Generalized automatic classification of cancer cell migratory patterns

Johannes Kumra Ahnlide

2021

Master's thesis in
Biomedical Engineering

Supervisor: Christian Antfolk

LUND
UNIVERSITY

Faculty of Engineering LTH
Department of Biomedical Engineering

Abstract

Automated fluorescence microscopy is an emerging technique that enables researchers to generate large amounts of data which in turn allows for large scale quantitative studies. In previous small scale quantitative studies, migration modes found by qualitatively observing migrating lung cancer cells have been described. Since such a method starts from qualitative observations it might not generalize to other modes of migration or cells. Furthermore it is not obvious that this approach captures the full complexity of the system.

In order to perform a fully quantitative study of cancer cell migration modes large data sets are needed which requires a method that allows us to detect cells of interest in low magnification. This way, larger samples can be scanned in an achievable time frame and high magnification data collected only on relevant cells. The aim of this thesis project was therefore to create a pipeline that extracts features from large data sets of low magnification live fluorescence imaging data and uses machine learning to cluster the cells into subpopulations that could be of interest for further study. Additionally, it would be of interest to see if such a method could recover the migration modes found in the previously mentioned small scale quantitative studies.

To run the pipeline a server and a database for storing metadata and analysis results were set up. I developed scripts for automated analyses such as cell tracking and extraction of morphology data. For the purpose of interaction with the data, a web server serving both a graphical and a programmatic interface was created.

The created infrastructure was successfully used for post-acquisition analysis. The clustering methods that I have evaluated tend to cluster dead cells. Further analysis of the other clusters and more data acquired by real-time classification is required for a generalized automatic method.

1 Introduction

Cell migration is a key biological process essential to the development and maintenance of the body as well as the spread of cancer [1]. An example of cell migration is the invasion of white blood cells into wounded tissue [2, 3].

Cancer invasion and metastasis are enabled by cell migration and the migration characteristics of the cells can determine how a tumour develops [4]. One way in which some cancer cells are known to migrate is called mesenchymal migration. In this mode of migration cells are spindle-shaped and move by extending in the direction of movement, attaching to the substrate, contracting and finally detaching the attachments at the back of the cell [2]. The mesenchymal migration mode is dependent on enzymes that break down the extracellular matrix (ECM), a network of macromolecules that is essential for biological tissue structure. Due to the dependence of the mesenchymal migration mode on enzymes, inhibitors of these have been developed as cancer therapeutics. So far clinical trials testing these inhibitors have been unable to show that they suppress metastasis. One possible explanation is that the cells switch to another mode of migration in the absence of these enzymes [5].

Certain composites of the ECM, such as fibronectin, provide physical leverage for migrating cells. Fibronectin has been shown to influence migration of cells by altering the strength with which the cells adhere and thereby affecting migration speed.

The migratory patterns of lung cancer cells seem to vary between cells [6]. Because of the role cell migration has in tumour invasion, a better understanding of how cells migrate may in the future help in pharmaceutical development. There exist various ways in which cell migration can be studied and straightforward methods involving optical techniques, such as fluorescence microscopy, are common. [7]. This technique uses fluorescence, the property of certain molecules to absorb electromagnetic radiation at a certain wavelength and emit it at another. By filtering out all but the emission wavelength a high contrast image of the molecules with this fluorescent property, called fluorophores, is acquired [8]. Various techniques can be used to label other molecules or structures inside a cell with fluorophores so that these can be visualised.

Most of the analyses performed in the biomedical field using fluorescence imaging are either highly specific or qualitative. Emerging

techniques, including automatic image acquisition, give researchers the ability to generate large amounts of data. The generated data sets allow for larger scale quantitative studies when combined with automated analysis [9].

While techniques for automatically digitizing certain single-cell parameters of spatiotemporal cell processes have existed for over 30 years [10], recent developments enable the automatic acquisition and statistical analysis of multiple single-cell parameters. Since multiple parameters can be integrated for each cell in the same study, a more holistic approach can be taken that can both describe correlations between these parameters and find subpopulations that explain cell to cell heterogeneity. This approach is called systems microscopy [9]. Previous small scale quantitative studies have started from a qualitative assessment of which migration modes a kind of cell exhibits and manually acquired images of these modes [6]. However, it is not obvious that the way a researcher chooses to divide the migration modes captures the full complexity of the system. There might be additional modes of migration, subgroups within each mode or completely different ways of dividing up the cells that better explain the observed heterogeneity. Studies starting from qualitative observations might not apply to other modes of migration and might not generalize to other cells.

Additionally, while the techniques of systems microscopy can be employed on relatively small samples, the focus on heterogeneity highlights the benefits of having a high-throughput system. When more data is available, what looked like an outlier might be discovered to be a distinct cell state and what looked like noise in one continuously varying population might be discovered to be distinct subpopulations.

One study describes two migration modes of migrating lung cancer cells [6]. Through high magnification fluorescence imaging and image analysis they thereafter extract quantitative parameters for each cell and describe how these parameters vary between the modes.

To be able to further study these and other phenotypes more data is required. In this study we want to acquire such data using a method that can be generalised. That is, using a fully automated approach that does not presuppose migration modes so that it is readily applicable to other cells and modes. Furthermore, the imaging pipeline should be high-throughput so that the heterogeneity of the cell population can be fully captured.

The aim of this thesis project was thus to create a pipeline that ex-

tracts features from low magnification live fluorescence imaging data and uses machine learning to cluster the cells into sub-populations that could be of interest for further study and see if such a method could recover the groups described in previous studies.

2 Background

This section describes the theoretical concepts employed in the analysis.

2.1 Cell tracking

Cellular movement can be studied in various ways, ranging from the study of the molecular processes that enable each cell to generate force to the stresses in a collection of cells closing a wound [11, 12].

A straightforward way to study migration is by determining the rate and direction of movement of the individual cells. This consists of two steps: finding the positions of the cells in each frame and linking the positions of the same cell together between frames. In this project, images of cells are acquired in time series at each predetermined field of view that can contain an arbitrary number of cells. A fluorescent dye labelling the cell nucleus has been used for tracking since it is easy to get a good, clearly localised signal from such labels. Besides, the inertia of the nucleus results in a stable position whereas the edges of the cell are constantly fluctuating. To obtain the locations of cells from this signal the first step is typically segmentation, i.e. finding a binary mask of the cell nuclei. The segmentation can be done in various ways, e.g. by selecting all pixels above a threshold. The next step is determining which of the masks correspond to cell nuclei and what their center is. Once again multiple methods exist. One early study used chain code [10], an algorithm that can encode connected components in an image by representing the boundary [13]. In this study the Laplacian of Gaussian is used to detect blobs in the binary image [14].

The positions of each cell in consecutive frames are linked together by optimizing the assignment of points in one frame to the next. In general, the problem of finding globally optimal cell tracks over the entire time is an NP-hard problem [15]. In effect, this means that the problem is unlikely to be solved in a reasonable computational time. Various techniques exist that find an approximate solution, often by

using a greedy approach, i.e. breaking the problem up into several steps and finding a locally optimal solution at each step [16, 17].

This study uses an algorithm that minimizes the sum of the distance of all points to their respective points in the next frame, subject to certain constraints, by using a solver for the linear assignment problem [16]. The linear assignment problem is a combinatorial optimization problem in which the objective is to find a mapping between two given sets together with a weight function W by minimising a cost function C. In this case the two sets T_i and T_{i+1} are positions of cells in two adjacent time frames and the weight function is the Euclidean distance.

Tracks can appear or disappear at any point in a time series due to various practical issues. Cells can move in and out of the image field of view, they can divide through mitosis and they can be undetectable at certain time frames due to a low signal. To represent the potential start or end of a track, T_i not only contains the existing cell positions, but also virtual start elements for each existing cell position in T_{i+1}. In the same manner, T_{i+1} contains virtual end elements for each existing position in T_i in addition to the existing cell positions of T_{i+1}.

2.2 Tracking features

Once the cell tracks have been identified, various features can be extracted from these that describe their overall movement patterns. Figure 1 shows how a track can be represented as a series of displacement vectors $\mathbf{v_i}$. The rate ($|\mathbf{v_i}|$) is a fundamental cell tracking feature that can give enough information to differentiate between modes of migration. However, many other features are often used to provide a more complete description of the cellular movement. Common track metrics are described below.

The expression for the directionality ratio DR is

$$\mathrm{DR} = \frac{d_{\mathrm{net}}}{d_{\mathrm{tot}}}$$

where d_{net} is the net distance, i.e. the distance between the first and final point in a track and d_{tot} is the sum of the rates. The directional autocorrelation DA and velocity correlation VA is

$$\mathrm{DA} = \frac{\mathbf{v_i} \cdot \mathbf{v_{i+1}}}{|\mathbf{v_i}|\,|\mathbf{v_{i+1}}|} \text{ and } \mathrm{VA} = \mathbf{v_i} \cdot \mathbf{v_{i+1}}$$

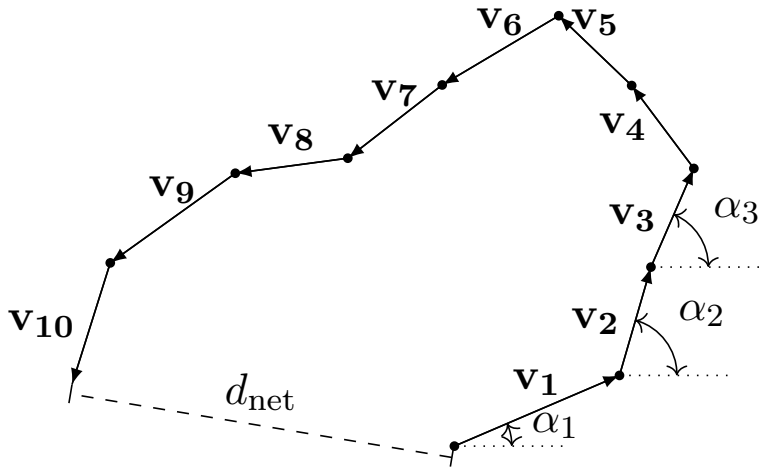

Figure 1: **Example track showing the basic metrics used in the track features.** The vectors $\mathbf{v_i}$ are the displacement vectors of the cells position from frame i to $i+1$. The net distance (d_{net}) is the the distance between the cells initial and final position. The angles α_i is the direction in a global reference frame.

respectively. These three provide insight into the cell's directional persistence, i.e. the cell's tendency to move in the same direction, in slightly different ways. Further features are calculated from these by calculating statistical descriptors, e.g. the arithmetic mean or standard deviation of the rate, over a defined period of time. Several of these kinds of features are used in this project.

2.3 Dimensionality reduction

A large number of tracking features can be extracted from image time series data. Techniques to condense this information and evaluate the importance of these features for a given data set are described below.

Feature projection aims to find a transform, to a smaller set of variables, that captures and removes redundancies between the original features [18]. One example of a feature projection is principal component analysis (PCA). The PCA model assumes that the observed features result from an orthogonal linear transform $\mathbf{W}$ of underlying variables that are assumed to have a Gaussian distribution. The PCA transform then selects $\mathbf{W}$ so that the retained variance is maximized under the projection. The ability to project the features down to a

2-dimensional space is essential for visualization and PCA is used extensively for this purpose. It can also be used as the first step of a classification or clustering pipeline to make it more computationally efficient and prevent it from fitting to noise.

In feature selection, the number of dimensions is reduced by simply excluding features. Features are excluded based on a relevancy metric and the techniques used are often supervised, i.e. they measure the relevance of a feature in relation to a specified output that the feature should be used to predict [18]. However, the intrinsic information redundancy between features can be taken into account when performing the selection. Additionally, feature selection algorithms exist that are used for unsupervised tasks [19], i.e. tasks where there is no predefined output. Feature selection is performed to remove superfluous features so that the models can be trained more efficiently and generalize better [20]. Finding the minimal set of features to describe a population can also immediately provide insight to researchers by indicating the properties that distinguish observations from each other.

2.4 Decision trees and random forests

Decision tree learning is an approach to machine learning that represents the acquired knowledge in a tree, creating a flow chart-like structure. For a classification task the leaves in the decision trees are the target classes, each internal node of the tree represents a test that branches to different subtrees depending on the value of one feature. This simple and explicit representation of the model makes decision trees easy to understand and allow them to be used for feature selection in addition to prediction [21]. To achieve better generalization on new data randomness can be applied to create many different trees and combine their predictions. A random forest is such a collection of trees where each tree is grown independently of the others. In random forest classifiers, as originally conceived, each tree casts a unit vote for one class and the mode is used to determine the prediction [22]. Here the implementation from Scikit-learn is used which instead uses the average of the probabilistic predictions of each tree to determine the predicted class [23].

2.5 Deep learning models

Certain aspects of my cell migration analyses involve deep learning models with various specialised network architectures. An artificial neural network consists of nodes that perform a non-linear operation on weighted inputs from connected nodes [24, ch. 6]. The nodes in a neural network are typically organized into layers and machine learning involving models with multiple layers is called deep learning. One kind of neural network, that has been shown to perform well on data with a grid or array-like organization, is the convolutional neural network (CNN). It applies the mathematical operation convolution on each layer. By using this operation it finds features that are invariant to translation and simplifies the interdependence of nodes in a way that is very natural for images [24, ch. 9]. CNNs were used in this project in order to extract information from the raw image data. The architectures these models were based on are described below.

2.5.1 Autoencoder

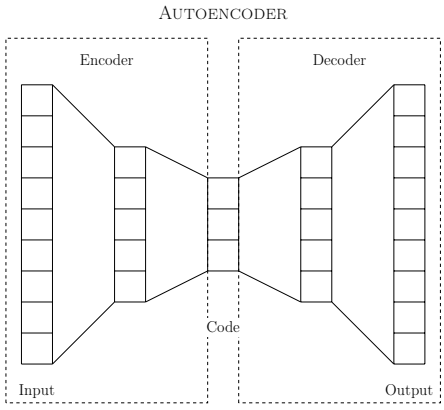

Figure 2: **Schematic of an autoencoder neural network architecture.** An autoencoder is trained to reproduce its input as its output. The schematic shows how the data can be encoded in a code with a lower dimensionality than the input. The encoder part of this architecture can then be used on new data e.g. for dimensionality reduction.

An autoencoder is a neural network that is trained to reproduce its input as its output with limitations in how it can represent the

data along the way [24, ch. 1]. This limitation forces the autoencoder to find structure in the data. The internal representation is called the code and the part of the network that precedes the code is the encoder. The encoder learns to create an efficient code subject to certain constraints, e.g. the dimension of the code is lower than the dimension of the input, and the part of the network following the code, the decoder, learns to recreate the input from the code. The encoder can be used as a non-linear dimensionality reduction for visualization purposes or to generate representation of features for further analysis.

2.5.2 Xception

Xception is a convolutional neural network architecture that has been shown to perform well on image classification tasks [25]. Xception uses depthwise separable convolutions, which is the factorization of a convolution into first a spatial convolution for each channel followed by a pointwise convolution that integrates the information from all channels. Xception also uses residual connections; blocks of layers wherein the input of one block is added to its output before being fed to the next block.

2.5.3 U-Net

U-Net is an architecture that consists of multiple convolutions and max poolings that contract an input image, followed by a series of upsamplings and convolutions to bring the dimension of the image back to that of the original image. It has been used primarily for segmentation of both microscopic and macroscopic objects such as cells and organs [26, 27].

3 Methods and Results

3.1 Infrastructure

Efficient and automated running of the pipeline requires a server. The infrastructure that was created for this project is outlined in figure 3. The server fetches image files from the image store and handles these images as well as experimental metadata submitted by the uploader. When a new experiment is fetched to the server a custom cell tracking algorithm (Supplement S1) is run automatically and the result is stored in the database for further analysis. The server was setup to

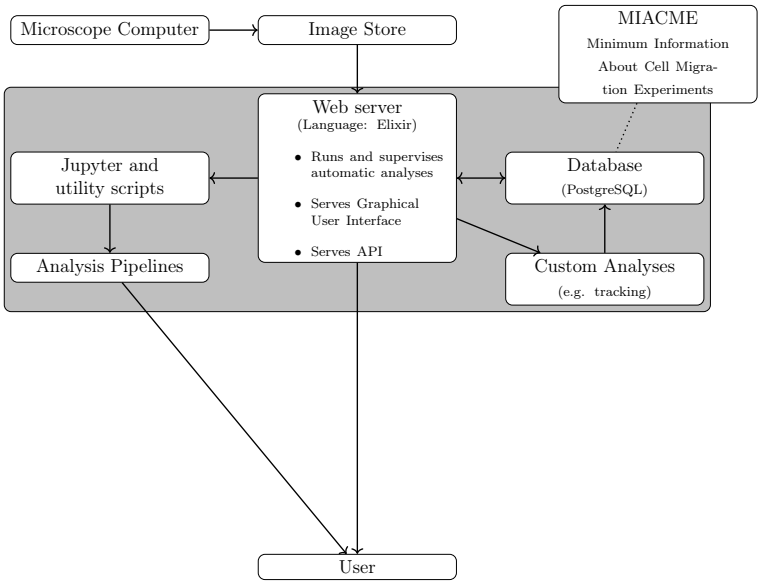

Figure 3: **Schematic of the developed server pipeline.** The grey area shows the physical server and arrows indicate the flow of data. Users upload images from the microscope computer to a central image repository. They can then interact with the graphical user interface of the web server to submit experimental metadata, run analyses, and view images or results. Metadata is stored in a PostgreSQL database and the schema is created to comply with the Minimum Information About Cell Migration Experiments reporting guideline [28].

run JupyterLab [29], a user interface for working with Jupyter notebooks, a document format that can record a computational workflow together with both its results and its description [30]. From the Jupyter notebooks the data on the server is accessed through utility scripts calling the web server API. This enables both manual and semi-automated processing of the automatically generated data. Additionally a web based graphical user interface, one part of which is shown in figure 4, was created to allow viewing of the images and extracted data.

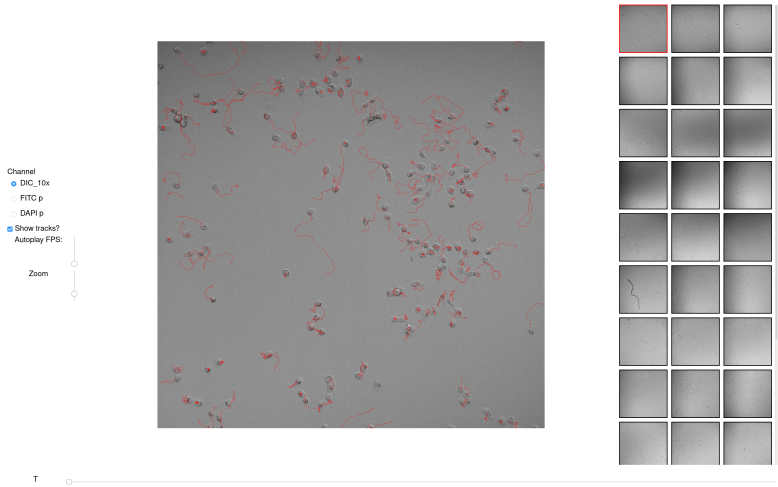

Figure 4: **The image viewing page of the graphical user interface created for the web server.** This allows users to view the different fields of views and channels acquired in an experiment.

3.2 Method

An outline of the method that has been used in this project is shown schematically in figure 5. Single-cell features were extracted from the microscopy data, initially in the form of tracks. Machine learning algorithms were applied to these features in order to find subpopulations of cells that have similar migration patterns. The found subpopulations were thereupon used for training a model that could identify each subpopulation using only a subset of the raw microscopy data, such as the first few time frames. The idea is to use the features extracted from this model to automatically find cells corresponding to a specific subpopulation in the microscope. The data used and the resulting models developed along the way are shown in figure 6.

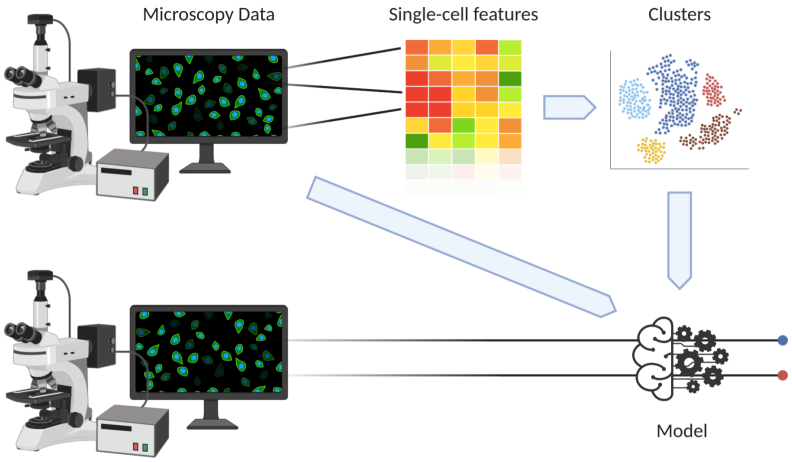

Figure 5: **Overview of method pipeline.** The schematic depicts the structure of the working process developed for this study. The goal was to create a classification model that can be used to automatically find cells corresponding to a specific subpopulation in the microscope. To begin with, single-cell features were extracted from microscopy data. Unsupervised machine learning algorithms were applied to these features in order to cluster the data. The clusters should correspond to subpopulations of cells that have similar migratory patterns. The clusters were then used as labels to train a model that could identify each subpopulation using only a subset of the raw microscopy data.

Some extracted tracks and the images of the corresponding cells are shown in figure 7. The cells used are from a human non-small cell lung carcinoma cell line (H1299). Fluorescent dyes are used to label

the cell nucleus and the intracellular space, i.e. the cytoplasm. Images were also acquired using Differential Interference Contrast (DIC), a microscopy technique that uses interference to create an image that relates to the spatial gradient of the optical path length through the sample along a specific direction. A microscopy slide divided into 8 chambers (wells) was used. The wells were coated with varying concentrations of fibronectin. The concentrations used were 0.1 µg/ml, 1.0 µg/ml, 10 µg/ml, and 50 µg/ml with two replicates for each concentration. The concentration of fibronectin was varied since this has been shown to affect migration and might shift the distribution of migratory phenotypes. Each well was divided into 36 fields of view imaged at 10x magnification. An image of each field of view was acquired every 10 minutes for just short of 10 hours with a total of 60 frames acquired. More details about image acquisition can be found in figures S1 and S2.

3.3 Clustering

Since the goal is to find migratory patterns the tracks were first studied in isolation from any other data from the images. From the tracks, 42 features were calculated using the mean values of the features described in section 2.2 with directionality ratio being calculated for step lengths from 2 to 39. The features were standardized so that each feature had mean zero and unit variance and PCA was performed. The distribution of the first two principal components for each well is shown in figure 8. This figure shows an obvious cluster of cells present in most wells around PC1 ≈ -5 and PC2 ≈ 0. Moreover, this figure also highlights differences between the replicates particularly in the first and the third column. In addition to this cluster to the left in the plots, many wells seem to have a cluster to the far right while the first well in the top row and the second well in the bottom row have a more even spread along the first principal component. It was of interest to find these clusters in the original data and explore whether the center region made sense as a cluster separate from the ones on the extremes of PC1. Hierarchical agglomerative clustering with the number of clusters set to 3 was thus performed.

Agglomerative clustering methods initially consider each singleton set, containing one data point, to be a cluster. The method iteratively replaces two sets in the partition with their union according to some criterion. This procedure is carried out until the grouping has the

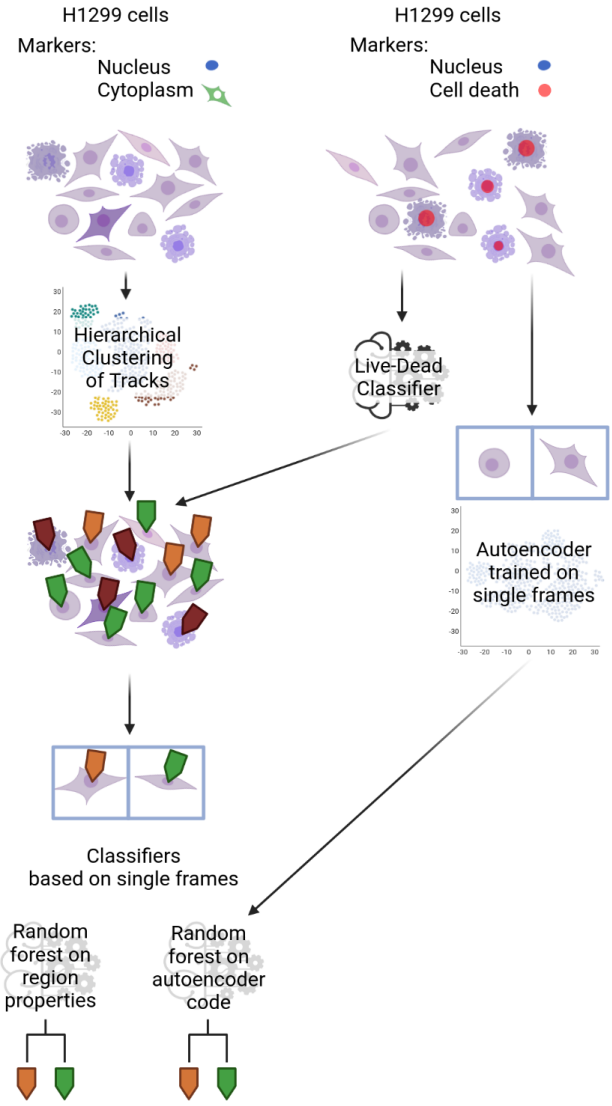

Figure 6: **Overview of models.** The schematic depicts the sources of data, how they were used to develop models, and transformed by these models. Two different experiments with different fluorescent markers were used: the main experiment, analyzed to find and characterize migratory behavior, and an auxilliary experiment used to train two neural networks. The auxilliary models feed into the main analysis at various points as indicated by the arrows.

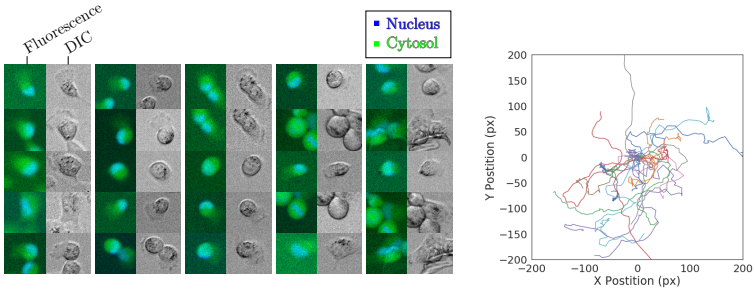

Figure 7: **A sample of the cells studied and their corresponding tracks.** To the right in the figure the tracked positions of 25 cells are shown. For each such track an 80px × 80px region around the first coordinate of the track is shown in the input fluorescence and DIC images to the left.

desired number of clusters. Ward's minimum variance method is one agglomerative clustering method and its criterion for joining two clusters is that the choice should minimise the total of the error sum of squares within each group. This method was chosen because it has empirically been shown to often be a suitable clustering method and tends to create clusters of similar size.

The result of clustering using Ward's method with the number of clusters set to 3 is shown in figure 9. Tracks sampled from each cluster are also shown. When studying these tracks it seems like Cluster 0 contains cells that move quickly in a fairly straight path. The cells in Cluster 1 move but seem slower and prone to get stuck around a point. In Cluster 2 the cells seem entirely static.

A possible explanation for the immobility of the cells in Cluster 2 could be that this cluster consists predominantly of dead cells. To examine this another experiment was performed with a fluorescent staining that indicates cell death. Using the data from this experiment a convolutional neural network based on the Xception architecture was trained to predict if an H1299 cell is dead. The Xception architecture was created for the ImageNet dataset with 299×299 RGB images while a more reasonable size for an image of a cell in our case was 80×80 with 2 channels. Furthermore, the ImageNet dataset contains 1000 classes and the current task only requires picking one of two classes. The entry and exit flow of the architecture was modified accordingly while the middle flow was the same as the original Xcep-

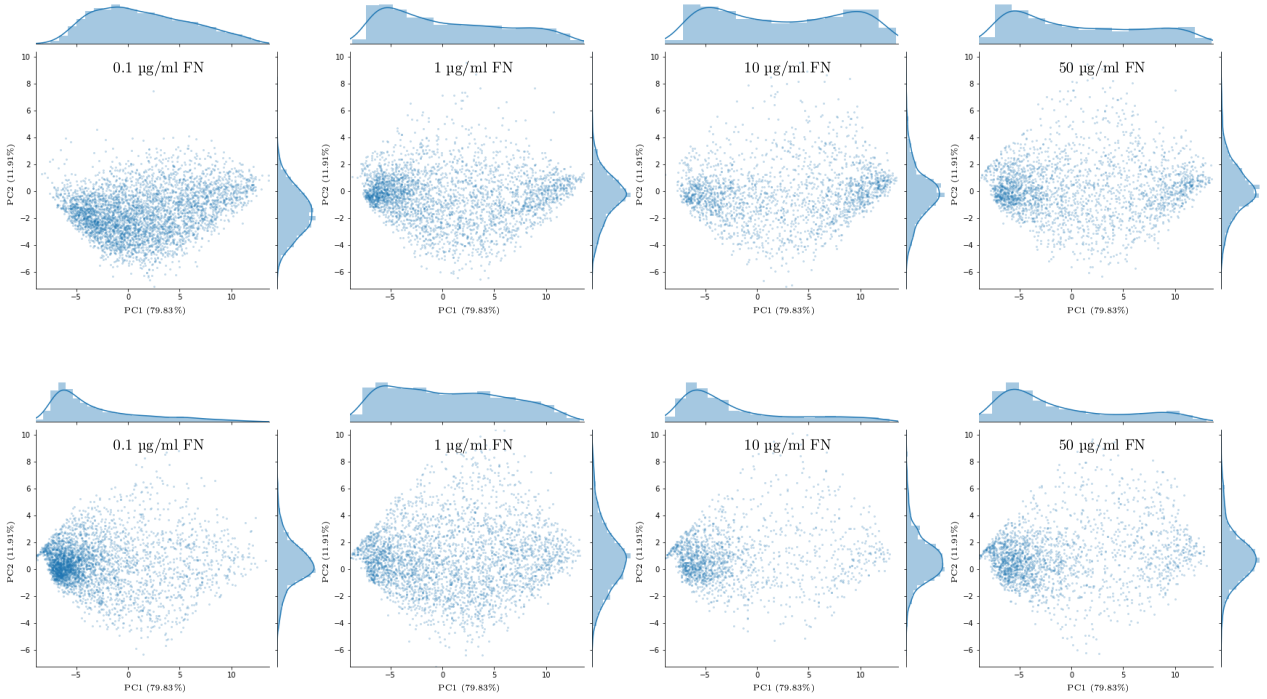

Figure 8: **Principal component analysis (PCA) of standardized track features.** The first two principal components are plotted for each well. The plots are arranged so that each column contains two replicates with the same fibronectin concentration and the fibronectin concentration increases from left to right. A cluster around PC1 ≈ -5 and PC2 ≈ 0 is apparent in most wells. There exists a visible variance between the replicates.

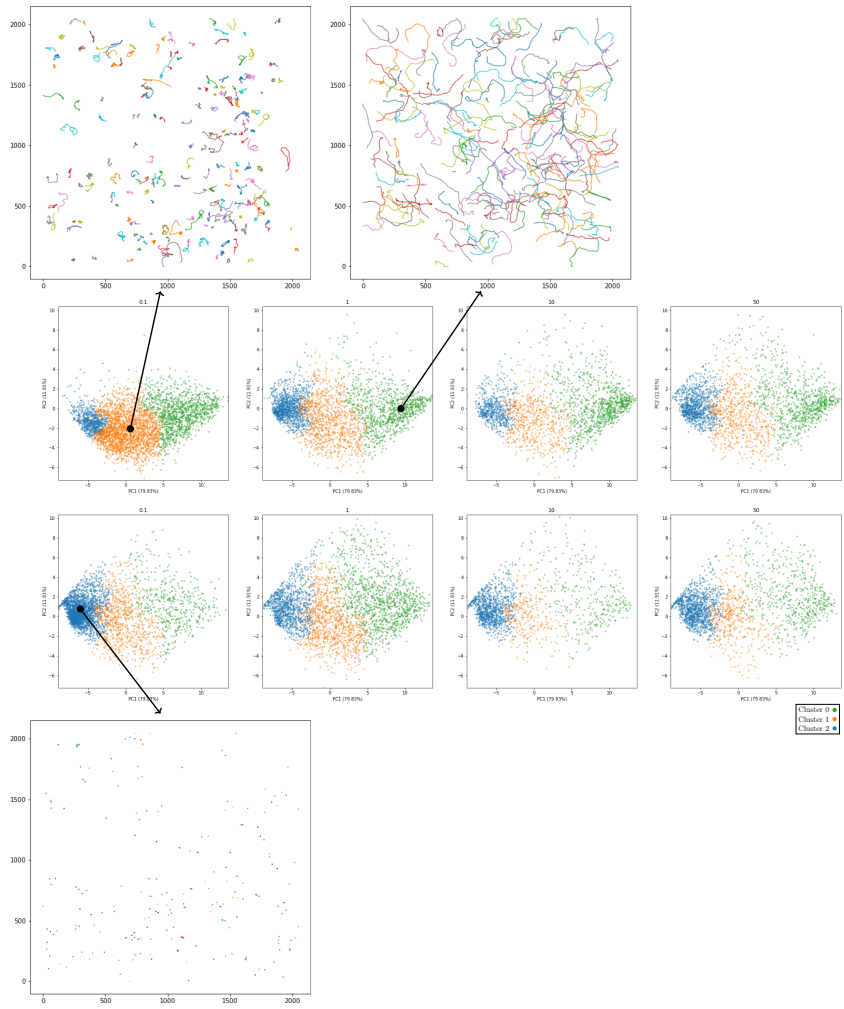

Figure 9: **Agglomerative clustering on track features**. The central figure shows PCA on track features for each well with the marker color indicating the cluster as found by agglomerative clustering using Ward's method with number of clusters set to three. Samples of tracks belonging to each of the three clusters are shown as indicated by the arrows. The tracks of Cluster 2 (blue) seem entirely static. A possible explanation for this could be that this cluster consists predominantly of dead cells.

tion. An evaluation of the performance of this neural network can be found in figure S3. The result of applying this network to the clusters in figure 9 is shown in figure 10. These results demonstrate that the cells the network predicted to be dead mostly belong to Cluster 2. They can however be found in the other clusters. The cells that the network predicted to be alive are almost evenly distributed among the clusters but slightly less common in Cluster 2. The tracks of the cells in Cluster 2 that were predicted to be alive are not static and qualitatively different from the examples shown in figure 9.

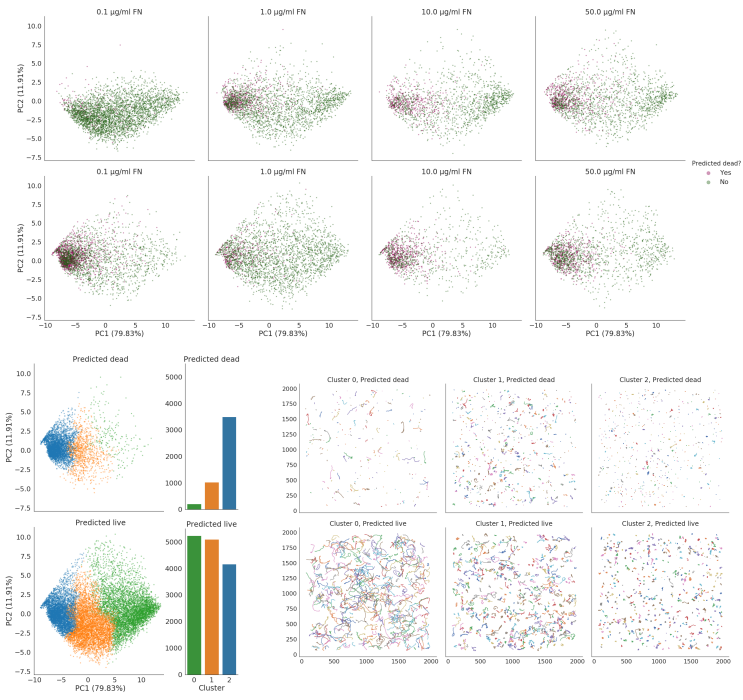

Figure 10: **Live-Dead classification in relation to the tracking data and the found clusters.** The top figure shows PCA on track features. The marker color indicates the prediction of the live-dead classifier applied to the raw image data of the tracked cells. Evaluation of the classifier is shown in figure S3. The cells that are predicted to be dead seem to be concentrated around the mid-left of the plot. It is evident that the number of cells predicted to be dead varies between wells, even between wells with the same fibronectin concentration. In the bottom left figure the data from all wells is shown split by the live-dead prediction This figure shows that tracks corresponding to cells that the network predicted to be dead lie in Cluster 2 in most cases but can also be found in the other clusters. The tracks of cells that the network predicted to be alive are more evenly distributed among the clusters but slightly less common in Cluster 2. A sample of tracks split by both cluster and live-dead classification are shown in the bottom right. The tracks of the cells in Cluster 2 are qualitatively different between the predicted live and predicted dead.

3.4 Prediction

For the final step, determining whether a model could be created that would find cells belonging to these clusters based on data from a single image, the dead cells were excluded as well as the cells in Cluster 2. The task for the models would be to predict whether a living cell belonged to Cluster 0 or Cluster 1. Figure 6 shows how the sources of data and earlier models feed in to this final step.

The first model used simple morphological data extracted from the images by segmenting the cells in the fluorescent channel of the cytoplasmic marker as well as the nuclear marker. For the segmentation mask of the cytoplasmic marker geometric properties and intensity-based properties were calculated and used as features. The geometric properties were: area, major axis length, and minor axis length. The intensity-based properties were: mean, standard deviation, maximum, minimum, and first moment of intensity as a function of distance to center. For the nucleus the same geometric properties were used but no intensity-based properties were calculated. A random forest classifier was fit to 80% of the data and tested against the other 20%. The result of applying the model to the test data is shown in figure 11. The accuracy is probably the most useful metric in this case. The purpose of doing this prediction is to be able to target either group in the microscope and the groups are fairly balanced; for both training and test data the ratio of cells belonging to Cluster 0 is $\approx$54 %. An accuracy of $\approx$61 % is better than chance but will not be very useful for getting high magnification images of cells belonging to the different clusters. By looking at the precisions we see that this model would be about equally useful for finding cells from either cluster.

For the second model, a convolutional autoencoder generating a 128-dimensional code was trained on the DIC and nuclei data of the dataset used for training the live-dead classifier. The network architecture can be seen in figure S4. The encoder part was then used to create features from the images of the cells in the original data. A random forest classifier was trained on this data, once again with a 80:20 train-test split. From the results in figure 12 we can see that, in this case, all metrics are better for this model than for the model using a small number of morphological features. Even though the error rate is still high this model could be useful for targeting cells belonging to a specific cluster.

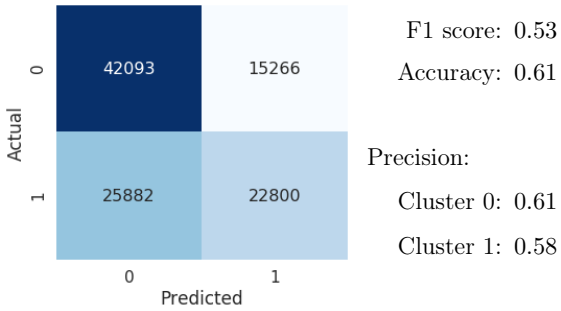

Figure 11: **Confusion matrix and metrics for random forest on image region properties.** The confusion matrix shows the frequencies of prediction that a cell belongs to Cluster 0 or Cluster 1 and the corresponding actual cluster the cell belonged to. The predictions were made by a random forest trained on predefined properties of a cytoplasmic segmentation mask and a nuclear segmentation mask.

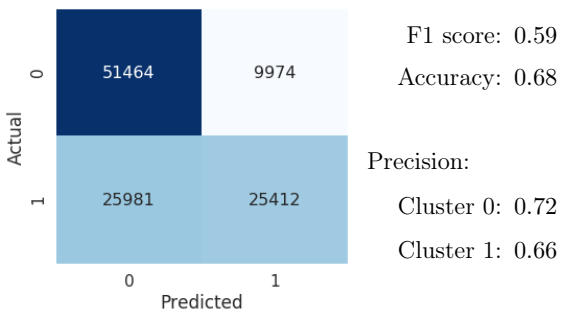

Figure 12: **Confusion matrix and metrics for random forest on autoencoder code.** The confusion matrix shows the frequencies of prediction that a cell belongs to Cluster 0 or Cluster 1 and the corresponding actual cluster the cell belonged to. The predictions were made by a random forest trained on an autoencoder code encoding the DIC and nuclear channel.

4 Discussion

The inability of the image based models to clearly differentiate between the classes may in part be attributed to experimental difficulties. One issue that affects the first model is that many cells, perhaps because they are dead or dying, do not have any fluorescent signal from the cytoplasmic dye.This makes the morphological data noisy. Generally, all the data analysis is affected by the high number of dead cells in many of the wells. In particular it makes it difficult to discern what happens to the distribution of migratory behaviours as fibronectin concentration is varied, something the literature indicates could be useful for seeing different migration modes. Even though the live-dead classification model gives us some ability to filter out dead cells, the large number of dead cells as well as the underlying reason for the cell death may still affect the other cells in the sample. Furthermore the network is trained on images of cells where the live-dead stain has been activated enough to give a strong signal, something that likely happens at the end stage of cell death and may never occur for some cells. For future studies, to filter cells in the early processes leading to cell death and improve the performance, the model could be trained on the frames leading up to the frame where a dead staining is detectable.

A factor that likely affects the result and could be improved in future studies is the handling of cell contact. The cells, in particular the dead and dying cells, clump together and migrating cells sometimes collide and pull on each other. These factors make segmentation difficult, influences the neural networks by giving them images containing multiple cells and affects the migration tracks in unexpected ways. To improve this a more advanced segmentation method, e.g. a CNN based on the U-Net architecture, could be used. A rigorous way of excluding cells that are affecting each other through contact should be found. Efforts should also be made to as much as possible minimize these issues in future experiments.

One of the aims of the study was to see if an unsupervised approach could recover previously described modes of cell migration. Based on the data presented here it is not possible to determine such a correspondence since these previous studies used higher magnification data. However, through the models produced, targeted high magnification data can be collected and analyzed further to determine if such a correspondence exists.

5 Conclusion

The pipeline that was implemented in this project has enabled efficient processing of a large dataset and was able to use clustering to find qualitatively different subpopulations of cells. The created predictive models could allow us to refine one group or the other using targeted microscopy and indicate that there are visible properties of the cell that correlate with its migratory behavior. The results can not at this point confirm the existence of clearly distinct types of cell migration in the analyzed samples but show avenues along which further inquires into this question may proceed. As described in the discussion, the experimental setup can be improved and further aspects added to the analysis. Once these improvements have been made the goal is to apply this model in real-time in the microscope to automatically classify the migratory subpopulations of cells.

6 Acknowledgements

I would like to thank Christian Antfolk for supervising this project, making me create a detailed project schedule and allowing me to throw it out the window. I would like to thank my other supervisor Pontus Nordenfelt as well as all the members of Nordenfelt Lab for their guidance and support. A very special thanks to Oscar André for acquiring the data used in this project, and providing the Enya used in writing. Finally, a thank you to my wife Vibha for introducing me to quantitative immunobiology and always knowing how to move forward.

References

[1] P Friedl and E B Bröcker. "The biology of cell locomotion within three-dimensional extracellular matrix." In: *Cellular and Molecular Life Sciences* 57.1 (Jan. 2000), pp. 41–64. DOI: 10.1007/s000180050498. URL: http://dx.doi.org/10.1007/s000180050498 (visited on 09/25/2019) (cit. on p. 1).

[2] D A Lauffenburger and A F Horwitz. "Cell migration: a physically integrated molecular process." In: *Cell* 84.3 (Feb. 1996), pp. 359–369. DOI: 10.1016/s0092-8674(00)81280-5. URL:

http://dx.doi.org/10.1016/s0092-8674(00)81280-5 (visited on 09/26/2019) (cit. on p. 1).

[3] A R Huber et al. "Regulation of transendothelial neutrophil migration by endogenous interleukin-8." In: *Science* 254.5028 (Oct. 1991), pp. 99–102. DOI: 10.1126/science.1718038. URL: http://dx.doi.org/10.1126/science.1718038 (visited on 09/26/2019) (cit. on p. 1).

[4] Peter Friedl and Katarina Wolf. "Tumour-cell invasion and migration: diversity and escape mechanisms." In: *Nature Reviews. Cancer* 3.5 (May 2003), pp. 362–374. DOI: 10.1038/nrc1075. URL: http://dx.doi.org/10.1038/nrc1075 (visited on 01/19/2017) (cit. on p. 1).

[5] Katarina Wolf et al. "Compensation mechanism in tumor cell migration: mesenchymal-amoeboid transition after blocking of pericellular proteolysis." In: *The Journal of Cell Biology* 160.2 (Jan. 2003), pp. 267–277. ISSN: 0021-9525. DOI: 10.1083/jcb.200209006. URL: http://dx.doi.org/10.1083/jcb.200209006 (visited on 03/20/2017) (cit. on p. 1).

[6] Hamdah Shafqat-Abbasi et al. "An analysis toolbox to explore mesenchymal migration heterogeneity reveals adaptive switching between distinct modes." In: *eLife* 5 (Jan. 2016), e11384. DOI: 10.7554/{eLife}.11384. URL: http://dx.doi.org/10.7554/%7BeLife%7D.11384 (visited on 08/26/2019) (cit. on pp. 1, 2).

[7] Sylvia E Le Dévédec et al. "Systems microscopy approaches to understand cancer cell migration and metastasis." In: *Cellular and Molecular Life Sciences* 67.19 (Oct. 2010), pp. 3219–3240. DOI: 10.1007/s00018-010-0419-2. URL: http://dx.doi.org/10.1007/s00018-010-0419-2 (visited on 01/06/2020) (cit. on p. 1).

[8] Jeff W Lichtman and José-Angel Conchello. "Fluorescence microscopy." In: *Nature Methods* 2.12 (Dec. 2005), pp. 910–919. DOI: 10.1038/nmeth817. URL: http://dx.doi.org/10.1038/nmeth817 (visited on 09/27/2019) (cit. on p. 1).

[9] John G Lock and Staffan Strömblad. "Systems microscopy: an emerging strategy for the life sciences." In: *Experimental Cell Research* 316.8 (May 2010), pp. 1438–1444. DOI: 10.1016/j.

yexcr.2010.04.001. URL: http://dx.doi.org/10.1016/j.yexcr.2010.04.001 (visited on 09/27/2019) (cit. on p. 2).

[10] R M Donovan et al. "A computer-assisted image-analysis system for analyzing polymorphonuclear leukocyte chemotaxis in patients with diabetes mellitus." In: *The Journal of Infectious Diseases* 155.4 (Apr. 1987), pp. 737–741. DOI: 10.1093/infdis/155.4.737. URL: http://dx.doi.org/10.1093/infdis/155.4.737 (visited on 09/29/2019) (cit. on pp. 2, 3).

[11] Weigang Wang et al. "Tumor cells caught in the act of invading: their strategy for enhanced cell motility." In: *Trends in Cell Biology* 15.3 (Mar. 2005), pp. 138–145. DOI: 10.1016/j.tcb.2005.01.003. URL: http://dx.doi.org/10.1016/j.tcb.2005.01.003 (visited on 10/08/2020) (cit. on p. 3).

[12] M Poujade et al. "Collective migration of an epithelial monolayer in response to a model wound." In: *Proceedings of the National Academy of Sciences of the United States of America* 104.41 (Oct. 2007), pp. 15988–15993. DOI: 10.1073/pnas.0705062104. URL: http://dx.doi.org/10.1073/pnas.0705062104 (visited on 10/08/2020) (cit. on p. 3).

[13] Herbert Freeman. "On the encoding of arbitrary geometric configurations". In: *IEEE Transactions on Electronic Computers* EC-10.2 (June 1961), pp. 260–268. ISSN: 0367-7508. DOI: 10.1109/{TEC}.1961.5219197. URL: http://ieeexplore.ieee.org/document/5219197/ (visited on 10/11/2019) (cit. on p. 3).

[14] Lars Bretzner and Tony Lindeberg. "Feature Tracking with Automatic Selection of Spatial Scales". In: *Computer Vision and Image Understanding* 71.3 (Sept. 1998), pp. 385–392. ISSN: 10773142. DOI: 10.1006/cviu.1998.0650. URL: https://linkinghub.elsevier.com/retrieve/pii/S1077314298906506 (visited on 10/14/2019) (cit. on p. 3).

[15] Patrick Emami et al. "Machine Learning Methods for Solving Assignment Problems in Multi-Target Tracking". In: *arXiv* (Feb. 2018). URL: https://arxiv.org/abs/1802.06897 (visited on 10/15/2019) (cit. on p. 3).

[16] Khuloud Jaqaman et al. "Robust single-particle tracking in live-cell time-lapse sequences." In: *Nature Methods* 5.8 (Aug. 2008), pp. 695–702. ISSN: 1548-7105. DOI: 10.1038/nmeth.1237. URL:

http://dx.doi.org/10.1038/nmeth.1237 (visited on 10/15/2019) (cit. on p. 4).

[17] Nicolas Chenouard et al. "Objective comparison of particle tracking methods." In: *Nature Methods* 11.3 (Mar. 2014), pp. 281–289. DOI: 10.1038/nmeth.2808. URL: http://dx.doi.org/10.1038/nmeth.2808 (visited on 10/15/2019) (cit. on p. 4).

[18] John A. Lee and Michel Verleysen, eds. *Nonlinear Dimensionality Reduction*. New York, NY: Springer New York, 2007. ISBN: 978-0-387-39350-6. DOI: 10.1007/978-0-387-39351-3. URL: http://link.springer.com/10.1007/978-0-387-39351-3 (visited on 10/20/2019) (cit. on pp. 5, 6).

[19] E P Xing and R M Karp. "CLIFF: clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts." In: *Bioinformatics* 17 Suppl 1 (2001), S306–15. DOI: 10.1093/bioinformatics/17.suppl_1.s306. URL: http://dx.doi.org/10.1093/bioinformatics/17.suppl%5C_1.s306 (visited on 10/21/2019) (cit. on p. 6).

[20] Norbert Jankowski and Krzysztof Grabczewski. "Learning Machines". In: *Feature Extraction*. Ed. by Isabelle Guyon et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 29–64. ISBN: 978-3-540-35487-1. DOI: 10.1007/978-3-540-35488-8_2. URL: http://link.springer.com/10.1007/978-3-540-35488-8%5C_2 (visited on 10/21/2019) (cit. on p. 6).

[21] J. R. Quinlan. "Induction of decision trees". In: *Machine learning* 1.1 (Mar. 1986), pp. 81–106. ISSN: 0885-6125. DOI: 10.1007/{BF00116251}. URL: http://link.springer.com/10.1007/%7BBF00116251%7D (visited on 10/15/2020) (cit. on p. 6).

[22] L Breiman. "Random forests". In: *Machine learning* (2001). URL: https://link.springer.com/article/10.1023/a:1010933404324 (visited on 10/15/2020) (cit. on p. 6).

[23] Fabian Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* (2011). URL: https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html (visited on 10/15/2020) (cit. on p. 6).

[24] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. illustrated. MIT Press, 2016. ISBN: 9780262035613. URL: https://books.google.se/books?id=%7BNp9SDQAAQBAJ%5C%7D&printsec=frontcover%5C&dq=goodfellow+deep+

learning%5C&hl=en%5C&sa=X%5C&ved=%7B0ahUKEwjdosPu%5C_8HmAhUBm1wKHZD3B54Q6AEIKDAA%5C%7D#v=onepage%5C&q=goodfellow%5C%20deep%5C%20learning%5C&f=false (visited on 12/19/2019) (cit. on pp. 7, 8).

[25] Francois Chollet. "Xception: Deep Learning with Depthwise Separable Convolutions". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017, pp. 1800–1807. ISBN: 978-1-5386-0457-1. DOI: 10.1109/{CVPR}.2017.195. URL: http://ieeexplore.ieee.org/document/8099678/ (visited on 10/17/2019) (cit. on p. 8).

[26] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Ed. by Nassir Navab et al. Vol. 9351. Lecture notes in computer science. Cham: Springer International Publishing, 2015, pp. 234–241. ISBN: 978-3-319-24573-7. DOI: 10.1007/978-3-319-24574-4_28. URL: http://link.springer.com/10.1007/978-3-319-24574-4%5C_28 (visited on 07/14/2020) (cit. on p. 8).

[27] Hao Dong et al. "Automatic Brain Tumor Detection and Segmentation Using U-Net Based Fully Convolutional Networks". In: *Medical image understanding and analysis*. Ed. by María Valdés Hernández and Víctor González-Castro. Vol. 723. Communications in computer and information science. Cham: Springer International Publishing, 2017, pp. 506–517. ISBN: 978-3-319-60963-8. DOI: 10.1007/978-3-319-60964-5_44. URL: http://link.springer.com/10.1007/978-3-319-60964-5%5C_44 (visited on 10/21/2019) (cit. on p. 8).

[28] Alejandra Gonzalez-Beltran and Philippe Rocca-Serra. "CellMigStandOrg/MIACME: MIACME v1.1". In: *Zenodo* (2019). DOI: 10.5281/zenodo.3457561. URL: https://zenodo.org/record/3457561 (visited on 10/20/2020) (cit. on p. 9).

[29] *jupyterlab/jupyterlab: JupyterLab computational environment.* WEBSITE. URL: https://github.com/jupyterlab/jupyterlab (visited on 10/08/2020) (cit. on p. 10).

[30] Thomas Kluyver et al. "Jupyter Notebooks - a publishing format for reproducible computational workflows". In: *Positioning and Power in Academic Publishing: Players, Agents and Agen-*

das. Ed. by Fernando Loizides and Birgit Scmidt. IOS Press, 2016, pp. 87–90. URL: https://eprints.soton.ac.uk/403913/ (visited on 10/08/2020) (cit. on p. 10).

[31] Alex Bäuerle, Christian van Onzenoodt, and Timo Ropinski. "Net2Vis – A Visual Grammar for Automatically Generating Publication-Ready CNN Architecture Visualizations". In: *arXiv* (Feb. 2019). URL: https://arxiv.org/abs/1902.04394 (visited on 10/20/2020) (cit. on p. 32).

Supplementary Material

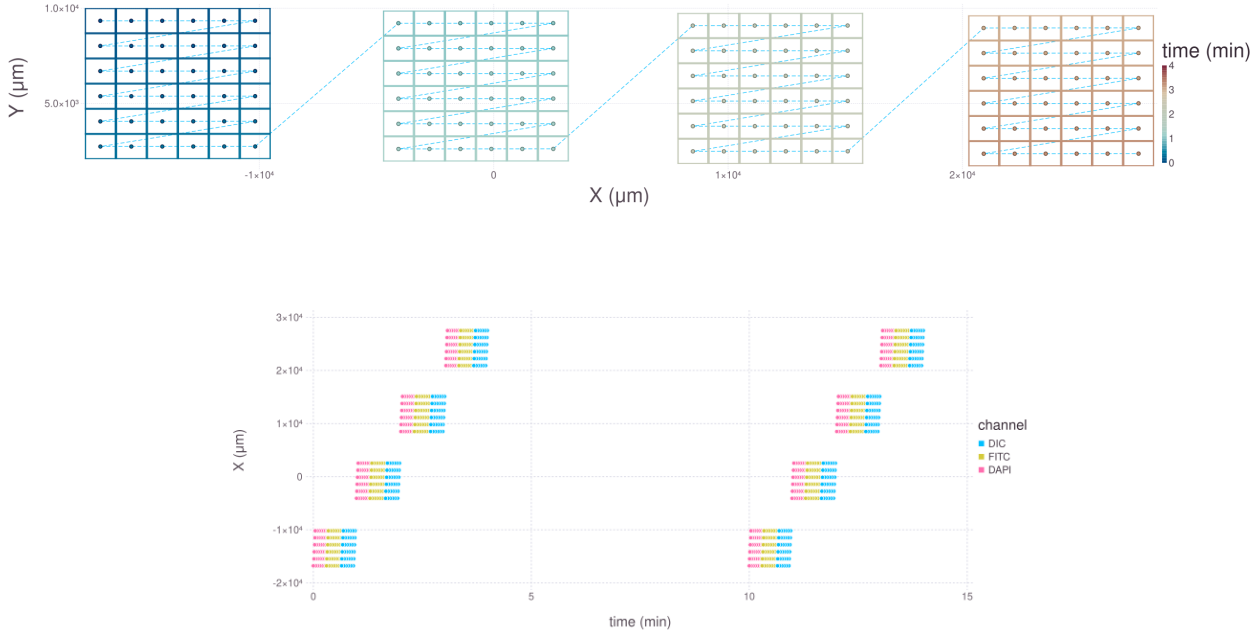

Figure S1: **Description of the acquisition process.** The top figure shows the positions and sizes of the fields of view in the microscope stage coordinate system. The blue dashed line indicates the order the positions are visited in; for each well the stage is moving in the x-direction first. The colors of the fields of view show the point in time when an image was acquired for one particular channel during the first run through the sample. In the bottom figure the x-coordinate of the stage position is shown as a function of time for the first two frames with the color indicating what channel the microscope was capturing at the time. After capturing all fields of view in one well in "DAPI" (fluorescent marker, nucleus) the microscope switches to "FITC" (fluorescent marker, cytoplasm) and the stage moves to the first field of view of the well. Capturing all wells takes just under 4 minutes after which the microscope waits for 6 minutes before starting the next round of acquisition yielding a framerate of 0.1/min.

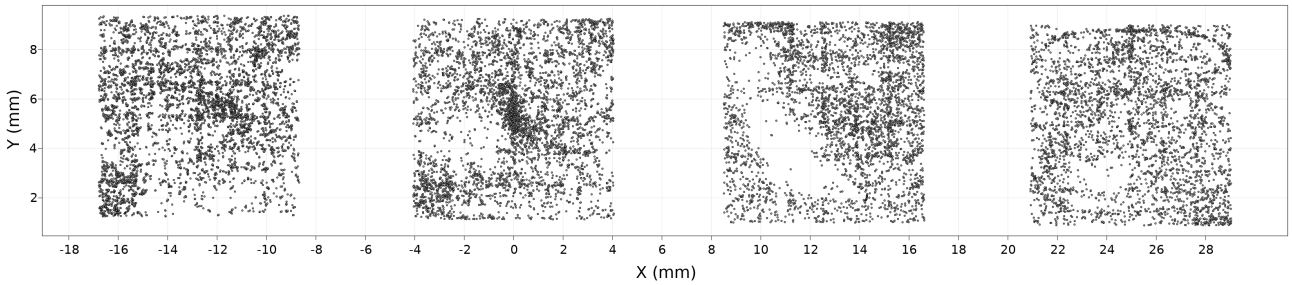

Figure S2: **Distribution of cells within wells.** Scatter plot of positions of detected cells in the first frame of the image time series.

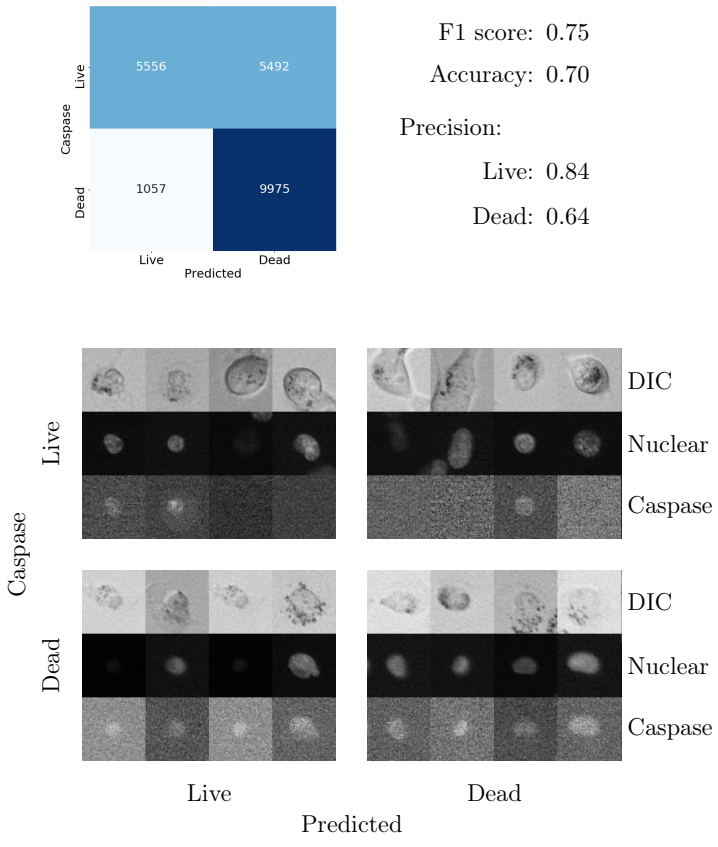

Figure S3: **Confusion matrix and prediction examples for live-dead classifier.** The confusion matrix shows the frequencies that a cell is dead or alive as predicted by a neural network model and measured from the caspase signal. The images show examples of cells randomly sampled from the corresponding element in the confusion matrix.

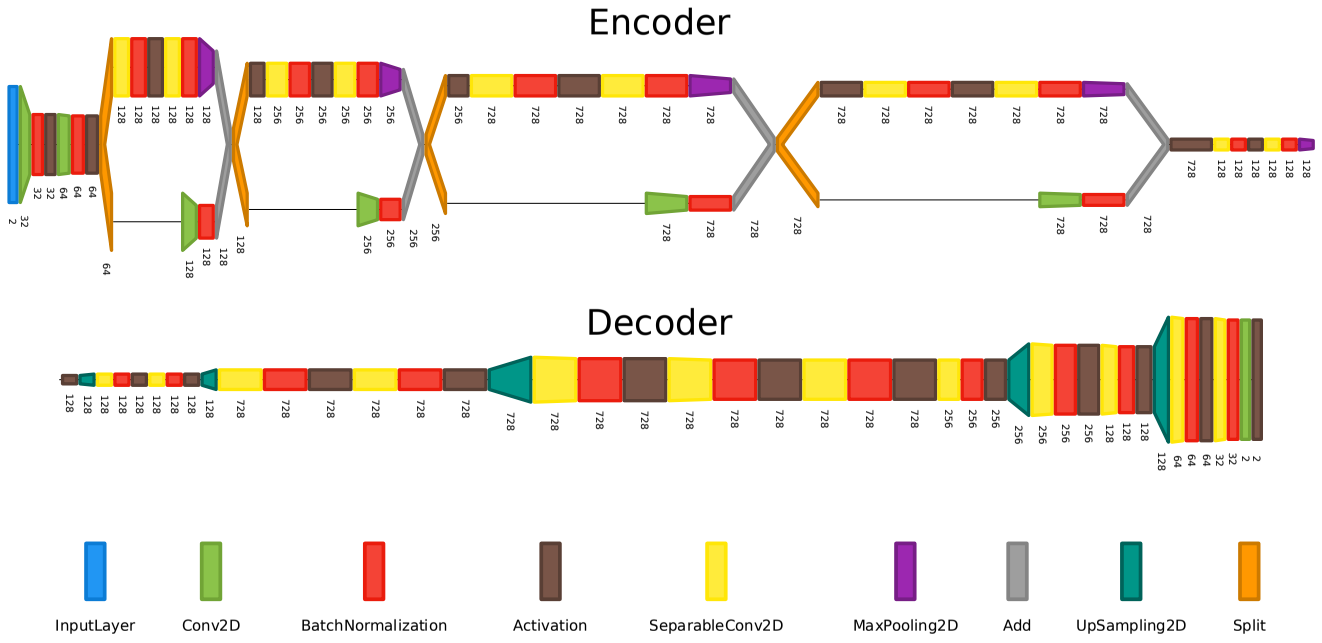

Figure S4: **Schematic of actual autoencoder architecture.** A visualization of the autoencoder architecture used for dimensionality reduction. The visualization was generated using Net2Vis [31].

S1 Tracking code

```
from multiprocessing import Pool
from collections import defaultdict
from pathlib import PosixPath
from argparse import ArgumentParser
from psycopg2 import connect
from psycopg2.extensions import register_adapter
import os

import numpy as np
from nd2reader import Nd2
from scipy.spatial.distance import cdist
from skimage.feature import blob_log
from skimage.filters import threshold_local

from lapjv import lapjv

db = 'dbname=analysis_dev user={db_user} host=localhost password={db_pass}'.format(**os.environ)
blocking = 1e300
register_adapter(PosixPath, lambda p: str(p.resolve()))


class Frame:
    def __init__(self, fh, channel, field_of_view, frame):
        self.fh = fh
        self.channel = channel
        self.fov = field_of_view
        self.frame = frame

    def get(self):
        return self.fh.get_image(self.frame, self.fov, self.channel, 0)


def get_args():
    """Parses the command line arguments"""
    parser = ArgumentParser()
    parser.add_argument('well_id', type=int, help='Id of well')
    parser.add_argument('-j', type=int, default=8, help='Number of threads')
    parser.add_argument('--force', '-f', action='store_true', help='Number of threads')
    return parser.parse_args()


def my_blob(im):
    return blob_log(im, min_sigma=8, max_sigma=12, threshold=0.45, overlap=0.5)


def blobs_in_frame(img):
    img = img > threshold_local(img, 501, offset=-50)
    return my_blob(img)


def track_cells(frames, max_dist=50, threads=1):
    with Pool(threads) as p:
        blobs_list = p.map(blobs_in_frame, (f.get() for f in frames))

    pos = [(0, [x], [y]) for x, y in blobs_list[0]]
    ind = list(range(len(blobs_list[0])))

    for frame_i in range(len(blobs_list) - 1):
        prev = blobs_list[frame_i]
        blobs = blobs_list[frame_i + 1]
        n = len(prev)
        m = len(blobs)
        indnxt = [None] * m
        top_left = cdist(prev, blobs, 'sqeuclidean')
        calt = top_left.max() * 1.05 if top_left.size > 0 else 0
        cmin = top_left.min() if top_left.size > 0 else 0
        top_left[top_left > max_dist**2] = blocking
        top_right = np.empty((n, n), dtype='double')
        bottom_left = np.empty((m, m), dtype='double')
        top_right.fill(blocking)
        bottom_left.fill(blocking)
        np.fill_diagonal(top_right, calt)
        np.fill_diagonal(bottom_left, calt)
```

```
        aux = top_left.T.copy()
        aux[aux != blocking] = cmin
        cost_mat = np.empty((n + m, n + m), dtype='double')
        cost_mat[:n, :m] = top_left
        cost_mat[:n, m:] = top_right
        cost_mat[n:, :m] = bottom_left
        cost_mat[n:, m:] = aux

        row_ind, col_ind, _ = lapjv(cost_mat) if n > 0 or m > 0 else ([], [], [])
        for r, c in enumerate(row_ind[:n]):
            if c < m:
                pos[ind[r]][1].append(blobs[c, 0])
                pos[ind[r]][2].append(blobs[c, 1])
                indnxt[c] = ind[r]
        for r, c in enumerate(row_ind[n:]):
            if c < m:
                pos.append((frame_i, [], []))
                pos[-1][1].append(blobs[c, 0])
                pos[-1][2].append(blobs[c, 1])
                indnxt[c] = len(pos) - 1
        ind = indnxt
    return pos


def row_generator(file_id, fov_id, pos):
    for track_id, (start, y, x) in enumerate(pos):
        for i, (pos_x, pos_y) in enumerate(zip(x, y), start):
            yield file_id, fov_id, track_id, i, pos_x, pos_y


def already_tracked(c, file_id, fov):
    c.execute("SELECT exists(SELECT 1 FROM objects WHERE file_id=%s AND fov=%s)", (file_id, fov))
    return c.fetchone()[1]


def all_sequences(nd2, channel):
    return [[Frame(nd2, channel, i, t) for t in nd2.frames] for i in nd2.fields_of_view]


def process_sequence(well_id, threads, force):
    with connect(db) as conn:
        with conn.cursor() as c:
            c.execute("SELECT f.id, path, c.name FROM channels AS c "
                      "INNER JOIN files AS f ON (c.file_id = f.id) "
                      "WHERE (c.name ~ 'DAPI' OR path ~ 'ChannelDAPI') "
                      "AND f.well_id = %(id)s", {'id': well_id})

            files = c.fetchall()
            for i_file, (file_id, fname, channel) in enumerate(files):
                for fov, frames in enumerate(all_sequences(Nd2(fname), channel)):
                    if not force:
                        if already_tracked(c, file_id, fov):
                            continue
                    pos = track_cells(frames, threads=threads)
                    objects = defaultdict(list)
                    for row in row_generator(file_id, fov, pos):
                        objects[row[0:3]].append(row[3:])
                    ids = []
                    for k in objects:
                        c.execute('INSERT INTO objects (file_id, fov, inserted_at, updated_at) '
                                  'VALUES (%s, %s, NOW(), NOW()) RETURNING id', k[0:2])
                        ids.append(c.fetchone()[0])
                    for (k, track), objid in zip(objects.items(), ids):
                        c.executemany('INSERT INTO tracks (object_id, frame, pos_x, pos_y) '
                                      'VALUES (%s, %s, %s, %s)', ((objid, *t) for t in track))

                    conn.commit()
                    print((i_file + fov / len(all_sequences)) / len(files), flush=True, end='')


def main(args):
    process_sequence(args.well_id, args.j, args.force)


if __name__ == '__main__':
    main(get_args())
```