match with the number of possible labels in the test. This means that if in the training set we had images that have a number on AgNORss numbers from 0 to 10, there must be at least one image in the test with which at least one of the training labels is associated.

Unfortunately, having tested this model, its accuracy is not satisfactory, amounting to about 25 percent.

```
train_data_generator = train_datagen.flow_from_dataframe(
    dataframe=train_image_data_df,
    directory='/Users/Vincenzo/PycharmProjects/Tesi/agnor/
       agnor_x_train_resized_1',
    x_col='image',
    y_col='class',
    target_size=(90, 90),
    batch_size=8,
    class_mode='sparse')

repeat for the test set data generator

model = keras.Sequential([
    keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(90,
       90, 3)),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Conv2D(64, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Conv2D(128, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Flatten(),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(8, activation='softmax')])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(
    train_data_generator,
    epochs=50,
    validation_data=test_data_generator)
```

7 Data Science and Computer Vision for COVID-19 Diagnosis based on Chest X-Ray

Countries around the world are still feeling the effects of the pandemic after more than three years. COVID-19 was the greatest significant public health disaster in more than a century, causing a global economic crisis with a lasting impact throughout global society. COVID-19 continues to claim lives, many people are enduring long-term repercussions because of the virus, and health-care systems are attempting to recover from the tremendous disruption. These negative consequences highlight the need for smart investments to strengthen the resilience of healthcare systems – to protect underlying population health, fortify health system foundations, and strengthen front line health workers – giving countries the agility to respond not only to evolving pandemics but also to other shocks, whether natural or manmade. [19] [20]

7.0.1 Introduction

COVID-19 vaccination has substantially altered the course of the pandemic, saving tens of millions of lives globally. Lidia Fonseca, Chief Digital and Technology Officer for Pfizer, the biggest manufacturer of COVID-19 vaccines, stated that they leveraged digital innovation to accelerate their COVID-19 vaccine development. Given the necessity to move the vaccine clinical trial forward quickly as possible as while maintaining the safety and well-being of the researchers and clinical trial participants, among other solutions, they used artificial intelligence and machine learning to more swiftly quality-check and analyze the trial's massive volumes of data.[22]

Therefore, it is clear that artificial intelligence and machine learning are tools used by even the biggest companies in healthcare to develop their products and are already having massive, positive effects on our lives.

A further advanced application of machine learning in the fight against COVID-19 is proposed below: chest X-ray scans of healthy patients and patients with COVID-19 will be used with similar logic to that presented in the main body of the research. An advanced convolutional neural network will identify which features are recurrent in each class of scans and the predictive capabilities of the model will be assessed. Unlike Pap smear slides, it is particularly complicated for the author to identify the distinguishing features of each class by eye, reducing the possibility of confirmation bias.

Unlike the style of discussion in the main body of the research, in this appendix, more emphasis will be placed on the code, explaining why certain technical solutions were adopted rather than others.

7.0.2 Data Analysis

The development environment was Google Colab with Python 3.10. The code was written with a view to ensuring maximum reproducibility.

```
if not os.path.isdir("dataset"):
  os.mkdir("dataset/")
  os.mkdir("dataset/train")
  os.mkdir("dataset/test")
  os.mkdir("dataset/train/covid")
  os.mkdir("dataset/train/notcovid")
  os.mkdir("dataset/test/covid")
  os.mkdir("dataset/test/notcovid")
```

The first code block creates a "dataset" directory, with subdirectories "train" and "test", containing 75% and 25% of the observations respectively. In each of the subdirectories, a further branch is created, which distinguishes between images classified as "normal" and those classified as "covid", respectively those depicting plates of healthy and infected patients respectively.

```
!git clone https://github.com/ieee8023/covid-chestxray-dataset.git
```

The *git clone* command executable from the terminal using the shortcut *!* allows you to clone a GitHub repository locally.

The *covid-chestxray-dataset* repository of user Joseph Paul Cohen a.k.a. *ieee8023* aims to create an open public dataset of chest X-ray and CT images of patients with positive or suspected of COVID-19 or other viral and bacterial pneumonias (MERS, SARS, and ARDS). Data were collected from public sources and through indirect collection from hospitals and physicians. This project is approved by the University of Montreal Ethics Committee #CERSES-20-058-D [7].

The repository also has a metadata.csv file that contains accompanying information on the images, of which it not only specifies features but also provides information on the patient's state of health like their vital parameters and clinical notes.

Cleaning of the dataset was achieved by:

- Reformatting the columns of interest, mainly by manipulating the strings.
- Dropping the columns not used.
- Filtering the observations related to COVID-19 patients and frontal images.

Figure 21: Chest X-ray of a Covid-19 positive patient

```
df = pd.read_csv("/content/covid-chestxray-dataset/metadata.csv")

df["location"] = df["location"].str.rsplit(',', n=1)
                    .str[-1].str.rstrip().str.lstrip()
df["finding"] = df["finding"].str.rsplit('/', n=1)
                    .str[-1].str.rstrip().str.lstrip()

df = df[(df["finding"] == "COVID-19") & (df["view"] == "PA")]

df.drop(columns = ["doi", "url", "license",
                    "Unnamed: 29", "date", "view"],
    inplace = True)
```

The dataset consists of 24 columns and 196 observations, one for each image in the dataset. The images refer to 139 unique patients. **It may be noted that there are twice as many images of men than women, so we may conclude that COVID-19 affects men twice as frequently as women. Given the small sample size, it will not be possible to draw statistically relevant conclusions on the univariate analysis that follows, so nothing can be concluded about this disproportion.**

Table 3: Dataset information

#	Column	Non-Null Count
0	patientid	196 non-null
1	offset	169 non-null
2	sex	177 non-null
3	age	136 non-null
4	finding	196 non-null
5	RT_PCR_positive	192 non-null
6	survival	69 non-null
7	intubated	58 non-null
8	intubation_present	56 non-null
9	went_icu	92 non-null
10	in_icu	76 non-null
11	needed_supplemental_O2	27 non-null
12	extubated	10 non-null
13	temperature	26 non-null
14	pO2_saturation	44 non-null
15	leukocyte_count	6 non-null
16	neutrophil_count	13 non-null
17	lymphocyte_count	15 non-null
18	modality	196 non-null
19	location	173 non-null
20	folder	196 non-null
21	filename	196 non-null
22	clinical_notes	156 non-null
23	other_notes	77 non-null

There are numerous null values in the dataset, which are particularly concentrated in the columns relating to white blood cell counts (*leukocyte*, *neutrophil*, *lymphocyte*). Furthermore, there are no observations with no null values. For the time being, however, these items will not be addressed.

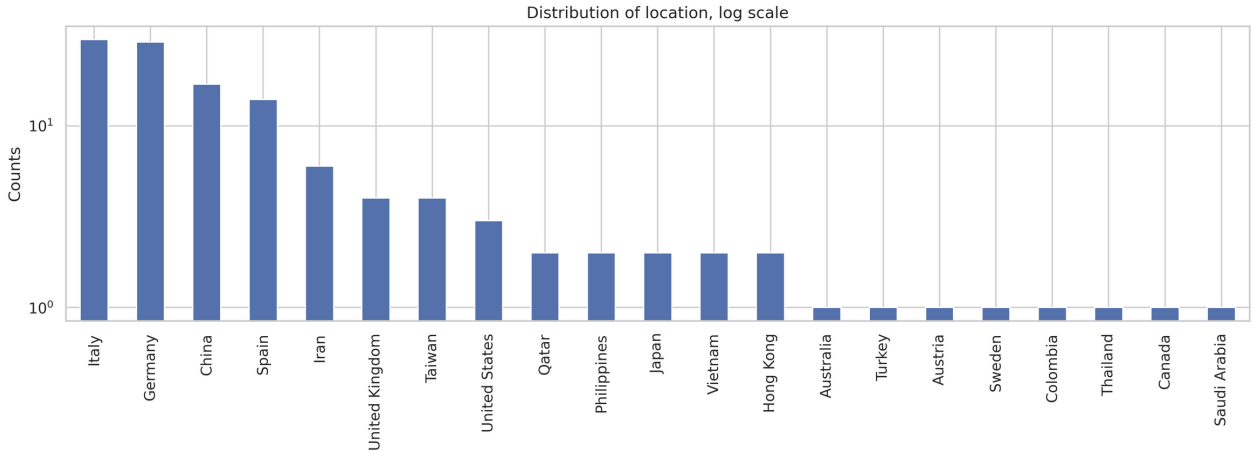

Figure 22: Log distribution of location

Most of the images come from Italian (30), German (29), Chinese (17) and Spanish (14) hospitals.

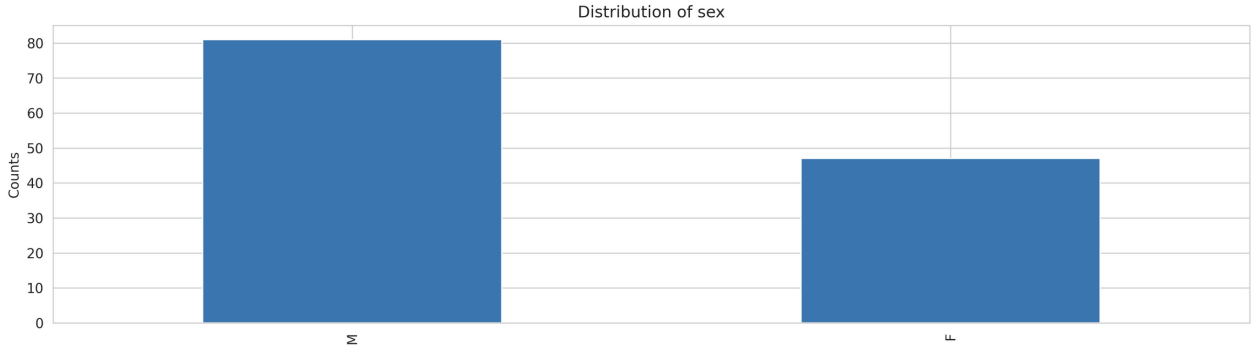

Figure 23: Distribution of sex

Images relates to 81 males and 47 females.

Most of the patients were between 60 and 80 years old, with a mean age of 55.50 and a median age of 56. While the maximum age is 84, the minimum age is 20, with only one case. Although this is an exception, the fact that even a young patient required an X-ray suggests that not all young people experience only mild symptoms when infected by Covid-19.

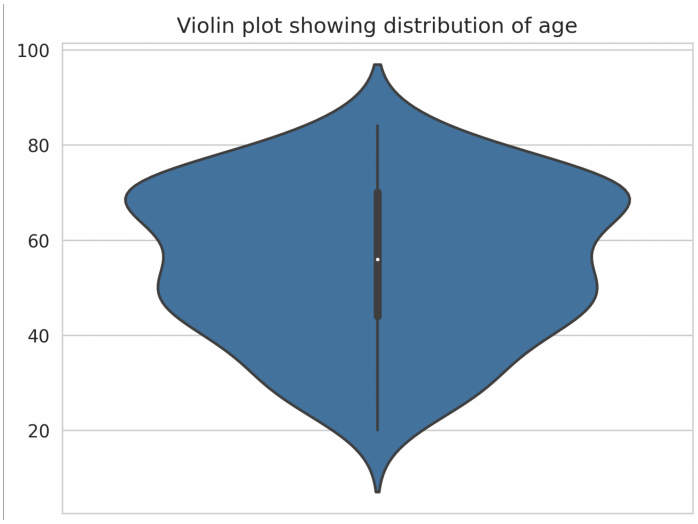

Figure 24: Distribution of age

7.0.3 Natural Language Processing on Clinical Notes

Of particular interest is the "clinical notes" column, which gives the patient's anamnestic picture, i.e. their clinical condition, in human language. The following is an example of a clinical note reported in the dataset:

> *On January 22, 2020, a 65-year-old man with a history of hypertension, type 2 diabetes, coronary heart disease for which a stent had been implanted, and lung cancer was admitted to the emergency department of Cho Ray Hospital, the referral hospital in Ho Chi Minh City, for low-grade fever and fatigue. He had become ill with fever on January 17, a total of 4 days after he and his wife had flown to Hanoi from the Wuchang district in Wuhan, where outbreaks of 2019-nCoV were occurring. He reported that he had not been exposed to a "wet market" (a market where dead and live animals are sold) in Wuhan. Progressive infiltrate and consolidation*

There are several clinical notes for the same patient, each reporting verbatim the previous one, plus any updates. In order to avoid duplication, it was decided to keep only the last clinical note for each patient: thus, 107 clinical notes exist for the same number of patients.

A **word cloud** is a visual representation of text data, where words are displayed in different sizes based on their frequency or importance in the given text. It is a popular technique used to analyze and present textual information in a concise and visually appealing manner. This technique developed in the field of *natural language processing*, will be used to analyze the most frequent terms in each patient's clinical notes.[45]

```
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS
nlp_df = df.dropna(subset="clinical_notes").drop_duplicates("patientid",
                                                            keep = "last")
                                                            ["clinical_notes"]
                                                            .reset_index(drop = True)


join_nlp_df = ' '.join(nlp_df)

stop_words = STOPWORDS.update(["COVID", "patient", "day",
                               "history", "performed", "days",
                               "showed"])

wordcloud = WordCloud(width=1920,
                      height=1080,
                      stopwords = stop_words,
                      normalize_plurals=True,
                      background_color='white').generate(join_nlp_df)

plt.figure(figsize=(16, 9), dpi = 300)
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```

The different clinical notes are merged to calculate the frequency of each term. Thus, stopwords are removed from the corpus: those frequently used words that will be ignored in the word cloud, such as prepositions, articulations and conjunctions. In addition to common English stopwords, words that are expected to be very frequent, such as "COVID" and "patient", will also be removed.

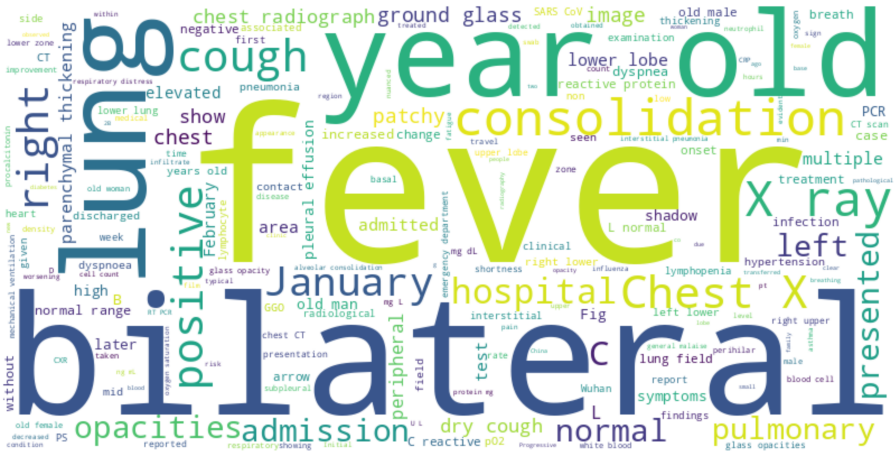

Figure 25: Wordcloud of clinical notes

Of particular interest is the word "fever", one of the most frequent symptoms of COVID-19 disease, but also "old", which might suggest that patients are more frequently elderly people, as well as "bilateral", a term referring to pneumonia, one of the most frequent symptoms in hospitalized patients.

Based on the doctors' impressions and thus, on the description of the patient's clinical condition in human language, an algorithm will be created to determine whether a patient will be intubated. The analysis that follows will only be notional, as we are subject to numerous limitations:

- The format of the notes is not standardized since they come from countries. In addition, it is possible for two doctors even within the same hospital to have completely different note styles: some might write exclusively numerical data while others would use a more accurate description.
- We are not sure whether the condition "intubated", our target variable, occurred at the same as the clinical note was written, or to a past and already resolved condition.
- It is possible that the word "intubated" is present in the clinical notes, introducing information about the target variable that the model should not know.
- We have no information about how they were translated from languages other than English.
- A small sample size: We have the clinical notes of 107 patients, but we only know whether 31 were intubated.

The first step in the classification pipeline consisted of the cleaning of the textual data: all information not strictly necessary for classification should be removed, such as punctuation, superfluous blank spaces, and stop words as well as but also all inflected forms of the same root word or any words that are semantically identical but use a different capitalization. The machine, in fact, is unable to establish that the words "were" and "is", although morphologically different, belong to the same semantic field and would, therefore, classify them as very different. The same applies to the words "HELLO" and "hello".

The clean-up was carried out thanks to the NLTK (Natural Language Toolkit) library, which contains a list of English language stop words, as well as the PorterStemmer function, which is useful for reducing inflected words to the root form.[24]

```
nltk.download('stopwords')

stemmer = PorterStemmer()
stop_words = set(stopwords.words('english'))
translator = str.maketrans('', '', string.punctuation)

def preprocess_text(text):
    text = text.translate(translator)
    text = ' '.join([stemmer.stem(word) for word in text.split()
                     if word.lower() not in stop_words])
    return text
```

```
classification_df['processed_text'] = classification_df['clinical_notes']
    .apply(preprocess_text)
```

Before cleaning:

progressive diffuse interstitial opacities and consolidation in the bilateral parahilar areas and lower lung fields

After cleaning:

progress diffus interstiti opac consolid bilater parahilar area lower lung field

```
x = classification_df['processed_text']
y = classification_df['intubated']
```

Natural language must be represented numerically in order to be processed. Text vectorization is the process of converting text into a numerical feature. It returns the uncommon term high weight and the common term low weight.[25] Creating a Tf-Idf matrix has two steps:

- Creating a sparse frequency matrix, where each row refers to each clinical note and each column to all unique words in each note. The cells take a numerical value equal to the number of times a certain word is present in the text.
- Each value is recalculated so that the least frequent values assume a higher weight, while the most frequent assume a lower one.

At the end of the Tf-Idf transformation, we obtain a matrix representing each clinical note's information in numerical form in addition to the target variable of interest. Later, it is possible to apply a logistic regression.

Logistic regression is a statistical procedure used to solve binary classification problems, in which the goal is to estimate the likelihood of an event or outcome falling into one of two categories[32].

lung	*old*	*intubated*
0.283	0	Y
0	0.561	Y
0.1128	0.131	N

```
vectorizer = TfidfVectorizer()

x_tf_idf = vectorizer.fit_transform(x)

lr = LogisticRegression()
```

K-fold cross-validation is a machine learning method used to evaluate the predictive performance of a model. It entails separating the available dataset into k subsets of equal size. The model is then trained and tested k times, each time with a different fold serving as the validation set and the remaining folds serving as the training set. The performance metrics are calculated at each training iteration and at the end of the process, the mean and standard deviation of the metrics is taken as a reference.[37]

```
kfold = KFold(n_splits=3, shuffle=True)

cv_metrics = []

for idx_train, idx_test in kfold.split(x_tf_idx):
    x_train, x_test = x_tf_idf[idx_train], x_tf_idf[idx_test]
    y_train, y_test = y[idx_train], y[idx_test]

    lr.fit(x_train, y_train)

    y_pred = lr.predict(x_test)

    cv_metrics.append(accuracy_score(y_test, y_pred))

mean_cv_metrics = np.mean(cv_metrics)
```

7.0.4 Computer Vision

In order to obtain a balanced dataset through which to train our model, it is necessary to provide it with an equal number of observations of healthy patients. The Kaggle dataset "Chest-X-ray" provides numerous images of healthy patients and pneumonia patients. From this dataset 196 images of healthy people are randomly extracted, with no accompanying metadata unlike in the case of COVID-19 images.[15]

To interact easily with the Kaggle dataset, its API *Application Programming Interface* was used. This tool that allows people to interact with a server and download files from it without manually having to do a traditional local download. To do this, it was necessary to configure a .json file containing the data needed for authentication: in fact, in order not to clog up the servers at their disposal and control traffic, API providers require users to have a unique username and password.

The code block below, installs the Kaggle library, makes sure that there is no "Kaggle" subdirectory in the root directory, then creates a .json file in which it inserts the username and password, setting read and write permissions via the chmod command. Next, the dataset is downloaded, and its contents unzipped.

```
!pip install kaggle

user = "username"
key = "key"
if '.kaggle' not in os.listdir('/root'):
  !mkdir ~/.kaggle
!touch /root/.kaggle/kaggle.json
!chmod 666 /root/.kaggle/kaggle.json
with open('/root/.kaggle/kaggle.json', 'w') as f:
  f.write('{"username":"%s", "key":"%s"}' % (user, key))
!chmod 600 /root/.kaggle/kaggle.json

!kaggle datasets download -d paultimothymooney/chest-xray-pneumonia

!unzip -q chest-xray-pneumonia.zip
```

As with the cervical images, a strict tree structure was set for file management: a *dataset* folder was divided into train and test and in each sub-directory, the images were divided based on their label. The function below, split the "covid" or "not covid" whole dataset in its random 75% and 25%. Each split was respectively moved into the train and the test folder.

```
def move_images(file_list, source_folder, destination1, destination2):

  totalfiles = len(file_list)
  filestomove = int(0.75 * total_files)
  filestomove = random.sample(file_list, filestomove)

  for file_name in filestomove:
      source_path = os.path.join(source_folder, file_name)
      destination_path = os.path.join(destination1, file_name)
      shutil.move(source_path, destination_path)

  for file_name in file_list:
      if file_name not in filestomove:
          source_path = os.path.join(source_folder, file_name)
          destination_path = os.path.join(destination2, file_name)
          shutil.move(source_path, destination_path)

move_images(file_list_covid,
            '/content/covid-chestxray-dataset/images',
            '/content/dataset/train/covid',
            '/content/dataset/test/covid')
```

Once the data were appropriately organised, the variables train data and test data were initialised, assigning them a dataset containing two columns: one column referring to the image identifier and another column with its label, equal to 0 in the case of healthy patients or 1 in the case of COVID-19 patients.

```
data = Path('/content/dataset')
train = data/"train"
test = data/"test"

notcovidcases = train/"notcovid"
covidcases = train/"covid"

list_notcovidcases = notcovidcases.glob('*.jpeg')
list_covidcases = covidcases.glob('*.jpeg')

train_data = []

for img in list_notcovidcases:
    train_data.append((img,0))

for img in list_covidcases:
    train_data.append((img, 1))

train_data = pd.DataFrame(train_data, columns=['image',
'label'],index=None)

train_data = train_data.sample(frac=1.).reset_index(drop=True)
```

So, they could be compared, the images in the dataset were standardized by reducing them to the same size, the same number of channels, and dividing the value of each pixel by 255, normalizing it between 0 and 1.

```
for img in normal_cases:
    img = cv2.imread(str(img))
    img = cv2.resize(img, (224,224))
    if img.shape[2] ==1:
        img = np.dstack([img, img, img])
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = img.astype(np.float32)/255.
    label = to_categorical(0, num_classes=2)
    valid_data.append(img)
    valid_labels.append(label)
```

Image augmentation imparts noise and variability to original pictures by applying random trans-

formations such as rotation, scaling, or flipping. This creates new training examples that can help the model to learn invariant characteristics and decrease overfitting. In our model, we have used the *imgaug* library, which applies a series of random transformations through a simple and readable code.. The function *SomeOf* applies only some of its children, or transformations, to images.[46]

```
seq = iaa.SomeOf([
    iaa.GaussianBlur(sigma=(0, 3.0))
    iaa.Affine(rotate=15)
    iaa.LinearContrast((0.75, 2))
    iaa.Multiply((1.3, 1.5))])
```

The ImageDataGenerator and the architecture of the CNN are provided by Aakash Nain in his solution to the Chest X-Ray Kaggle challenge.[18] The solution proposed by Nain aims to create a batch of data and labels to be trained, composed of the original images plus augmented images of the least represented class. While that could be a smart strategy to avoid class imbalance, we think that it could be also applied to balanced problems.

```
def data_gen(data, batch_size):
    n = len(data)
    steps = n//batch_size

    batch_data = np.zeros((batch_size, 224, 224, 3), dtype=np.float32)
    batch_labels = np.zeros((batch_size,2), dtype=np.float32)

    indices = np.arange(n)

    i = 0
    while True:
        np.random.shuffle(indices)

        count = 0
        next_batch = indices[(i*batch_size):(i+1)*batch_size]
        for j, idx in enumerate(next_batch):
            img_name = data.iloc[idx]['image']
            label = data.iloc[idx]['label']

            encoded_label = to_categorical(label, num_classes=2)
            # read the image and resize
            img = cv2.imread(str(img_name))
            img = cv2.resize(img, (224,224))

            if img.shape[2]==1:
                img = np.dstack([img, img, img])
```

```
            orig_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            # normalize the image pixels
            orig_img = img.astype(np.float32)/255.

            batch_data[count] = orig_img
            batch_labels[count] = encoded_label

            if label==0 and count < batch_size-2:
                aug_img1 = seq.augment_image(img)
                aug_img2 = seq.augment_image(img)
                aug_img1 = cv2.cvtColor(aug_img1, cv2.COLOR_BGR2RGB)
                aug_img2 = cv2.cvtColor(aug_img2, cv2.COLOR_BGR2RGB)
                aug_img1 = aug_img1.astype(np.float32)/255.
                aug_img2 = aug_img2.astype(np.float32)/255.

                batch_data[count+1] = aug_img1
                batch_labels[count+1] = encoded_label
                batch_data[count+2] = aug_img2
                batch_labels[count+2] = encoded_label
                count +=2

            else:
                count+=1

            if count==batch_size-1:
                break

        i+=1
        yield batch_data, batch_labels

        if i>=steps:
            i=0
```

An interesting solution is proposed for the model, which uses mainly *Separable Convolutional Layers* and not "vanilla" ones.

The primary distinction between a convolutional layer and a separable convolutional layer isthe way they analyze input data and conduct convolutions.

The convolution process of a typical convolutional layer, also known as a "vanilla" convolutional layer, is accomplished using a full set of filters for each channel of the input. The layer's filters have the same dimensions as the input and are convolved with it to produce feature maps. This process is

computationally expensive, especially in deep neural networks with enormous input volumes.

A separable convolutional Layer, on the other hand, divides the convolution operation into two steps: depthwise convolution and pointwise convolution.

Depthwise convolution: In this stage, a single filter is applied to each input channel independently, resulting in a set of feature maps. It conducts a distinct convolution for each input channel, rather than combining them. The goal of this stage is to individually capture spatial information inside each input channel.

Pointwise convolution: The feature maps acquired from the depthwise convolution are subjected to a 1x1 convolution, also known as a pointwise convolution, in this phase. To obtain the final result, pointwise convolutions are utilized to linearly merge the channels. They compute a weighted total of the depthwise convolution feature maps, allowing for cross-channel interaction and feature combining.

When compared to regular convolutional layers, separable convolutional layers significantly reduce the number of parameters and calculations.

```
def build_model():
    input = Input(shape=(224,224,3))
    x = Conv2D(64, (3,3), activation='relu', padding='same')(input)
    x = Conv2D(64, (3,3), activation='relu', padding='same')(x)
    x = MaxPooling2D((2,2))(x)

    x = SeparableConv2D(128, (3,3), activation='relu', padding='same')(x)
    x = SeparableConv2D(128, (3,3), activation='relu', padding='same')(x)
    x = MaxPooling2D((2,2))(x)

    x = SeparableConv2D(256, (3,3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = SeparableConv2D(256, (3,3), activation='relu', padding='same')(x)
    x = MaxPooling2D((2,2))(x)

    x = SeparableConv2D(512, (3,3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = SeparableConv2D(512, (3,3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = SeparableConv2D(512, (3,3), activation='relu', padding='same')(x)
    x = MaxPooling2D((2,2))(x)

    x = Flatten()(x)
    x = Dense(1024, activation='relu')(x)
    x = Dropout(0.7)(x)
    x = Dense(512, activation='relu')(x)
    x = Dropout(0.5)(x)
```

```
    x = Dense(2, activation='softmax')(x)

    model = Model(inputs=input_img, outputs=x)
    return model

model =  build_model()

batch_size = 16
nb_epochs = 20

# Get a train data generator
train_data_gen = data_gen(data=train_data, batch_size=batch_size)

# Define the number of training steps
nb_train_steps = train_data.shape[0]//batch_size

print("Number of training and validation steps: {} and {}"
        .format(nb_train_steps, len(valid_data)))

opt = Adam(lr=0.0001, decay=1e-5)
model.compile(loss='binary_crossentropy', metrics=['accuracy'],
                  optimizer=opt)
```

Two of the arguments used in the fitting step are of particular interest: *callbacks* and *class_weight*.

- Callbacks are used to perform various actions at specific points during training, such as saving the model's weights or adjusting the learning rate. In our specific case, the *EarlyStopping* callback is used to interrupt the fitting process to save time and avoid over fitting once after K-patience epochs the performance does not improve.[43] *ModelCheckpoint* enables the saving of the model's weights during training in order to reuse the best weight in another dataset.[44]

- The *class_weight* argument enables the weighting of the loss function and deals with an unbalanced dataset: in binary classification, class weights could be represented just by calculating the frequency of the positive and negative class and then inverting it so that when multiplied to the class loss, the underrepresented class has a much higher error than the majority class.[13]

```
es = EarlyStopping(patience=5)
chkpt = ModelCheckpoint(filepath='best_model_todate', save_best_only=True,
save_weights_only=True)

history = model.fit(train_data_gen,
                    epochs=nb_epochs,
                    steps_per_epoch=nb_train_steps,
```

```
validation_data=(valid_data, valid_labels),
callbacks=[es, chkpt],
class_weight={0:1.0, 1:0.4})
```

The accuracy of the validation set is 81%, giving a satisfactory result for the diagnosis of COVID-19 from X-ray imaging.

8 Conclusions

The aim of this thesis was to make additional tools available to medical research, and measure their effectiveness, with the goal of helping doctors in the research and subsequent treatment of cervical abnormalities. At present, the Pap test is the elective method for the diagnosis of such abnormalities although its imperfect sensitivity is well-known, resulting in a fair number of false negatives; thus women are convinced they are healthy and so will never be treated with the risk that such abnormalities may degenerate into cancerous formations. Among the tools used are mainly convolutional neural networks i.e., artificial neural networks whose architecture allows the analysis and prediction of two-dimensional data as opposed to the more common artificial neural networks to which one-dimensional or tabular data are subjected. Neural networks serve several purposes: one of them image detection i.e., detecting known subjects and assigning them a label in an image containing noise. In the Yolov8n chapter, a complex convolutional neural network highlighted cells showing abnormalities in slides containing several dozen cells. The reason for using such a tool lies in a potential problem of under-sampling the cervical cell collection: we estimated that one of the probable reasons related to the low sensitivity of the Pap smear is linked to the small number of cells on which doctors make a diagnosis. By expanding the number of cells analyzed, it will be possible to make more meaningful generalizations about the patient's health status. The results should be considered satisfactory but can be further improved by obtaining more detailed images with less noise through the technique of image augmentation, which creates altered data compared to the originals while retaining their key features with the goal of submitting more images to the machine.

We also investigated the predictive ability of cells treated with the silver technique. Such staining is not widespread in cytology and gynecology probably due to the lack of standardization in the process. However, there is evidence suggesting that the number of AgNORs, elements that are highlighted in the cell nucleus by this technique, is positively correlated with the aggressiveness of carcinomas in other organs.

On this occasion, a new approach to semi-learning aimed at minimizing human intervention even during the manual labeling phase was theorized.

The dataset related to images of AgNOR cells did not contain the number of AgNORs for each cell; thus, no classification or regression problem would have been possible. Part of the dataset, therefore, was subjected to manual labeling. It was necessary to count the number of AgNORs for each image, an operation that required a considerable amount of time and in any case, encountered the same issue of subjectivity, this time the developer had to decide whether to count a certain area of the nucleus as abnormal or not. Transporting this technique to the clinical setting, the labeling operation would have to be repeated for each sample to be analyzed, resulting in an unfeasible technique. The new approach proposed — Optimized Derived Feature Semi-Supervised Learning — aims to shift the repeated time commitment of the physician to the developer who will need to count the number of AgNOR only once developing the model.

1. Part of the dataset will be labeled by the developer.

2. The same instances will be subjected to a labeling model. In our case, a clustering algorithm and

counting of connected components.

3. To optimize the hyperparameters of such a transformation and minimize the difference between the actual label and the produced label, the counting function is subjected to an optimization algorithm, in the case of a genetic algorithm, that will find the best parameters to best generalize the counting model and thus, regardless of the starting input return an accurate count.

4. Once the predicted label is given, we can create a classification problem taking into account the predicted number of AgNORs.

Therefore, we built two classification models using two convolutional networks identical in structure changing only the label y and for one model it was the predicted label and for another the actual label. Both models settle their accuracy performance on the test set at an unsatisfactory 24%. This result does not necessarily imply that the number of AgNORs does not necessarily contradict what it asserted earlier. The reasons for such a low performance may be related to an inefficient network structure, poor image resolution, or an extremely poor sample size. Then, we evaluated the predictive performance of the image as a whole trying to determine whether there was an association between the visual characteristics of the cell and thus not necessarily the AgNOR number and the patient's diagnosis. An additional convolutional neural network was able to correctly classify whether the patient was lesion-positive or not in 71% of the cases: an extremely satisfactory result even with the limitations in the dataset set out above.

In addition, the overlapping of the predictive performance of the two models one with predicted labels and one with actual labels suggests that the labeling machine returns accurate classification, confirming the notion that it is possible to speed up and automate the supervision process in a semi-supervised approach thanks to an optimization algorithm.

In the last instance, the predictive performance of images belonging to the SIPakMeD dataset with respect to patient diagnosis was evaluated. Given the same sample size, class distribution, and nearly overlapping neural network architecture but being able to rely on high-resolution images, this neural network was able to correctly predict diagnoses in 90% of cases, an extremely satisfactory result in our research.

The last chapter of the paper was devoted to testing the effectiveness of techniques similar to those just discussed in different diagnostic applications: chest X-ray for COVID-19 detection. Overall, the proposed convolutional neural network returned a prediction accuracy of 81%. However, we consider the section on the design of an NLP algorithm working on patient records to be of greater interest. Although it was not possible to create a workable algorithm due to the lack of a variable to interpolate with the vectors corresponding to each text, such an approach allows predictions to be made about a patient's clinical condition from data that accurately represent a physician's assessments, and therefore do not need to be adapted to fit into a table or transformed into numerical form, potentially losing some of the crucial information captured by the physician's *clinical eye*. Thanks to the proposed method, in fact, it is possible to predict the course of a sick patient from the text representing the doctor's assessment 'verbally'. The AI, therefore, assists the doctor in the field and also wears, at least virtually, the white coat, as if listening to a colleague and helping him in his assessments. The next step in the

realisation of the algorithm, could be to transpose the vocal audio of the doctor examining the patient to the computer, and apply the NLP algorithm on the text.

References

[1] AIRC. Pap test. URL: https://www.airc.it/cancro/affronta-la-malattia/guida-agli-esami/pap-test.

[2] Joao Gustavo Atkinson Amorim, Luiz Antonio Buschetto Macarini, Andre Victoria Matias, Allan Cerentini, Fabiana Botelho De Miranda Onofre, Alexandre Sherlley Casimiro Onofre, and Aldo Von Wangenheim. A novel approach on segmentation of agnor-stained cytology images using deep learning. In *2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS)*. IEEE, July 2020. doi:10.1109/cbms49503.2020.00110.

[3] Manjit Singh Bal, Ashok Kumar, Kuldeep Singh, Gurjeet Kaur, Neha Gupta, and Inderjit Kaur. Detection of abnormal cervical cytology in papanicolaou smears. *Journal of cytology*, 29(1):45–47, 2012. doi:10.4103/0970-9371.93222.

[4] Janice Britton-Davidian, Bernard Cazaux, and Jordi Catalan. Chromosomal dynamics of nucleolar organizer regions (nors) in the house mouse: micro-evolutionary insights. *Heredity*, 108(1):68–74, 2012. doi:10.1038/hdy.2011.105.

[5] Calculator.net. Sample size calculator. URL: https://www.calculator.net/sample-size-calculator.html.

[6] Cleveland Clinic. Cytology. URL: https://my.clevelandclinic.org/health/diagnostics/21714-cytology.

[7] Joseph Paul Cohen, Paul Morrison, Lan Dao, Karsten Roth, Tim Q Duong, and Marzyeh Ghassemi. Covid-19 image data collection: Prospective predictions are the future. *arXiv 2006.11988*, 2020. URL: https://github.com/ieee8023/covid-chestxray-dataset.

[8] International Agency for Research on Cancer. Histopathology of the uterine cervix - digital atlas. URL: https://screening.iarc.fr/atlasglossdef.php?key=Koilocyte&lang=1.

[9] Google. Google colab. URL: https://colab.research.google.com/.

[10] Steven B Hoath and Dorothy G Leahy. The organization of human epidermis: functional epidermal units and phi proportionality. *Journal of Investigative Dermatology*, 121(6):1440–1446, 2003. doi:10.1046/j.1523-1747.2003.12606.x.

[11] Jonathan Hui. map (mean average precision) for object detection. URL: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173.

[12] IBM. Cos'è la computer vision? URL: https://www.ibm.com/it-it/topics/computer-vision.

[13] Angel Igareta. Dealing with imbalanced data in tensorflow: Class weights. URL: https://towardsdatascience.com/dealing-with-imbalanced-data-in-tensorflow-class-weights-60f876911f99.

[14] Investopedia. P-value: What it is, how to calculate it, and why it matters. URL: https://www.investopedia.com/terms/p/p-value.asp.

[15] Kang; Goldbaum Michael Kermany, Daniel; Zhang. Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification. 2018. doi:10.17632/rscbjbr9sj.2.

[16] Suki Lau. Image augmentation for deep learning. URL: https://towardsdatascience.com/image-augmentation-for-deep-learning-histogram-equalization-a71387f609b2.

[17] MathWorks. What is the genetic algorithm? URL: https://it.mathworks.com/help/gads/what-is-the-genetic-algorithm.html.

[18] Aakash Nain. Beating everything with depthwise convolution. URL: https://www.kaggle.com/code/aakashnain/beating-everything-with-depthwise-convolution#Augmentation.

[19] OECD. Investing in health systems to protect society and boost the economy: Priority investments and order-of-magnitude cost estimates (abridged version). URL: https://tinyurl.com/OECDcovidcosts.

[20] Jaspreet Toor Alexandra B Hogan Peter Winskill Azra C Ghani Oliver J Watson, Gregory Barnsley. Global impact of the first year of covid-19 vaccination: a mathematical modelling study.

[21] World Health Organization. Cervical cancer. URL: https://www.who.int/news-room/fact-sheets/detail/cervical-cancer.

[22] Pfizer. Applying the latest digital technology to optimize covid-19 vaccine efforts. URL: https://www.pfizer.com/sites/default/files/investors/financial_reports/annual_reports/2021/story/latest-digital-technology-to-covid-vaccine-efforts/.

[23] Marina Plissiti, P. Dimitrakopoulos, Giorgos Sfikas, Christophoros Nikou, O. Krikoni, and Antonia Charchanti. Sipakmed: A new dataset for feature and image based classification of normal and pathological cervical cells in pap smear images. pages 3144–3148, 10 2018. doi:10.1109/ICIP.2018.8451588.

[24] NLTK Project. Sample usage for stem. URL: https://www.nltk.org/howto/stem.html.

[25] Luthfi Ramadhan. Tf-idf simplified. a short introduction to tf-idf vectorizer. URL: https://towardsdatascience.com/tf-idf-simplified-aba19d5f5530.

[26] Abdou Rockikz. How to use k-means clustering for image segmentation using opencv in python. URL: https://www.tensorflow.org/api_docs/python/tf/keras/metrics/mean_absolute_error.

[27] Olavo Ribeiro Rodrigues, Leila Antonangelo, NeideYagi, Hélio Minamoto, Jr Schmidt, Aurelino Fernandes, Vera Luiza Capelozzi, Saul Goldenberg, and Paulo Hilário Nascimento Saldiva. Prognostic Significance of Argyrophilic Nucleolar Organizer Region (AgNOR) in Resected Non-small Cell Lung Cancer (NSCLC). *Japanese Journal of Clinical Oncology*, 27(5):298–304, 10

1997. arXiv:https://academic.oup.com/jjco/article-pdf/27/5/298/5124626/27-5-298.pdf, doi:10.1093/jjco/27.5.298.

[28] Adrian Rosebrock. Opencv connected component labeling and analysis. URL: https://pyimagesearch.com/2021/02/22/opencv-connected-component-labeling-and-analysis/.

[29] Adrian Rosebrock. Opencv thresholding (cv2.threshold). URL: https://pyimagesearch.com/2021/04/28/opencv-thresholding-cv2-threshold/.

[30] Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 way. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[31] Prajakta Sathawane, Sagar Kedia, Sushma Bobhate, Narayan Bagde, Swapnil Jajoo, and Amar Taksande. Nuances of the papanicolaou stain. *CytoJournal*, 19(43), 2022. doi:10.25259/CMAS_03_18_2021.

[32] scikit learn. Linear models. URL: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression.

[33] scikit learn. scikit-learn.metrics.mean_absolute_error. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_error.html.

[34] scikit learn. sklearn.manifold.tsne. URL: https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html.

[35] scikit learn. sklearn.metrics.accuracyscore. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html.

[36] scikit learn. sklearn.metrics.confusion_matrix. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html.

[37] scikit learn. sklearn.model_selection.kfold. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html.

[38] scipy. scipy.ndimage.label. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.label.html#scipy.ndimage.label.

[39] scipy. scipy.stats.pointbiserialr. URL: https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.pointbiserialr.html.

[40] Bay Area Women's Specialists. Pap test. URL: https://www.bayareawomensspecialists.com/patient-resources/paptest/.

[41] statistics4u. Optimization methods - brute force approach. URL: http://www.statistics4u.com/fundstat_eng/cc_optim_meth_brutefrc.html.

[42] TensorFlow. Mean absolute error. URL: https://www.tensorflow.org/api_docs/python/tf/keras/metrics/mean_absolute_error.

[43] Tensorflow. tf.keras.callbacks.earlystopping. URL: https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping.

[44] Tensorflow. tf.keras.callbacks.modelcheckpoint. URL: https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ModelCheckpoint.

[45] Cambridge Advanced Learner's Dictionary Thesaurus. Word cloud. URL: https://dictionary.cambridge.org/dictionary/english/word-cloud.

[46] Cambridge Advanced Learner's Dictionary Thesaurus. Word cloud. URL: https://imgaug.readthedocs.io/en/latest/source/examples_basics.html.

[47] Davide Trerè. Agnor staining and quantification. *Micron*, 31(2):127–131, 2000. doi:10.1016/S0968-4328(99)00069-4.

[48] ultralytics. Yolov8. URL: https://docs.ultralytics.com/.

[49] Valohai. What is a machine learning pipeline? URL: https://valohai.com/machine-learning-pipeline/.

[50] Wikipedia. Artificial neural network. URL: https://en.wikipedia.org/wiki/Deep_learning.

[51] Wikipedia. Backpropagation. URL: https://en.wikipedia.org/wiki/Backpropagation.

[52] Wikipedia. Deep learning. URL: https://en.wikipedia.org/wiki/Deep_learning.

[53] Wikipedia. Machine learning. URL: https://en.wikipedia.org/wiki/Machine_learning.

[54] Wikipedia. Mathematical optimization. URL: https://en.wikipedia.org/wiki/Mathematical_optimization.

[55] Wikipedia. Point-biserial correlation coefficient. URL: https://en.wikipedia.org/wiki/Point-biserial_correlation_coefficient.

[56] Wikipedia. Semi-supervised learning. URL: https://en.wikipedia.org/wiki/Semi-Supervised_Learning#Semi-supervised_learning.