Miguel Alexandre Figueiredo

Automated analysis of electrophoresis gels

Dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Computer Science and Informatics Engineering

Adviser: Ludwig Krippahl, Assistant Professor,
NOVA University of Lisbon

Examination Committee

Chairperson: Prof. Doctor João Miguel da Costa Magalhães
Members: Prof. Doctor José Ricardo Ramos Franco Tavares
Prof. Doctor Ludwig Krippahl

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

September, 2021

Automated analysis of electrophoresis gels

This document was created using the (pdf)LaTeX processor, based in the "novathesis" template[1], developed at the Dep. Informática of FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Acknowledgements

My sincere thanks to Professor Ludwig Krippahl for teaching me about machine learning, suggesting me this theme that provided a challenge to my capabilities, and for all the guidance and help in finding the best approaches to the obstacles in the development of this thesis.

I would like to thank Professor Ricardo Franco for sharing with me all the gel electrophoresis recordings that served as the basis for the several datasets created across the whole experimental development process.

I am profoundly grateful to my parents, for all the time and money they have invested in me and, above all, the emotional support throughout the years. Everything that I've ever achieved and will achieve in the future is but a reflection of their effort in raising me.

Lastly, a "thank you" goes out to my closest friends, colleagues, and especially to my partner, Marta Oliveira, for always supporting me and helping me push through the hardships, making sure I'd finish this chapter of my life feeling accomplished and ready for the next one.

Abstract

Gel Electrophoresis is an important technique used to divide molecules such as proteins, DNA, RNA, and others, into isolated components, by placing them in a gel and applying an electric field, that will make them migrate towards the end of the gel and separate while doing so. This procedure is usually used as an intermediate step in many experiments, to obtain a particular component. Generally, neither the samples or the migrations are visible to the naked eye and it's necessary to use dyes and development processes to observe the results post gel electrophoresis, meaning any analysis in the course of the experiment is limited or non-existent.

The application of gold nanoparticles to the gel electrophoresis has changed that, since their red colored stain can be used as a marker, making it so that it's feasible to observe all of the samples' migrations, from start to finish, and record them. This way, gel electrophoresis becomes a useful method to learn more about each sample and its migration rates, since it can be used in samples interacting with gold nanoparticles to analyse the visible migrations, so as to obtain the migration parameters.

This thesis focuses on the analysis of video recordings of the gel electrophoresis procedure when applied to samples interacting with gold nanoparticles. Two different prototypes were developed, in order to divide the videos into frames, analyse those frames and identify each sample's location throughout time, making it possible to obtain their migration paths and starting points. For this, several machine learning models were implemented, all focused on different areas of computer vision.

By the end, both prototypes managed to identify the majority of the migration paths, with the second one having a much higher success rate and with lower number of false positives, while taking a tenth of the time to analyse each video, compared to the first one. As such, this prototype was chosen as the best choice to solve the initial problem.

Success in this work may lead to an easier process of extraction of meaningful data from gel electrophoresis' video recordings in the future and an eventual estimation of samples' migration rate parameters.

Keywords: Gel Electrophoresis, Machine Learning, Computer Vision, Image Segmentation, Image Classification, Neural Networks, Deep Learning...

Resumo

Eletroforese em gel é uma técnica usada para dividir moléculas como proteínas, ADN, ARN, entre outros, em componentes isolados, colocando-as num gel e aplicando um campo elétrico, que as fará migrar para o fim do gel e separar enquanto isso. Este procedimento é normalmente usado como um passo intermédio em muitas experiências, para obter um componente específico . Geralmente, tanto as amostras como as migrações não costumam ser visíveis a olho nu, sendo necessário o uso de corantes e técnicas de revelação para observar os resultados pós eletroforese, o que significa que qualquer análise no decurso da experiência é limitada ou não-existente.

A utilização de nanopartículas de ouro na eletroforese em gel possibilitou o desaparecimento dessa limitação, já que a mancha corada vermelha das nanopartículas pode ser usada como um marcador, tornando possível observar as migrações de todas as amostras, de início ao fim, e gravá-las. Desta forma, a eletroforese em gel torna-se um método útil para uma aprendizagem mais profunda sobre as amostras e as suas taxas de migração, visto que pode ser usada em amostras com nanopartículas de ouro, com o intuito de analisar as migrações visiveis, de modo a obter os parâmetros da migração.

Esta tese foca-se na análise de vídeos do procedimento de eletroforese em gel com amostras que possuem nanopartículas de ouro. Foram desenvolvidos dois protótipos, de modo a dividir os vídeos em imagens, analisar essas imagens e identificar a localização de cada amostra durante o tempo, tornando possível obter as suas faixas de migração e os pontos de partida. Para tal, vários modelos de aprendizagem automática foram implementados, todos eles específicos a diferentes áreas de visão computacional.

No fim, ambos os protótipos foram capazes de identificar a maioria das faixas de migração, com o segundo tendo uma maior taxa de sucesso e um número menor de falsos positivos, levando apenas um décimo do tempo a analisar cada vídeo, comparado ao primeiro. Assim sendo, este protótipo foi escolhido como sendo a melhor opção à resolução do problema inicial.

O sucesso neste trabalho possibilitará o desenvolvimento de um processo mais simples de extração de dados de vídeos de eletroforese em gel no futuro, tal como uma eventual estimação dos parâmetros de taxa de migração de amostras.

Palavras-chave:

Eletroforese em gel, Aprendizagem Automática, Visão Computacional, Segmentação de Imagem, Classificação de Imagens, Redes Neuronais, Aprendizagem Profunda...

Contents

List of Figures xiii

List of Tables xvii

Acronyms xix

1 Introduction 1
- 1.1 Gold nanoparticles in gel electrophoresis . . . 1
- 1.2 Machine learning . . . 3
- 1.3 Computer vision . . . 4
- 1.4 Deep learning and neural networks . . . 6
- 1.5 Objectives and contributions . . . 8

2 State of the art 11
- 2.1 Approaches . . . 11
- 2.2 First approach: Image classification . . . 11
  - 2.2.1 Dimensionality reduction techniques . . . 11
  - 2.2.2 Convolutional Neural Networks for image classification . . . 13
- 2.3 Second approach: Image segmentation . . . 14
  - 2.3.1 Image segmentation techniques . . . 14
  - 2.3.2 Fully Convolutional Networks for image segmentation . . . 15
  - 2.3.3 U-Net . . . 16
  - 2.3.4 DeepLabv3+ . . . 17
  - 2.3.5 Encoder VGG16 . . . 18

3 Experimental Work 21
- 3.1 Video recording and key frames . . . 21
- 3.2 Binary classifier . . . 21
  - 3.2.1 Reducing dimensions with PCA and UMAP . . . 23
  - 3.2.2 Fragments dataset . . . 25
  - 3.2.3 CNN as binary classifier . . . 26
  - 3.2.4 Identifying the lanes . . . 27
  - 3.2.5 Preliminary results analysis . . . 30

3.3 FCN and image segmentation 32
3.3.1 Image segmentation techniques 33
3.3.2 FCN for image segmentation 34
3.3.3 Segmenting frames and identifying lanes 37
3.3.4 Preliminary analysis of results 39

4 Results **41**
4.1 Image classification results 41
4.2 Image segmentation results 43
4.3 Results comparison 48

5 Conclusion **51**

Bibliography **53**

List of Figures

1.1 Loading sample into a well in the gel [5]. 2
1.2 An image of a gel post electrophoresis. The ladders on each side can be used to infer the approximate size of each band. EtBr was added to the wells before electrophoresis. The gel was exposed to UV light and the picture taken with a gel documentation system [5]. 2
1.3 2D image analysis paradigm [14]. 4
1.4 Image segmentation. a-c) Original images. d-f) Segmentation results based on a mixture of 5 normal distributions. g-i) Segmentation results based on a mixture of K t-distributions [12]. 5
1.5 Distribution of published papers that use deep learning in subareas of health informatics [21]. 6
1.6 Schematic representation of an artificial neural network. Each perceptron, denoted as a circle, receives n inputs from the layer before and outputs a result using the nonlinear function. The bias is a value used as the output of the neural network when it has zero input. This value is applied to all perceptrons in the hidden layer [23]. 7
1.7 Percentage of most used deep learning methods in health informatics [21]. . 8

2.1 Plot of several datasets using UMAP [32]. 12
2.2 Convolutional Neural Networks [20]. 14
2.3 Result of the Otsu's method when applied to a renal biopsy image [41]. . . . 14
2.4 Fully Convolutional Network's architecture for image segmentation [45]. . . 15
2.5 A spatial map is outputed by switching the original fully connected layers for convolutional layers [45]. 16
2.6 Improvement of spatial detail by fusing information from different layers through skip connections [45]. 16
2.7 U-Net architecture[49]. 17
2.8 Result difference between deconvolutions and atrous convolution [50]. . . . 17
2.9 DeepLabv3+ encoder-decoder architecture. The encoder module encodes multi-scale contextual information, while the decoder module refines the segmentation results along object boundaries [51]. 18

2.10 Depthwise separable convolution consists of (a) a depthwise convolution and (b) a pointwise convolution [51]. 18
2.11 Architecture of VGG16. 19

3.1 Electrophoresis gel procedure with corrected orientation. The wells are placed on top of the image. 22
3.2 Examples of bands and wells used in the dataset. 22
3.3 Examples of background used in the dataset. 22
3.4 Dataset mapping using PCA. The labeled examples are depicted as red, while the rest of the data is represented as green. 23
3.5 Dataset mapping using UMAP. The labeled examples are depicted as red, while the rest of the data is represented as green. 24
3.6 Map divided into 7 clusters by K-means. Out of the three clusters with labeled data, the purple and red ones were discarded, with only the orange one being majorly consisted of bands and wells. 25
3.7 Single frame of video at the start of the migration. 27
3.8 Single frame of video after 10 minutes of migration. 27
3.9 Single frame of video after 15 minutes of migration. 27
3.10 Single frame of video by the end of migration. 27
3.11 Result of the sum of all the analysed frames of a video. Contrast was enhanced to provide better visualization of the lanes. 28
3.12 Result of the sum of all the analysed frames of a video. Contrast was enhanced to provide better visualization of the lanes. Every dot represents a center identified by the model across all frames of a video. 28
3.13 Result of the sum of all the analysed frames of a video by model 1. Contrast was enhanced to provide better visualization of the lanes. All lanes have been correctly identified, with only 10 false lanes. 29
3.14 Result of the sum of all the analysed frames of a video by model 2. Contrast was enhanced to provide better visualization of the lanes. All lanes have been correctly identified. There are more than 15 false lanes. 29
3.15 Result of the sum of all the analysed frames of a video by model 3. Contrast was enhanced to provide better visualization of the lanes. Most lanes have only been half identified, starting in the middle of the lane and ending at the end of the box. There are more than 20 false lanes. 30
3.16 Result of the sum of all the analysed frames of a video. Contrast was enhanced to provide better visualization of the lanes. The red cable, red wall of the box and light reflected on the gel might be the cause of false positives. 31
3.17 Result of the sum of all the analysed frames of a video. Contrast was enhanced to provide better visualization of the lanes. The background is clearer and without obstructions, possibly improving the results. 31

3.18 Result of the sum of all the analysed frames of a video. Contrast was enhanced to provide better visualization of the lanes. The starter point of the lanes was identified and all the lines with different start were discarded. Notice that some of the lines that were correct also got deleted, due to the failure of the classifier on identifying the correct start. 32
3.19 Result of background subtraction for image segmentation. 34
3.20 U-Net architecture[49]. 36
3.21 Result of the segmentation of a frame executed by the trained model. 37
3.22 Result of the sum of all the analysed frames of a video. Contrast was enhanced to provide better visualization of the lanes. Each yellow dot represents the center of a well/band that was identified in the segmented frames. 38
3.23 Result of the sum of all the analysed frames of a video. Contrast was enhanced to provide better visualization of the lanes. The starter point of the lanes was identified and all the lines with different start were discarded. Each yellow line represents the lane that was created by aligning the centers of the bands/wells. 38

4.1 Key frame obtained by dividing a video into frames. 42
4.2 Key frame after wells are identified and its centers marked. 42
4.3 Overlapped frames with vertical lines representing the lanes. 42
4.4 Overlapped frames with vertical lines that start in the same "y". 42
4.5 Result of the sum of all the analysed frames of a video. Contrast was enhanced to provide a better visualization of the lanes. Outside of the box limits, the first problem is apparent, where many false lanes were identified. Inside of the box limits, the second problem is visible, with many false lanes making it impossible to group the correct ones resulting from the overlap together. . . 43
4.6 Key frame obtained by dividing a video into frames. 44
4.7 Key frame segmented by subtracting the background. 44
4.8 Key frame segmented by the trained FCN. 44
4.9 Key frame after bands are identified and centers are calculated. 44
4.10 Overlapped frames with all the bands and wells' centers across time. 44
4.11 Overlapped frames with vertical lines representing the lanes. 44
4.12 Segmentation result of a frame of a video. Both the first two and the last two bands are fused together, being treated as one entity with an incorrect center. 45
4.13 Lane results with only the centers drawn. These centers were the ones calculated, with no neighbouring being applied, in order to better show the problems of this approach. The green box presents the result of the band fusion, where several centers are located in between lanes, instead of being on top of them. 45
4.14 Segmentation result of a frame of a video. The box interferes with the segmentation, fusing some bands and fragmenting almost all of them. 46

4.15 Lane results with only the centers drawn. these centers were the ones calculated, with no neighbouring being applied, in order to better show the problems of this approach. The green box presents the region of the box that interferes with the segmentation, resulting in band fusion, where several centers are located in between lanes, instead of being on top of them. 46
4.16 Reconstruction of a segmented frame after the original one was divided into small squares, with these being segmented by the Simple U-Net and used for the reconstruction. 47
4.17 Lane results with only the centers drawn. These centers were the ones calculated, with no neighbouring being applied, in order to better show the problems of this approach. The low success rate in proper segmentation makes the algorithm identify several false centers and subsequently false lanes. 47

List of Tables

3.1 CNN architectures and results. The convolutional blocks' columns (Conv.1, Conv.2, Conv.3) represent the number of filters used in each layer (N/A when the layer was not used). The last dense layer is always used for classification. The loss values refer to the validation loss of each model. 26
3.2 Results of the training and validation of the segmentation models chosen. . . 35
3.3 Results of the training and validation of the simple U-Net. 37

4.1 Comparison of the results of both approaches. 48

Acronyms

ANN	Artificial Neural Network.
AuNP	Gold Nanoparticle.
CNN	Convolutional Neural Network.
DNA	Deoxyribonucleic Acid.
EtBr	Ethidium Bromide.
FCN	Fully Convolutional Neural Network.
GE	Gel Electrophoresis.
IoU	Intersection-Over-Union.
mIoU	Mean Intersection-Over-Union.
NN	Neural Network.
PCA	Principal Component Analysis.
RNA	Ribonucleic Acid.
SPR	Surface Plasmon Resonance.
UMAP	Uniform Manifold Approximation and Projection.
UV	Ultra Violet.

Chapter 1

Introduction

1.1 Gold nanoparticles in gel electrophoresis

Gel Electrophoresis (GE) [1, 2] is a laboratory procedure, most commonly used to separate samples [3], such as proteins and DNA, into the different components that constitute them. Generally, it can separate around 15 to 20 samples in less than an hour [4]. The core of this procedure is the application of an electric field onto a gel, making its ends polarized. The electrically charged samples are loaded on small holes on the gel, named wells (figure 1.1), and will start to migrate towards the end of the gel with inverse charge [5]. This migration is possible thanks to small pores that exist in the gel [5]. During the movement, the sample's different components will separate and move with different velocities according to the voltage applied to the gel, with higher values implying higher velocity, and the sample's size [6], with smaller components moving faster and further than bigger ones.

By the end of the procedure, the gel will have separated bands or spots, with each one being a component of the original sample. Since they are not visible to the naked eye, there is a need to add a dye or an agent, such as Ethidium Bromide (EtBr) or SYBR Green, to the samples in the wells, after the GE [7]. This will stain the bands, so as to be able to observe them with the help of development procedures (figure 1.2).

In some cases, the final objective is to identify the approximate size of the components post GE. To analyse the results it is required the use of a ladder. This ladder is a sample with known components and their respective sizes, that can be placed onto the gel to serve as a control sample. By the end of the procedure, one can compare the unknown components of each sample to the ladder and relatively conclude the size of each component [5]. It is also possible to extract a particular component to be used on further procedures.

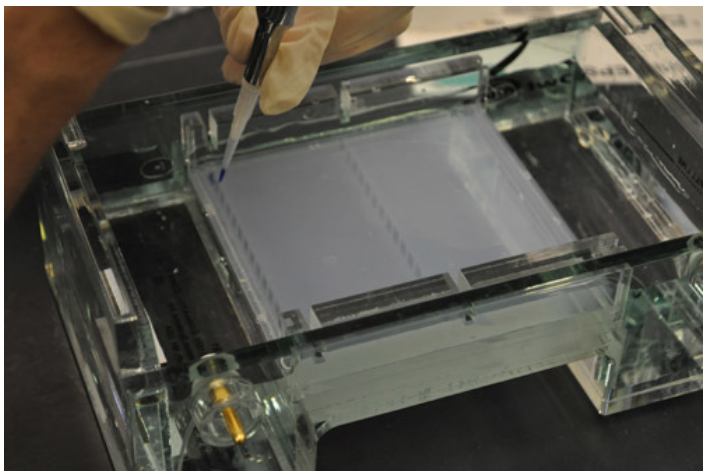

Figure 1.1: Loading sample into a well in the gel [5].

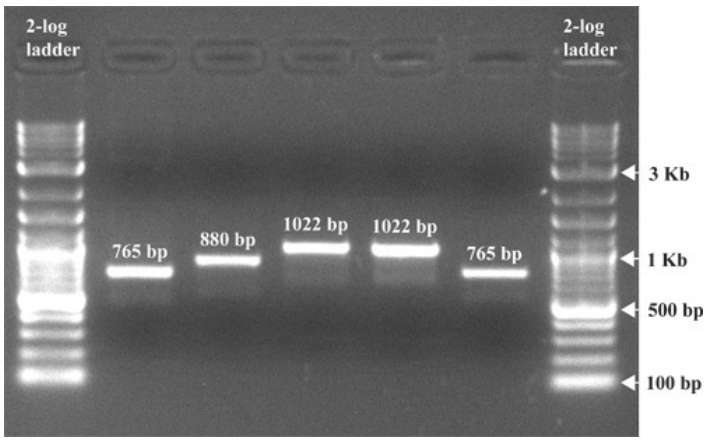

Figure 1.2: An image of a gel post electrophoresis. The ladders on each side can be used to infer the approximate size of each band. EtBr was added to the wells before electrophoresis. The gel was exposed to UV light and the picture taken with a gel documentation system [5].

The main limitation when it comes to the GE procedure is the poor observation conditions that limit the overall analysis of the process. Usually, since GE is not visible by itself, the records of the experiment are images of the bands post electrophoresis. Although this enables the relative identification of the components, it's impossible to extract information regarding what happened during the electrophoresis, incapacitating any attempt of analysing in depth a sample's migration process.

The application of Gold nanoparticles (AuNPs), however, is able to solve the previous limitation. When interacting with electromagnetic waves (light), there's an oscillation of electrons on the AuNPs's surface, causing a phenomenon called Surface Plasmon Resonance (SPR), "...leading to an absorption band in the visible spectrum. Gold nanoparticles of 15 nm in diameter have the SPR absorption peak at around 520 nm, which corresponds to a transmitted red color (620-750 nm)." [8]. This means that, if instead of loading normal samples in the wells, samples of proteins interacting with AuNPs are

loaded, the migration will be displayed with a red color, being visible to the naked eye. With this, it's possible to give a new meaning to the gel electrophoresis procedure, other than to separate macromolecules. By inducing a migration and observing/recording the whole gel electrophoresis procedure with AuNPs, it's possible to make an analysis with the objective of estimating the dimensions and charges of the samples through their migration rates.

To reach that point, it's first necessary to create a way to analyse the video recordings. Although it's possible to do it one by one, it would not be efficient, since there could be quite a lot of videos and the analysis will be mostly the same for all of them. In that case, the best solution is to create a prototype capable of analysing any video autonomously. This can be achieved with the use of a machine learning model trained in computer vision, to be able to analyse visual data and extract useful information. The development of this software is the main focus of this thesis.

1.2 Machine learning

Machine learning is the scientific discipline that focuses on creating computational models capable of understanding complex relationships between data. The usual process involves training a computational model with the data available at the moment and, by doing so, making the model capable of correlating the input data to the output data the most correct way possible [9]. A trained model can not only show how the data interact, but also predict the most probable relation of future data.

Although a great number of models can be used to solve the same problem, each one works in different ways and will have different advantages and disadvantages. In that sense, it's always recommended to experiment several models and compare the results to understand which one has the best performance and is the best to solve the particular problem [10].

Machine learning can be divided into several areas, such as:

1. **Supervised Learning:** a model receives a labeled set of training data. The model will learn how to best correlate the input data to the output data [10].

2. **Unsupervised Learning:** requires only the input feature values to train on and it's up to the model to discover the hidden structure in the training data based on them [9].

3. **Reinforcement Learning:** reigns over the models that learn through trial and error in a dynamic environment, while trying to achieve an objective or maximizing an outcome [11].

Since the dataset used for training and testing is based on the video recordings, which is unlabeled data, the first approach to the problem uses unsupervised learning as a

helpful tool to create a labeled dataset for a supervised model. Since this approach might not be the best, the second one focuses only on supervised learning. This way a comparison between the approaches can be made. All the required models need to be able to handle image data, which means they were chosen from a list of models that are used for computer vision. Before checking the best machine learning model architectures for the problem at hand, it's important to understand exactly what is computer vision.

1.3 Computer vision

Computer vision's main goal is to analyse and extract useful information from images [12]. To be more specific, a computer-based visual system has to be capable of receiving input data that has been derived from an image and, using machine learning algorithms, not only process the data, but also interpret it, with the intuit of giving meaning or information about the initial image [13]. These "observing machines" are quite used nowadays in the medical field, radiology and microscopy, and industrial field, industrial inspection and robot guidance [14].

For the processing of images, it's first required to convert them into digital format or data. This format is a "discrete array of numbers representing brightness or color values at a discrete grid of points in the image plane-or, more precisely, average values in the neighborhoods of these points. (...) The elements of the array are usually called pixels..." [14]. The generalized process of computer vision is summarized in figure 1.3. Note that the schematic focuses on 2D images. Although the process of analysing 3D images is more complex, its early stages can benefit from the application of some 2D image analysing techniques.

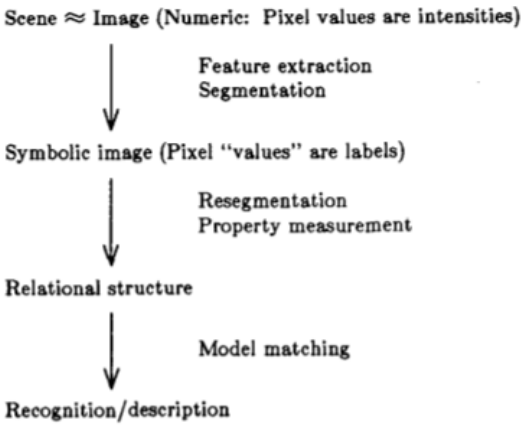

Figure 1.3: 2D image analysis paradigm [14].

Computer vision encompasses several areas related to image data. It's important to understand some of them, since the concepts can sometimes be hard to distinguish and different machine learning models can focus on these different areas to solve the same problem.

The one probably most commonly known is image classification [15, 16]. Image classification is simply used to decide whether an image belongs to a certain category [17]. For example, if the category is "car" and an image depicts a parked car in a street, then that image would be classified as belonging to that category. To achieve this classification, a classifier can be applied. The classifier can be trained with a dataset of images manually labeled, in order to classify an image in a category.

Another area is image segmentation. Image segmentation (figure 1.4) is the concept of separating or partitioning an initial image into regions. The two basic rules of partitioning, which are the core of several partitioning methods, are discontinuity between groups of pixels and homogeneity between pixels of a group [18]. In other words, when analysing an image for segmentation, pixels which exhibit similar features in the local neighborhood are grouped together into a partition and when neighbour pixels show a difference in their features, they are considered to belong to different partitions. The toughest topic when it comes to segmentation is really which method is the most accurate in partitioning an image, with many articles exploring, testing and comparing them frequently. After the segmentation comes the extraction of attributes and its analysis/interpretation [13].

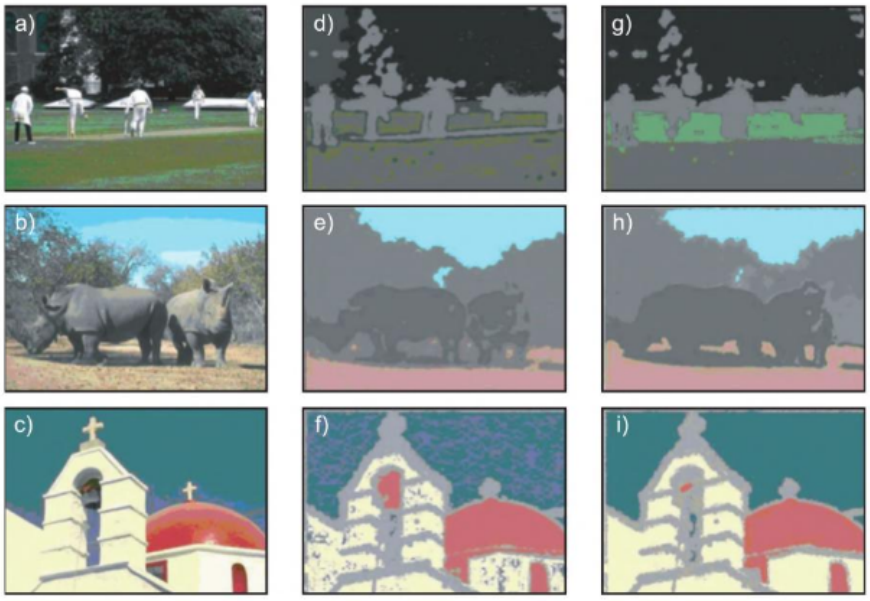

Figure 1.4: Image segmentation. a-c) Original images. d-f) Segmentation results based on a mixture of 5 normal distributions. g-i) Segmentation results based on a mixture of K t-distributions [12].

Finally, there's object recognition/detection, which is the detection of certain segments of an image, which properties satisfy certain constraints pertaining to an object [14]. While segmentation focuses on separating the image without care for the shape of the partitions, object recognition will use the shape as one of the parameters by which it might detect a certain segment of the image as a possible object.

As mentioned in the previous section, there are two different approaches. The first

one focuses on image classification, while the second one dives into image segmentation. With these concepts in mind, it's time to understand a bit more about the best model architectures when dealing with image data.

1.4 Deep learning and neural networks

When it comes to image data in general, although there are several classic models that can be trained to achieve whichever task it's supposed to achieve, all of the current state-of-the-art models are based on neural networks. This is because most, if not all, of these models are capable of autonomously detect features, which was a major difference from the classic models that had the need for manual feature selection. They also have high accuracy, making them the most successful types of models when working with image data nowadays. Since this thesis will focus on several models that apply neural networks, it's best to start by analysing this model and the area of machine learning that focuses on its uses, Deep Learning.

Deep Learning is a form of machine learning, which focal point is to train a model to better solve problems a human being could solve, such as identifying objects in an image, by computing the data similar to the way a human brain would do it and without human involvement [19]. In the past 10 years, Deep Learning's applications skyrocketed, with it being used as the state-of-the-art in areas such as speech recognition, computer vision and others [20]. Figure 1.5 describes the sudden escalation in the interest of deep learning in health informatics over the years of 2010 to 2015, with a huge increase in the number of published papers regarding that theme [21]. Some of the factors that led to the growth of deep learning's usage were the evolution of hardware, which increased the computing power overall and a large quantity of data available for training models [20].

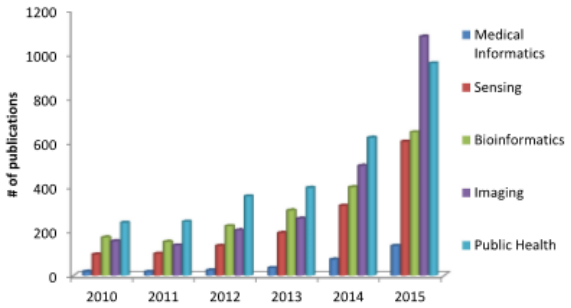

Figure 1.5: Distribution of published papers that use deep learning in subareas of health informatics [21].

The application of Deep Learning involves the use of Artificial Neural Networks (ANNs) or simply Neural Networks (NNs) [20] with the caveat of needing multiple hidden layers in order to profoundly analyse the input data and extrapolate the output [21]. A neural network is a machine learning model inspired by the biological neural network

that composes the human brain [22]. The core of this neural network is the perceptron, the digital equivalent of the neuron. By receiving an input and applying a function to it, the perceptron computes an output of 0 or 1. In other words, it's a binary classifier [21]. The biggest problem with using this method is the inability to compute complex algorithms.

A neural network is the name given to the connection of several perceptrons. These are distributed into layers, with the first one being the input layer, the last one being the output layer and everyone in between being called the hidden layers (Figure 1.6).

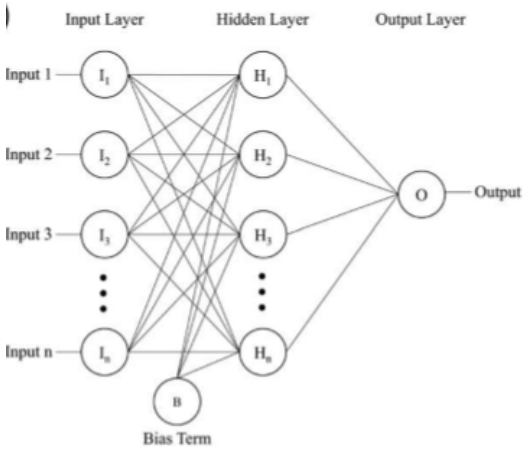

Figure 1.6: Schematic representation of an artificial neural network. Each perceptron, denoted as a circle, receives n inputs from the layer before and outputs a result using the nonlinear function. The bias is a value used as the output of the neural network when it has zero input. This value is applied to all perceptrons in the hidden layer [23].

Each perceptron receives an input value from the previous layer and maps it onto a nonlinear function [23]. The resulting value will be passed to the next layer of perceptrons which will, in turn, use it as input, continuing the process until it reaches the output layer, with the final result being output. This whole process is called forward-propagation. Moreover, each perceptron also possesses an inherent value named "weight", which is a tunable parameter, starting at random and varying during the training of the model, so as to adjust and diminish the error [21]. To tune this value, each time the model outputs a result, a comparison is made to the desired output. It's then possible to do a back-propagation, where a loss function will be computed to calculate the error value and with it compute the necessary changes to every weight, in order to decrease the error on the next forward-propagation.

Several models of neural networks have been developed over time, each own with better affinity towards certain problems. Figure 1.7 depicts the most used derivations of neural networks in the field of health informatics. Some of these models were the ones used to solve the problem in question.

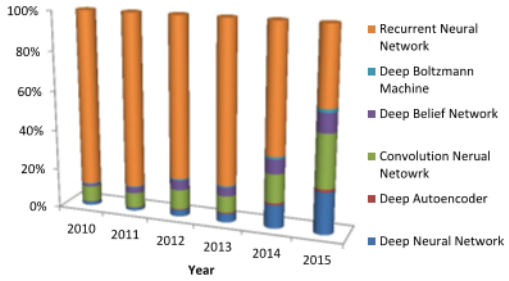

Figure 1.7: Percentage of most used deep learning methods in health informatics [21].

1.5 Objectives and contributions

There are already many research papers involving gel electrophoresis analysis and many of them employ machine learning as a helpful tool to rapidly and accurately evaluate results and compare approaches [24, 25, 26, 27, 28, 29]. However, most, if not all of these, focus on the final results of the GE procedure, with the separation of the samples into several bands, the detection of each band and the estimation of its size, but not as much on what happens during the process, specifically what can be learned from the migration of each sample. As referred to previously, one of the reasons for this to happen is the inability to observe and record the whole process coherently, due to the low, or even, nonexistent visibility of the samples during the migration.

With the introduction of AuNPs, it became possible to observe and record all of the GE procedure and this, in turn, provides a great chance to obtain new information regarding each sample and its behaviour during the experiment. With a future goal of studying the migrations and learn their characteristics, it's essential to be able to analyse the recordings efficiently, identify certain zones of interest and extract data from those regions.

This thesis focuses on developing a software that is capable of autonomously analysing each video and successfully find the wells and the bands corresponding to each sample throughout time, since with these two it's possible to determine each sample's lane and starting point. This software deals with a great amount of data, so the best way to achieve the objective was by applying a machine learning model, specific to computer vision, to be trained to find the bands and wells, so it can be used in all the videos without much manual effort. To analyse the video throughout time, the video recordings were fragmented into frames (still images that constitute the video), which were used from then on as the basis of the datasets.

The first approach involved training a machine learning classifier with labeled images of small dimensions, so that it could be used to sweep a frame little by little, using a sliding window with overlap approach, and evaluate each small section, classifying it as either bands/wells or the background. Every section evaluated as the former had its center's coordinates stored. Finally, the frames that were analysed were overlapped, creating an image with the progression of the bands across time and the stored coordinates were drawn on top of it, so as to visually evaluate the results. The labeled dataset used to train

and validate the model used both background and bands/wells examples. While there is an immense number of background examples, in order to obtain the other type, the frames were fragmented into small squares, with some bands and wells being manually labeled. Using dimensionality reduction techniques to map out these squares, the neighbours of the labeled data were extracted, since they had a high chance of also being bands and wells. The best components were then used as ground truth for the classifier to learn from. Data augmentation techniques were also implemented to increase the number of bands and wells' examples, making the dataset evenly split.

The other approach to the problem was to implement a model capable of supervised learning, this time in the area of image segmentation. The model learnt to segment the bands and the wells from the background of the key frames. The segmented images were analysed, with the centers of each band and well being calculated and its coordinates stored for estimating the lanes later. Finally, the original frames that had been segmented were overlapped, creating an image with the progression of the bands across time and the stored coordinates were drawn on top of it, so as to visually evaluate the results. This approach required pixel-wise annotations on the key frames, to distinguish the bands and wells from the rest, serving as ground truth to train the model in image segmentation. For that, classic methods of segmentation were applied, creating a few examples that were manually evaluated and, if correct, were saved for training.

With no computational way to analyse the results of both approaches, these were visually evaluated and compared, to understand which one achieved the best performance in terms of correctly identifying the wells and bands across time, by which the lanes of each sample can be calculated, from the starting well, to the last position of the band.

The usage of both classification and segmentation models with the objective of locating samples throughout time led to the final contribution of this thesis, a prototype capable of identifying the whole region where a sample migrates, which can be used in the future to extract useful information regarding the migration process, such as the concentration profile of each sample throughout the gel, throughout time. Overall, it's a good opportunity to learn more about each sample and its behaviour when submitted to gel electrophoresis.

Chapter 2

State of the art

2.1 Approaches

As defined before, there are two very different approaches to solve the same problem of identifying the zones of interest in images. Each approach requires different machine learning models, algorithms and techniques to achieve the proposed task. The next section will be used to present and argue the choices made for the models, algorithms and techniques of the first approach, followed by another section for the second approach. Other options will also be presented, based on the fact that they show great promise at solving the same problems as their peers, according to the literature referenced, and so, might also be used as auxiliaries to improve results when deemed needed.

2.2 First approach: Image classification

2.2.1 Dimensionality reduction techniques

In this approach, a model will learn to distinguish between two categories of small-sized images. To train this model, it's necessary to have many examples of each category, which need to be extracted from the frames. After dividing them into small sections, there's a need to computationally divide this unlabeled data and a way to do that is by reducing the dimensions of the data. With this reduction, the data becomes both easier to visualize and to analyse its patterns, making the process of finding similar examples of the same category a question of finding clusters and evaluating neighbours.

2.2.1.1 PCA

Principal component analysis is one of the most well-known dimensionality reduction algorithms. It's a simple and non-parametric method that "...reveals simple underlying structures in complex data sets using analytical solutions from linear algebra."[30]. Its task is to determine linear combinations with the maximum amount of variance in the data, which are called principal components. PCA calculates these principal components and checks how many are needed to be considered to best represent the original data, taking into account the next bigger principal component if the ones before it are not enough[31]. In order to visualize the data after reduction, two to three principal components are used.

Since it's a simple and fast method, PCA will be tested, although other algorithms will be prioritized, due to being considered as better performers than this one.

2.2.1.2 UMAP

Uniform Manifold Approximation and Projection [32] is a manifold learning and non-linear dimensionality reduction technique specifically used for data visualization. This method reduces the number of features to the most relevant ones, so that it becomes easier to visualize the data in a 2D or 3D plot (figure 2.1). This is done with the usage of graph layout algorithms, firstly constructing a high dimensional graph representing the data and secondly creating the optimal low dimensional graph that is the most structurally similar to the former one.

It's considered very effective for visualizing clusters of data and their respective proximities and is scalable, performing better than other techniques, such as PCA [33] and t-SNE [34], being both faster and more general-purposed. This technique will be the one mainly used for the dimensionality reduction of the dataset.

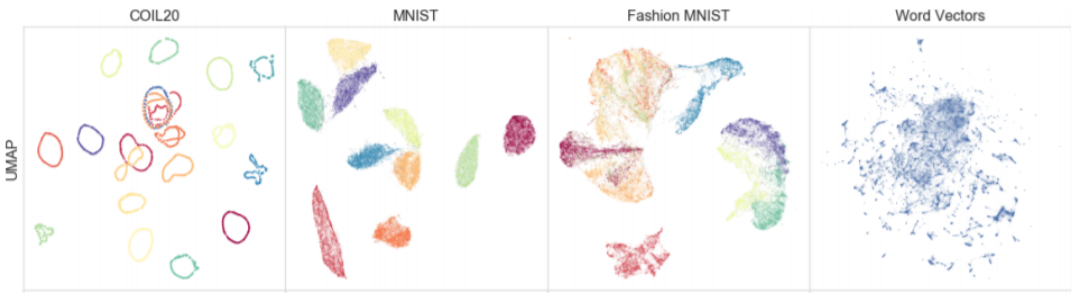

Figure 2.1: Plot of several datasets using UMAP [32].

2.2.1.3 Autoencoder

An autoencoder[35] is a type of neural network, which objective is to be able to produce an output that is equal to the input. An autoencoder is composed of two parts, an encoder

responsible for compressing the input into a latent-space representation and a decoder, that will reconstruct the input from the latent space representation.

The interesting bit about this type of model is not the output, but in the hidden or latent representations of the data that the autoencoder is able to discover. When encoding, the input data will be mapped to the hidden or latent variables, reducing the dimensions. With some changes, this autoencoder becomes a dimensionality reduction technique. According to Hinton[36], it's a "...nonlinear generalization of PCA that uses an adaptive, multilayer "encoder" network to transform the high-dimensional data into a low-dimensional code and a similar "decoder" network to recover the data from the code.". The author also states that "...work much better than principal components analysis as a tool to reduce the dimensionality of data.", which further incentives its usage over PCA.

Since there are several variations of autoencoders, the one used would have to be a convolutional autoencoder [37], which switches the fully-connected layers for convolution layers, in order to accept images.

Although this model seems like a viable option, there's a need to implement and train it in order to achieve the results, which is more time-consuming than the other options. It is also quite prone to errors, if the initial weights are not close to optimal, since "With large initial weights, autoencoders typically find poor local minima; with small initial weights, the gradients in the early layers are tiny, making it infeasible to train autoencoders with many hidden layers..."[36]. With this in mind, the autoencoder approach should only be used in case the other techniques prove to be unsuccessful.

2.2.2 Convolutional Neural Networks for image classification

As for the model that will be classifying the images, a Convolutional Neural Network (CNN) is the best choice. These models are considered to be the most successful when working with image data [38] even out of all the variants of neural networks. As an ANN is based on the human brain, a CNN is inspired on the neurobiological model of the visual cortex. This structure consists of maps of local receptive fields that decrease in granularity as the cortex moves anteriorly [21]. The general process consists of implementing convolution layers to convolve the input image, using small filters, to map the features (creating feature maps) and then, subsample the result of the convolution to reduce the spatial size of the representation and computational complexity (figure 2.2). The result is used as the new input, repeating the process until the features are extracted. A well-known example of its applications is self-driving cars, which utilize CNNs to analyse video and images of scenes and recognize, for example, people or traffic signs [39].

Since CNNs in general are state-of-the-art when working with image data and computer vision, this model seems to be the correct one to experiment with. It's expected to be able to achieve high accuracy in the classification and there is no need to manually intervene, for example, for feature selection. In order to obtain the best performing CNN, several models will be trained, validated and tested, with the number of layers and the

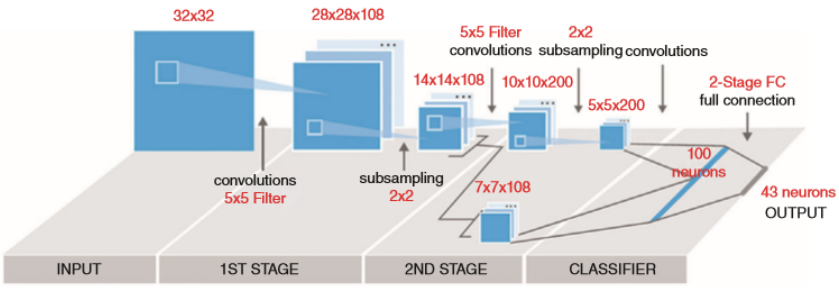

Figure 2.2: Convolutional Neural Networks [20].

size of each one varying.

2.3 Second approach: Image segmentation

2.3.1 Image segmentation techniques

In this approach, a model will learn to segment the bands and wells of each frame from the background. To train this model, it's required to have examples of already segmented frames, to be used as ground truth, and to achieve that, several segmentation algorithms will be used on the frames, with the results being evaluated and the correct ones being stored to create the training and validation dataset.

2.3.1.1 Otsu's method

The first runs will use Otsu's method [40] for creating the segmented images. This method is a thresholding technique, that analyses a greyscale histogram of an image and automatically iterates through all the possible threshold values, calculating a measure of spread for the pixel levels on each side of the threshold. The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum. Figure 2.3 shows the result of using the Otsu method on a medical image.

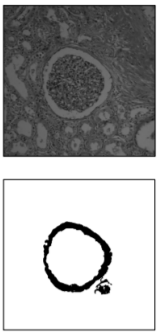

Figure 2.3: Result of the Otsu's method when applied to a renal biopsy image [41].

Although this method also has its flaws, such as difficulty in dealing with small-sized

objects when compared to the background, several authors have managed to correct it and enhance its performance. One example [41] is by analysing the histogram with wavelet transform to correctly segment renal biopsies. It has also been used for image segmentation of gel electrophoresis and has achieved good results [27, 42] with some slight adjustments, such as Particle Swarm Optimization. With this in mind, Otsu's method will be experimented with.

2.3.1.2 Background subtraction

A pretty common technique used in video footage, background subtraction permits the detection and segmentation of objects that do not belong to the background, normally moving ones. This process is based on the comparison between a real image and an estimate of that image if it contained no objects of interest[43]. These estimates are generated by analysing several frames of the video and labeling the objects that did not change position as "background". Then, comparing that image to the next one, the objects that did move are thresholded, with the background being subtracted, resulting in a segmented image.

As all the images used come from a video, where nothing moves except for the samples, this technique seems like a viable option for the segmentation and will be tested alongside the previous one. The main concerns here are eventual changes in the alignment of the camera and fluctuations in the illumination of the scene throughout the recording process[44].

2.3.2 Fully Convolutional Networks for image segmentation

One of the models that will be used for image segmentation is a Fully Convolutional Network (FCN) [45]. This model replaces all the fully connected layers of regular CNNs, such as GoogleLeNet [46], AlexNet [47] and VGG [48], which were typically used for classification, by convolutional layers (figure 2.4). This permits the classification network to output a spatial map (figure 2.5). Using this with differential interpolation layers and spatial loss "... produces an efficient machine for end-to-end pixelwise learning" [45]. It is also able to receive an image input of any size, creating an output of the segmented image with the same size. This is possible due to backward convolutions and skip connections between layers (figure 2.6).

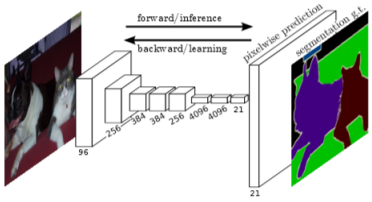

Figure 2.4: Fully Convolutional Network's architecture for image segmentation [45].

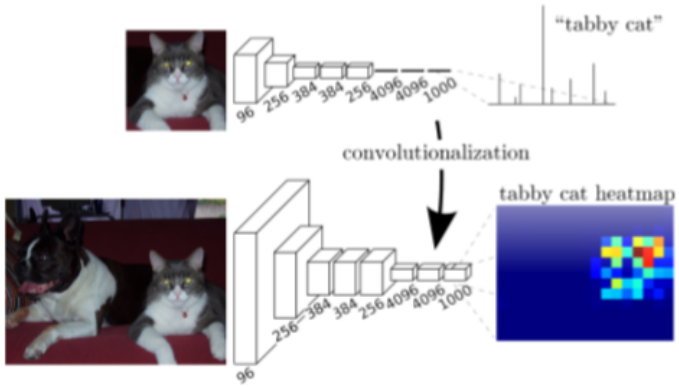

Figure 2.5: A spatial map is outputed by switching the original fully connected layers for convolutional layers [45].

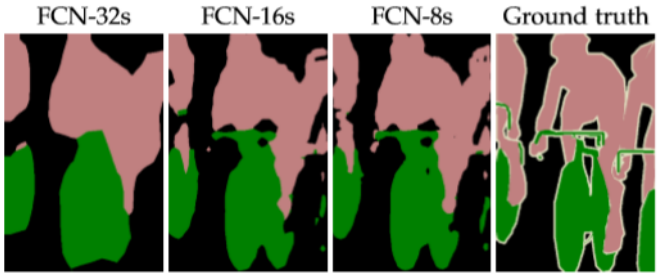

Figure 2.6: Improvement of spatial detail by fusing information from different layers through skip connections [45].

The reason why the model chosen was an FCN is because this model was created for the sole purpose of image segmentation and its performance is one of the best even nowadays.

2.3.3 U-Net

U-Net[49] was created as an extension of the FCN for biomedical images. Built to be able to obtain more precise segmentations from small-sized datasets, this model's standard architecture (figure 2.7) has ten layers, with the first four being the encoder part where, like in an FCN, the data is put through convolution and pooling layers for downsampling, then a middle layer and then the other four layers, decoder part, where the data is up-sampled with up-convolutions and concatenated with cropped feature maps from the downsampling segment of the network, in order to not lose pattern information. Finally, the last layer generates the segmentation map by doing a 1x1 convolution on the feature maps.

As there is a chance of the number of quality examples of segmented frames created by the segmentation techniques being low, this model may guarantee better results than the other options, so instances of it will be trained, validated and tested with.

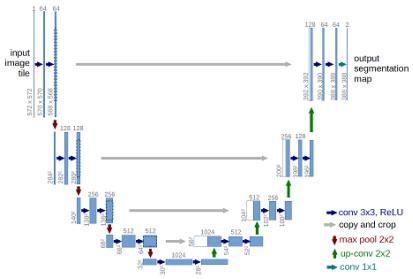

Figure 2.7: U-Net architecture[49].

2.3.4 DeepLabv3+

Although FCNs were created for image segmentation, there have been many advancements on that topic. One model that started as an FCN and has evolved throughout several iterations is the Deeplab [50]. Deeplab's implementation differs from the previous model, with the use of atrous convolutions instead of backward convolutions (figure 2.8) and Atrous Spatial Pyramid Pooling (ASPP) and many other changes that have been added in each iteration. The last version of the model is named DeepLabv3+ [51], which extends on the Deeplabv3 [52] model, with the change to an encoder-decoder architecture (figure 2.9), with a modified Align Xception [53] as its main feature extractor and the use of atrous separable convolutions (figure 2.10).

The reason why it's interesting to explore this model is, firstly, because it's one of the most recent models, in general, dedicated to image segmentation (published in 2018) and, secondly, because it's the latest iteration on a long series of models that have faced many challenges in the area of computer vision and, time and time again, have surpassed them, improving its performance along the way. This state-of-the-art model is a good alternative to the FCN and it should be considered a viable option. This, however, will be left as an alternative, since it's a quite complex solution to a most likely simple problem.

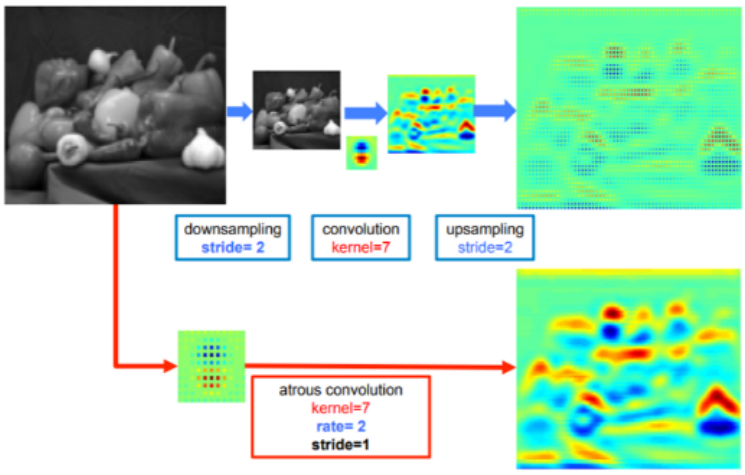

Figure 2.8: Result difference between deconvolutions and atrous convolution [50].

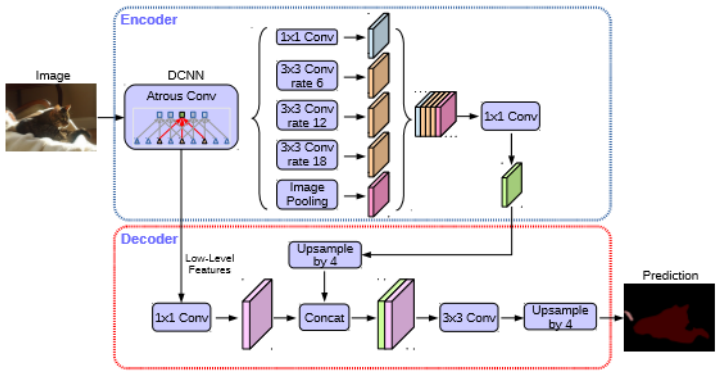

Figure 2.9: DeepLabv3+ encoder-decoder architecture. The encoder module encodes multi-scale contextual information, while the decoder module refines the segmentation results along object boundaries [51].

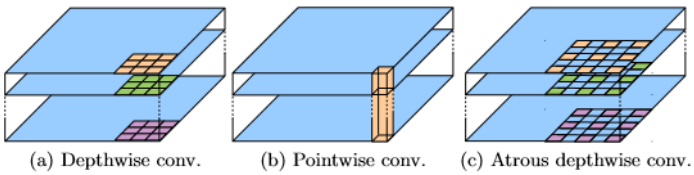

Figure 2.10: Depthwise separable convolution consists of (a) a depthwise convolution and (b) a pointwise convolution [51].

2.3.5 Encoder VGG16

VGG16[48] is a famous convolutional neural network created for complex classification problems, such as classifying 14 million images belonging to 1000 classes. Its architecture is shown in figure 2.11. As presented, the model only accepts images of dimensions 224x224 and passes them through five blocks of convolutions and max pooling, using "ReLu" as the activation function. The convolutions use filters with size 3 by 3 and the pooling uses a stride of 2 by 2, improving on the implementation of other models, such as AlexNet [47], which uses larger filter sizes. After these blocks of convolutions and max pooling, there are three fully connected layers, also using "ReLu", with the third one having 1000 channels in which to perform classification. Finally, there is a "Softmax" layer for the output.

This model will be used, not as a standalone for classification, but because it's also a success when being used as part of a segmentation architecture. By replacing the fully connected layers for convolution and max-pooling blocks, it becomes a fully convolutional network, capable of feature extraction. By joining it with a decoder, the feature maps obtained with the usage of the filters can be turned into segmentation maps, returning a segmented version of the input.

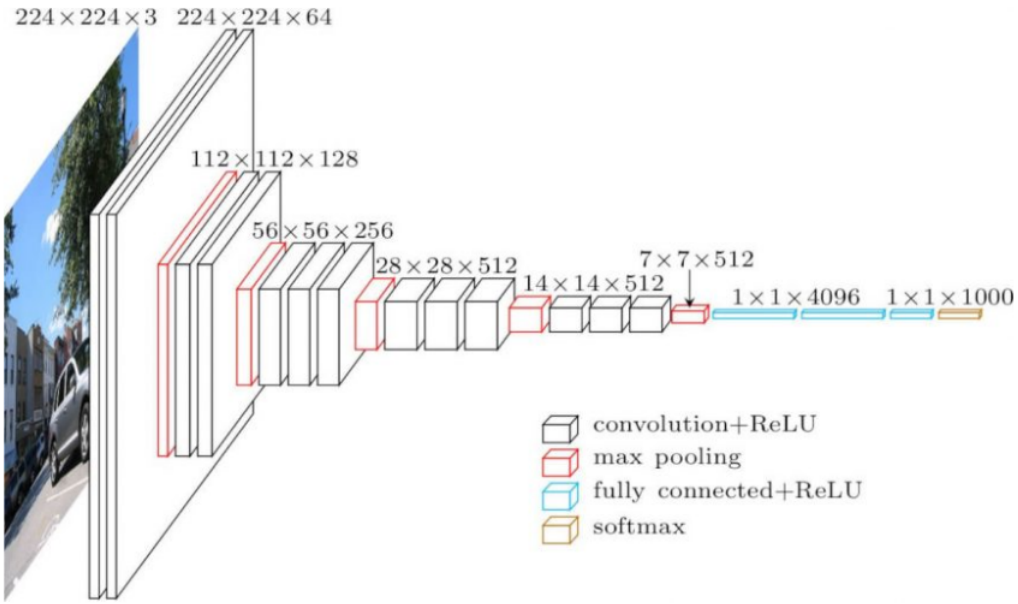

Figure 2.11: Architecture of VGG16.

Chapter 3

Experimental Work

3.1 Video recording and key frames

With the final goal of identifying each sample's lanes on each video, two different approaches will be implemented and its results compared. Both will focus on examining the frames of each video to have a temporal analysis, which will then be used to calculate the final product.

Before diving into the different paths of work, it's necessary to obtain the basis of the data, as in, dividing every video into frames and identifying which frames are worth analysing. In terms of value, the best frames will be I-Frames (intraframes), also known as key frames, which are single frames that the compressor examines independently from the frames that precede and follow it and stores all of the data needed to display that frame. These frames possess the most information out of all the frames, meaning they should be the correct choice to use from here on. In order to obtain these, FFmpeg[54] was used. FFmpeg is an open-source software run in the command line, that can be used to record, convert and tweak both video and audio. With it, all the key frames were extracted and stored. Finally, one third of the videos and their respective frames were reserved for future testing, while the rest were used as the basis of datasets for training and validation of both approaches.

With the frames now saved and separated, it's time to focus on each approach and its details.

3.2 Binary classifier

The first approach involves using a classifier to analyse one fragment of a frame at a time and distinguish it between a band/well and any other object, such as gel, cables, glass and

others..., considered as background. By fragmenting every frame and feeding each small section to this classifier, it's possible to identify the coordinates of the bands/wells in each frame and, finally, use these to draw vertical lines that correspond to each well/band's lane. For this, there's a need to create a binary dataset of these fragments to train and validate the model.

First of all, 27 frames, one from each video belonging to the training set previously mentioned, were randomly selected and their orientation was corrected, so that every image would have the wells on the top (figure 3.1). Then, every frame was fragmented into 50x50 pixels images, that being the approximate size of a well/band, starting from the top-left corner and in a horizontal direction. Adding to it, the slicer would only move 25 pixels after each cut, meaning there were fragments with overlaps, in order to increase the chances of obtaining fragments with only wells/bands (figure 3.2) or background (figure 3.3), instead of images with both. This first dataset totaled 59850 fragments.

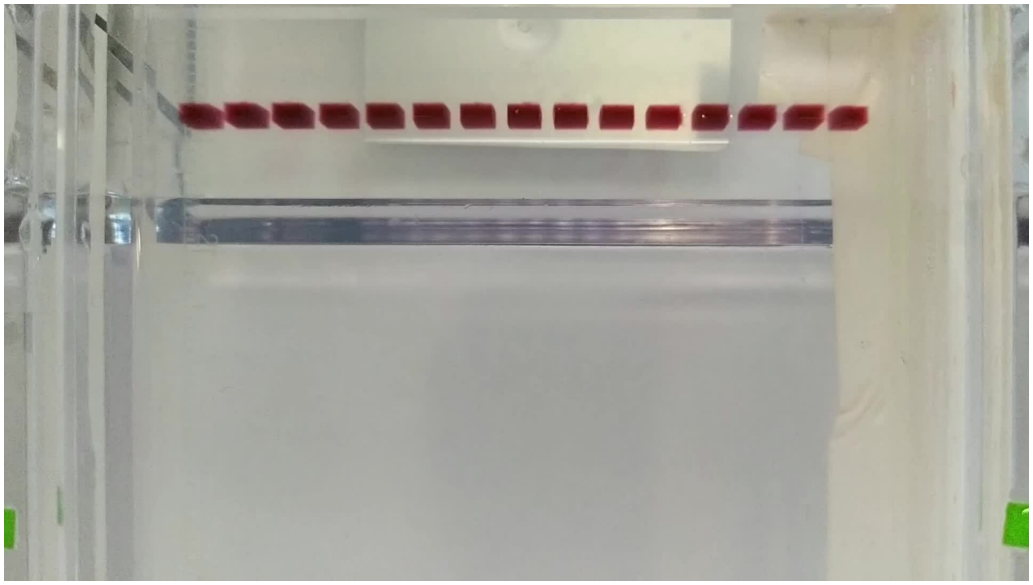

Figure 3.1: Electrophoresis gel procedure with corrected orientation. The wells are placed on top of the image.

Figure 3.2: Examples of bands and wells used in the dataset.

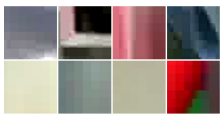

Figure 3.3: Examples of background used in the dataset.

3.2.1 Reducing dimensions with PCA and UMAP

This dataset needed to be divided into two labels, bands/wells and background, so that it could be used for training. While it's easy to obtain a huge number of background examples from that pool, even without using any techniques, due to them occupying the majority of each frame, there's a need to find a good number of bands and wells' examples, which is a more complex matter. These, however, have inherent features that are similar in almost every video, which means it's possible to group them by these features to obtain the needed examples.

For that, PCA and UMAP were used. Being tools capable of reducing the dimensionality of the data, using them on any image dataset will create a map where images with similar features will be grouped together. By manually identifying several examples of bands and wells and passing them alongside the rest of the previously mention dataset to these algorithms, it's possible to observe where do these labeled examples congregate and, after, it's merely a question of obtaining the closest images to them and evaluating them on whether they are bands/wells as well or not, saving them for the future labeled dataset or discarding them respectively.

PCA was tested with first. Figure 3.4 refers to the mapping of the dataset after the reduction of the dimensions. As depicted, there is an overlap of both categories of data, with the labeled examples, red dots, being scattered everywhere. Although there are defined regions of data conglomerates, these are likely to be background images, which is not as useful as clusters of bands and wells.

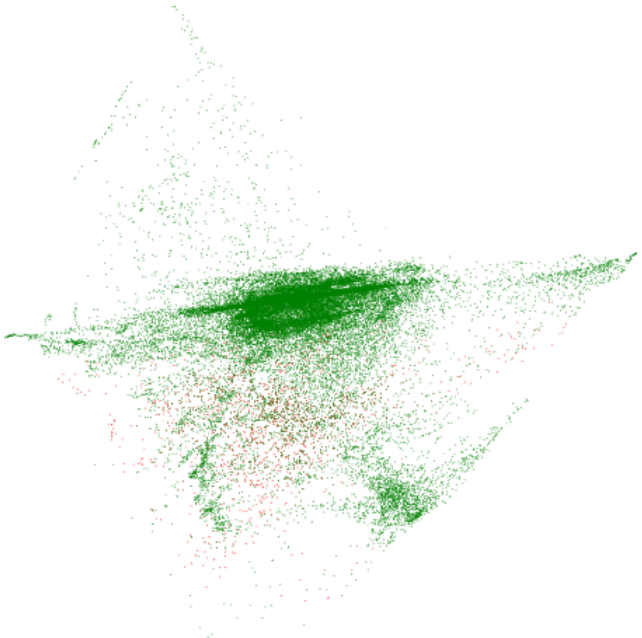

Figure 3.4: Dataset mapping using PCA. The labeled examples are depicted as red, while the rest of the data is represented as green.

Umap was tested secondly. Figure 3.5 shows the mapping of the dataset after the reduction of the dimensions. The selected "band" examples, red dots, are majority congregated in the middle of the image, so that's the region of interest that must be obtained.

Figure 3.5: Dataset mapping using UMAP. The labeled examples are depicted as red, while the rest of the data is represented as green.

The next step in order to obtain the bands and wells' examples was to identify and evaluate the neighbours of the labeled data. For this, K-means was applied, with the number of clusters to be identified being increased in every run. The goal was to let K-means separate the data into clusters and then evaluate the clusters that also possessed the labeled bands. A low number of clusters, however, meant these were bigger in size, increasing the chance of background images being present in them. That is why the number was increased several times, until the size of the clusters that had the labeled data stopped decreasing. Figure 3.6 represents the clusters in the mapping of the dataset that were used. This format was the first to distinguish the region which had the labeled bands from the bigger clusters with a lot of background samples. Lastly, the three clusters with the labeled bands were identified and the samples extracted. After manual evaluation, two of the clusters were discarded, since the majority of the samples weren't bands or wells and the last cluster was manually filtered, with the few incorrect samples being discarded.

Figure 3.6: Map divided into 7 clusters by K-means. Out of the three clusters with labeled data, the purple and red ones were discarded, with only the orange one being majorly consisted of bands and wells.

3.2.2 Fragments dataset

After eliminating the incorrect samples, the number of bands and wells that were extracted totaled 1519. These numbers furtherly decreased to about 400, removing images that showed only half or less of the band or well and others which had two bands and gel in between. The only images left were those that consisted of about 80% of the image being a band or well, so that the model could learn with more precision to identify fragments that possessed bands or wells on the center and ignore those that are background, but still show a small part of a band or well.

The disparity between this number and the number of all "background" fragments is enormous, meaning to create a big enough dataset, there was the need to increase the number of images regarding the label "bands". For that, several techniques of data augmentation were applied, such as rotations and transformations to the images in terms of color, contrast and light. The final product was a 20000 entries binary dataset, with the labels "band" and "background".

This dataset was split and used to train and validate several models. A test dataset was also manually created out of the videos and their respective frames that had been

Model	Conv. 1	Conv. 2	Conv. 3	Flatten	Dense layer	Dense layer	Loss
1	256	N/A	N/A	Flatten	N/A	1	8.48
2	64	N/A	N/A	Flatten	64	1	13.07
3	32	64	N/A	Flatten	64	1	14.33
4	64	128	256	Flatten	64	1	16.97
5	64	128	N/A	Flatten	64	1	17.82
6	32	64	128	Flatten	64	1	28.05
7	16	32	N/A	Flatten	32	1	31.12

Table 3.1: CNN architectures and results. The convolutional blocks' columns (Conv.1, Conv.2, Conv.3) represent the number of filters used in each layer (N/A when the layer was not used). The last dense layer is always used for classification. The loss values refer to the validation loss of each model.

saved up for testing. This one was simply used as a practical test to confirm the models could identify the correct label of the test images after being trained and validated. As so, this test dataset was small, with around 50 images in total (25 bands and 25 background images).

3.2.3 CNN as binary classifier

All the models trained and validated are sequential models from keras, with the input being the 20000 entries binary dataset of "background" and "band" images of 50x50 pixels. In order to obtain the best models, every model ran 200 epochs, with each epoch's validation loss being evaluated and the model's state saved, if the value was better than the one that had been saved previously.

The constitution of the models varied greatly so as to find the best model possible for the problem at hand within the capabilities of the hardware used. The convolution and max-pooling blocks varied in number, between one and four, with the convolution's filters ranging between 32 and 512. After flattening, the number of dense layers used was either one or two, with one of them always being used for classification as normal. These experiments were executed with loops.

Table 3.1 depicts the seven best performing models out of the more than thirty trained and validated models. These models were ranked according to their validation loss, with the best one having only one convolution block (convolution and by max-pooling) with 256 filters, followed by flattening and the one dense layer of 1 unit, to output the classification. The table also presents the architecture of each model, with the number of convolution blocks, the number of filters per block, the number of dense layers and the number of units per dense layer.

All convolutional layers had a kernel size of 3 by 3 and all max-pooling had a pool size of 2 by 2. The convolutional layers and the extra dense layer with varying size used activation function "ReLu", with the dense layer used for classification applying a "Sigmoid" function. The models were always compiled with "binary crossentropy"

as the loss function and "adam" as the optimizer. The batch size was fixed on 64 and the validation split was 0.3 with shuffling enabled. Figures 3.7, 3.8, 3.9 and 3.10 are examples of single frames that were used as input in some of these models, having their bands identified.

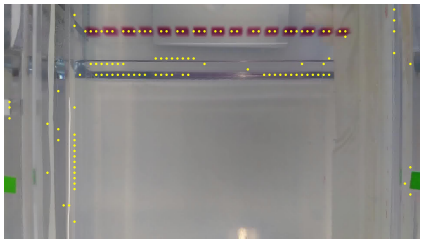

Figure 3.7: Single frame of video at the start of the migration.

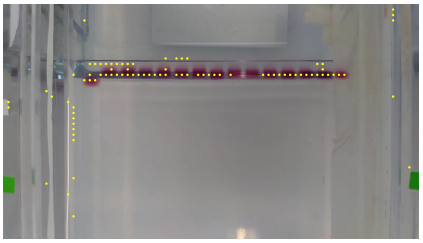

Figure 3.8: Single frame of video after 10 minutes of migration.

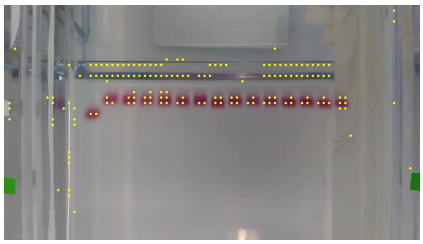

Figure 3.9: Single frame of video after 15 minutes of migration.

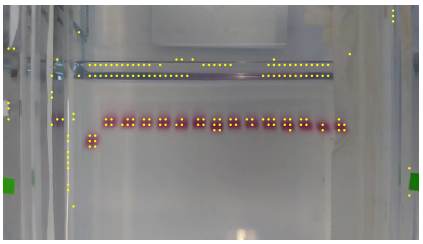

Figure 3.10: Single frame of video by the end of migration.

3.2.4 Identifying the lanes

After running several models, the three with the lowest validation loss and highest accuracy were picked and used to analyse some of the frames from each video. As every video had close to 2000 frames, with most of them showing almost no temporal progression, the frames were filtered, with only one out of fifty frames being used (around 40 frames per video).

There was also the problem of trembling during the video recordings, which made some frames not aligned with the majority, harming the coordinates that would be stored after classification. To fix it, the CV2 library[55] was used, which can compare an image with incorrect orientation to a correct one, find regions that are equal in both, calculate the transformations necessary to move those regions from the first to the second image and then apply those transformations to the whole image, correcting the orientation. However, this implementation failed quite a few times, most likely due to the identification of wrong regions and so it was left out of the final result. As all the trembles occurred during the beginning of the recordings, due to the shifting of the recorder to its correct position, the first 49 frames were discarded and the classification started from the 50th frame onward.

Every fragment classified as "band" had its center's coordinates stored (figure 3.12) and at the end of the analysis of all the frames of a video, all the coordinates with the same x value were used to draw vertical lines, representing the lanes, on one of the frames. In order to help evaluate whether the lines were placed correctly on top of the lanes, all the frames used were summed with an opacity value, so that a final image with all the bands across time would be used to draw the lines. Since the more frames used for classification, the lower the quality of the summed image, the contrast was increased to improve visibility of the bands, creating figure 3.11.

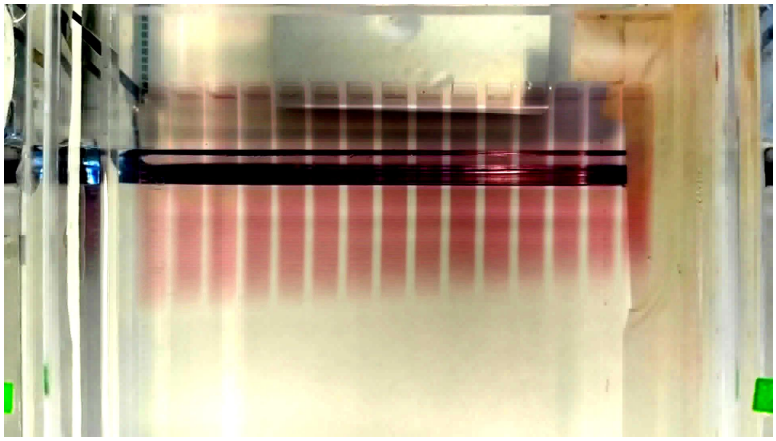

Figure 3.11: Result of the sum of all the analysed frames of a video. Contrast was enhanced to provide better visualization of the lanes.

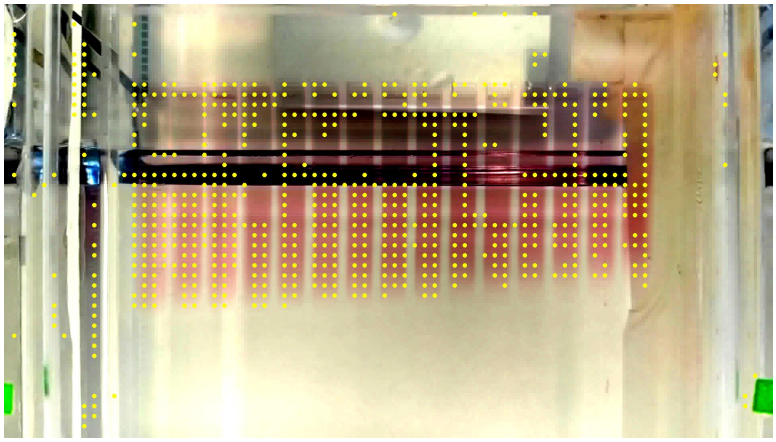

Figure 3.12: Result of the sum of all the analysed frames of a video. Contrast was enhanced to provide better visualization of the lanes. Every dot represents a center identified by the model across all frames of a video.

The three picked models were models 1, 2 and 3 from table 3.1. Model 1 is the simplest of the three, with only one convolution block and one dense layer for classification. Model 2 also has one convolution block, but it has a dense layer after the flatten and before the dense layer for classification. Lastly, model 3 is the most complex, with two convolution blocks and the two dense layers, just like the previous one. In terms of validation loss,

model 1 was superior to the other two, with them having similar values.

After using all models for classification, the results regarding the same video were compared, to check whether model 1 truly was the best option out of the three. Figures 3.13, 3.14 and 3.15 show the results of the classification of a particular video, done by model 1, 2 and 3 respectively. As expected, model 1 is both capable of identifying a higher number of true positives and a lower number of false positives. Many more results were compared to be certain this pattern kept occurring and it was not a one-of. The vast majority of comparisons served to support the claim that model 1 was, indeed, the best performing model for the classification approach and so, its results were the ones used henceforth.

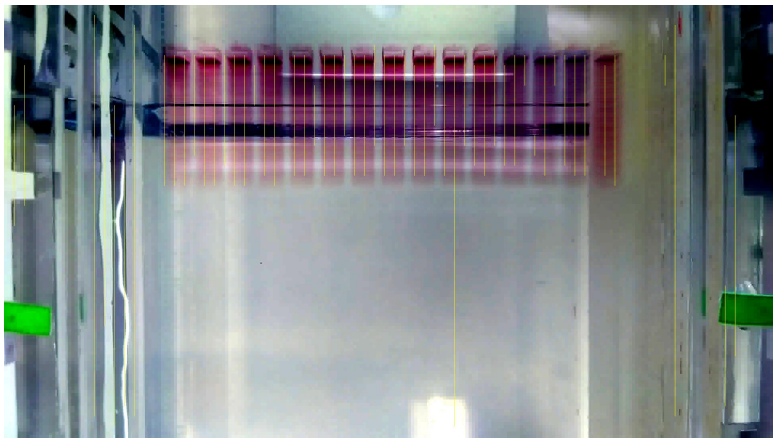

Figure 3.13: Result of the sum of all the analysed frames of a video by model 1. Contrast was enhanced to provide better visualization of the lanes. All lanes have been correctly identified, with only 10 false lanes.

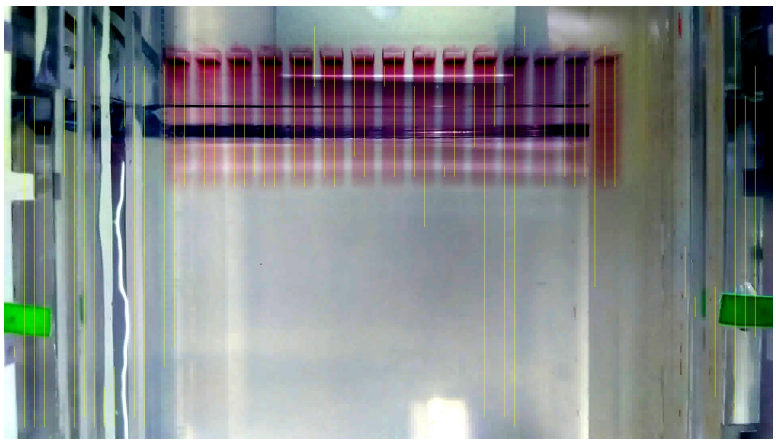

Figure 3.14: Result of the sum of all the analysed frames of a video by model 2. Contrast was enhanced to provide better visualization of the lanes. All lanes have been correctly identified. There are more than 15 false lanes.

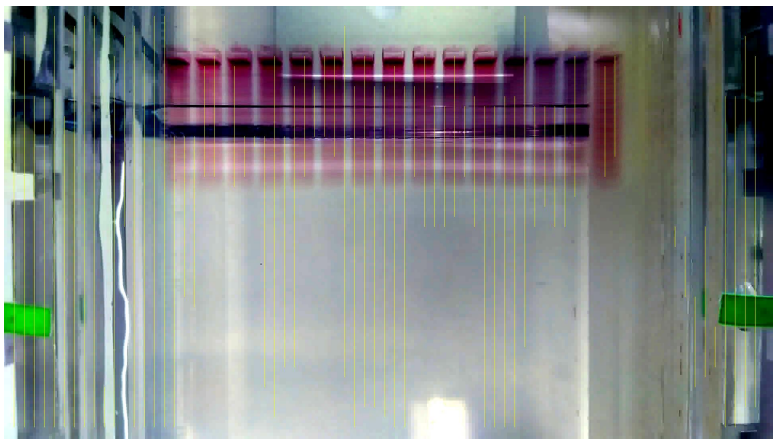

Figure 3.15: Result of the sum of all the analysed frames of a video by model 3. Contrast was enhanced to provide better visualization of the lanes. Most lanes have only been half identified, starting in the middle of the lane and ending at the end of the box. There are more than 20 false lanes.

3.2.5 Preliminary results analysis

Model 1 was able to predict between 12 to 15 full-sized lanes out of 15 in more than 50% of the videos, while predicting half-sized lanes on the others. This might be due to the difference in setup and quality of the videos, with the ones with higher error rate (group 1) having a background with intense colors, reflecting light on the gel and box, red cables, which the model can wrongly identify as bands and one side of the box also being red (figure 3.16). On the contrary, the video with the lowest error rate (group 2) has a glass box, the reflection of light is less intense and there are no surrounding cables (figure 3.17). The visibility of the bands is also a factor, with the first videos having bands with low visibility to start with and that decreases even more by the end of it, while the bands of the second group of videos clearly show the wells and bands, even by the end of the procedure.

The main errors can be divided into two groups:

1. Incorrect placed lines;

2. Incorrect sized lines.

The incorrectly placed lines are due to false positives and can either occur far away from the lanes, as in the limits or outside of the box, or in between lanes. Incorrect sized lines usually occur due to false negatives, with lines being cut short or even starting lower than expected.

Several methods were tested in order to improve the results of model 1, with most being discarded either to being impractical or simply failing. Methods such as limiting the classifier analysis to the inside of the box, discarding all lines that are too far from the rest and selecting only the biggest lines are not viable, since those parameters (size of the box, distance between lanes, number of lanes, number of analysed frames, . . .) are

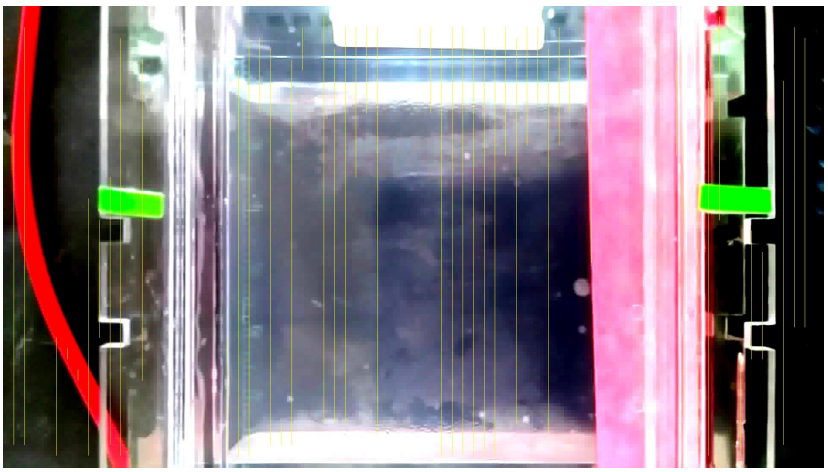

Figure 3.16: Result of the sum of all the analysed frames of a video. Contrast was enhanced to provide better visualization of the lanes. The red cable, red wall of the box and light reflected on the gel might be the cause of false positives.

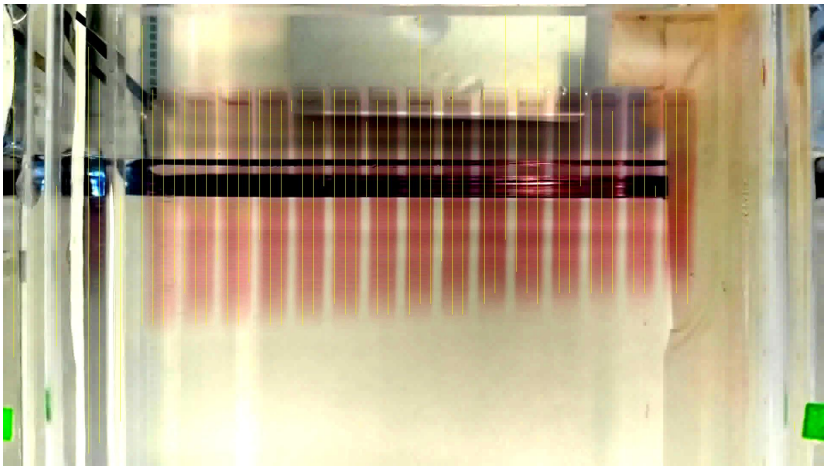

Figure 3.17: Result of the sum of all the analysed frames of a video. Contrast was enhanced to provide better visualization of the lanes. The background is clearer and without obstructions, possibly improving the results.

specific to each video. The solutions need to be abstract enough that they would work on any video used, even outside of this dataset.

The best solution found was to identify the "y" value with the highest number of starting points and discard all lines that do not start at that value. This works relatively well, since all wells are aligned at the start of the procedure and the classifier analyses the frame horizontally, meaning all lanes will start at the same "y". Even more, the color is most intense while the samples are restricted in the wells, so the likelihood of the classifier correctly identifying all of them is high. In practice, group 2's videos had most, if not all, incorrect lines disappear (figure 3.18). Unfortunately, this method can delete correct lines if the classifier failed to identify the starter point correctly, which happens mostly on group 1's videos.

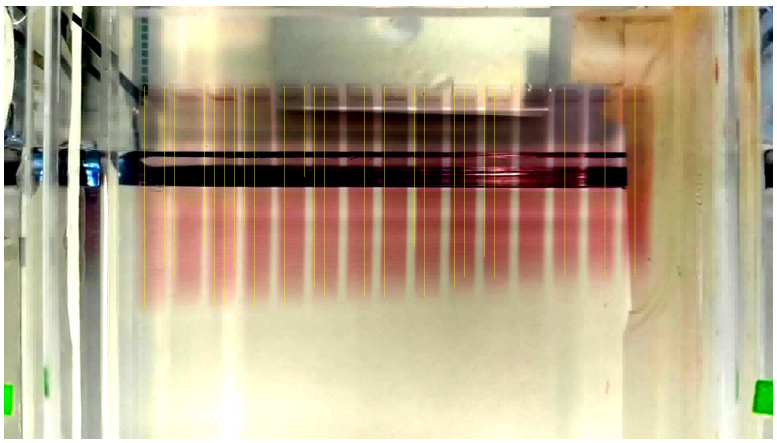

Figure 3.18: Result of the sum of all the analysed frames of a video. Contrast was enhanced to provide better visualization of the lanes. The starter point of the lanes was identified and all the lines with different start were discarded. Notice that some of the lines that were correct also got deleted, due to the failure of the classifier on identifying the correct start.

Another interesting solution was to discard all lines that were constituted by less than half of the number of frames being analysed. In theory, a perfect classifier would identify a particular band or well at least one time in the analysis of a particular frame. Considering the fact of the classifier also analysing the frames with an overlap of 25 pixels both vertically and horizontally, one band/well could even be identified multiple times. That means, the final lines should have at least one point for each frame. Since the classifier is not perfect, it will miss some of the bands and wells so, given a margin of half the points, discarding the lines with less than that amount should work relatively well. Unfortunately, while it seems to be a success in group 2, group 1 has a huge error rate, making this approach worse than the first one.

While these methods help improve the results somewhat, they are worthless when it comes to incorrect sized lines, as there is no way to identify the correct length of a lane, meaning no corrections are possible. Adding onto that, there are some lines existing on the limits of a lane or even in between lanes, which have the correct starter point, making them harder to discard depending on which method is used. Some of these could be fixed, however, with further processing, by knowing specific details that video's gel electrophoresis process, namely the number of samples used and the distance between wells. These two can enable the removal of extra lanes that exist in between wells, although, since it's information that changes from video to video, this approach was not tested with.

3.3 FCN and image segmentation

The second approach to solving the initial problem focuses on the use of a fully convolutional neural network to produce semantic segmentations of the frames of each video,

which tend to have better results and be less prone to errors in more complex regions of an image, when compared to simply segmenting using segmentation techniques. In doing so, there will be a clear distinction between the bands/wells and the rest of the objects, the background. Each segmented frame can then be analysed, with the center of each band/well being calculated and grouped to other centers close by. Finally, these groups are connected vertically to groups of the other frames, marking the lanes of each sample.

To accomplish this task, it was first required to obtain a fully convolutional neural network capable of correctly segmenting the frames, meaning there was a need to create a dataset with a minimum amount of examples of segmented frames and the original frames and feed them into the model, so that it could be properly trained and validated.

3.3.1 Image segmentation techniques

Even though the quality of segmentation using segmentation techniques is lower than using a network, since there's a requirement of segmentation examples, these can be used to create them, as long as the results are evaluated. This evaluation was done manually, by observing the segmented frame and comparing it to the original, pointing out the errors and concluding whether they were crass enough to deem the image as unusable, or if it could be saved to be used in the training and validation phase.

The starting dataset was comprised of the training set mentioned at the start. From that, around 90 images from each video were selected (the first and last frame and every frame multiple of 25), totaling close to 2300 frames, and used to create the potential examples of image segmentation that would be used further on.

To produce the segmentations from the original frames, the following techniques were implemented:

1. Otsu's method;
2. Background subtraction.

Otsu's method proved to be incapable of correctly segmenting the majority of the images, with only 758 segmentations being partially correct, so as to say most of the image being correctly segmented, but still needing manual corrections that proved quite time-consuming.

The most efficient technique was background subtraction, where all frames of a video were analysed by the algorithm BackgroundSubtractorMOG2, from the library CV2, that in turn identified the moving objects and the still objects, by comparing images and noting the regions that remained the same and those that changed in each one. This way, the algorithm was able to segment the moving objects (bands and wells) from the still objects (background) quite effectively (figure 3.19). This algorithm also permits the identification of shadows but, in this case, most of the shadows ended up being wells and bands or just plain background, so this feature was not used.

Although background subtraction proved to be the best semantic segmentation technique by far, there were still quite a few images that needed manual corrections and many that were deemed unusable for the next phase. By the end, 1322 segmented images were obtained, forming a final dataset of 2644 images, with the addition of the original frames.

Figure 3.19: Result of background subtraction for image segmentation.

3.3.2 FCN for image segmentation

As mentioned previously, even though the frames can be segmented using segmentation techniques, fully convolutional neural networks tend to have better results in image segmentation overall. With this in mind, the techniques were used to obtain viable data, which was then manually filtered, and could now be used to train the segmentation models.

All the models trained and validated were fully convolutional neural networks, used for image segmentation, from the open-source library "image-segmentation-keras"[56]. This library permits the training, validation and testing of several FCN models, that use pre-trained encoders, and also helps with the preparation of the dataset, augmentation of data, evaluation of the models and other features.

The dataset used for the training and validation steps consisted of the 2644 images mentioned previously, all with the same orientation, with the wells' starting position being on the top of the image and moving downwards across time. All the images were 1920x1080, which poses a problem for the pre-trained encoders, which need the dimensions to be multiple of 32, due to the size of the filters and the number of convolution blocks. To fix this issue, the library resizes the original input into an acceptable size before training and validation. Since the size of the images was far bigger than usual for this kind of operation, there was a chance of it increasing the error rate. As such, several

Segmentation Model	Input Dimensions	Usage of Data Augmentation	mIoU(%)
FCN 8	1920x1056	No	82.08
FCN 8	1344x768	No	86.17
FCN 8	1280x736	No	85.91
FCN 8	640x352	No	85.48
FCN 32	1280x736	No	84.20
U-Net	1920x1056	No	79.69
U-Net	1344x768	No	87.26
U-Net	1280x736	No	89.03
U-Net	640x352	No	88.96

Table 3.2: Results of the training and validation of the segmentation models chosen.

attempts were made at training and validation, each with the frames being resized with different measures. The values used were 1920x1056, 1344x768, 1280x736 and 640x352. The smaller the images, the bigger the number of images that could fit in a batch, with the batch size of each try being 1, 2, 3 and 12 respectively. In other words, the closer the dimensions were to their original size, the slower the training and validation process was.

The following FCN models were trained and validated:

1. Segmentation model FCN 8 with VGG16 as encoder;
2. Segmentation model FCN 32 with VGG16 as encoder;
3. Segmentation model U-Net with VGG16 as encoder.

Since the library has a large number of model architectures available and it's only needed to select which one to train, it accelerates the whole process, making it possible to test more models than expected. All of these networks' architectures involved using a pre-trained VGG16 as the encoder part, followed by the usual decoder section of each particular network. It is worth noting that these networks were also tested with Resnet 50, as a substitute to VGG16, to increase the likelihood of finding an even better model, but all attempts at training failed. This could be due to the machine's hardware limitations, not being able to handle the training, or issues within the library itself, since some other functionalities were also not working properly. As so, the three mentioned networks were the only ones trained and validated from this library.

The evaluation was made by the use of the library's resources, namely the use of mean Intersection over Union (mIoU). IoU is commonly used to evaluate image segmentation, it calculates the area of overlap between the predicted segmentation and the ground truth and divides it by the area of union between the predicted segmentation and the ground truth. In this case, since it's a binary segmentation, the mIoU is the average of the IoU of each class. The results can be observed in table 3.2.

According to these results, the model chosen to execute the segmentation of the frames from each video was the U-Net model, with the input being resized to 1280x736.

3.3.2.1 Simple U-Net

The common denominator out of all the previous tested models was the usage of an encoder that was created to solve complex problems. The VGG16 is capable of such feats as classifying images into 1000 object categories, which is a much bigger scale when compared to the present problem of segmenting bands and wells. In order to further increase the pool of results, a simpler model should also be trained and validated, to prove whether it is enough to solve this and even if it somehow produces close results to such a powerful network as the VGG16.

Since, according to table 3.2, the best model is a U-Net with VGG16, the model used for this particular test will be a U-Net, from the open-source library "unet"[57], with no pre-train and with slight changes to its architecture. The original U-Net (figure 3.20) is constituted by ten layers, with the last one returning the output and the rest of them being divided into encoder (convolution/pooling) and decoder (up-convolution/merge). By removing the two last layers from the encoder and the respective two layers from the decoder that merged with them, the model becomes a six-layer U-Net, with the middle layer having 256 perceptrons, instead of 1024.

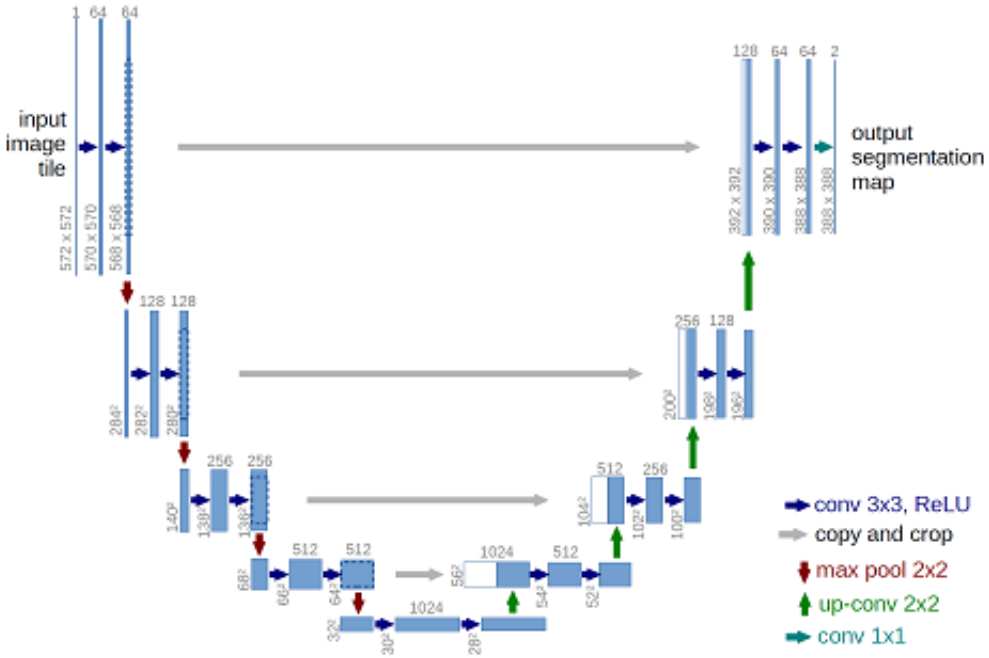

Figure 3.20: U-Net architecture[49].

As for the input, although the dataset is the same, instead of resizing the input such as before, an alternative was tested. Since, usually, the inputs tend to be small, in order to facilitate the learning process, the 1920x1080 images and respective annotations were divided into squares of 120x120 pixels. This makes it so that there are approximately two bands per square. From the results, around 20000 images and respective annotations were randomly selected and the model was trained and validated with these squares

Segmentation Model	Input Dimensions	Usage of Data Augmentation	mIoU(%)
Simple U-Net	120x120	No	48.57

Table 3.3: Results of the training and validation of the simple U-Net.

(validation split of 0.3), with the version presenting the best validation loss being saved for later use. The objective was to prepare a model that segments the small images of a frame, with these, in turn, being reattached to create a segmented version of the original frame. These can then be analysed the same way the results of the pre-trained U-Net would be. Table 3.3 presents the results of the training and validation of the Simple U-Net.

3.3.3 Segmenting frames and identifying lanes

As in the first approach, around 40 images were used per video (every 50th frame) in the segmentation process, with the resulting segmentations being stored (figure 3.21). The next step was to find the center of each well/band and save its coordinates for calculating the lanes, meaning, there was a need to identify all the pixels that presented the particular value (color) used during the segmentation. Normally, since the output function of these models is sigmoid, meaning the results are values in a range of 0 and 1, there would only be the need to use a threshold value of 0.5, in order to separate the pixels into two classes, creating a binary version of the frame, that could then be analysed. However, when finishing every segmentation, instead of returning a frame where each pixel's value would vary between 0 and 1, the library would use a scale to assign RGB values to each pixel, depending on the output value of the sigmoid function. This way, the segmentation would still be correct, although it would be a colored image. Due to this peculiarity, each segmented frame had to be turned into a greyscale version, which was then turned into a binary version, by using the mean value between the highest value of background and the lowest value of band registered as a threshold, dividing the pixels correctly into two distinct pixel values.

Figure 3.21: Result of the segmentation of a frame executed by the trained model.

With the frames now ready for the analysis, the cv2 function "findCountours" was applied, being able to identify the limits of the several bands and wells, saving the groups of coordinates, which were used in the calculation of the centers. These centers were stored until the last frame of the video was scanned and were then drawn in the sum of the frames, just like on the first approach (figure 3.22). While on the first approach the centers were already aligned and vertical lines could be drawn from one center to the next, in this case, there was a need to align the centers first. For this, a simple function was created to group the centers, accepting a distance as input, checking whether every center had another center in its vicinity (less than or equal to the input distance) and creating a mean center from those two. After every center was checked, the process was reiterated, this time with the new mean centers instead of the original ones and kept going until no new mean centers were created. With the centers now properly aligned, the vertical lines can be drawn on the sum of the frames (figure 3.23).

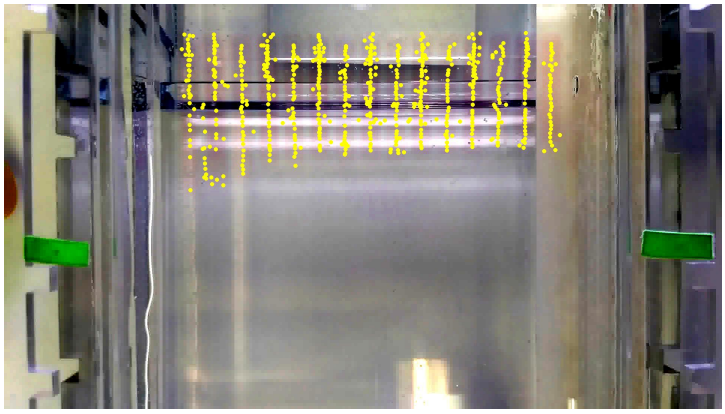

Figure 3.22: Result of the sum of all the analysed frames of a video. Contrast was enhanced to provide better visualization of the lanes. Each yellow dot represents the center of a well/band that was identified in the segmented frames.

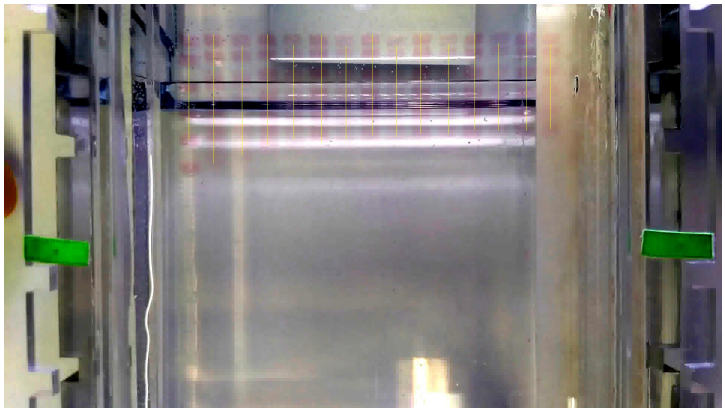

Figure 3.23: Result of the sum of all the analysed frames of a video. Contrast was enhanced to provide better visualization of the lanes. The starter point of the lanes was identified and all the lines with different start were discarded. Each yellow line represents the lane that was created by aligning the centers of the bands/wells.

3.3.4 Preliminary analysis of results

Overall, both U-Net models mentioned previously were able to properly segment most of the frames of each video. The quality of these segmentations, in general, was quite high and the algorithm used to locate the regions of interest had no problem identifying almost every well and band across time and calculating each sample's lane.

Just as it happened in the previous approach, there were visible differences between the results of the first half of videos (group 1) and the rest of them (group 2). In this case, however, the problems did not arise from the exterior of the box or even from the box itself. The models had no difficulties dealing with the red cables and the red wall of the box, only being challenged in identifying the samples after they left the wells. Most of the bands were not segmented close to the end of these videos, meaning the percentage of lanes identified from the wells to the end position of the sample is way lower than the percentage of lanes identified from the start to the middle position of the sample. This is probably due to the decrease in the intensity of the color across time, since the samples diffuse in the gel, which alongside a background with intense colors and the reflecting light on the gel can make it harder for the models to correctly distinguish between sample and background.

The main errors observed are:

1. Fused bands in the last segmented frames;
2. Fused bands in the middle segmented frames.

Both problems are similar when observed, but they stem from different events and majorly occur on the videos of group 2, which have better image quality, a cleaner background, less light reflection on the gel and so, better visibility of the samples across time. The first problem occurs by the end of the videos, where the bands are quite diffused, yet identifiable due to the favourable visibility conditions. The models modify the threshold value to properly segment the bands, but it also segments some of the adjacent gel. This leads to large segmented bands that tend to appear fused, even though in reality, they did not. The second problem is caused by a specific region of the box, below the gel, which the models cannot segment properly when the samples are above it. The result is segmented bands divided horizontally into two and some even fusing with their neighbours.

An extra problem was identified in the segmentation results of the Simple U-net, whereas the small squared sections passed onto it for segmentation have their corners segmented as if they had a portion of a band or a well, even if they do not. This makes it so that when all segmented sections are put back together, the resulting frame has a lot of small segmented regions that are incorrect.

Even with these errors, the quality of the results seemed more than acceptable, and so, no post-processing was done to improve them.

Chapter 4

Results

With the two approaches completed, it's time to evaluate the results as well as analyse the biggest flaws in both procedures. This way, it will be easier to compare them and reach a conclusion about which one is the best at solving the initial problem or whether none of these can accomplish that which they were supposed to achieve. It's important to note that, since there is no algorithm or technique to confirm whether an identified lane is real or fake, or if their starting and ending points are correct, or even if a calculated center is a true or false positive, it becomes quite troubling to confer the results in a computational way, so the best way to evaluate them is through observation. The next sections will offer a brief summary of each process alongside several images of the intermediate and final results, to help the reader analyse them properly. Afterward, statistics will be presented, although these were compiled through visual observation and manual counting, being empirically correct, but not computationally certain.

4.1 Image classification results

Starting with the first approach, the objective was to train a neural network for binary classification of small chunks of a frame. The model was then used to classify every chunk or fragment of multiple frames of every video (figure 4.1), applying an overlap in order to increase the likelihood of finding the centers of the bands/wells. The coordinates of the center of each fragment were saved, if these were classified as band/well (figure 4.2). Finally, the saved coordinates were used to draw vertical lines, with every lane consisting of one to three lines, depending on the thickness of the band/well and the proximity of the camera to the gel when recording (figure 4.3).

To decrease the occurrence of false positives, several attempts were made, with the most promising one being the selection of the lines by starting point, where the "y" value

with the highest number of starting points was found and all the lines that did not start at that "y" value were discarded (figure 4.4). Although this was the option with the best results, which were quite an improvement from the original ones, they still proved to be lacking, with the price of deleting several wrong lines meaning the disappearance of some correct lines. Nonetheless, these will be the results analysed from now on.

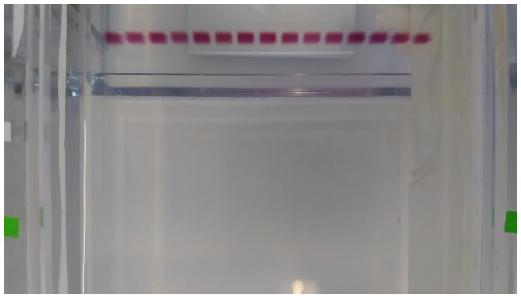

Figure 4.1: Key frame obtained by dividing a video into frames.

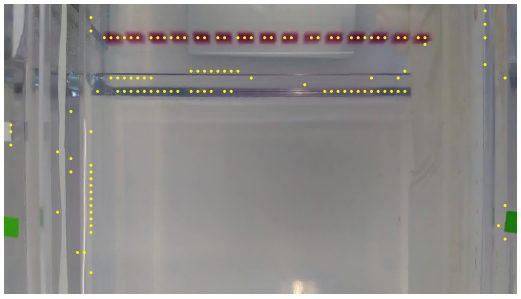

Figure 4.2: Key frame after wells are identified and its centers marked.

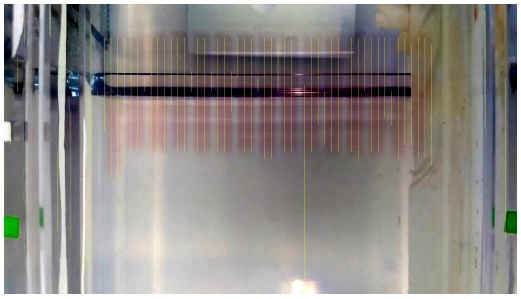

Figure 4.3: Overlapped frames with vertical lines representing the lanes.

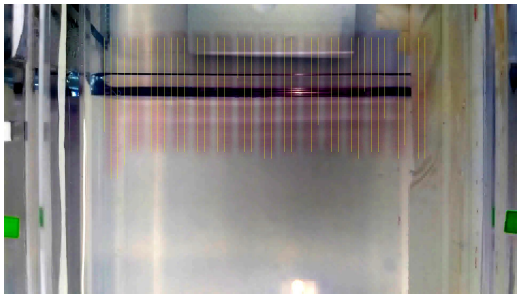

Figure 4.4: Overlapped frames with vertical lines that start in the same "y".

The simplest and perhaps biggest flaw of this approach is in the model's accuracy. The amount of false positives registered is too high, negatively impacting the end results. Even with the corrections made previously, there are still some false positives and true positives are also affected, which continues to be a negative outcome.

The other major issue is with the overlapping of fragments to be classified. The idea with this was to increase the odds of a band/well being fully sliced in the fragment that is being classified and just like that, having the center of the lane revealed. Then it was only a matter of grouping all the centers by the distance of the overlap and the coordinates that belonged to the same band/well would become unified whilst being separated from the other bands/wells. This, however, did not work as intended, since the accuracy of the model was lower than expected. With false positives being detected in between bands/wells, the distance method of separating the coordinates saved stopped working, as there was no way to determine if a particular coordinate is fake and the distance between the real ones of the different bands/wells are equidistant to it. This makes it so that instead of uniting two or three vertical lines to create the final predicted

lane, some would unite more than five lines, because the distance between them was all the same, with false positives in the mix. This problem managed to ruin quite a few predictions, creating predicted lanes between real lanes or even one predicted lane per two or three real ones.

Figure 4.5 presents both problems, with the first one happening outside of the limits of the box and the second one plaguing the inside of the box.

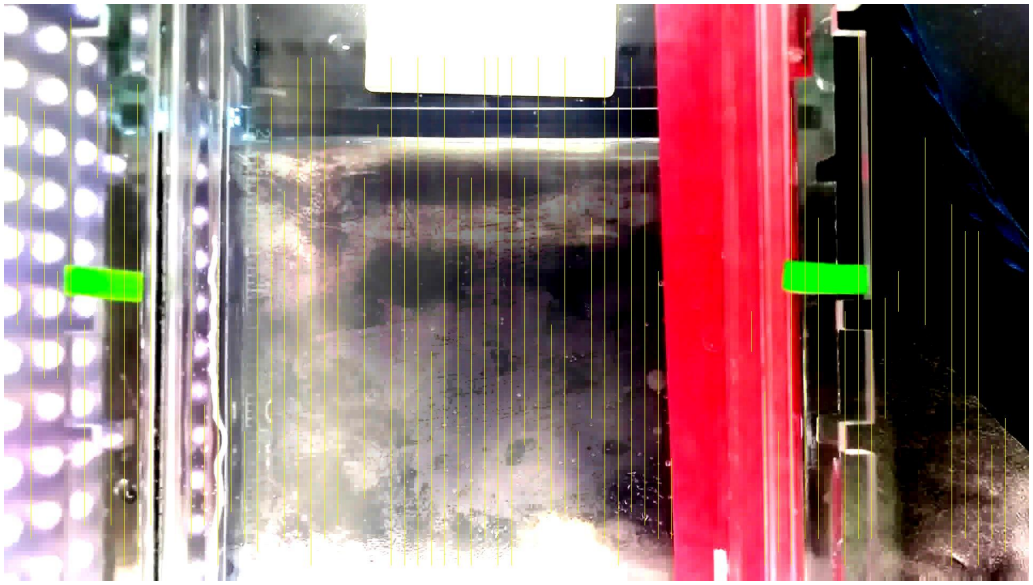

Figure 4.5: Result of the sum of all the analysed frames of a video. Contrast was enhanced to provide a better visualization of the lanes. Outside of the box limits, the first problem is apparent, where many false lanes were identified. Inside of the box limits, the second problem is visible, with many false lanes making it impossible to group the correct ones resulting from the overlap together.

4.2 Image segmentation results

Onto the second approach, the objective was to train a fully convolutional neural network in semantic segmentation, using some examples created with simple segmentation techniques (figure 4.7), in order to segment the frames of each video (figure 4.6), separating the bands/wells from the background (figure 4.8).

The segmented frames were then swept and the bands/wells identified, with the center of each one being calculated and saved (figure 4.9). Finally, the saved coordinates were grouped with other points at close distance and a mean was made to better align them (figure 4.10), creating the vertical lines that represent the lanes (figure 4.11).

This approach's two biggest flaws occur during the segmentation process. The first one is regarding the segmentation of bands at later stages of electrophoresis. Since the samples diffuse during the procedure, the bands tend to become larger along the time, but also become harder to distinguish from the background, which makes it harder for

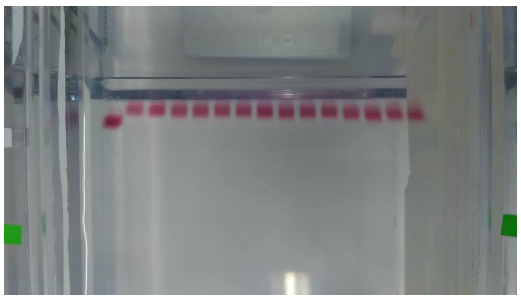

Figure 4.6: Key frame obtained by dividing a video into frames.

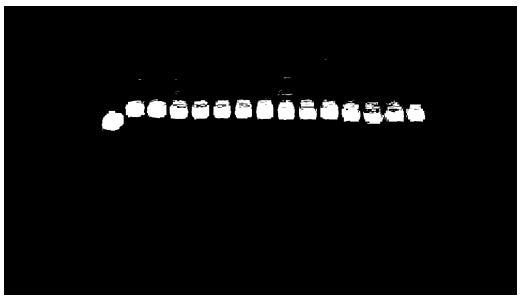

Figure 4.7: Key frame segmented by subtracting the background.

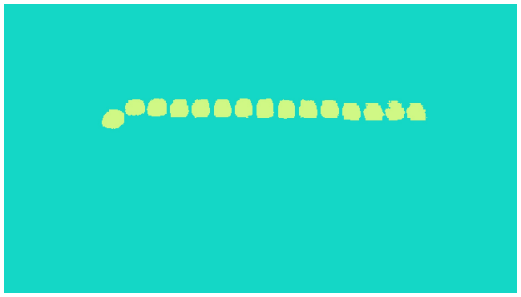

Figure 4.8: Key frame segmented by the trained FCN.

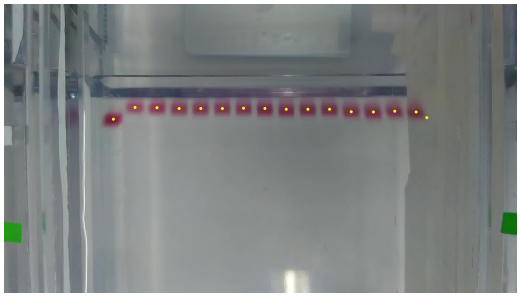

Figure 4.9: Key frame after bands are identified and centers are calculated.

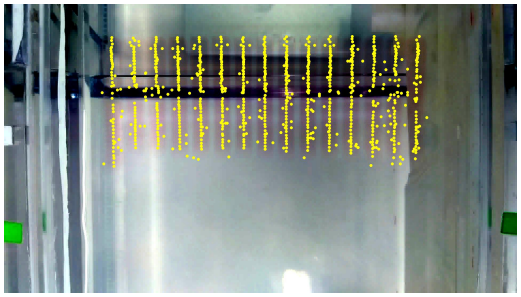

Figure 4.10: Overlapped frames with all the bands and wells' centers across time.

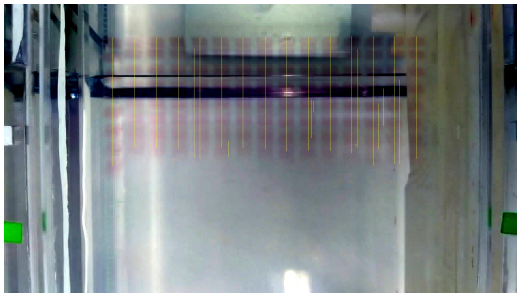

Figure 4.11: Overlapped frames with vertical lines representing the lanes.

the model to segment properly. To adapt to this change, the threshold value is modified, segmenting not only the band, but also the adjacent gel. The problem comes from the fact that, by the end of the video, the segmented bands are so large that they fuse with other bands (figure 4.12). The algorithm used to identify the bands will then find one band instead of two or even three, miscalculating the center. As this happens by the end, the centers will not be aligned correctly, which will affect the end point of the vertical line that represents the lanes, making them end before the correct spot (figure 4.13).

The second problem is akin to the first one, where there is a mixing between segmented bands, creating the same problem. However, this one happens not at the end, but in the middle and is caused by the box in which the procedure is done. This box has a horizontal part in the middle, below the gel, that is quite visible and has a different tone

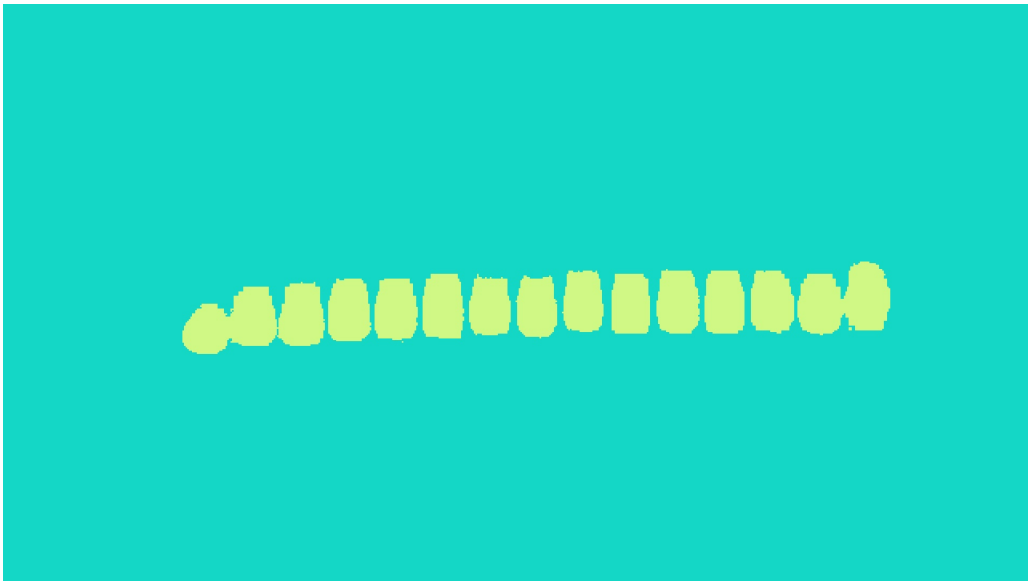

Figure 4.12: Segmentation result of a frame of a video. Both the first two and the last two bands are fused together, being treated as one entity with an incorrect center.

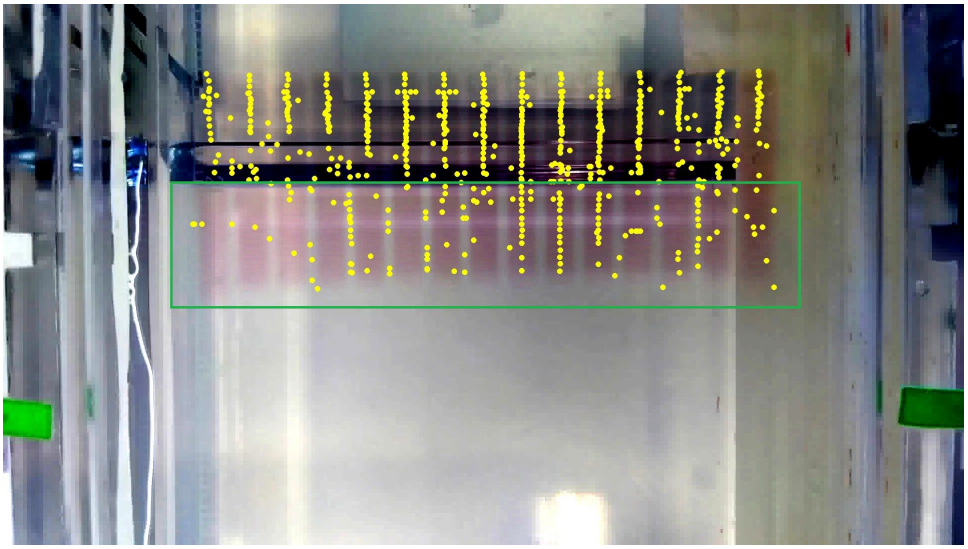

Figure 4.13: Lane results with only the centers drawn. These centers were the ones calculated, with no neighbouring being applied, in order to better show the problems of this approach. The green box presents the result of the band fusion, where several centers are located in between lanes, instead of being on top of them.

than the rest of the box, making it stand out during the segmentation process. While manual corrections were made to the training and validation dataset, in order to properly teach the model and, in doing so, limiting the impact of this part on the segmentation results, when the bands stand aligned and directly above it, it becomes hard to ignore it and it influences the segmentation, in some cases fusing the bands (figure 4.14). Although this happens often, it's not as much of a problem as the previous point, since there are still plenty of centers before and after that are aligned correctly, so the ones in the wrong

tend to be correctly aligned after the neighbour search and mean calculation. Even in the worst cases, when the distance applied to the neighbouring algorithm is not enough to align them, the points can be ignored, with the vertical lines using the centers before and after (figure 4.15).

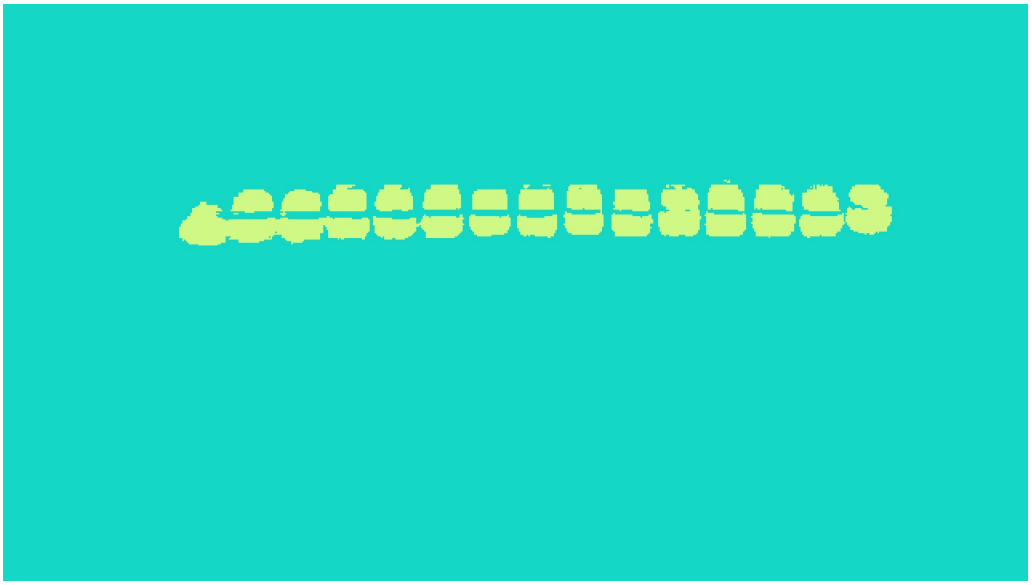

Figure 4.14: Segmentation result of a frame of a video. The box interferes with the segmentation, fusing some bands and fragmenting almost all of them.

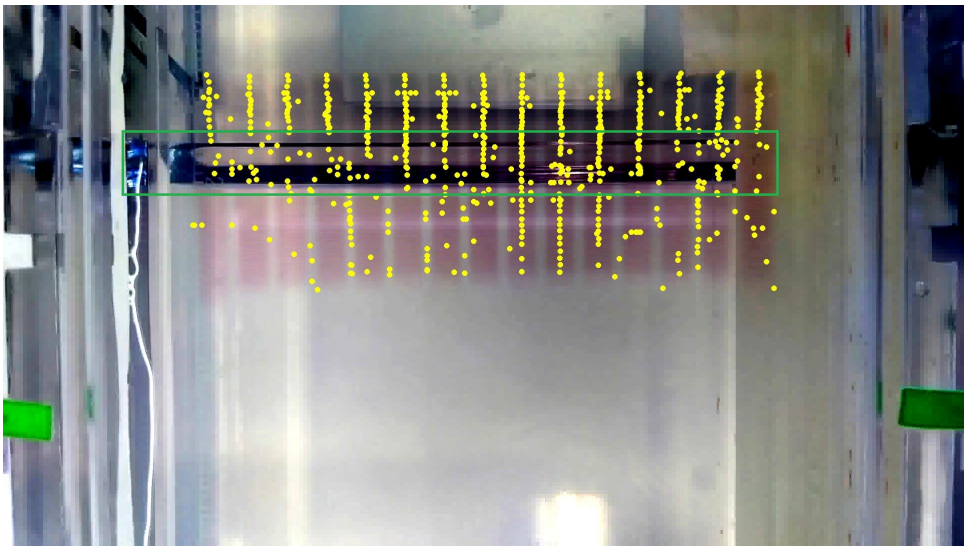

Figure 4.15: Lane results with only the centers drawn. these centers were the ones calculated, with no neighbouring being applied, in order to better show the problems of this approach. The green box presents the region of the box that interferes with the segmentation, resulting in band fusion, where several centers are located in between lanes, instead of being on top of them.

When it comes to the Simple U-Net mentioned previously, not only are these two problems present in the segmentation portion, but the model also tends to segment the

corners of each small square that it receives as input, as if being a band or a well. This makes it so that when the frame is reconstructed from these squares, it will have a huge amount of segmented dots (figure 4.16), which in turn will be considered as bands or wells by the algorithm and the centers will be stored and used to calculate the final lanes. Adding onto that the fact that the model shows a low mIoU, the segmentation results are significantly negatively affected (figure 4.17), by which point it's safe to assume the pre-trained U-Net to be the better model to consider in the final results.

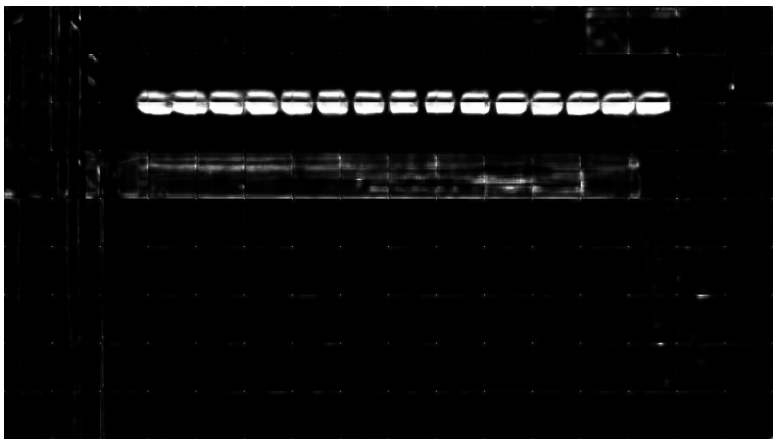

Figure 4.16: Reconstruction of a segmented frame after the original one was divided into small squares, with these being segmented by the Simple U-Net and used for the reconstruction.

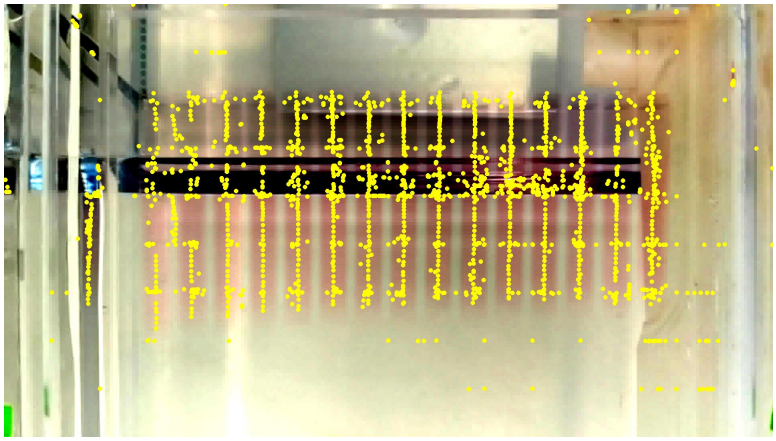

Figure 4.17: Lane results with only the centers drawn. These centers were the ones calculated, with no neighbouring being applied, in order to better show the problems of this approach. The low success rate in proper segmentation makes the algorithm identify several false centers and subsequently false lanes.

	1st approach		2nd approach	
Nº of lanes across all videos	466	100%	466	100%
Nº of identified lanes	412	88.4%	477	102.4%
Nº of correctly identified lanes	378	81.1%	466	100%
Nº of incorrectly identified lanes	34	7.3%	11	2.4%
Nº of fully correctly identified lanes	218	46.8%	304	65.4%
Time taken to analyse 1 video	120 min	-	12 min	-

Table 4.1: Comparison of the results of both approaches.

4.3 Results comparison

After analysing both approaches' results and their main flaws, it's time to put them against each other and draw the final conclusions. Since only one model of each approach is needed, the Simple U-Net will be cast aside, due to its results being sub-par to the pre-trained U-net.

The empirical results are shown in table 4.1. The description of every row is presented below:

1. Number of lanes across all videos: the sum of the number of lanes in every video. All videos had between 10 and 15 lanes;

2. Number of identified lanes: the sum of all lanes that were identified by the algorithm, whether they match with the real lanes or are false positives.

3. Number of correctly identified lanes: the sum of all lanes that were identified by the algorithm and match the position of a real lane, even if the start and/or end points do not match the ones from the real one;

4. Number of incorrectly identified lanes: the sum of all lanes that were identified by the algorithm, but do not match the position of a real lane at all, meaning, a false positive;

5. Number of fully correctly identified lanes: the sum of all lanes that were identified by the algorithm and fully match the position of a real lane, from its starting point to its end point.

6. Time taken to analyse 1 video: Amount of time, in minutes, that an approach takes to analyse 1 video, from the moment the model receives the input to the moment the algorithm draws the calculated centers/vertical lines ends.

As per the table, in the first approach, even though the number of lanes correctly identified is near 81%, close to half of these did not start and/or end at the correct points, which cannot be considered good results. There were also quite a few false positives, which decreases, even more, the usefulness of the algorithm. It's also important to note

that these results were already improved, as explained earlier, when the lanes with incorrect starting position were discarded. Even if, taking into account that this solution also eliminated some correct lanes, the original results were to be used instead of the improved ones, increasing the number of correctly identified lanes and maybe even fully correctly identified lanes, the number of false positives would skyrocket, leading to even poorer performance. In matters of time, this approach needs to chop the frames into small fragments and then classify them, which is quite time-consuming. The whole process, from adding all the frames into a final image, to the sweeping, cropping, classification and the selection and drawing of the vertical lines into the final image took around 2 hours for 1 video of 40 frames. Since every step excluding the cropping and classification is of low computational complexity, the time needed for these tasks is negligible, so it's safe to say that each frame takes around 3 minutes to be analysed.

As for the second approach, all the real lanes were identified to some extent, with 11 extra lanes being false positives. The number of fully correctly identified lanes is, however, still quite lacking, with only 65%. This ends up diminishing the value of the algorithm. When it comes to time, the two biggest tasks are the segmentation of the frames and the identification of the bands/wells and consequent calculation and grouping of neighbour centers. Both the addition of the images and the drawing of the lines are of low computational complexity, so these can be disregarded in terms of time. The segmentation of 40 frames (1 video) takes around 30 seconds, while the identification, calculation and grouping take around 11 minutes, making the total time of finding the lanes in a video around 12 minutes.

It's quite clear which of these methods is superior. Not only is the segmentation approach better in terms of results, whether it being a lower amount of errors or higher accuracy, but it's also 10 times faster. It's then safe to assume that the second approach is fully functional and the best option to the initial problem.

That being said, both approaches are still show a certain amount of errors and would benefit from post-processing of the results. It's worth pointing out that there were pre-processing processes that could have been applied to the frames before being used as input that would theoretically improve the results. One such process would be to crop out of each video everything that was outside of the box where the gel electrophoresis occurred. This would, at least, prevent any false lane from being identified outside of the gel, which would decrease the error rate. However, as the objective of both approaches was always to create an algorithm that would use the frames of any video regarding gel electrophoresis as input, without having to do any manual, one-by-one, pre-processing, these processes were not practically applied.

Chapter 5

Conclusion

The main goal of this work was to find a way to automatically identify the migration paths of samples used in the gel electrophoresis procedure. This was only possible, due to the existence of gold nanoparticles within the samples placed on the gel, which made them visible to the naked eye.

Two approaches were taken in order to obtain the best prototype possible, one using a binary classifier to sweep each frame and classify small sections as having or not the samples and the other using a segmentation model to distinguish the regions of interest from the background.

To train and validate these machine learning models, datasets had to be created. Forming these manually was not viable, and so, several techniques had to be used to facilitate the creation process. Since these do not possess higher enough accuracy to obtain the best possible data, their results were manually analysed and filtered. On the first approach, frames were fragmented, with these being put through dimensionality reduction techniques, so as to easily find agglomerates of examples of the different categories. The one for the second approach was a produce of segmentation techniques applied onto the frames to create the examples needed.

Both approaches were able to reach the goal, with varying levels of success. In order to decrease the error rate on the first approach, post-processing of the results was done. Although most of the false positives were discarded, no coherent way of increasing the success rate in the classification of true positives was found. The improved results still fell short when compared to the second approach.

The segmentation algorithm developed was superior in both results and speed of execution, making it the clear best option to solving the initial problem.

There were still a considerable amount of errors in finding every sample across time. This, however, should not be viewed as a failure since, with both the starting, ending and

some of the middle positions of each sample, it's already possible to calculate the pathing of the migration. This approach is indeed functional and capable of completing the initial problem.

The end result was a fully functional prototype that uses a machine learning model capable of image segmentation, in order to segment the frames of each video, making it possible to identify wells and bands and use their locations across time to calculate the migration paths of the samples.

The correct identification of the lanes of the samples used in the gel electrophoresis can now be used alongside their color intensity profiles, to measure the migration of these samples across time. All of this data might possibly be used in the future to fit several models, with the goal of modeling the migration of samples during the gel electrophoresis procedure.

Bibliography

[1] J. Jeppson, C. Laurell, and B. Franzén. "Agarose gel electrophoresis." In: *Clinical Chemistry* 25.4 (1979), pp. 629–638.

[2] J. A. Meyers, D. Sanchez, L. P. Elwell, and S. Falkow. "Simple agarose gel electrophoretic method for the identification and characterization of plasmid deoxyribonucleic acid." In: *Journal of bacteriology* 127.3 (1976), pp. 1529–1537.

[3] D. Voytas. "Agarose gel electrophoresis." In: *Current protocols in molecular biology* 51.1 (2000), pp. 2–5.

[4] B. Johansson. "Agarose gel electrophoresis." In: *Scandinavian Journal of Clinical and Laboratory Investigation* 29.sup124 (1972), pp. 7–19.

[5] P. Y. Lee, J. Costumbrado, C.-Y. Hsu, and Y. H. Kim. "Agarose gel electrophoresis for the separation of DNA fragments." In: *JoVE (Journal of Visualized Experiments)* 62 (2012), e3923.

[6] M. Ünlü, M. E. Morgan, and J. S. Minden. "Difference gel electrophoresis. A single gel method for detecting changes in protein extracts." In: *Electrophoresis* 18.11 (1997), pp. 2071–2077.

[7] R. Westermeier. "Gel electrophoresis." In: *e LS* (2001).

[8] D. J. Peitinho, M. P. de Almeida, E. Pereira, L. Krippahl, and R. Franco. "Interaction between gold nanoparticles and blood proteins to define disease states." In: *Annals of Medicine* 51.sup1 (2019), pp. 37–37.

[9] Y. Baştanlar and M. Özuysal. "Introduction to machine learning." In: *miRNomics: MicroRNA Biology and Computational Analysis*. Springer, 2014, pp. 105–128.

[10] J. A. Cruz and D. S. Wishart. "Applications of machine learning in cancer prediction and prognosis." In: *Cancer informatics* 2 (2006), p. 117693510600200030.

[11] L. P. Kaelbling, M. L. Littman, and A. W. Moore. "Reinforcement learning: A survey." In: *Journal of artificial intelligence research* 4 (1996), pp. 237–285.

[12] S. J. Prince. *Computer vision: models, learning, and inference*. Cambridge University Press, 2012.

[13] A. Hanson. *Computer vision systems*. Elsevier, 1978.

[14] A. Rosenfeld. "Computer vision: basic principles." In: *Proceedings of the IEEE* 76.8 (1988), pp. 863–868.

[15] W. Chong, D. Blei, and F.-F. Li. "Simultaneous image classification and annotation." In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, pp. 1903–1910.

[16] D. Lu and Q. Weng. "A survey of image classification methods and techniques for improving classification performance." In: *International Journal of Remote Sensing* 28.5 (2007), pp. 823–870. DOI: 10.1080/01431160600746456.

[17] M. Guillaumin, J. Verbeek, and C. Schmid. "Multimodal semi-supervised learning for image classification." In: *2010 IEEE Computer society conference on computer vision and pattern recognition*. IEEE. 2010, pp. 902–909.

[18] S. Kamdi and R. K. Krishna. "Image Segmentation and Region Growing Algorithm." In: 2012.

[19] J. B. Heaton, N. G. Polson, and J. H. Witte. "Deep learning for finance: deep portfolios." In: *Applied Stochastic Models in Business and Industry* 33.1 (2017), pp. 3–12.

[20] L. Zhang, S. Wang, and B. Liu. "Deep learning for sentiment analysis: A survey." In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.4 (2018), e1253.

[21] D. Ravì, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, and G.-Z. Yang. "Deep learning for health informatics." In: *IEEE journal of biomedical and health informatics* 21.1 (2016), pp. 4–21.

[22] M. Lai. "Deep learning for medical image segmentation." In: *arXiv preprint arXiv:1505.02000* (2015).

[23] G. B. Goh, N. O. Hodas, and A. Vishnu. "Deep learning for computational chemistry." In: *Journal of computational chemistry* 38.16 (2017), pp. 1291–1307.

[24] E. Bettens, P. Scheunders, J. Sijbers, D. Van Dyck, and L Moens. "Automatic segmentation and modelling of two-dimensional electrophoresis gels." In: *Proceedings of 3rd IEEE International Conference on Image Processing*. Vol. 2. IEEE. 1996, pp. 665–668.

[25] A. M. Machado, M. F. Campos, A. M. Siqueira, and O. S. De Carvalho. "An iterative algorithm for segmenting lanes in gel electrophoresis images." In: *Proceedings X Brazilian Symposium on Computer Graphics and Image Processing*. IEEE. 1997, pp. 140–146.

[26] R. S. Sengar, A. K. Upadhyay, M. Singh, and V. M. Gadre. "Segmentation of two dimensional electrophoresis gel image using the wavelet transform and the watershed transform." In: *2012 National Conference on Communications (NCC)*. IEEE. 2012, pp. 1–5.

[27] E. Kostopoulou, E. Zacharia, and D. Maroulis. "An effective approach for detection and segmentation of protein spots on 2-d gel images." In: *IEEE Journal of Biomedical and Health Informatics* 18.1 (2014), pp. 67–76.

[28] Guo Dong and Ming Xie. "Color clustering and learning for image segmentation based on neural networks." In: *IEEE Transactions on Neural Networks* 16.4 (2005), pp. 925–936.

[29] M. Berth, F. M. Moser, M. Kolbe, and J. Bernhardt. "The state of the art in the analysis of two-dimensional gel electrophoresis images." In: *Applied microbiology and biotechnology* 76.6 (2007), pp. 1223–1243.

[30] J. Shlens. "A tutorial on principal component analysis." In: *arXiv preprint arXiv:1404.1100* (2014).

[31] A. Maćkiewicz and W. Ratajczak. "Principal components analysis (PCA)." In: *Computers & Geosciences* 19.3 (1993), pp. 303–342.

[32] L. McInnes, J. Healy, and J. Melville. "Umap: Uniform manifold approximation and projection for dimension reduction." In: *arXiv preprint arXiv:1802.03426* (2018).

[33] S. Wold, K. Esbensen, and P. Geladi. "Principal component analysis." In: *Chemometrics and intelligent laboratory systems* 2.1-3 (1987), pp. 37–52.

[34] L. v. d. Maaten and G. Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[35] P. Baldi. "Autoencoders, unsupervised learning, and deep architectures." In: *Proceedings of ICML workshop on unsupervised and transfer learning*. 2012, pp. 37–49.

[36] G. E. Hinton and R. R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." In: *science* 313.5786 (2006), pp. 504–507.

[37] M. Chen, X. Shi, Y. Zhang, D. Wu, and M. Guizani. "Deep features learning for medical image analysis with convolutional autoencoder neural network." In: *IEEE Transactions on Big Data* (2017).

[38] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez. "A survey on deep learning in medical image analysis." In: *Medical image analysis* 42 (2017), pp. 60–88.

[39] P. Sermanet and Y. LeCun. "Traffic sign recognition with multi-scale convolutional networks." In: *The 2011 International Joint Conference on Neural Networks*. IEEE. 2011, pp. 2809–2813.

[40] N. Otsu. "A threshold selection method from gray-level histograms." In: *IEEE transactions on systems, man, and cybernetics* 9.1 (1979), pp. 62–66.

[41] J. Zhang and J. Hu. "Image segmentation based on 2D Otsu method with histogram analysis." In: *2008 International Conference on Computer Science and Software Engineering*. Vol. 6. IEEE. 2008, pp. 105–108.

[42] M. M. Noor, Z Hussain, K. Ahmad, and A. Ainihayati. "Gel electrophoresis image segmentation with otsu method based on particle swarm optimization." In: *2011 IEEE 7th International Colloquium on Signal Processing and its Applications*. IEEE. 2011, pp. 426–429.

[43] A. M. McIvor. "Background subtraction techniques." In: *Proc. of Image and Vision Computing* 4 (2000), pp. 3099–3104.

[44] P. Spagnolo, M Leo, A. Distante, et al. "Moving object segmentation by background subtraction and temporal analysis." In: *Image and Vision Computing* 24.5 (2006), pp. 411–423.

[45] E. Shelhamer, J. Long, and T. Darrell. *Fully Convolutional Networks for Semantic Segmentation*. 2016. arXiv: 1605.06211 [cs.CV].

[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. *Going Deeper with Convolutions*. 2014. arXiv: 1409.4842 [cs.CV].

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[48] K. Simonyan and A. Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. arXiv: 1409.1556 [cs.CV].

[49] O. Ronneberger, P. Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

[50] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*. 2016. arXiv: 1606.00915 [cs.CV].

[51] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. *Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation*. 2018. arXiv: 1802.02611 [cs.CV].

[52] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. *Rethinking Atrous Convolution for Semantic Image Segmentation*. 2017. arXiv: 1706.05587 [cs.CV].

[53] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. "Deformable Convolutional Networks." In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017.

[54] S. Tomar. "Converting video formats with FFmpeg." In: *Linux Journal* 2006.146 (2006), p. 10.

[55] G. Bradski. "The OpenCV Library." In: *Dr. Dobb's Journal of Software Tools* (2000).

[56] zhixuhao. *Implementation of deep learning framework – Unet, using Keras*. Oct. 2017. URL: https://github.com/zhixuhao/unet.

[57] D. Gupta. *Image Segmentation Keras : Implementation of Segnet, FCN, UNet, PSPNet and other models in Keras*. Mar. 2021. URL: https://github.com/divamgupta/image-segmentation-keras.