Feasibility Study for Autovalidation of Blood Cells on a Digital Morphology System

Elin Branzell, Linnea Hellholm

Master's thesis
2019:E23

LUND UNIVERSITY

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

CENTRUM SCIENTIARUM MATHEMATICARUM

Master's Thesis in Biomedical Engineering

Feasibility study for autovalidation of blood cells in a digital morphology system

by

Linnea Hellholm and Elin Branzell

Abstract

This thesis was carried out at CellaVision, a company within digital blood analysis, with the aim of investigating the possibility of autovalidation of blood cells using neural networks. The thesis started with preparing a dataset for training and validation, including cell features and binary labels indicating if the cell is easy or difficult to classify. This dataset consisted of 122 227 cell images. The binary labels were generated through several different methods such as using neural networks, statistics from CellaVision's system's classification result and manual classification by a morphology expert.

Three kinds of neural networks were tested with the aim of separating easy cells from difficult ones: a binary Artificial Neural Network (ANN), an autoencoder for anomaly detection and a Self-Organizing Feature Map (SOFM) visualizing the position of difficult cells in clusters. The methods were compared and the performance were evaluated. It was found that the ANN was not useful for this task, while the autoencoder could be used for successfully autovalidating 74% of the cells. With better labeling techniques for the dataset, the performance could potentially be improved. The SOFM was not used for anomaly detection in this study, but for visual analysis of the clusters and labels. However, it may become relevant in the future to look further into this method's possibility of anomaly detection, as patterns in the clusters were apparent.

BME14
Master's Thesis
May 29, 2019
Supervisors: Karl Åström and Kent Stråhlén
Examiner: Mikael Nilsson

LUND UNIVERSITY

LTH
FACULTY OF ENGINEERING

Popular science summary

Blood cell autovalidation- will machines replace humans in the future?

We're about to enter an exciting time where autovalidation will change the game of industries, become a part of the technologies we use in our everyday life and affect the whole society. Artificial Intelligence (AI) have already changed the map of where tech can replace human minds and no one knows the limits of what the future holds. One area where AI has begun to conquer land is the difficult, time consuming and vital analysis of blood samples done at hospitals. Will AI machines be able to do the job of humans in blood analyzes in the future?

The implementation of autovalidation means that processes which are normally analyzed and controlled by humans can instead be validated by a machine. This can lead to major savings in time and costs, especially in pressured areas like the healthcare sector, while providing unbiased results based on statistics and experience.

CellaVision is a company which have taken blood analysis towards a digital transformation. They now want to answer the question if it is possible to autovalidate the digital blood cell analysis using AI-methods. Applying autovalidation to the systems of CellaVision would mean that some analyzes can be completely done by computers, resulting in improved efficiency and savings in time and cost at hospitals and laboratories around the world. If created and implemented correctly, this type of technology could save lives.

Several different types of AI-methods were tested for autovalidation of blood cells. The task for the model was to separate easy cells from difficult ones, which is the first step in autovalidation. We found that the most successful method was a module detecting abnormal patterns among cells. It could be used for successfully autovalidating 74% of the cells. It is also probable that the result could have been even better if a better technique had been used to label the cell images which the module was trained on.

In the end of the study, an algorithm arranging cells with similar appearance in clusters was used for visualizing the results. It would definitely be interesting to look further into this technique in the future to see if it could be used for autovalidation itself. You could already see patterns of how the difficult cells were located among the normal cells, which makes this approach very promising.

All in all, the results indicates a promising future. Who knows, maybe one day the machines will have replaced the humans when it comes to blood analysis?

Acknowledgement

This thesis could never had been accomplished without the help of others. Therefore we would like to start by saying thanks to those who have supported us during the work. Firstly, a big thank you to Kent Stråhlén at CellaVision and Karl Åström at the Department of Mathematics, our supervisors who have helped us during the entire study and always been there when in need of new ideas, knowledge within specific areas and support with everyday matters. Secondly, we would like to thank Annelie McCorkindale for classifying cells in the speed of light. We would also like to thank Jesper Jönsson for all the help with generating data through CellaVision's system. Last but not least, we would like to thank all the personnel at CellaVision for welcoming us to their team from day one. It has been a pleasure to work with you.

Glossary

Batch	A set of patches/inputs
Epoch	One epoch is completed when all batches have been passed through the model
SOM/SOFM	A type of ANN
Label	The class of the object
Supervised Learning	Training performed with access to labels
Unsupervised Learning	Training performed without access to labels
Loss	Another word for error
Accuracy	Proportion of data with the same predicted label as the validation label
Specificity	Probability of a negative test that the test truly is negative
Sensitivity	Probability of a positive test that the test truly is positive
Miss-rate	Probability of a negative test being truly positive
Internal labeling	Assigning the task of labeling data to an in-house team
Data programming	Assigning a task to be solved by a computer with a programmed software

Acronyms

CBC	Complete Blood Count
ANN	Artificial Neural Network
NN	Neural Network
ReLU	Rectified Linear Unit
SOFM	Self-Organizing Feature Map
BMU	Best Matching Unit
AE	Autoencoder
PCA	Principal Component Analysis
PRC	Precision-Recall Curve

Contents

Chapter 1

Introduction

1.1 Motivation

1.1.1 CellaVision's History and Aim

CellaVision was founded in 1994 by Christer Fåhraeus with the aim of developing an instrument for automated blood analysis replacing the use of the manual microscopes. The year of 2001, the first instrument of CellaVision was sold in Europe. CellaVision offers digital solutions for medical microscopy, replacing the traditional microscopes with instruments based on digital image analysis, artificial intelligence and IT. The digital microscopy enhances diagnostics while streamlining the work flow and lowering costs (1, p. 8).

1.1.2 Using CellaVision's System

When a patient is suspected to suffer from a hematological disease, a complete blood count (CBC) is often the first test ordered by the health care. The CBC is one of the most common diagnostic tests in the world, used routinely to get an overall status of the different cells present in the blood of the patient. The analysis provides information about the distribution and morphology of the blood cells, such as size, color and shape (2). It is estimated that four billion CBC analyses are performed in the world each year, and approximately 15% of these requires further analysis. This is where CellaVision's system enter the process.

CellaVision's analysis is often required because of the presence of immature or sick cells in the patient's blood. This could be the case for hematological diseases such as anemia, lack of platelets, leukemia or various tumor diseases e.g. lymphoma. The systems of CellaVision are developed with these particular tests in mind. The result of the analysis, including images of the white blood cells, is presented to the user on a screen. The instruments includes applications for analysis of blood and other body fluids, and software enabling examination of analysis results remotely.

CellaVision has not yet reached a completely automatic system, which was the goal from the start. Today it is used as a decision support for diagnosis. To move towards the goal, CellaVision wants to find a way to create an autovalidating system instead of a decision support system.

1.1.3 Previous Work

Autovalidation is the automatic release of test results into the patients record without any manual intervention (3). The results are only released if they meet the criterias defined by the lab, otherwise manual verification is done. The autovalidation process can be explained as a rule-based system for validation of laboratory test results, with an algorithm based on clear rules of decision-making from the laboratory (4). The system makes the validation of the result more uniform and objective as long as the rules are clearly and unambiguously stated. The use of autovalidated processes are becoming more and more important as the requests of laboratory test increases and due to the chance of minimizing the risk of human errors (4).

Autovalidation is used for several processes in clinical chemistry today. One example where the use is well-established is when performing complete blood counts (CBC). In a study from 2014, it was concluded that the use of autovalidation in CBC processes decreased the workload by 7.7 - 11.6 hours per 3 000 test results (5). In this study the autovalidation was done by the laboratory information system according to set criterias, meaning that the critical values for the results was known from the start.

Since the 1960's, numerous projects have attempted to develop computer assisted and autovalidation techniques for the aim of screening women for cervical cancer. The developed techniques has been able to recognize cells and smears as normal or abnormal, which have been found to effectively reduce the workload in the laboratories. The developed techniques uses image analysis approaches based on manual feature extraction. These features are often mathematical features determining the morphology, texture, intensity, borders or the neighbouring area around a pixel (6, p. 1-6) (7, p. 1-2, 6-11). These projects have been successful in detecting abnormal cells and smears through image analysis, although feature detection and analysis completely performed by neural networks has not been used for these projects. It would thereby be interesting to test the use of neural networks on projects combining image analysis and autovalidation.

In 2016, a study was done trying to use neural networks as a method of autoverifying the results in a biochemistry laboratory (8). The method included building a neural network model which was able to create a decision algorithm. This algorithm would signal when the results in a test indicated that a patient was sick. The data for the study consisted of laboratory test results from patients which gave several parameters for each patient. Examples are the level of glucose, level of sodium or the age of the patient. Altogether the parameters represented the well-being of the patient. These parameters had pre-determined critical values and the aim of the project was to find a model which could identify the cases when the values were critical and the result indicated that the patient was sick. The neural network model could thereby train on the parameters with known critical values.

There is a big difference in the purpose of autovalidation for previous studies and this. In the previous ones, autovalidation is used as a way of finding tests with alarming results, or results which are very "off", indicating a sick patient or a broken machine. These kind of alarming result have specific critical values in the parame-

ters. This study aims to create an autovalidation as a way of finding results where the system is potentially wrong in its conclusions, specifically to find events where a cell has a risk of being classified into the wrong cell class. These results do not have specific critical values in the same way, and are therefore more difficult to get at. The results will not be "alarming" in any way; it will look like a normal result but will be wrongly concluded. In this case, the parameters used are features from cell images, which are extracted from neural networks. It is not possible to know its critical values, and the model therefore has to do the autovalidation without known limits.

To be able to do the autovalidation the study aims for, the system should determine which results the system is sure of, and which might be faulty and thereby needs to be manually re-evaluated. For CellaVision, this means that the autovalidation algorithm needs to find which cells are "easy to classify" and which are "difficult to classify".

A study in Dordrecht used the probabilities from CellaVision's image classification as critical values of where to draw the line for the need of manual re-evaluation. These probabilities were extracted directly from the neural network model, and the critical value of these probabilities were decided based on certain percentages of the resulting probabilities. This method turned out to work well in Dordrecht, but it was found that the same critical values did not give good results in other laboratories. For this to work, new critical values needed to be calculated specifically for each laboratory. That would imply uneccessary work and resources. A more general method to find critical values was needed.

Based on our search of the literature, a general autovalidation method for blood cell classification has not yet been tested using the ANN approach.

1.2 Aim

The aim of this thesis is to investigate the possibility of creating a binary classifying module. It should be able to separate white blood cells which are easy to classify from those that are difficult, with the purpose of autovalidating the easy cells. This by building different ANN's and training them on data from CellaVision's database. The module should be independent of which hospital has made the blood cell preparation.

1.3 Outline of the Thesis

In this report, the chapters Methodology, Results and Discussion are divided into two different phases, referred to as phase 1 and phase 2, see figure 1.1. The results and conclusions from section 1 have been used as the input data for phase 2, and it is therefore recommended to read all three chapter's sections related to phase 1 before continuing with phase 2. How phase 1 and phase 2 are connected in the project is described in the flowchart in figure 3.1 on page 23. There are also two flowcharts further explaining the two separate phases and the relations between their different

sections and subchapters: figure 3.3 on page 29 describes phase 1 and 3.6 on page 35 describes phase 2.

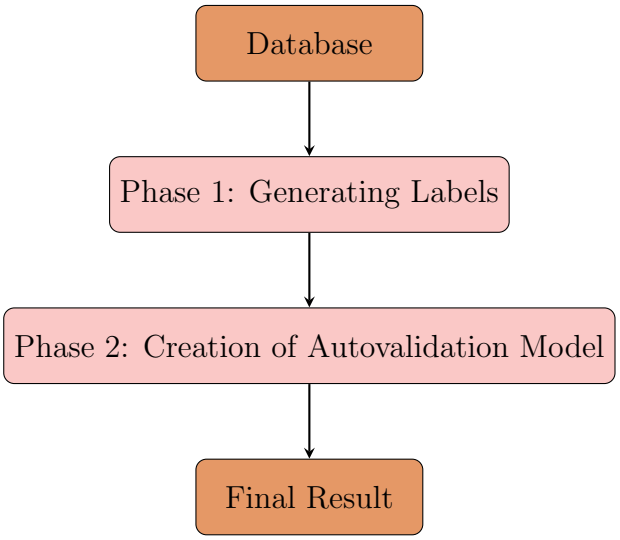


Figure 1.1: An overview of the general structure of the study.

Chapter 2

Theoretical Background

2.1 Blood Cells

The blood consists of plasma and cells, where the plasma constitutes 60% of the blood while white blood cells (WBCs), red blood cells (RBCs) and platelets together constitutes 40% of the blood. These blood cells are all formed in the bone marrow where they develop from stem cells into WBCs, RBCs or platelets. White blood cells can be divided into three main categories: lymphocytes, monocytes and granulocytes. The granulocytes can further be divided into neutrophils, eosinophils and basophils (9). The neutrophils can be seen in two stages, band neutrophil, where the nucleus is one intact band, and segmented neutrophil, where the nucleus has separated into several segments. These cells are displayed in a-f in figure 2.1. Mature red blood cells in mammals lacks nucleus, since this provides the cell with more room to store the oxygen-binding protein hemoglobin (9). Nucleated red blood cells called erythroblasts (10) and greatly enlarged platelets called giant thrombocytes (11) can also be present in the blood. These cells are displayed in g-h in figure 2.1. There could also be a wider range of maturation or variants of these cells present in the blood.

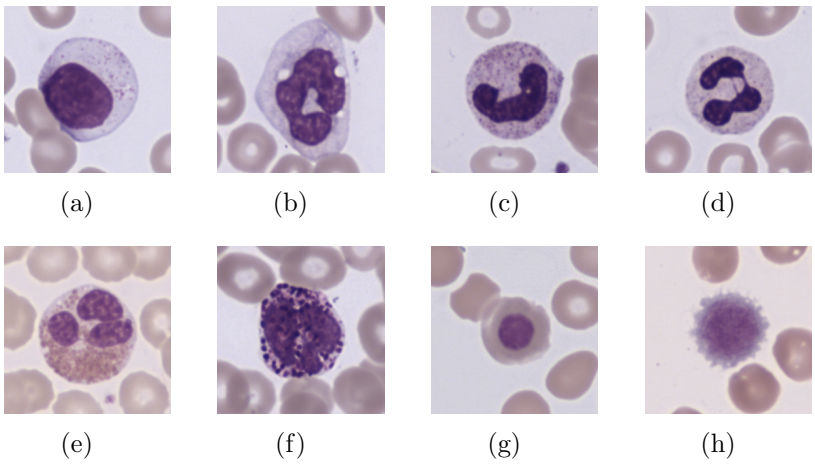


Figure 2.1: Examples of cell classes included in the dataset: (a) Lymphocyte, (b) Monocyte, (c) Band Neutrophil, (d) Segmented Neutrophil, (e) Eosinophil, (f) Basophil, (g) Erythroblast and (h) Giant Thrombocyte.

2.2 Artificial Neural Networks (ANN)

Artificial neural networks (ANN) is a branch of machine learning algorithms inspired by the human brain (12, p. 3). ANN's consists of layers of connected neurons, where the input layer and output layer is connected with one or more layers, called hidden layers. The idea is that the ANN should be able to take input data and, by propagating it through the network, learn to interpret and map this input into outputs (13, p. 2-3). Haykin (14, p. 24) uses the following definition for ANN's:

"A neural network is a massively parallel distributed processor made up of simple processing units that has a natural propensity for storing experimental knowledge and making it available for use. It resembles the brain in two aspects:

1. *Knowledge is acquired by the network from its environment through a learning process.*
2. *Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge."*

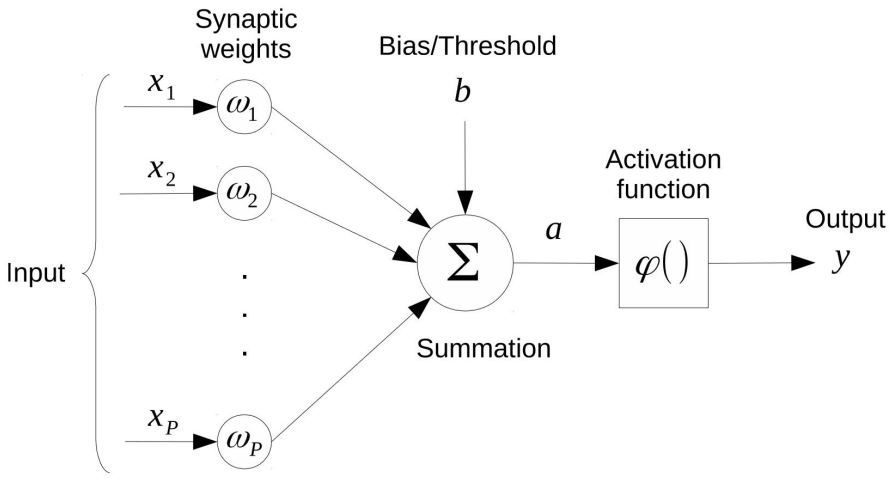


Figure 2.2: A neuron, which is the basic element of all ANN.

Figure 2.2 shows a neuron, which is the basic element of all ANN's. The output y is calculated according to equation (2.1). The neuron in the ANN calculates a weighted sum of it's input and sends a new output forward through the activation function,

$$\begin{aligned} \sum_{k=1}^{P} w_k x_k + b, \\ y = \varphi(a). \end{aligned} \tag{2.1}$$

By connecting several neurons with each other, a neural network is created.

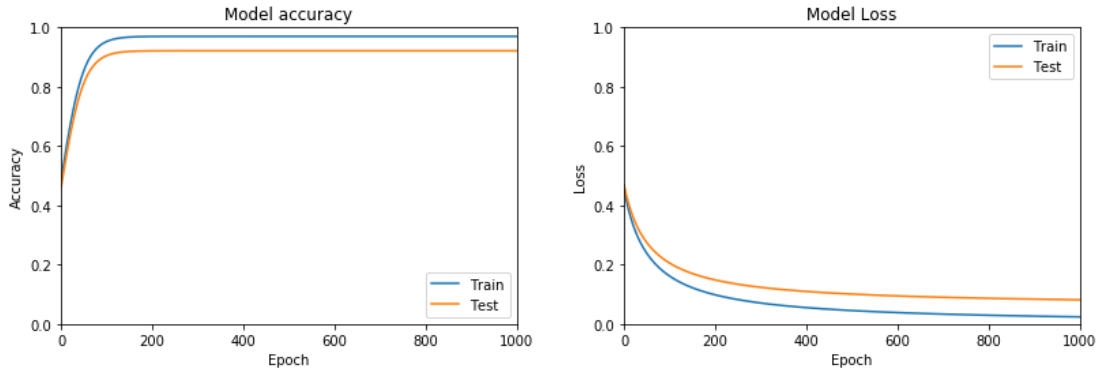
There are few guidelines on how to build the best-performing neural network architecture for a specific problem. Therefore using systematic experimentation is a common way to discover what architecture seems to perform well for that specific problem and set of data (15).

2.2.1 Training the Network

When training an ANN, the most common case is to use data (e.g. features) and corresponding labels. This type of training, i.e. with labels, is called supervised learning. The opposite, called unsupervised training, is training without access to labels and thus without answer of what is the correct solution (12, p. 3). Unsupervised training thus focuses on finding common patterns and trends in the data. During supervised training, the model computes the error of it's output. This is the difference between the output labels of the model compared to the input labels and is commonly referred to as loss. The aim of the training is to minimize the loss. This is achieved by using the computed error to modify the weights between the connected neurons in the network (12, pp. 23-24, 27) (16, p. 200). The accuracy is the proportion of the data which was predicted to the same label as the input label, therefore a high accuracy is desired.

2.2.2 Generalization, Overfitting and Underfitting

When working with neural networks it is important that the model performs well not only on the training data but also on new, previously unseen data. The measurement of the performance on this type of data is known as the generalization of the model (16, p. 108). In the case of this thesis, the training data consists of features from the cell images and their corresponding labels being the name of the cell class. After training the model, it is tested on previously unseen data, referred to as validation data. Since the model has not trained on the validation dataset, this test will provide an indication of the model's ability to generalize (16, p. 108)(12, pp. 44-45). When training an ANN using Keras, a deep learning library, a useful tool is available which splits the training dataset into a set for training and a set for testing. This tool allows the user to track the training-loss and the test-loss at the same time, where the test error is an approximation of the validation loss. An example of what these plots can look like is displayed in figure 2.3.



(a) (b)

Figure 2.3: Examples of the plots simultaneously describing the performance of the model on both training and test data: (a) An example of training and test accuracy; (b) An example of training and test loss.

To ensure that a model will be able to generalize well, it is of importance to keep the training loss small while keeping the gap between training and test loss small.

This is strongly related to the concepts of overfitting and underfitting. If the model cannot achieve a low enough training loss, the model is said to be underfitted. If the model has achieved a low enough training loss but the gap between the training loss and the test loss is too large, the model is said to be overfitted (16, pp. 109-110). An exemple of an overfitted model is shown in figure 2.4 and an illustration of the behavior of the loss and accuracy during overfitting is provided in figure 2.5. Note that these are two separate plots only illustrating the behaviour of the curves, generated from two different models. Overtraining is another common word for overfitting.

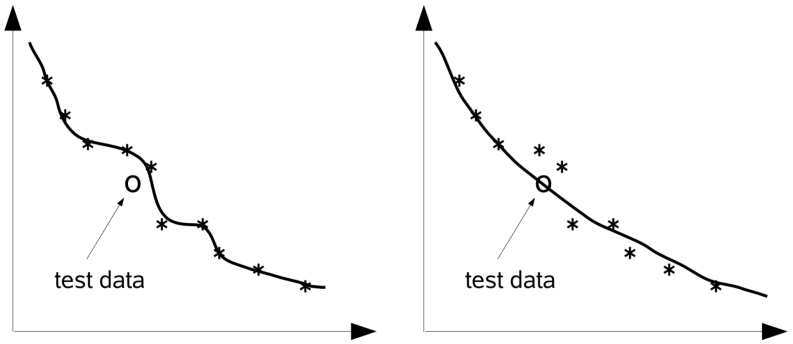
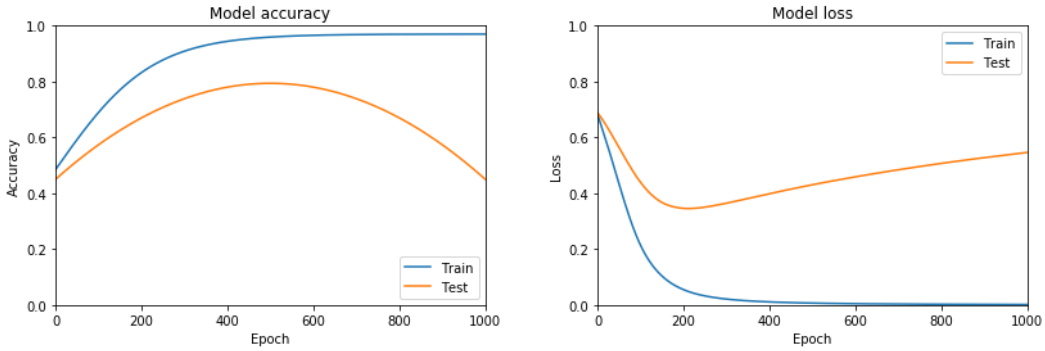


Figure 2.4: The left plot is an example of overfitting, while the right plot is generalizing well (12, p. 45).



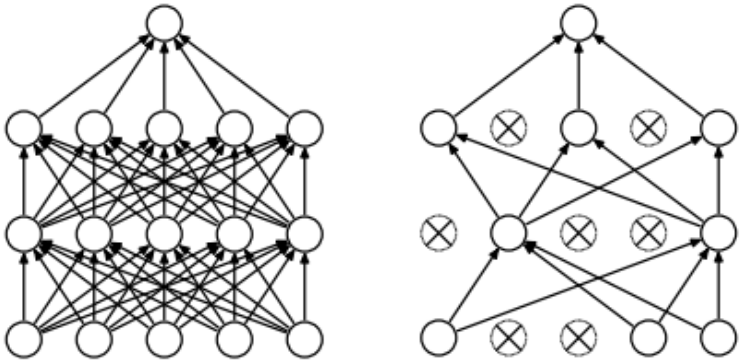
(a) (b)

Figure 2.5: An illustration of the behaviour of the loss and accuracy in an overfitted model: (a) Training and test accuracy at overfitting; (b) Training and test loss at overfitting.

2.2.3 Regularization

Regularization is a term occurring when talking about ways to combat the problem of overfitted models. Goodfellow, Bengio and Courville (16, p. 117) uses the following definition: *"Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not it's training error."* In figure 2.5(b), the overfitting occurs around epoch 200 when the two curves diverge. An easy way of avoiding overfitting the model is called "early stopping", where the technique is to simply stop training before overfitting occurs. The point where to stop can be found by finding the minima of the test loss curve (12, p. 50-51). Again,

note that the loss and accuracy curves in figure 2.5 are not generated from the same model and thus the minimum test loss does not occur at the same epoch as the maximum test accuracy.



(a) (b)

Figure 2.6: An illustration of the effect of applying dropout to the layers of a network: (a) A network before applying dropout; (b) A network after applying dropout.

Another regularization technique is called "dropout regularization". This technique works by temporarily removing nodes in a network. When defining dropout in a network, a dropout probability p is defined. This probability is the probability of staying, so $p = 0.3$ will result in the node being temporarily removed in 70% of the times, while $p = 0.8$ will result in the node being temporarily removed in 20% of the times. This will prevent the nodes in the network from being fully dependent on each other, reducing the network's ability of training on very complex patterns (which is characteristic for noise) and instead encouraging the network to be good at generalizing on the data. Dropout can be present in all layers of a network, except in the final output layer. The technique is only used during training to prevent overfitting, so while performing validation tests all nodes will be kept as usual (12, p. 51). An image from (17) showing the effect of applying dropout on the layers in a network can be seen in figure 2.6.

2.3 Self-Organizing Feature Maps (SOFM)

Self-Organizing Feature Map is a type of ANN which trains unsupervised (without labels) and uses competitive learning to improve its result (12, p. 122). It is presented as a grid with nodes representing the input data visually.

The process of training a SOFM is iterative competitive learning. Every iteration starts with choosing a random input vector and finding the node on the grid with a weight vector most alike the input vector, the winning node. This is called the best matching unit (BMU). The next step is to move the BMU closer in space to the input vector. After this, the neighbours of the BMU are identified and moved closer to the input vector as well, but not as much as the BMU. The last step of the iteration is to update the learning rate, before the process starts over with a new

random input vector and its BMU (18, p. 5).

When the training is finished, the result is a grid with nodes representing the input data. Data points which are similar have moved close to each other in the grid, while data points which are different have moved further away from each other. An image from (12, p. 134) showing an example of a SOFM can be seen in figure 2.7.

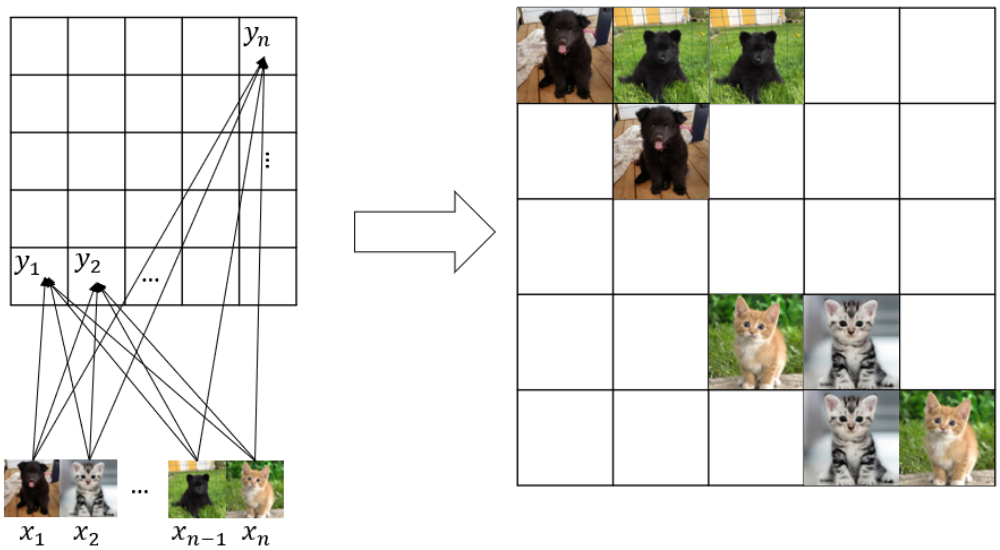


Figure 2.7: An example of a SOFM where the input $\boldsymbol{x}$ *is mapped on a grid with nodes* $\boldsymbol{y}$.

When training a machine learning process it is almost always important to pre-process the data by normalizing it, to avoid some dimensions to dominate. This is especially important for SOFM's, which trains and performs better with data scaled between zero and one (18, p. 4).

2.4 Autoencoder

An autoencoder is a special kind of ANN which has the same number of outputs as inputs. This is important, since the main idea of an autoencoder is that the output should predict the input. The model can have several hidden layers. The middle hidden layer, called the bottleneck, always has a smaller size than the input layer. The fact that it is smaller means that the model have to represent the input nodes in a lower dimensionality, namely the dimensionality of the bottleneck. The part of the model before the bottleneck is called the encoder and is used for transforming data of high-dimensional space into features of low-dimensional space. The part after the bottleneck is called the decoder, and is thus used to reconstruct the original high-dimensional data from the features of low-dimensional space. The autoencoder do not need any labels, since its answer is the input data itself (12, p. 112) (19, p. 121) (20, p. 3). The accuracy of the autoencoder is calculated just as for a regular ANN, by comparing the predicted labels with the validation labels. For the autoencoder, this means that the accuracy would be the amount of reconstructed features which are identical to the input data. Hence, a high accuracy would be expected only for identical output, and not similar, and therefore it does not make sense to use this as a reflection of the model's performance when a similar output is

sufficient. A more useful value to measure and minimize is the loss, since it reflects on the potential differences between the reconstructed output of the model and the original input (12, p. 113).

The image in figure 2.8, from (12, p. 112) shows an example of a small autoencoder.

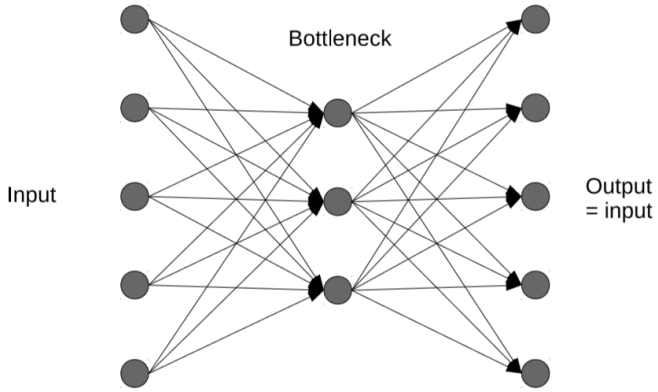


Figure 2.8: An example of a small autoencoder.

2.4.1 Autoencoder for Dimensionality Reduction

When an autoencoder has been trained such that the loss has become small, i.e. the difference between the reconstructed data and the original data is very low, the autoencoder can be used for dimensionality reduction. Since the first part of the autoencoder encodes the input data into a lower-dimensionality representation, this means that the important information from the input data is represented in the bottleneck layer. A representation of the data in a lower dimensionality will thereby be available simply by saving the output from the bottleneck layer of the autoencoder (21).

2.4.2 Autoencoder for Anomaly Detection

Using dimensionality reduction, such as the autoencoder, as a way of detecting anomaly is a common method. It is based on the assumption that the features of normal data correlate. This means that it is possible to find a subspace which can describe normal data. Data which is not normal would give a large reconstruction loss and can therefore be detected (22, p. 1). The autoencoder is therefore only trained on what is considered normal data, and will thereby become very good at reconstructing it. The large reconstruction loss will appear when the autoencoder is asked to reconstruct anomaly data since the autoencoder has not trained on this. When running a dataset consisting of both normal and anomaly data through the trained autoencoder, a reconstructed data point with a low reconstruction loss will indicate the data point being normal while a data point with a high reconstruction loss will indicate the data point being anomaly (23) (20, p. 4).

2.5 Pre-Processing of Data

Before inputting the data into an ANN, it is common to pre-process it. Pre-processing can be done in various ways and to varying degrees, depending on the data itself (12, p. 66). The goal is to improve the result by having a better input. One way of pre-processing data is to reduce the dimensionality, a process which was mentioned earlier in the context of autoencoder. Another way of pre-processing data is to normalize it. There are several ways of normalizing data, where using L1- and L2-norm are two ways. A norm is in short the magnitude of a vector (24), which can be used in normalization by dividing each component of the vector with the norm. Doing this with several vectors in a set gives them the the same length, but different directions, which makes them easier to compare (25). The equations for deriving L1-norm and L2-norm are shown in equation (2.2) and (2.3), where a and b are elements in the vector x. The norms are calculated by

$$L1 - norm : ||x|| = |a| + |b|\,, \tag{2.2}$$

$$L2 - norm : \|x\| = \sqrt{|a|^2 + |b|^2}. \tag{2.3}$$

Chapter 3

Methodology

3.1 General Methodology

A previously mentioned, the study was divided into two phases. The general idea of the study was to, as a first phase, use different methods to generate training and validation labels to the cell images with the value "easy to classify" or "difficult to classify". The methods were analyzed and the most suitable label vectors for training and validation data was determined. The chosen vectors were used for training networks in predicting easy and difficult cells, which implied moving into the second phase of the study.

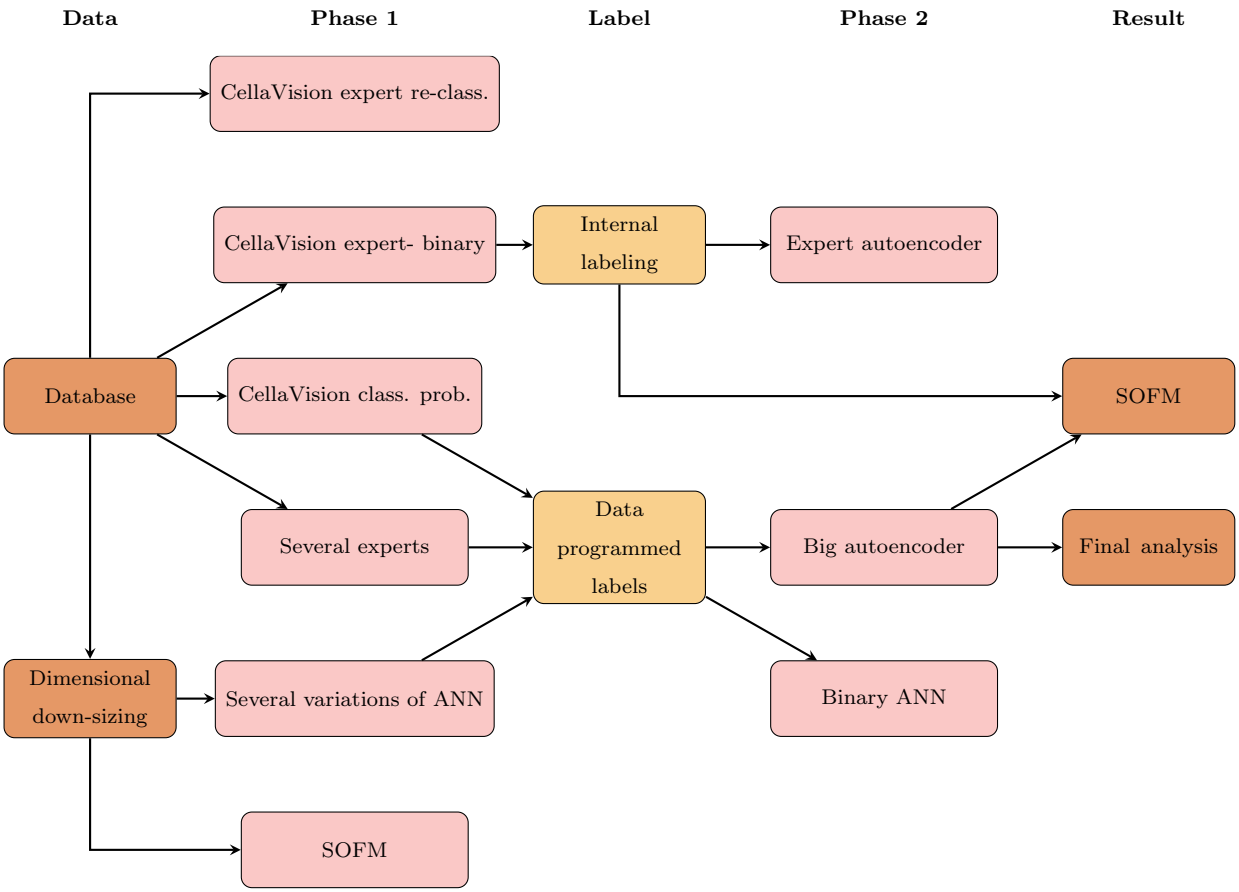


Figure 3.1: A flow chart of all sections in the study, describing how they connect with each other.

The second phase included testing of two methods to build a general model in finding easy and difficult cells. To validate the results of the methods, the validation labels as well as the positioning of the cells in an SOFM was used to compare and confirm the classification. An overview of the methodology can be seen in figure 3.1.

3.1.1 Data

The data used were 122 227 feature vectors from CellaVision's system, generated from images of cells. These originated from a database at CellaVision and were raw images in bmp format and of size 256x256 pixels. Ten randomly chosen images from this dataset are displayed in figure 3.2. The 122 227 images were run through CellaVision's classifying neural network and their features were extracted from a hidden layer just before the last neuron layer. This hidden layer had 13 800 nodes, which resulted in every image being represented by a feature vector of 13 800 elements. Therefore, the networks in this study were easily trained on the classification of cells since the distinctive features for the cells were already found by CellaVision's network.

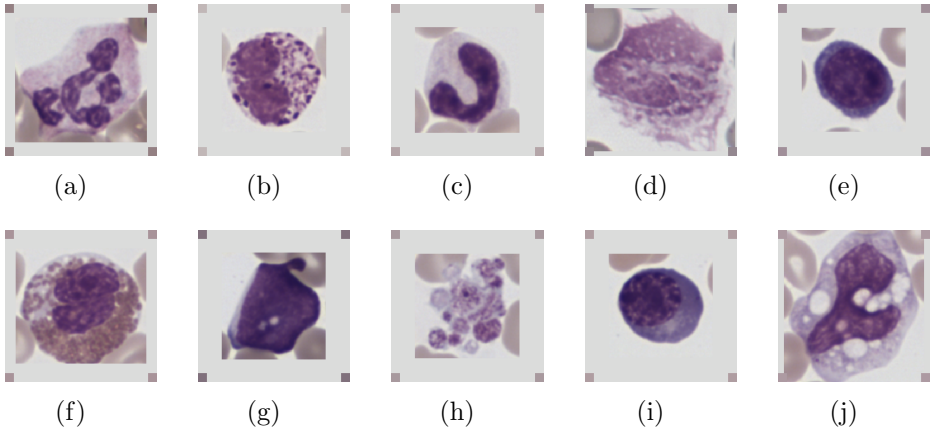


Figure 3.2: Ten randomly chosen cell images from the dataset. The images were labeled as following: (a) Segmented Neutrophil, (b) Basophil, (c) Band Neutrophil, (d) Smudge Cell, (e) Lymphocyte, (f) Eosinophil, (g) Blast Cell, (h) Thrombocyte Aggregation, (i) Erythroblast and (j) Monocyte.

For the two different phases of this project, in figure 1.1 referred to as "Phase 1" and "Phase 2", different types of data was required. During the first phase the task was to perform different kinds of classifications, thereby data consisting of cell features and corresponding labels stating the cell class was needed. During the second phase, the focus was instead to detect easy and difficult cells. Thereby cell features on the same format as in the first phase was of interest while the corresponding labels should instead be binary, stating if the cell was easy or difficult. Because of this, two different kinds of data labels were generated - one including cell class labels and one with binary labels.

In the original dataset, the cells were labeled by a number between 0 - 88. It turned out that this wide range of numbers were derived from different standards of how to label cells at CellaVision during different periods of time. The number of unique

Original label	New label	Amount	Cell class
0		11	Unidentified
1	0	99 455	Segmented neutrophil
2	1	11 864	Eosinophil
3	2	5 160	Basophil
4	3	33 040	Lymphocyte
5	4	29 071	Monocyte
6	5	8 676	Band neutrophil
8	6	2 174	Promyelocyte
9	7	8 880	Myelocyte
10	8	5 079	Metamyelocyte
11	9	11 219	Blast
12		55	Prolymphocyte
13	10	2 467	Plasma cell
14		17	Large granular lymphocyte
19		157	Promonocyte
21	11	18 816	Smudge cell
23	12	8 165	Erythroblast
24	13	8 319	Artefact
25	14	2 447	Giant thrombocyte
26		95	Megakaryocyte
27	15	2 104	Not classed
29	16	2 544	Thrombocyte aggregation
73		237	Other
85	17	9 855	Reactive lymphocyte
86	18	11 147	Abnormal lymphocyte
87	19	13 959	Large thrombocyte

Table 3.1: Translation of numbers indicating cell labels.

labels used were in fact 26, which can be seen in table 3.1 column 1. To make an efficient classifier of these different labels, it was required to translate these numbers into a range of 0 - 25 since this allowed the ANN to have 26 output nodes instead of 88.

Furthermore, the network of CellaVision was currently able to detect 19 types of cell classes, these are displayed in table 3.2.

Label	Name
0	Segmented Neutrophil
1	Band Neutrophil
2	Eosinophil
3	Basophil
4	Lymphocyte
5	Monocyte
6	Reactive Lymphocyte
7	Abnormal Lymphocyte
8	Promyelocyte
9	Myelocyte
10	Metamyelocyte
11	Blast
12	PlasmaCell
13	Erythroblast
14	Large Thrombocyte
15	Giant Thrombocyte
16	Smudge Cell
17	Artefact
18	Thrombocyte Aggregation

Table 3.2: Cells detectable by CellaVision's system.

When compared to 3.1 this means that there were 7 cell classes present in the dataset which could not be detected by CellaVision's system. These were:

- Unidentified
- Prolymphocyte
- Large granular lymphocyte
- Promonocyte
- Megakaryocyte
- Not classed
- Other

The class "not classed" included cells which the morphology expert had found extra difficult to class, and therefore these cells could be included in the binary dataset used during the second phase as examples of difficult cells. The cells from "not classed" were labeled as difficult cells. Since it would be interesting to see how the networks would class these previously not classed cells, they were also kept in the classification dataset under the label "not classed". The six other classes were removed from both datasets. The classes in the classification dataset were given new class labels of 0 - 19. These can be seen in table 3.1 column 2.

The data now consisted of 122 227 unique cell images which had been classed by experts 2 to 15 times each. This equals a dataset of 294 441 unique classifications of the 122 227 unique cells. The dataset was split into 133 sets, each containing 919 unique cells and all their different classes. The sets were divided into training and validation data, where the training data consisted of set 0 - 99 (75%) and the validation data consisted of set 100 - 132 (25%).

3.1.2 Building Neural Networks

When building the networks in this thesis, systematic experimentation with i.e. depth, number of neurons in the hidden layers, dropout rate, optimization techniques and activation functions were used to find the best performing architecture in terms of the lowest loss and highest accuracy.

3.1.3 Dimensionality Reduction

To be able to train the networks fast enough, it was necessary to reduce the dimensionality of the features extracted from CellaVision's network. In order to do this two methods were compared using literature studies; principal component analysis, or PCA, and the use of an autoencoder. It was decided to use an autoencoder since more information remains in the data compared to PCA and since this method gives higher accuracy for larger data sets (26).

After implementing an autoencoder, it was of interest to test the performance of an ANN trained on the original feature-dimensions and compare it with an ANN

Table 3.3: Architectures of the ANN's used to evaluate the dimensionality reduction.

ANN for full-sized data:

Dense	Units:	1 380	Activation fun.:	ReLU	Input dim.:	13 800
Dropout:	50%					
Dense	Units:	800	Activation fun.:	ReLU		
Dropout:	50%					
Dense	Units:	200	Activation fun.:	ReLU		
Dropout:	50%					
Dense	Units:	20	Activation fun.:	Softmax		

ANN for dimensionality reduced data:

Dense	Units:	80	Activation fun.:	ReLU	Input dim.:	100
Dropout:	50%					
Dense	Units:	60	Activation fun.:	ReLU		
Dropout:	50%					
Dense	Units:	40	Activation fun.:	ReLU		
Dropout:	50%					
Dense	Units:	20	Activation fun.:	Softmax		

trained on the dimensionality-reduced feature vectors generated by the autoencoder. This comparison would show how the performance of the ANN would be affected when reducing the dimensionality of the features. This test would thereby provide an indication whether this reduced dataset is suitable to perform the further classification tests on. An ANN was trained on a set of 22 541 images with the original feature dimension of 13 800 features, while another ANN was trained on the same set of images but where the autoencoder had been used for dimensionality reduction. Both datasets were scaled between 0 and 1 and L2-normalized over the feature vectors. Since the dimensionality of the features differed, the architecture of the two networks could not be identical but of similar proportions. The architectures of both networks were chosen to be what was considered as the best performing architecture found for these datasets. Both networks were trained on their data during 500 epochs and the accuracy, loss and resulting classification was observed.

3.1.3.1 Visual Evaluation Using SOFM's

The result of reducing dimensionality was tested using SOFM's. Two different SOFM's with alike architectures were built. It was decided to let the models train during 10 epochs, which took 12 hours for the SOFM training on the original data and one hour for the SOFM trained on dimensionality reduced data. For further details of how the SOFM's were built, see section 3.2.6.

3.2 Phase 1 - Generating Labels

Different methods of generating labels for training and validation data were tested and evaluated. A flowchart describing the relation between the different sections in phase 1 can be found in figure 3.3. The labels were binary, consisting of 0's and 1's, where a 0 indicated that the cell was easy while a 1 indicated that the cell was difficult.

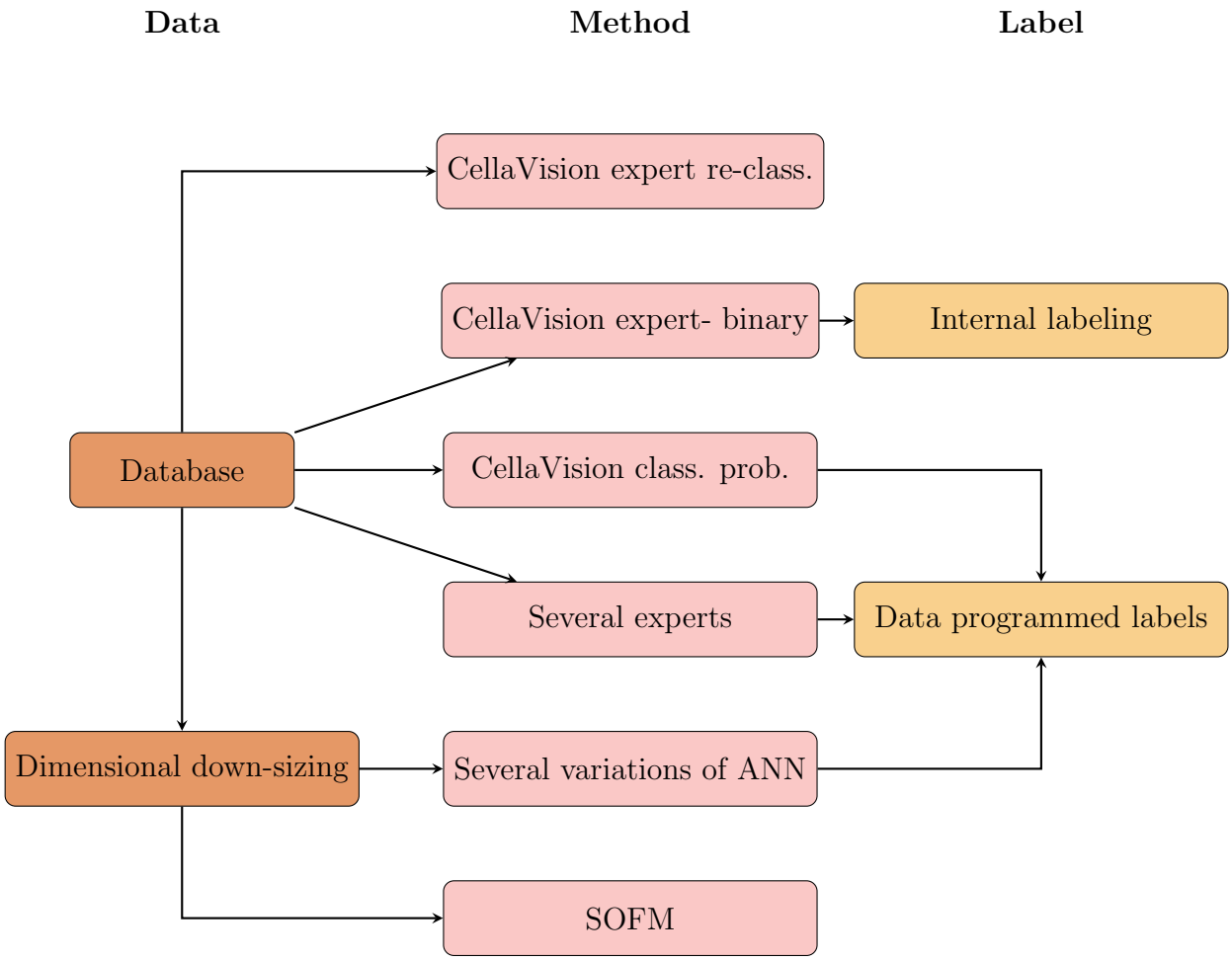


Figure 3.3: A flow chart showing an overview of the sections in phase 1: Generating Labels.

3.2.1 Several Experts

In this method, CellaVision's data classified to cell classes by morphology experts was used to generate labels indicating if the cell was easy or difficult to class. The data consisted of class labels from several experts which had classed the same cell image. In the case that all of the experts had chosen the same class label for the cell, the binary label was set to easy to classify. In the case that the experts had disagreed, in other words chosen different classes for the same cell, the binary label was set to difficult to classify.

3.2.2 Several Variations of ANN

Three different ANN's were created for the aim of generating 1) a binary label vector for the validation dataset and 2) binary labels for the training dataset. The task for the networks were to class cells into cell classes. Their results was compared, enabling the creation of binary labels describing cells as easy if all networks agreed on a cell class or difficult if they has classed the cell differently. The networks were built by composing different architectures of the hidden layers. These architectures were tested to perform equally well, while processing the data differently because of the differences in architecture, activation functions, dropout rate etc. The different ANN's were trained on different parts of the datasets for further variation of the networks. Since the networks were to be trained on data of equal dimensions and perform classifications of equal numbers of categories, the number of input nodes, input dimension and output nodes were the same for all networks. The architectures of the ANN's are described in figure 3.4.

3.2.2.1 Generating Binary Labels for Validation Data

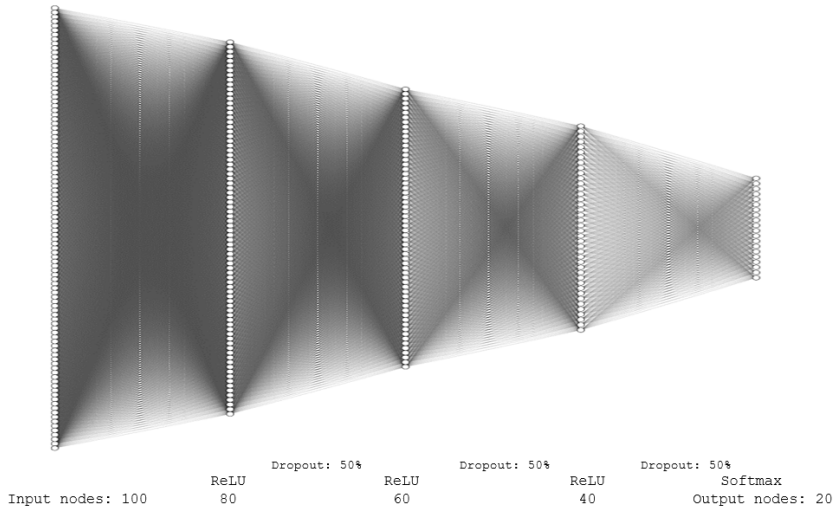
The three different ANN's, whose architecture is described in figure 3.4, were first trained on different parts of the training data in three rounds while always validating on the same validation set. How this training and validation were performed is described in detail in table 3.4. This layout thereby resulted in 9 different classifications of the validation set, one for each ANN and round. These classifications were compared by checking for which cells the networks had agreed by classing the same class, and for which cells the networks disagreed. The cells which the networks agreed on were labeled as "easy" while the cells where the networks disagreed were labeled as "difficult".

3.2.2.2 Generating Binary Labels for Training - Cross Validation

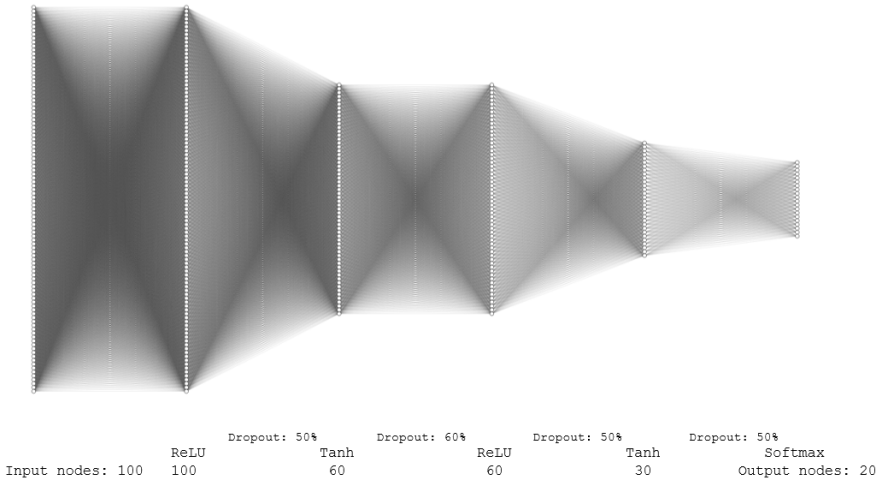
The same network architectures were then used for performing a cross validation for the training set. The layout of the cross validation is described in table 3.4. The cross validation thereby resulted in each network classing all cells in the training dataset twice, resulting in 6 unique classifications of the training dataset while avoiding using the same data for both training and validation in the same round. From this data, the classifications were compared and each cell was labeled as easy or difficult. This labeling were performed by the same approach as for the validation dataset, where the cells which the networks agreed on were labeled as "easy" while the cells there the networks disagreed were labeled as "difficult".

Table 3.4: The layout of the training and cross validation performed.

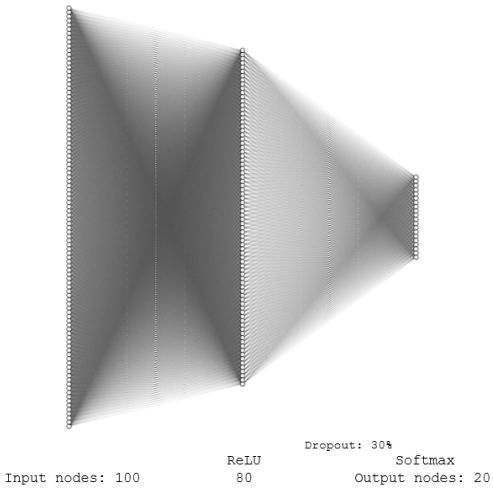
		Round 1	**Round 2**	**Round 3**
Training	Trained on:	10 - 39	40 - 69	70 - 99
	Validated on:	100 - 132	100 - 132	100 - 132
Cross	Trained on:	10 - 39	40 - 69	70 - 99
Validation	Validated on:	40 - 99	10 - 39, 70 - 99	10 - 69



(a) Network architecture of ANN1



(b) Network architecture of ANN2



(c) Network architecture of ANN3

Figure 3.4: Illustration of the architectures of the three different ANN's used for image classification in the ANN-part of phase 1.

3.2.3 CellaVision Expert- Binary Classification

Images of the data were shown to a morphology expert who marked the cell "easy to classify" or "difficult to classify" according to it's experience of classing cells.

3.2.4 CellaVision Expert Re-Classification

Images were run through CellaVision's network to be classified. After this, a morphology expert looked at the result. If the expert wanted to re-class the cell into a different class, it was labeled "difficult". Otherwise, it was labeled "easy".

3.2.5 CellaVision Classification Probabilities

When cell images are run through CellaVision's network, they come out with probabilities for each cell class. This was used in a way that if the highest probability (and thereby the class which the cell was classified into) was below a certain value, 0.999, the cell was classified as difficult. If the probability was equal to or above this certain value, it was classified as easy.

3.2.6 SOFM

As input to the SOFM the feature vector's of cell images which had been run through CellaVision's network and then reduced the dimensionality of was used. The vectors were scaled to a 0 - 1 number and L2-normalized over the feature vectors before put into the SOFM.

The SOFM was created using NeuPy, which is a python library for building different neural networks, such as SOFM, using Tensorflow as a computational back-end (27).

The network was trained on 199 574 images, corresponding to training on set 2 - 101. After the training was completed, 2 757 validation images (set 130 - 132) were run through the network, which predicted the position for these images in a grid.

The result of the clustering was plotted in matrices where a data point on a certain node was represented with a square of a color corresponding to the cell class of that particular data point. The matrix was configured such as if several cells (data points) were located on the same node, the square corresponding to that node was divided into smaller squares together representing all the cells placed on that particular node, appearing in the same order as in which they were placed on the node. This plotting is illustrated in figure 3.5.

The result of the SOFM was evaluated by visual analysis, meaning how easily clusters could be found and defined in the grid.

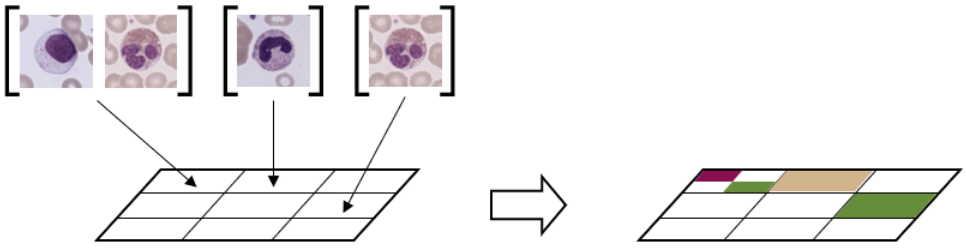


Figure 3.5: An illustration of how a SOFM was plotted.

3.3 Labeling Data

The possible labels were evaluated for the different applications later in the project. It was found in literature that internal labeling is more desirable compared to labels made from data programming if it is possible considering time and resources (28). For the methods autoencoder and binary ANN in phase 2, labels were needed for both training and validation. For the SOFM method, labels were only needed for the validation data.

To evaluate the different methods, the accuracy, sensitivity, specificity and miss-rate was derived for each resulting label vector. The equations used are described in (3.1). *Positives* are the cells which are labeled difficult in the data set, and *negatives* are the ones labeled easy. *True positives* are the cells which are difficult according to the validation label and also labeled as difficult by the method. *False positives* are cells which are easy according to the validation label, but labeled as difficult by the method. In the same way, *true negatives* are labeled as easy both according to the validation labels and the method, and *false negatives* are labeled as difficult by the validation labels but as easy by the method. A false negative occurs when the method misses to identify a difficult cell, this behaviour can be measured using a miss-rate (29, p. 2-3).

$$
\begin{aligned}
Accuracy &= \frac{True\ positives\ +\ True\ negatives}{Positives\ +\ negatives} \\
Sensitivity &= \frac{True\ positives}{Positives} \\
Specificity &= \frac{True\ negatives}{Negatives} \\
Miss-rate &= \frac{False\ negatives}{Positives}
\end{aligned}
\tag{3.1}
$$

3.3.1 Internal Labeling

It was possible to generate labels by internal labeling for a small part of the dataset, which would be used for the SOFM and for verification of the autoencoder. There were two possible ways to generate these, the "CellaVision expert- binary classification" and "CellaVision expert re-classing" from phase 1, two methods which were made only from an expert without adjustments or processing afterwards. Both alternatives were used as the answer for the other methods and the results were compared by calculating the accuracy, sensitivity and specificity. This comparison was done for set 132 including 919 images. It was decided to use "CellaVision expert- binary classification" as validation labels. The labels were created for set 130 - 132, thus for 2 757 cell images.

3.3.2 Data Programmed Labels

The labels for the binary ANN and the autoencoder was required to be generated in the same way for both training and validation data. Since this covers a large amount of data, using internal labeling by CellaVision's expert was ruled out due to lack of resources. Instead, it was decided to use data programmed labels.

To measure the quality of the methods "Several experts", "Several variations of ANN" and "CellaVision classification probabilities", the accuracy, sensitivity, specificity and miss-rate were derived using "CellaVision expert- binary classification" as the answer. The parameters were also derived for the merge of the three methods where the methods had to agree on a cell being easy for it to be labeled as easy, otherwise it was labeled as difficult. This quality measurement was done for set 132 including 919 images. The binary labels were generated for set 0 - 132, thus for 122 227 cell images.

3.4 Phase 2 - Autovalidation Model

A flowchart describing the relation between the different sections in phase 2 can be found in figure 3.6.

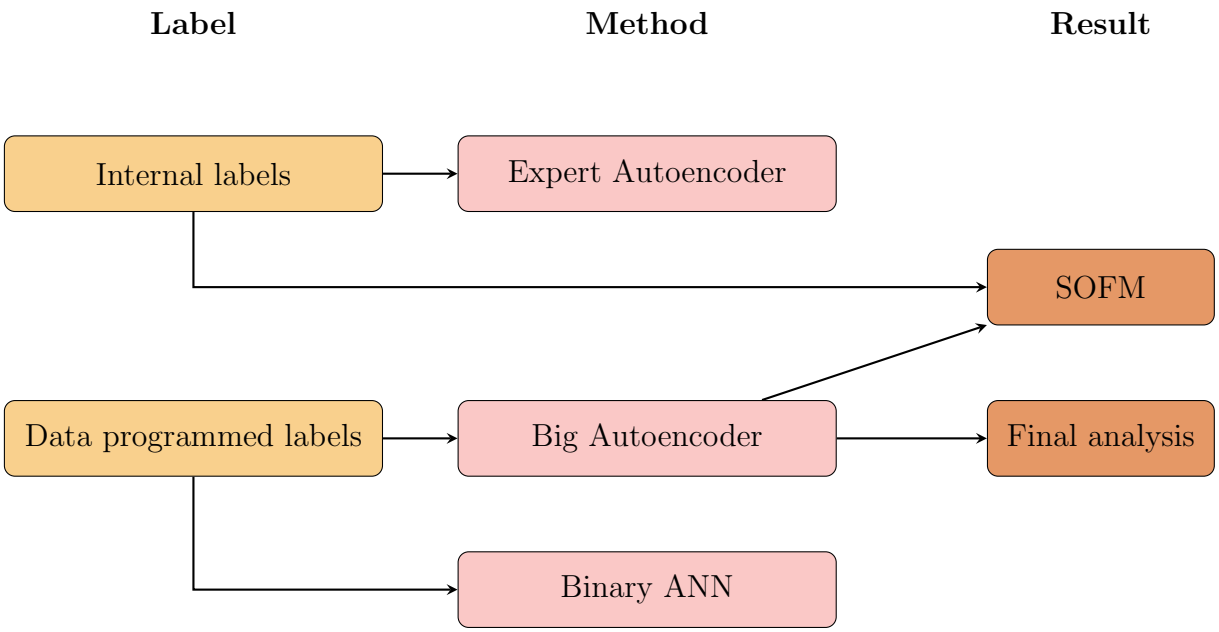


Figure 3.6: A flow chart showing an overview of the sections in phase 2: Autovalidation model.

3.4.1 Binary ANN

An ANN for binary classification was built for the purpose of classifying data into easy and difficult. Several different architectures were tested for the model and the best performing architecture, displayed in figure 3.7, was chosen. For both training and validation, labels from section 3.3.2 was used. The network trained 2 000 epochs on the training data and made a prediction of the validation data.

At this point the data was highly unbalanced since difficult cells accounted for only 18.2% of the training data and 18.0% of the validation data. To test what impact this imbalance had on the model, another test- and validation dataset was generated consisting of 50% easy cells and 50% difficult cells, i.e. 30 124 cells for training and 10 979 cells for validation. The architecture of the network was kept the same. The network trained 2 000 epochs on the balanced training data and made a prediction of the balanced validation data.

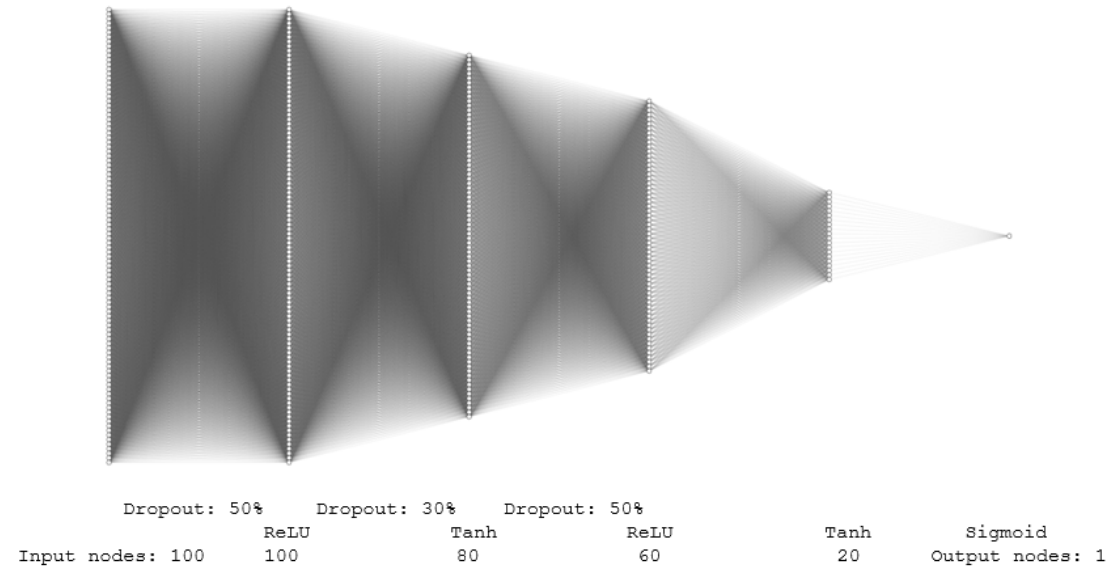


Figure 3.7: The architecture of the binary ANN used for classifying images as easy or difficult in the ANN used during phase 2.

3.4.2 Autoencoder

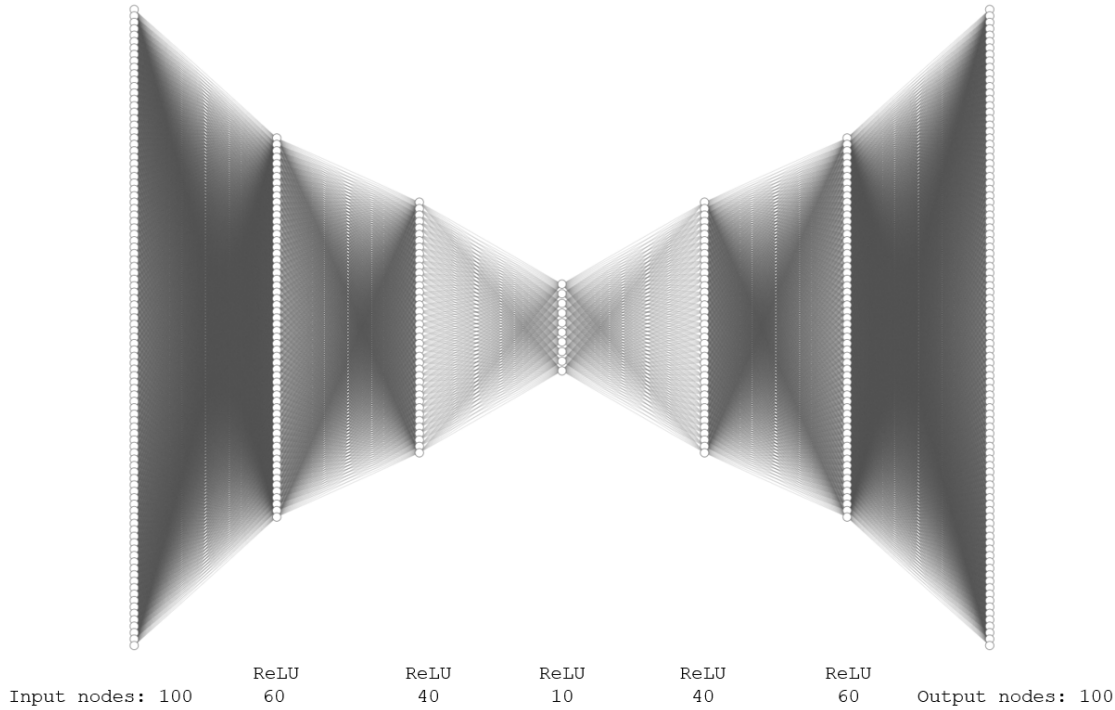


Figure 3.8: The architecture of the autoencoder used for anomaly detection.

An autoencoder was built to train on only easy classified cells from the merged data-programmed labels, see section 3.3.2. This autoencoder was called "Big-AE". The cells classified as "not classed" were removed from the training data since this cell class in itself indicated that the cell is difficult. The training dataset consisted of

cells from set 0 - 99, including 64 644 easy cells. The architecture of the autoencoder were chosen by systematic experimentation of the hyper-parameters and finally deciding on the model which demonstrated the lowest loss for a certain number of epochs. The architecture is described in figure 3.8.

The output of the autoencoder was a vector consisting of the recreation-errors per feature of each image in the validation dataset. L1- and L2-normalization was tested and the results were compared. L2-normalization was chosen as the best result and the normalized error vector was summed into a total recreation-error for each image. All the recreation-errors of the validation images were scaled between 0 - 1 and compared with the validation labels. This showed which cells in the validation dataset the autoencoder could recreate well i.e. with a low error, and which cells the autoencoder could not recreate so well i.e. with a high error. The sensitivity, specificity, miss-rate, accuracy and number of found difficult cells were calculated. Using these measurements of the model, several different autoencoders were trained and the best performing autoencoder was chosen.

The measurements were calculated for different delimiter values, and an optimal value was chosen. Recreation-errors below this value was converted into 0 (labeling the cell as easy) while errors above it were converted into 1 (labeling the cell as difficult).

A second autoencoder was built using identical architecture as the first one, see figure 3.8. This autoencoder was trained on the easy cells according to the internal labels generated in section 3.3.1, and thereby called "Expert-AE". This training data consisted of set 130 and 131, thus 1 684 easy cells. Set 132 was used for validation. The output was normalized and measurement were calculated as for the first autoencoder.

The results of the two autoencoders were compared.

3.5 Analyzing Phase 2

The binary ANN and the autoencoder resulted in a label vector of zeros and ones representing the easy and difficult classed cells. These vectors were compared to the validation label vector generated for the binary ANN and the autoencoder, see section 3.3.2. From this comparison, several parameters were calculated - the accuracy, sensitivity, specificity and miss-rate of this result. Confusion matrices were generated for both methods for comparison.

Several SOFM's were built as described in 3.2.6 and used to visualize the validation labels, see section 3.3.1, and the labels from the autoencoder. The SOFM's were trained on data features with reduced dimensionality from set 2 - 102 for 500 epochs and predicted the data from set 130 - 132. The different nodes were colorized according to cell class, see figure 3.5, and marked by coloring half of the cell's corresponding square black if they had been classified as difficult by the labels. This way of marking the cells are illustrated in figure 3.9. By being able to visualize

where the difficult cells were located in the grid of cell clusters, the autoencoder's result could be compared to the validation labels and thereby better understood.

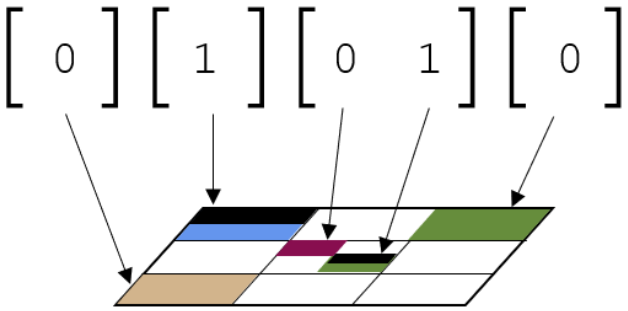


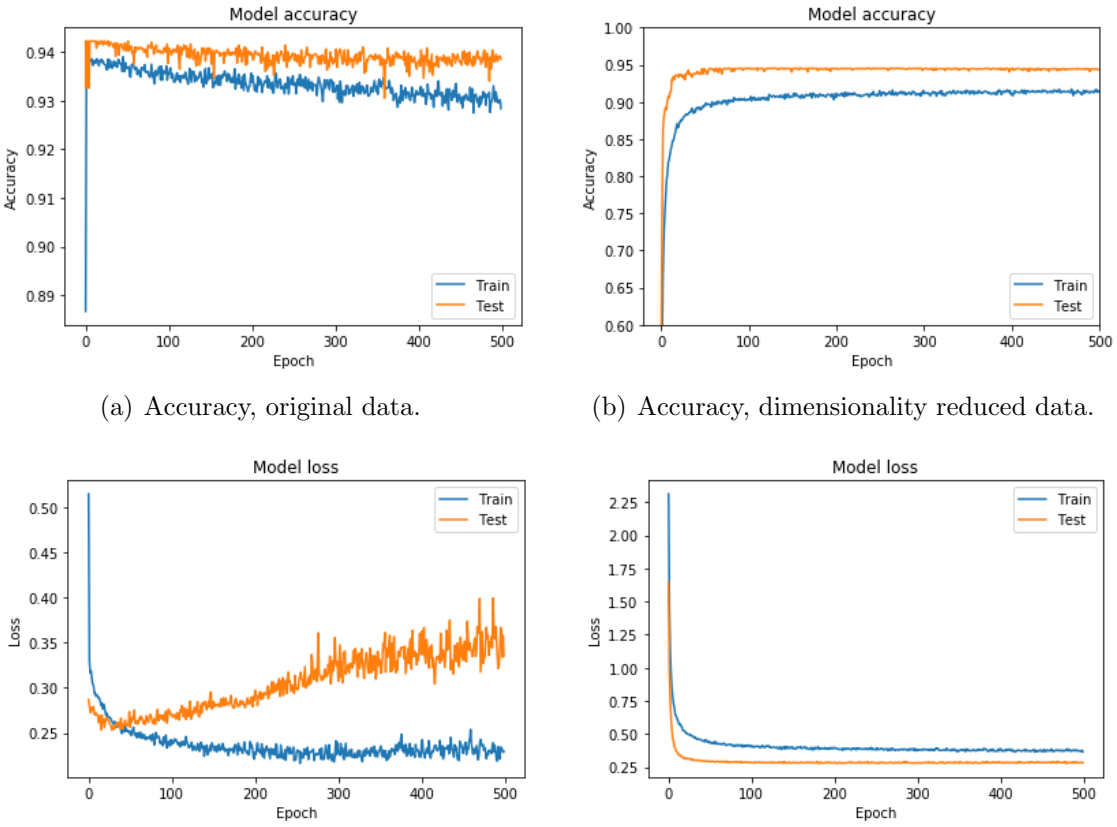
Figure 3.9: An illustration of how a difficult cell was marked by coloring half of the cell's corresponding square black. The numbers in the brackets are the labels, 0 *meaning easy and* 1 *meaning difficult.*

Chapter 4

Results

4.1 Autoencoder to Reduce the Dimensionality

An ANN was trained both on data with reduced dimensinality and on the original data. This resulted in plots of the accuracy and loss 4.1 and confusion matrices 4.2. Two SOFM's were trained on 9 190 cell images each during 10 epochs and displayed in figure 4.3 where 4.3(a) was trained on dimensionality reduced data to 100 features while 4.3(b) was trained on the original data of 13 800 features.



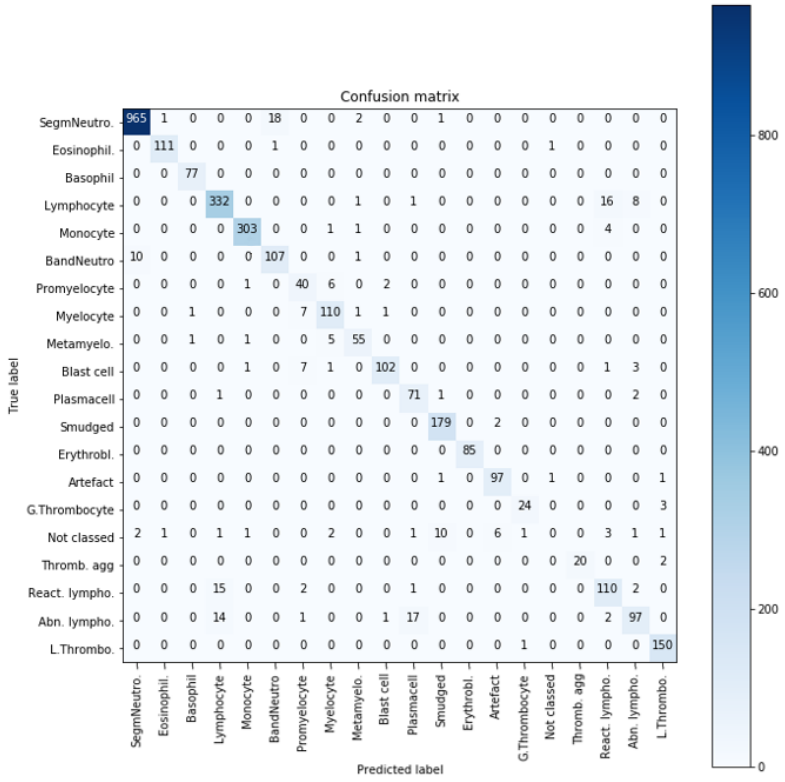
(a) Accuracy, original data.

(b) Accuracy, dimensionality reduced data.

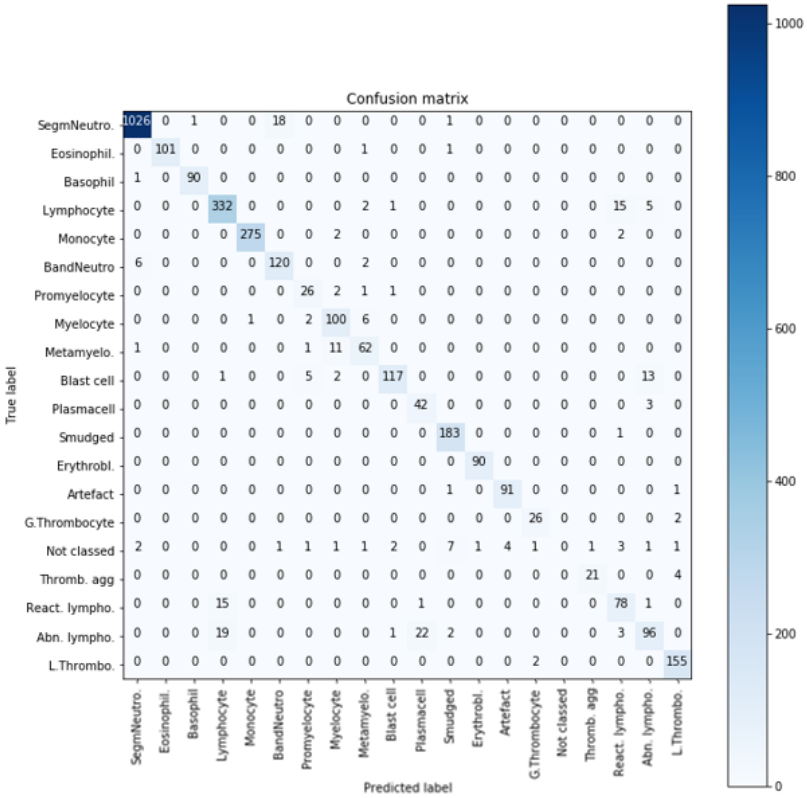
(c) Loss, original data

(d) Loss, dimensionality reduced data.

Figure 4.1: Plots comparing the accuracy and loss for the ANN trained on original data versus dimensionality reduced data after 500 *epochs.*

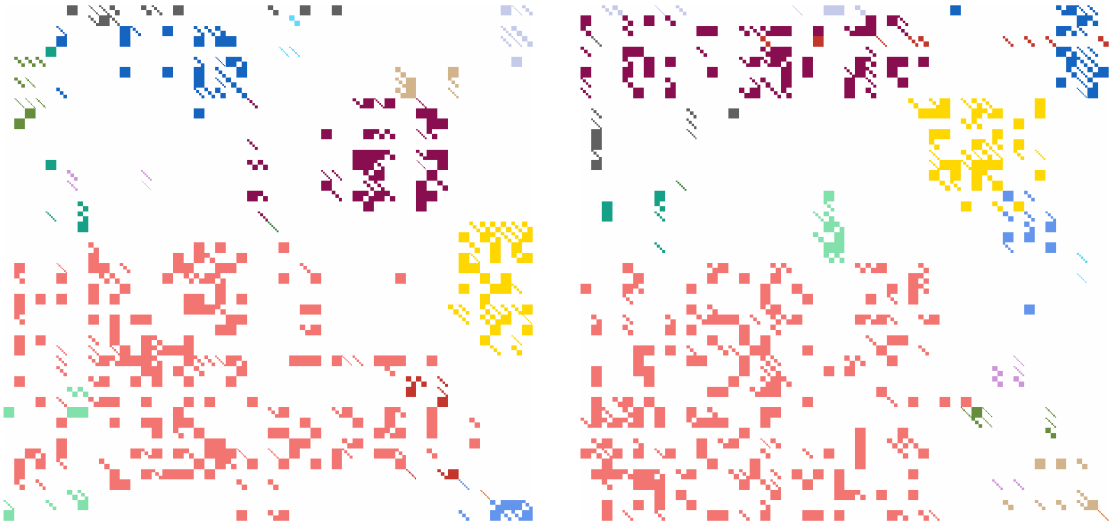


(a) Original data, 13 800 features.



(b) Dimensionality reduced data, 100 features.

Figure 4.2: Confusion matrices comparing the performance of the ANN for original data versus dimensionality reduced data. The models are trained 500 *epochs.*



(a) Dimensionality reduced data. (b) Original data.

Figure 4.3: Comparison of the clustering performance of a SOFM for original data versus dimensionality reduced data. Both maps are trained on 9 190 *images during* 10 *epochs.*

Figure 4.3 displays the clustering of 919 cell images from the validation dataset in an dimensionality reduced format, figure 4.3(a), versus the original format, figure 4.3(b). The cell classes are represented according to table 4.6.

4.2 Phase 1

For an overview of the different methods in phase 1 and how they connect with each other and the result, see figure 3.3.

4.2.1 Several Experts

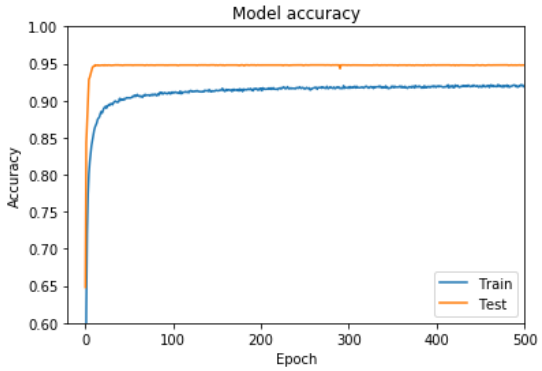
The comparison of several different expert's cell classifications resulted in an array with labels where each cell had a label of "easy" or "difficult" for the entire data set of 122 227 images.

4.2.2 Several Variations of ANN

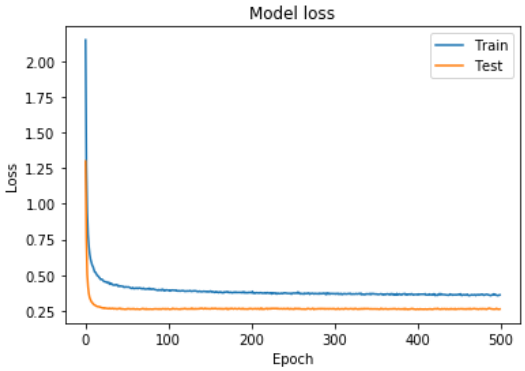
Examples of plots displaying the performance of the three ANN's are found in figure 4.4. Since each of the three models were trained on three different datasets, this in fact generated 18 plots all displaying an accuracy in the range 0.9 - 0.95 and a loss in the range 0.25 - 0.3. Here only one plot displaying the accuracy and one plot displaying the loss per model is presented, together covering the three different training datasets.

4.2.3 CellaVision Expert- Binary Classification

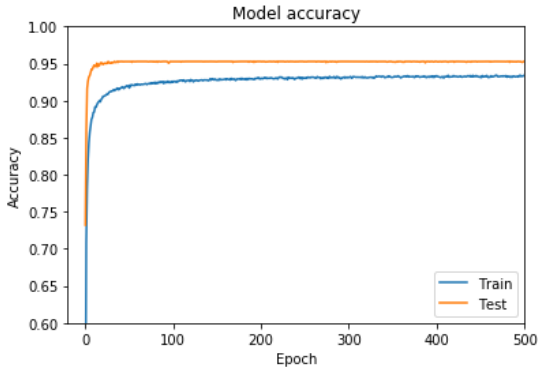
The images which were classified as "easy" or "difficult" by an expert resulted in an array with the labels belonging to each image. This was done for the datasets 130



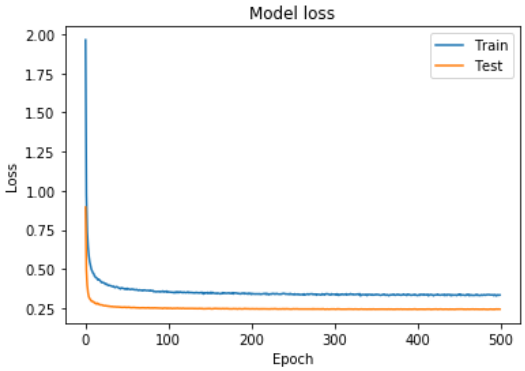
(a) Accuracy of Model 1, trained on set 10 - 39



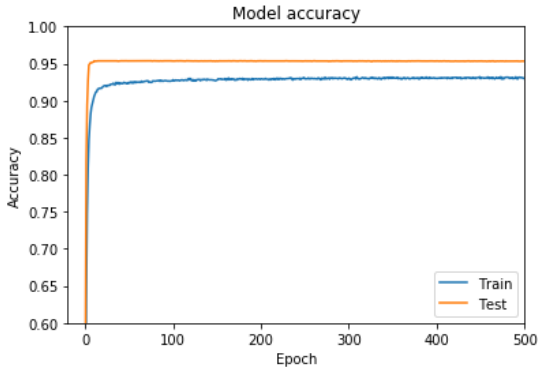
(b) Loss of Model 1, trained on set 10 - 39



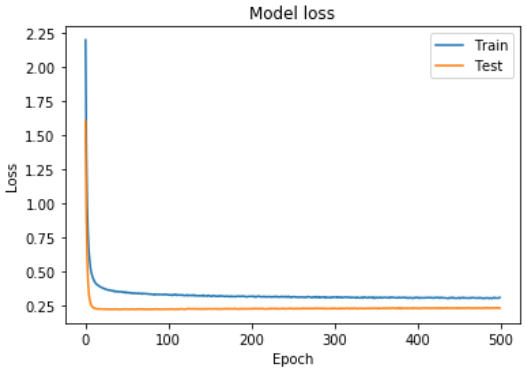
(c) Accuracy of Model 2, trained on set 40 - 69



(d) Loss of Model 2, trained on set 40 - 69



(e) Accuracy of Model 3, trained on set 70 - 99



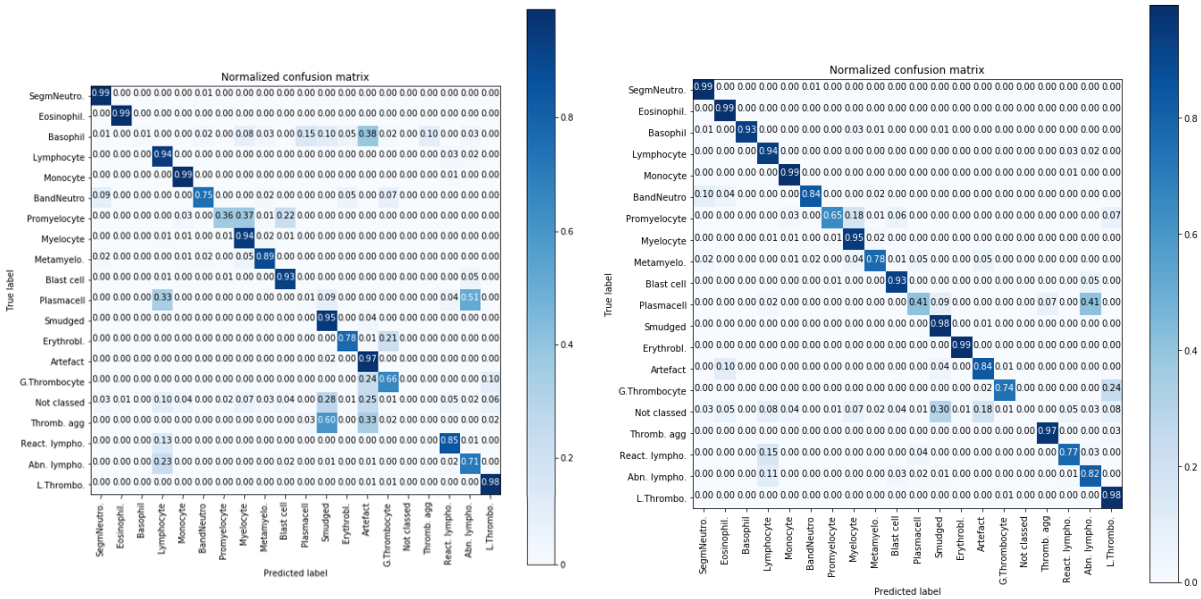
(f) Loss of Model 3, trained on set 70 - 99

Figure 4.4: Examples of the plots generated from the training of the three networks.

- 132, which means that it resulted in one label each for 2 757 images.

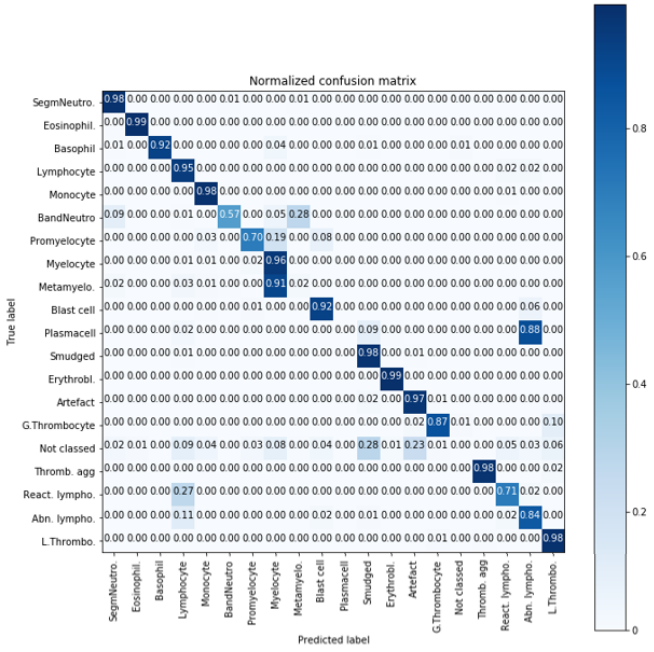
4.2.4 CellaVision Expert Re-Classing

The result of the images, which were run through CellaVision's network and then reclassified by an expert if needed, was an array with labels. Each cell had a label of "easy" or "difficult". This was done for the dataset 132 including 919 images.



(a) Model 1

(b) Model 2



(c) Model 3

Figure 4.5: Normalized confusion matrices displaying the classification performance of the three ANN's used in section 4.2.2

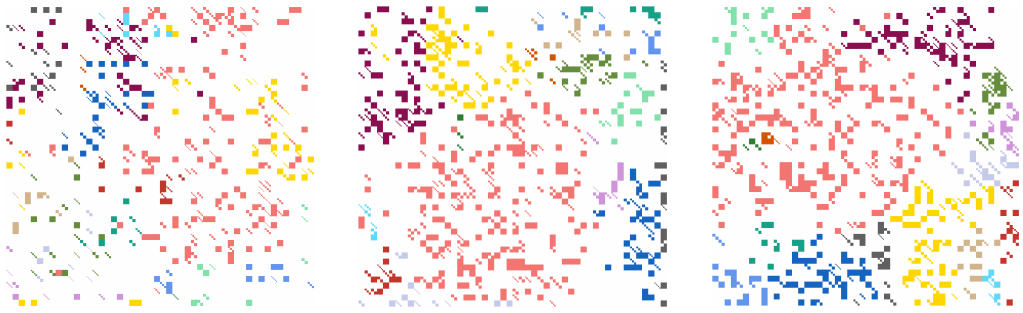
.

4.2.5 CellaVision Classification Probabilities

After running the entire data set of 122 227 images through CellaVision's network, extracting the different probabilities of each class and comparing them, the result was an array of binary labels for all of the images.

4.2.6 SOFM

It was discovered that the clustering performance of the SOFM was strongly correlated to the size of the dataset which the SOFM was trained on. Figure 4.6 displays this result by comparing three SOFM's trained on different amount of data sampled from the same dataset. 4.6(a) was trained on 919 cell images, 4.6(b) was trained on 9 190 cell images and 4.6(c) was trained on 91 900 cell images. All three models were trained 100 epochs.



(a) SOFM trained on 919 cell images

(b) SOFM trained on 9 190 cell images

(c) SOFM trained on 91 000 cell images

Figure 4.6: Three SOFM's trained on differently sized datasets sampled from the same original dataset.

4.3 Labeling Data

4.3.1 Expert Labels

The use of the two options of validation labels resulted in parameters according to table 4.1 and 4.2.

Table 4.1: Result of phase 1 using "CellaVision expert- binary classification" as validation labels. This label vector includes 111 *difficult labels.*

Method	No. difficult	Accuracy	Sensitivity	Specificity
Several exp	75	0.847661	0.207207	0.935643
Several ANN	94	0.837867	0.252252	0.918317
Class probabilities	9	0.873776	0.018018	0.991337

Table 4.2: Result of phase 1 using "CellaVision re-classing" as validation labels. This label vector includes 48 *difficult labels.*

Method	No. difficult	Accuracy	Sensitivity	Specificity
Several exp	75	0.890098	0.229167	0.926521
Several ANN	94	0.862894	0.166667	0.901263
Class probabilities	9	0.940152	0.020833	0.990815

4.3.2 Data Programmed Labels

The derivation of parameters for the different methods resulted in table 4.3.

Table 4.3: Result of phase 1. The label vector ("CellaVision binary class.") includes 111 *difficult labels.*

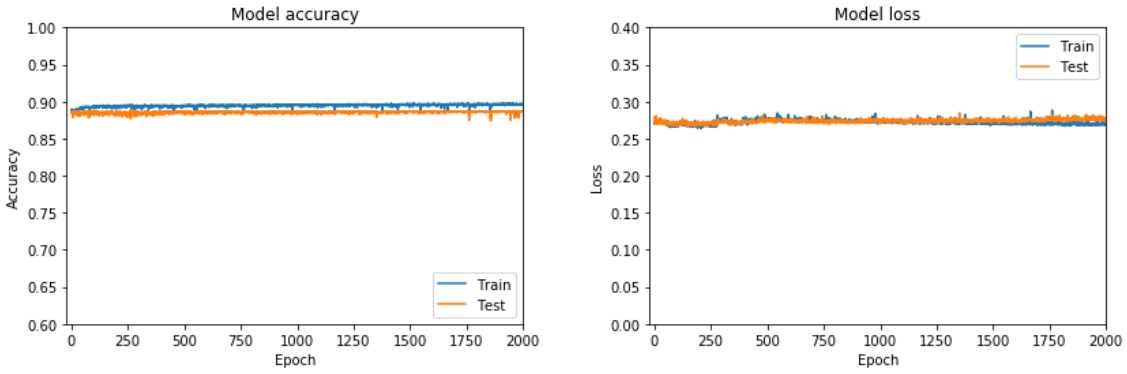
Method	No. difficult	Accuracy	Sensitivity	Specificity	Miss-rate
Several exp	75	0.847661	0.207207	0.935643	0.792793
Several ANN	94	0.837867	0.252252	0.918317	0.747700
Class probabilities	9	0.873776	0.018018	0.991337	0.981981
Merged methods	162	0.800870	0.405405	0.855120	0.594595

4.4 Phase 2

To get an overview of the different methods in phase 2 and how they connect with each other and the result, see figure 3.6.

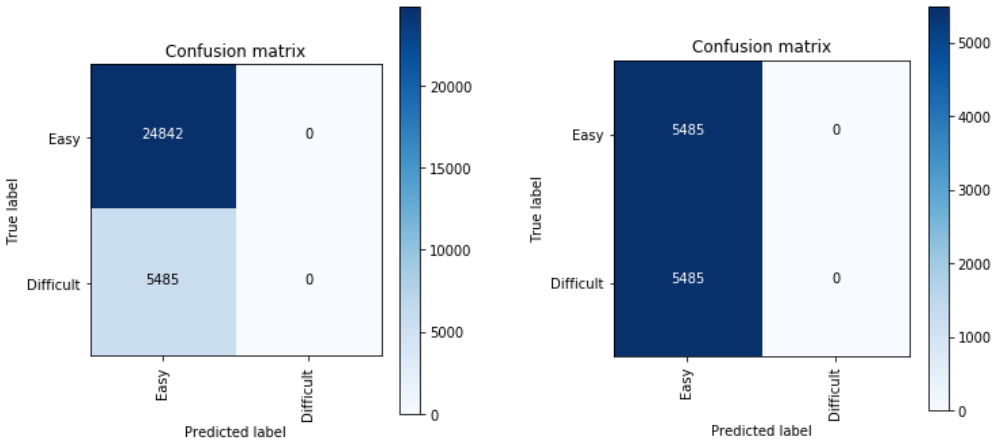
4.4.1 Binary ANN

The performance during training of the ANN for binary classification of the original imbalanced dataset is displayed in figure 4.7 while the prediction is presented in a confusion matrix in figure 4.8(a). The performance of the network during training on the balanced dataset is displayed in figure 4.9 while the prediction is presented in a confusion matrix in figure 4.8(b).



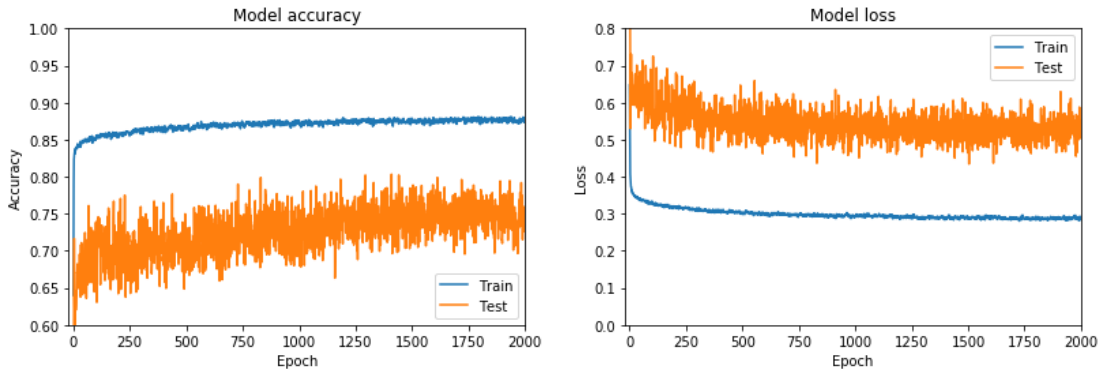
(a) Accuracy (b) Loss

Figure 4.7: Accuracy and loss during 2 000 epochs for the binary ANN trained on the labels "easy" and "difficult" from the original imbalanced dataset.



(a) Original imbalanced dataset (b) Balanced dataset

Figure 4.8: Two confusion matrices from validation of the binary ANN trained on the labels "easy" and "difficult" from two different datasets.



(a) Accuracy (b) Loss

Figure 4.9: Accuracy and loss during 2 000 epochs for the binary ANN trained on the balanced dataset.

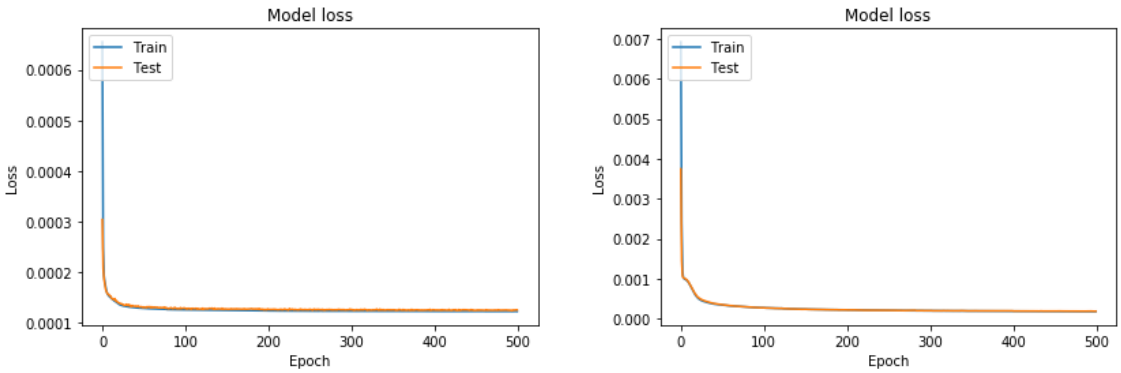
4.4.2 Autoencoder

Results of two autoencoders using different delimiters and normalization methods are shown in table 4.4, where the chosen method is highlighted in yellow.

Table 4.4: Result of the two autoencoders for different normalization methods and delimiters.

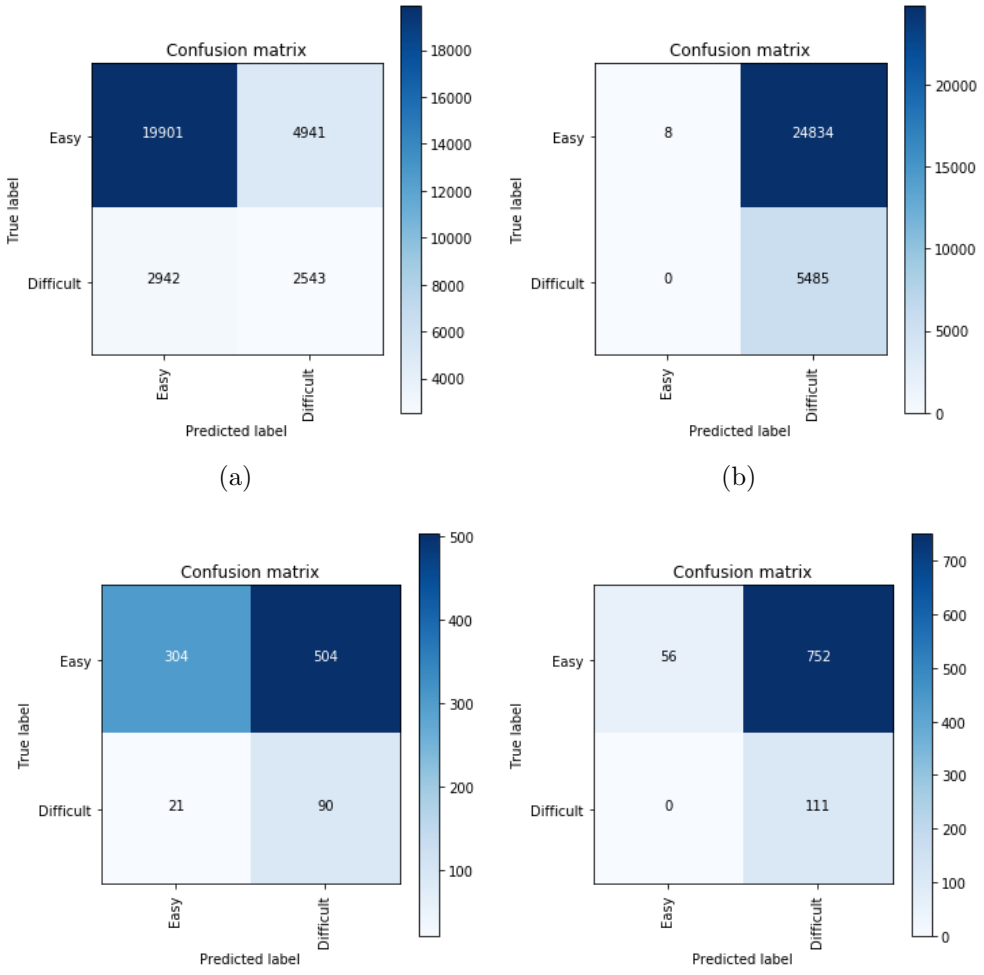
Model	Norm.	Delimiter	Accuracy	Sensitivity	Specificity	Miss-rate
Big-AE	L1	0.300	0.826887	0.345670	0.933137	0.654330
Big-AE	L2	0.300	0.822633	0.357885	0.925248	0.642115
Big-AE	L2	0.270	0.784647	0.415497	0.866154	0.584503
Big-AE	L2	0.260	0.764335	0.438469	0.836285	0.561531
Big-AE	L2	0.25	0.740067	0.463628	0.801103	0.536372
Big-AE	L2	0.240	0.714116	0.493163	0.762902	0.506837
Big-AE	L2	0.230	0.685330	0.525251	0.720675	0.474749
Big-AE	L2	0.010	0.181125	1.0	0.000322	0.0
Expert-AE	L2	0.250	0.428726	0.810810	0.376237	0.189189
Expert-AE	L2	0.113	0.177366	1.0	0.064356	0.0

The performance of two chosen autoencoders at two delimiter values is displayed in the confusion matrices in figure 4.11, and the training losses in figure 4.10. A Precision-Recall Curve (PRC) is an optimal way of analyzing a binary classification of unbalanced data (30). It plots the recall (the amount of all target cells which were found) on the x-axis and the precision (the proportion of the found target cells which were correctly classified) on the y-axis. Therefore such plots was generated for the autoencoders, seen in figure 4.12 and figure 4.13. Both for analyzing the classification of easy cells, and for analyzing the classification of difficult cells.



(a) (b)

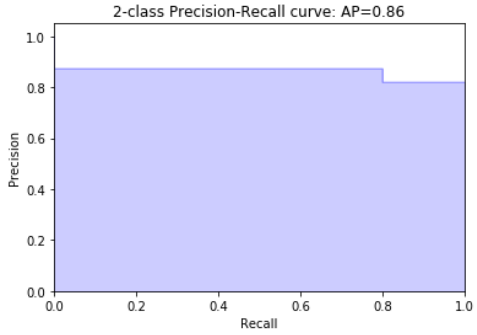
Figure 4.10: Plots demonstrating the loss during the training of the two autoencoders on reconstructing easy cell features, both trained 500 *epochs: (a) the loss of Big-AE; and, (b) the loss of Expert-AE.*



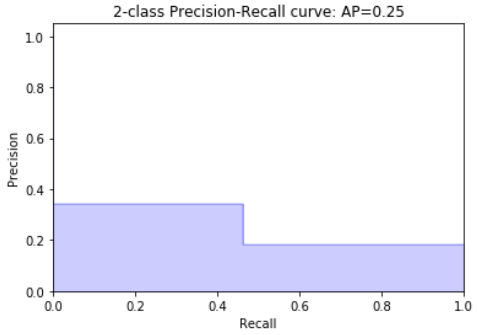
(a) (b)

(c) (d)

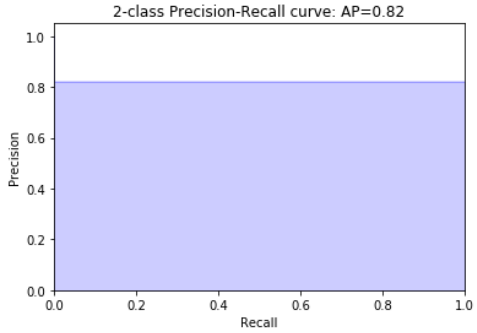
Figure 4.11: Confusion matrices of the performance of the two autoencoder at different delimiters: (a) Big-AE for delimiter 0.250*; (b) Big-AE for delimiter* 0.010*; (c) Expert-AE for delimiter* 0.250*; and, (d) Expert-AE for delimiter* 0.113*.*



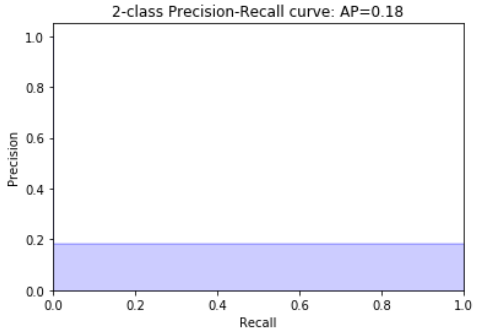
(a) Easy cells



(b) Difficult cells

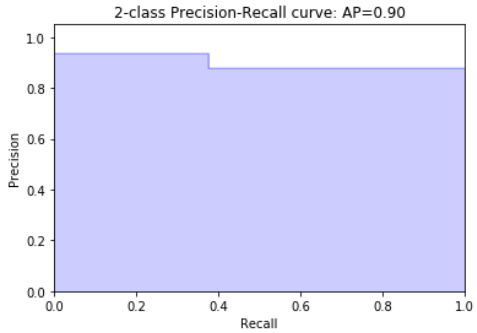


(c) Easy cells

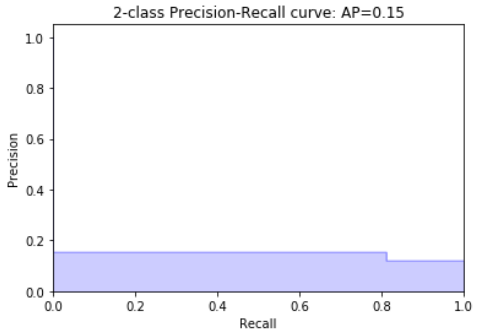


(d) Difficult cells

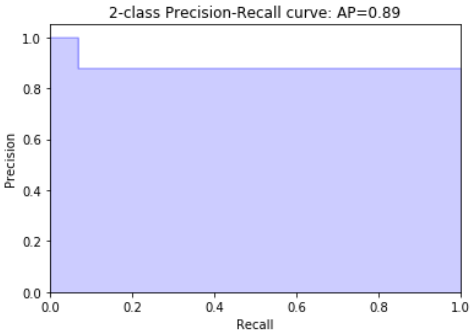
Figure 4.12: PRC's of the performance of the chosen Big-autoencoder: (a) and (b) at delimiter 0.250 *and (c) and (d) at delimiter* 0.010*.*



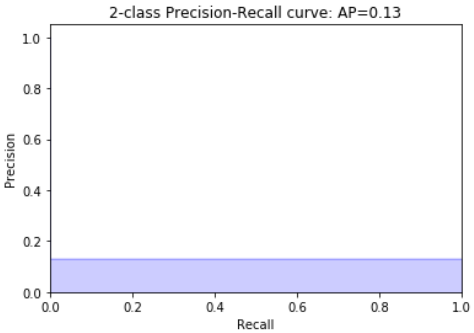
(a) Easy cells



(b) Difficult cells



(c) Easy cells



(d) Difficult cells

Figure 4.13: PRC's of the performance of the chosen Expert-autoencoder: (a) and (b) at the delimiter 0.250*; (c) and (d) at the delimiter* 0.113*.*

4.5 Analyzing Phase 2

Table 4.5 displays the results from the ANN and the autoencoder for comparison.

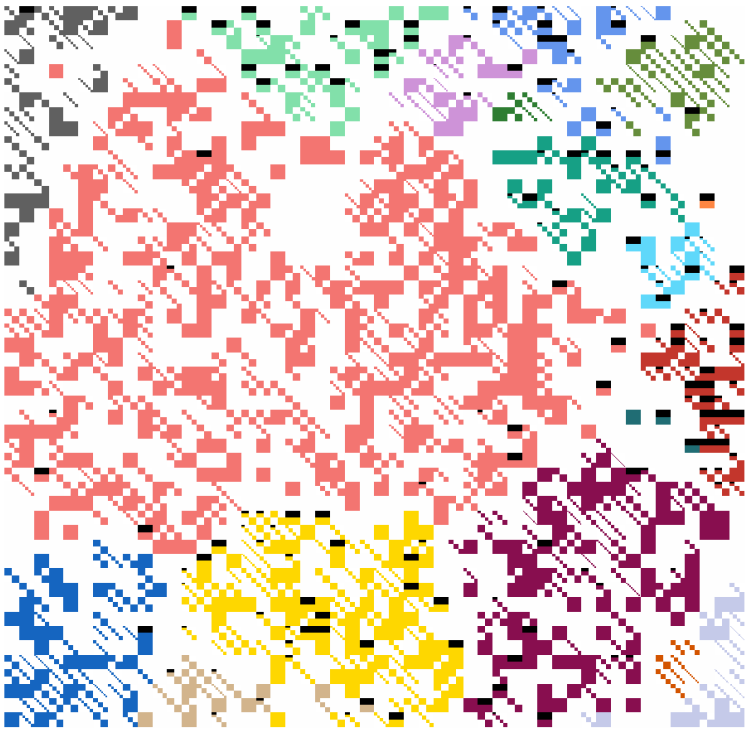
Table 4.5: Resulting parameters for the two models tested in phase 2.

Method	Norm.	Delimiter	Accuracy	Sensitivity	Specificity	Miss-rate
ANN	-	-	0.819138	0.0	1.0	1.0
Big-AE	L2	0.250	0.740067	0.463628	0.801103	0.536372

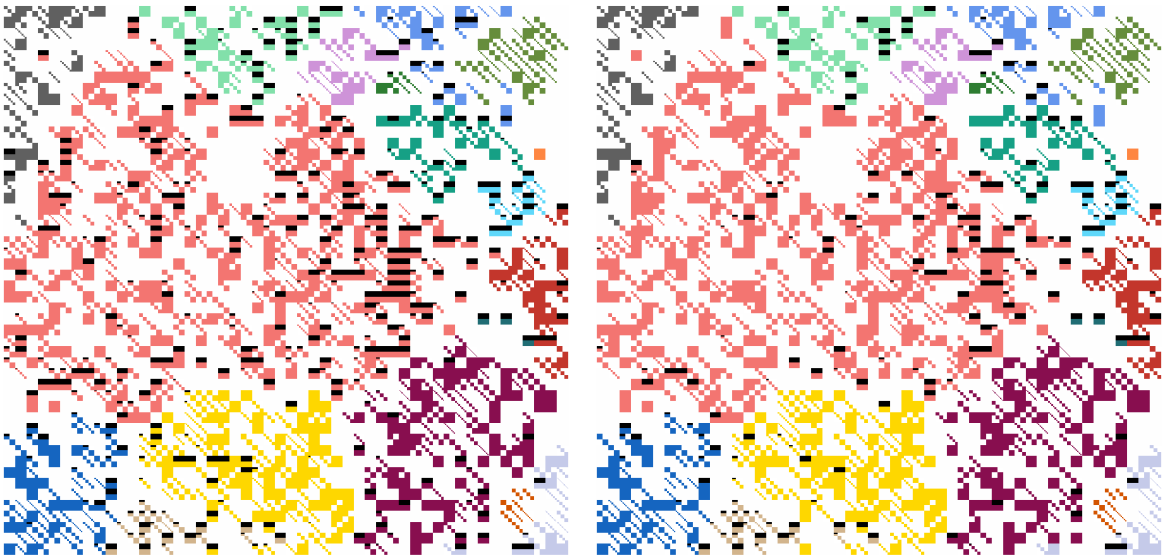
Figure 4.14 shows a SOFM with colors representing classes according to 4.6 where the difficult cells, according to the validation labels, have got half of their corresponding node colored black.

Table 4.6: Colors in SOFM.

Cell class
Segmented Neutrophil
Eosinophil
Basophil
Lymphocyte
Monocyte
Band Neutrophil
Promyelocyte
Myelocyte
Metamyelocyte
Blast
Plasma Cell
Smudge Cell
Erythroblast
Artefact
Giant Thrombocyte
Thrombocyte Aggregation
Reactive Lymphocyte
Abnormal Lymphocyte
Large Thrombocyte]



(a)



(b) (c)

Figure 4.14: SOFM trained 500 *epochs on* 91 900 *cell images, with colors representing the classes and black halves representing the difficult cells: (a) validation labels according to 3.3.1, (b) labels from the result of the autoencoder with delimiter* 0.25*, see table 4.4, (c) labels from the result of the autoencoder with delimiter* 0.30*, see table 4.4.*

Chapter 5

Discussion

5.1 General Discussion

A difficult aspect of the study has been the decision of what is easy to classify and what is not. This is because the answer often lies in the eye of the beholder. The divisions of cell classes are not distinctive and the classification of them often depends on the surrounding cells and not only the visual appearance of the cell itself. It is therefore difficult to decide which cells are difficult to classify only based on it's own visual appearance. An expert can can include many aspects in the classification of the cell. With many years of experience, it becomes easier to classify cells both based on morphology but also on qualified intuition. This is the reason that an expert have been used to create validation labels, see 3.3.1.

Since the aim of the project was to find true easy cells, it was decided early in the project that cells labeled as easy should, without any doubt, be easy. This resulted in that all methods in phase 1 used for generating the training and validation labels had to label a cell as easy for it to be classed as easy, otherwise it was classed as difficult. This made sure that the binary ANN and the autoencoder in phase 2 would only consider clearly easy cells as easy and everything else as difficult. These models were thereby exposed to minimal risk of learning patters of difficult cells and recognizing these patterns as patterns of easy cells. The models would thereby be expected to detect difficult cells, as all patterns which were not found on easy cells were recognized as difficult during training.

5.1.1 Data Selection

The results in figure 4.6 shows that the more data used, the better clustering in the SOFM. Figure 4.7 demonstrates the accuracy and loss for a network trained on a larger amount of data, while figure 4.9 demonstrates the accuracy and loss for the same network trained on a smaller amount of data derived from the same database. By comparing these figures, it can be seen that the accuracy is clearly higher while the loss is lower for the network trained on more data. This is generally the case for neural networks, and it would therefore be interesting to use an even larger data set in future tests and see if the results could be improved.

The dataset consisted of unevenly sized cell classes. This can be seen in table 3.1 by

comparing the amount of present cells. This distribution derives from natural blood samples from a wide range of patients and the distribution thereby reflects well on the common case for blood analysis on a normal patient. However, this provides the algorithm with less data for rare cell classes and it would thereby be less expected to provide confident analyses for these classes compared to more common classes. It would therefore be interesting to perform similar simulations for a larger dataset where all cells are represented to an equal amount.

As demonstrated in figure 4.1(a), 4.2(a) and 4.3(b), the networks in this study performed well on the original sized data. This is seen by a high training accuracy, well performed classification and clears clusters. However, a lot of time was required for training on this data. With more time and/or more powerful computational systems, it would be interesting to perform all tests from phase 2 on the original sized dataset and compare the result.

The division of the dataset during phase 1 resulted in a section of normal, easy cells and a section of more difficult cells. Because of the large size of this dataset while having limited access to morphology experts, it has not been possible to verify that this division was fully accurate. This means that there is a possibility of some easy cells being labeled as difficult and some difficult cells being labeled as easy. This might have influenced the model's capability of detecting the subtle patterns which separates easy cells from difficult. In the dataset, the cells marked as easy were very likely to be easy while the cells marked as difficult were not as likely to be difficult. This favored the model's capacities of labeling easy cells correctly, while being less confident in labeling difficult cells.

5.1.2 Building Neural Networks

Since all networks in this thesis were built using systematic experimentation, it is likely that the longer time spent on experimenting with architecture the better network architectures could probably have been found. Thereby, the performance of the networks was a trade-off between performance and time spent on this experimentation.

5.1.3 Dimensionality Reduction

The comparison of the two methods PCA and autoencoder for dimensionality reduction was only based on literature, which stated that the autoencoder was better for large data sets since it retained all input information and had higher accuracy than PCA (26). A more thorough comparison of the methods would have been interesting if there were more time.

The architecture of the ANN's which are testing the effect of dimensionality reduction could not be exactly equal, due to different input sizes, see table 3.3. For example for the full-sized dataset, the largest amount of nodes in the first dense layer that the computer could handle was 1 380 nodes, which corresponds to 10% of the 13 800 input features. For the dimensionality reduced dataset of 100 features, an amount of 80 nodes in the first dense layer were considered as the best performing

architecture for the network. This corresponds to 80% of 100 features. Because of this, the model comparison cannot reflect on the best possible performance of the two models but rather on the best available performance for these computers.

When plotting the accuracy and loss for the two networks, it was clear that the network trained on 13 800 dimensions had been overtrained already at epoch 30 of 500. This was considered early, since the network trained on only 100 dimensions showed no signs of overtraining after 500 epochs. Since overtraining occurs when the model starts to train on noise in the data (12, p. 45), this may indicate that a large amount of the 13 800 data points for each image are noise. It is believed that when using the AE to compress the data points from 13 800 to 100, these 100 data points are to the vast majority general properties while containing a small amount of noise. The network trained on the autoencoded features was better at generalizing, while the network trained on 13 800 dimensional features was trained on noise and thereby lost the ability of generalizing.

The time aspect of training the models was also important in this study. Considering that the network which trained on fewer dimensions finished 500 epochs in 10 minutes while the network trained on full-sized dimensions finished in 2 hours, it was therefore decided to continue with the dimensionality reduced data.

Testing the effect of dimensionality reduction using SOFM gave an unexpected result. The SOFM trained on the original dataset, see figure 4.3(a), displayed a better clustering of the cells compared to the SOFM trained on the dimensionality reduced features, see figure 4.3(b). The expected result was that the SOFM trained on the original features would have performed worse since the original data was suspected to be filled with unnecessary information and noise due to the performance of earlier ANN's during the previous tests.

5.2 Phase 1

To get an overview of the different methods in phase 1 and how they connect with each other and the result, see figure 3.3.

5.2.1 Methods

5.2.1.1 Several Experts

Removing all cells which had only been classed once by one morphology expert ensured that the cells had to be classed at least twice. These classifications were compared and the cells could be labeled as easy or difficult. Most of the cells in this dataset had been classed twice, while some cells was classed by up to 15 experts. This implies a certain imbalance in the dataset since one divergent classification of a cell will corresponds to a higher or lower percentage of the different classifications depending on the amount of experts who had classed that particular cell. It could be discussed whether a cell which for example was classed by 15 experts where 14 agreed on the cell class "Lymphocyte" while one claimed it was an "Abnormal Lymphocyte" should be considered as easy or difficult, since 93,3% of the experts

in fact agreed on the cell class. However, in this project it has been of importance to ensure that all cells which are labeled as easy are very easy while all ambiguous cells are labeled as difficult. Therefore, the decision to label all cells where at least one of the morphology experts classifications disagreed with the others is a decision consistent with the aim of the project.

5.2.1.2 Several Variations of ANN

The performance of all three networks were very even, with an approximate accuracy of 92.5% and a loss at 27.5%. This was considered as a good enough performance. Which level of performance to aim for when creating these networks was a trade-off between performance and resulting classification, since an accuracy of 100% and a loss of 0% for all networks would have resulted in all networks classing the validation equally and correct. Hence, the networks would not have disagreed on any cells, resulting in this method not generating a vector of difficult cells. Also the other way around, a very low accuracy and a high loss would probably have resulted in the networks classing most of the cells differently and wrong, would have resulted in a vector containing an unreasonably large amount of difficult cells.

5.2.1.3 CellaVision Expert Re-Classing

The result from using a morphology expert from CellaVision reclassing the cells was good. However, since it was better to use the result of an expert classing easy or difficult right away (4.1) and it takes too much time and resources to get enough data from this method, it was not used to generate training data in phase 1.

5.2.1.4 CellaVision Classification Probabilities

Since all probabilities of CellaVision's classification algorithm were very high, it was hard to make the decision on where to draw the line between what should be considered as easy or difficult. The probability value chosen as the delimiter was 0.999, see section 3.2.5. The fact that values below of a delimiter this high could still be considered as the difficult cells, shows that CellaVision's network was very sure of it's own classification.

5.2.1.5 SOFM

The importance of training a SOFM on a large dataset was discovered during early tests. When comparing the figures 4.6(a), 4.6(b) and 4.6(c), it was clearly seen that a better clustering is achieved when training the SOFM on a larger dataset in figure 4.6(c). The SOFM trained on 91 900 images were found to perform best, and it was thereby decided to further train all SOFM's on such large datasets to ensure a reasonably good test result.

The initial idea was to use the SOFM as a way to find which cells were easy to classify and which were not. If a cell was visualized in the middle of a cluster on the SOFM-grid, the thought was that this cell would be easy to classify versus a cell which was visualized between clusters. The distance between the neurons would therefore be a measure of how easy the cell was to classify. However, as can be seen

in figure 4.14 where the difficult cells are visible in the same grid as the clusters, the probability of a cell image being difficult does not seem to relate only to the distances from the cluster centers, but also very much to the cell type. Therefore, it was decided that the SOFM would not be used as a way to find if a cell was difficult or easy to classify, but rather to analyze the other methods.

5.2.2 Labeling Data

A SOFM performs its training process without labels and is therefore only dependent on having access to labels during validation, which makes the amount of required labels relatively low. It was thereby possible to use labels generated by a morphology expert at CellaVision for this task, which was desired since these kind of labels are of high quality (28). An ANN for classification on the other hand requires access to labels during training, and the labels should be generated before splitting the dataset into training and validation for the result to make any sense (31). The autoencoder in particular do not require labels during its training but, in this research, labels were required for generating the dataset which the autoencoder would train on. Therefore it would not make sense to use the CellaVision expert's labels for validation of the binary ANN or the autoencoder, since the labels which these methods were build upon would not have been generated in the same way as the labels for the training data.

5.2.2.1 Internal Labeling

Looking at table 4.1 and 4.2, it can be seen that the specificity in general was higher for 4.1. The sensitivity was higher in 4.1 for "Several ANN" but lower for the other two methods. This is related to that this method found more difficult cells than the other two methods. Finding too many difficult cells is better than finding too few difficult cells, which makes this high number a good thing as long as it is within a reasonable range. 111 difficult cells would be considered reasonable. The accuracy is rather similar for both methods, even though the second method do have better values. All together, the difference in sensitivity and number of found difficult cells outweighs the difference in other aspects, and the first option, "CellaVision expert-binary classification" is chosen as the validation labels to be used when mapping where difficult cells are placed in the SOFM.

5.2.2.2 Data Programmed Labels

The validation labels for the binary ANN and the autoencoder had to be generated in the same way for both training and validation, which means that labels were needed for a very large amount of data. The alternative of using CellaVision's morphology expert was therefore ruled out due to lack of resources. The labels were generated for a large dataset, which enhanced the training of the models, although labels generated by data programming tends to be of a lower quality (28).

When analyzing the three methods separately, it was found that the method "Classification probability" had the best accuracy and specificity. However, it had low sensitivity compared to the other two methods. This implied that the method labeled difficult cells as easy, which was an unwanted behaviour. The methods "Several

experts" and "Several ANN" were of equal performance in accuracy and specificity, although "Several ANN" had higher sensitivity and lower miss-rate. It is also the method which found the highest number of difficult cells. This is an advantage since it is better to classify too many cells as difficult than too few.

It was found that the three methods identified different kinds of difficult cells. It was therefore decided to merge these, generating a fourth label vector. The derived parameters for this label vector is seen as "Merged methods" in figure 4.3. When merging the methods, the resulting sensitivity was highly increased since all previously difficult labeled cells are added together in this method. This was a much better result than the methods one by one. The specificity was decreased, which also is a result of the higher number of difficult labeled cells. Even though this decrease was a deterioration, the overall parameters for the merge was the best obtained result and chosen to use in phase 2.

It should be pointed out that the sensitivity is rather low for all the methods. This is hard to get around, since the answer used is a subjective answer from the expert. The problem lies in the study area itself, since there is no objective answer to what is a difficult cell, which has been discussed, see section 5.1.

5.3 Phase 2

To get an overview of the different methods in phase 2 and how they connect with each other and the result, see figure 3.6.

5.3.1 Binary ANN

As seen in figure 4.8(a) and 4.8(b), the binary network could not detect any difficult cells neither when trained on the original imbalanced dataset nor the balanced dataset. However, the performance of the model during training on the original imbalanced dataset was good, as seen in figure 4.7. The accuracy was high and the loss relatively low for both training and test data, although the test performance was somewhat unstable for both accuracy and loss. The performance of the model during training on the balanced dataset was not as good, see figure 4.9. For the original data the training accuracy was high (87%) and the loss relatively low (28%) while both being unstable for the balanced data where the accuracy varied between 65% to 80%, and the loss between 50% and 60%.

When looking at the plots from the model trained on the original imbalanced data, the measurements from the model indicates that the model should perform quite well - while it in fact guessed all cells to be easy and no cells to be difficult. This indicates that the properties that makes the cell difficult is very individual from cell to cell. The model has not been able to detect any general patterns for what makes a cell difficult, it has only learned what was difficult for the difficult cells in the training data. The fact that the model labeled all cells as easy indicates that the difficult cells in the validation dataset displays tendencies and properties of easy cells.

When instead looking at the plots from the model trained on the balanced dataset with a much higher presence of difficult cells, the unstable plots indicates a difficult learning process for the network. Since the test loss does not increase while the training loss decreases as seen in figure 2.5, we can draw the conclusion that the poor performance is not related to the model being overfitted to the training data. This unstability could instead be a result of a higher presence of difficult cells, resulting in the model making wrong classifications of the difficult cells more often since they now correspond to 50% of the dataset.

5.3.2 Autoencoder

As seen in figure 4.10(a), the loss of the big autoencoder is very low (0.015%) when trained on reconstructing the easy cells. As seen in figure 4.10(b), the loss of the expert autoencoder is also low (0.05%). The low losses can be considered as indications of both autoencoders performing well as reconstruction models.

To decide a delimiter value for the big autoencoder, several different values were tested and the resulting parameters values were calculated, which can be seen in table 4.5. With a high delimiter value, the specificity is high and the miss-rate increases. With a low value both the miss-rate and the specificity decreases. The highest risk for the future use of this model would be that the difficult cells are identified as easy cells and therefore hidden from the user. This happens when the miss-rate is high. However, if the specificity is low, the model identifies many easy cells as difficult and thereby displays them in the interface. If too many cells are displayed, there is no point in using the model at all. A high specificity is therefore needed for the model to become useful for lab personnel. It becomes a trade-off where balancing these two values, the miss-rate and the specificity, is important. A low miss-rate and high specificity is wanted. It was found that a delimiter at 0.25 displayed a low miss-rate (around 50%) while the specificity was high (80%) e.g. reflecting on few false positives. This choice of delimiter was a number providing desirable results for this specific dataset for the big autoencoder.

The difference between using L1- and L2-normalization was small, but since L2 gave slightly better results it was chosen for this model.

By looking at table 4.5 and figure 4.11, 4.12 and 4.13 the two autoencoders Big-AE och Expert-AE can be compared. While Big-AE with delimiter 0.25 has a higher precision for the found difficult cells than the Expert-AE, it finds fewer and has a lower precision of the easy cells, figure 4.12(b) and 4.13(b). When comparing the delimiter value which gave 100% sensitivity for both autoencoders, i.e. predicting all difficult cells correctly, it is clear that the Expert-AE has a much higher specificity than the Big-AE. It finds 56 out of 808 easy cells, while Big-AE only finds 8 out of 24 842 easy. This means that the Expert-AE could be used autovalidate many of the cells and with the delimiter value 0.113 predict all difficult cells correctly. However, finding 56 out of 808 easy cells and hiding these would not decrease the workload for the lab personnel sufficiently. Besides, the specificity of the Expert-AE is lower than for Big-AE, see figure 4.5, meaning that the probability of the predicted difficult cells to truly be difficult, is lower. This is not desired. Considering that the

Expert-AE is trained on a much smaller dataset and received a larger loss than the Big-AE it can also be considered as more uncertain. Therefore, the Big-AE has been chosen for use in the tests in this study. However, it would definitely be interesting to see how well the Expert-AE could perform when trained on a larger dataset.

Figure 4.11 shows how the Big-AE worked as an anomaly detector. About 46% of the difficult cells were found. Figure 4.12(a) shows that the model found about 80% of the easy cells in the data (recall) while about 90% of the predicted easy cells were true easy cells (precision). When looking at figure 4.12(b) it can be seen that the model is better at identifying easy cells than difficult ones. About 45% of the difficult cells were found (recall), while only about 35% of the predicted difficult cells were true difficult cells (precision). This is not surprising, since the difficult cells are more unlike each other and therefore more difficult for the model to find. It is a benefit that the model is better at identifying the easy cells, since the future use of the model would imply hiding easy cells from a user interface. It is therefore more important to be sure of the classification of the easy hidden cells, than of the difficult displayed cells. However, if the model finds 90% correct easy cells this means that 10% of the hidden cells are in fact difficult, and would have been important to display. This number needs to be decreased for the model to be used in a real application.

Despite the low loss of the autoencoder during training, many of the difficult cells could not be detected. This may be a result indicating that some of the difficult cells could in fact be semi-easy cells, or that these cells have attributes making them very alike normal cells. The result supports the theory of difficult cells being composed by several separately normal attributes from different cell classes, making the true cell class difficult to decide.

When dividing the dataset into easy and difficult cells in the setup of this study, it was decided to make sure that all cells labeled as easy were definitely easy. All cells displaying any tendency of being difficult due to any method during phase 1 were labeled as difficult. This have affected the autoencoder by enhancing its capability of reproducing easy cells, while all cells in the borderline between being easy and difficult might be harder to recognize as difficult since these are partially patterns which the autoencoder was trained on. Therefore it is an expected result for the autoencoder to display a higher specificity than sensitivity. If the setup of the task was the opposite, meaning that it was instead required for a cell to be very difficult for it to be classed as difficult while labeling all easy and semi-easy cells as easy, the autoencoder could instead be expected to display a higher sensitivity since the patterns of the very difficult cells might not be present to the same extent in the easy or semi-easy cells. A disadvantage of this setup would instead be the risk of a lower specificity, reflecting on an increase of easy cells labeled as difficult. When using autovalidation in a real product, the purpose would be to decrease the workload for the lab personnel by hiding the normal, easy cells which were classed correctly by the software. Furthermore, since easy patters constitutes for the majority of the data, it makes sense to approach the problem of autovalidation by being sure of what is easy rather being sure of what is difficult.

5.3.3 SOFM

The resulting SOFM, see figure 4.14, shows clear clustering of the cell classes. This is a positive result that implies that the SOFM method works well for the cells. When it comes to identifying the difficult cells in the clusters, it is not obvious what conclusions can be drawn from the SOFM. In many cases the difficult cells seems to be in the outskirts of the clusters, which makes sense if one assumes that the difficult cells are the ones which are most unlike a specific cell type. It indicates that they are somewhere in between cell classes when it comes to appearance.

However, the difficult cells are not always positioned in the outskirts, there are cases seen when the cells are almost in the middle of clusters. It is difficult to understand how these cells can be difficult to label and at the same time looking very much alike many cells of the same cell class.

Furthermore, big differences can be seen when it comes to the proportion of difficult cells in different cell classes. Abnormal lymphocytes, reactive lymphocytes, myelocytes, monocytes and artefacts are all classes where a large proportion of the cells are difficult. It would probably be desired that these cell classes have a higher chance of being classified as difficult, since they show this tendency.

5.4 Future

5.4.1 Future Improvements

The parameters in this study have been selected based on the aim, models and time limits of the study. The parameters in models have been optimized due to the restrictions of the project, and the choices which led to these results can many times be seen as subjective. This means that with other circumstances, such as other data or time limits, it is possible that improved specificity, sensitivity, miss-rate and accuracy could be obtained.

The method "CellaVision classification probabilities" might have been able to give clearer results if the probabilities had been more separated and evenly spread over an interval between 0 and 1. In this study the probabilities were all very high, about 0.9. A further study trying to separate the high numbers would have been needed. One idea is to use layer normalization for this task. This was something that this study did not have the time to focus on, but could be an example of future work.

Furthermore, the results showed that the SOFM improved its clustering ability with a larger dataset. It would therefore have been interesting to see how much the SOFM could improve with an even larger data set than used in this study, but also if the result of the other methods would improve with more data.

For future studies, it would be of interest to put more work and detail into deciding which cells should be considered as easy and which should be considered as difficult when preparing the binary labels. Since this question is of a such subjective matter, it would be interesting to involve more morphology experts and create a larger set of

internal labels. Another complement to the methodology of preparing the database could be to perform interviews with morphology experts to get a better understanding of the appearance of difficult cells. This may result in an improved sectioning of the dataset, providing the models with better prerequisites and enhancing the autoencoder's chances to detect anomalies in data.

5.4.2 Future Work and Usage

Future work would include testing the model by implementing it as a module in the CellaVision system, where it could be used as a way for the user to hide different levels of easy cells. A user study should be carried out to investigate how much this reduces the work load for the users, as well as a study of how the module would affect the result of diagnoses made with help from it. An autovalidation module like this includes risks since cell classification is affected by the appearance of the cells nearby, which means that a morphology expert often have to view the whole blood sample to be able to draw conclusions. This is not taken into account in this study. There is a risk of hiding cells which are needed for the evaluation, which could lead to that the conclusions of the test might be incorrect. It is therefore important to do a proper clinical study of the module.

5.5 Ethical Considerations

When performing studies in the medical field, there is an ethical risk to consider due to patient integrity. In this study blood samples from patients have been used to generate data. However, it is not the actual blood which have been exposed for the tests but the images of the cells in the blood. For all samples, the blood have been anonymized which makes them impossible to trace back to the patients.

Chapter 6

Conclusions

The aim of this thesis was to investigate the possibility of creating a binary classifying module able to separate white blood cells which are easy to classify from those more difficult to classify. The models were thought to be trained on data from CellaVision's database, while being independent of which hospital has made the blood cell preparation. The results in this research was based on data from blood smears of different preparations, which made the model independent corresponding to the aim. The possibility of creating a binary classifier able to separate easy cells from difficult cells has been investigated while achieving indications of both successful and less successful ways of approaching this problem. The conclusions of the study will be presented in the following sections.

Firstly, the ANN for binary classification turned out not to work for this particular task of classifying cells as easy or difficult since no difficult cells could be detected with this method.

Secondly, the autoencoder was able to correctly detect 45% of the difficult cells and 80% of the easy cells. This indicates that the used approach was somewhat successful, although several future improvements was suggested which might improve the performance to become even better.

Furthermore, it was concluded that it was of importance to train SOFM's on large datasets to ensure a good clustering capacity. In the SOFM, the difficult cells were often placed on nodes at the outskirts of the clusters, although some difficult cells were located in the middle region of the clusters.

Additionally, it was found that reducing the dimensionality of the data with the use of an autoencoder did not have a negative impact on the performance during classification or anomaly detection. This since all important information seemed to be represented in the converted data. When also considering the time aspect of training networks, using dimensionality reduced features in neural networks and autoencoders was beneficial.

Bibliography

[1] CellaVision, "Årsredovisning 2018, CellaVision," 2018.

[2] M. Sandhya Pruthi and M. Rajiv K. Pruthi, "Complete blood count (CBC)." https://www.mayoclinic.org/tests-procedures/complete-blood-count/about/pac-20384919, 2018. Accessed 2019/04/12.

[3] G. Gerhardt, "Autovalidation for Patient Test Results in Chemistry- What it Means for the Patients and the Staff." http://apps.pathology.jhu.edu/blogs/pathology/autovalidation-for-patient-test-results-in-chemistry-what-it-means-for-the-patients-and-the-staff, 2010. Accessed 2019/04/10.

[4] V. Rimac, I. Lapic, D. Rogic, K. Kules, and M. Miler, "Implementation of the Autovalidation Algorithm for Clinical Chemistry Testing in the Laboratory Information System," *Laboratory Medicine*, vol. 49, pp. 284–291, 02 2018.

[5] P. Froom and M. Barak, "Auto-validation of complete blood counts in an outpatient's regional laboratory," *Clinical Chemistry and Laboratory Medicine (CCLM)*, vol. 53, no. 2, pp. 275–279, 2014.

[6] R. U. Deepak, R. R. Kumar, N. B. Byju, P. N. Sharathkumar, C. Pournami, S. Sibi, E. Bengtsson, and K. Sujathan, "Computer Assisted Pap Smear Analyser for Cervical Cancer Screening using Quantitative Microscopy," *Journal of Cytology Histology*, 04 2015.

[7] W. William, A. Ware, A. H. Basaza-Ejiri, and J. Obungoloch, "A pap-smear analysis tool (PAT) for detection of cervical cancer from pap-smear images," tech. rep., 02 2019.

[8] F. Demirci, P. Akan, A. R. Sisman, T. Kume, Z. Erbayraktar, and S. Sevinc, "Artificial Neural Network Approach in Laboratory Test Reporting: Learning Algorithms," *American Journal of Clinical Pathology*, vol. 146, pp. 227–237, 07 2016.

[9] L. Dean, *Blood Groups and Red Cell Antigens: Chapter 1, Blood and the cells it contains*. Bethesda (MD): National Center for Biotechnology Information (US), 2005.

[10] CellaVision, "Hematopoiesis, Erythropoiesis." https://www.cellavision.com/en/cellavision-cellatlas/hematopoiesis-cellatlas, 2019. Accessed 2019/03/01.

[11] T. Scordino, "Giant platelets." https://imagebank.hematology.org/image/60931/giant-platelets, 2016. Accessed 2019/03/01.

[12] M. Ohlsson, *Lecture Notes on Introduction to Articial Neural Networks and Deep Learning (FYTN14/EXTQ40).* Computational Biology and Biological Physics Department of Astronomy and Theoretical Physics, Lund University, 2018.

[13] D. Asboth, "Self-Organizing Maps: An introduction." http://blog.yhat.com/posts/self-organizing-maps-1.html, 2017. Accessed 2019/01/07.

[14] S. Haykin, *Neural Networks: A Comprehensive Foundation.* Prentice Hall PTR, 2nd edition ed., 1998.

[15] J. Brownlee, "How to Configure the Number of Layers and Nodes in a Neural Network." https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/, 2018. Accessed 2019/03/01.

[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016. http://www.deeplearningbook.org.

[17] A. Budhiraja, "Dropout in Deep Machine Learning." https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5, 2016. Accessed 2019/03/01.

[18] D. Asboth, "Self-Organizing Maps: In Depth." http://blog.yhat.com/posts/self-organizing-maps-2.htm, 2017. Accessed 2019/01/07.

[19] J. Wanga, H. Hea, and D. V. Prokhorovb, "A Folded Neural Network Autoencoder for Dimensionality Reduction," tech. rep., Department of Electrical, Computer and Biomedical Engineering, University of Rhode Island, Toyota Research Institute NA, 2012.

[20] J. An and S. Cho, "Variational Autoencoder based Anomaly Detection using Reconstruction Probability," tech. rep., 2015.

[21] C. Hu, X. Hou, and Y. Lu, "Improving the Architecture of an Autoencoder for Dimension Reduction," tech. rep., School of Information Science and Engineering, 2014.

[22] Z. Chen, C. Yeo, B. Lee, and C. Lau, "Autoencoder-based Network Anomaly Detection," tech. rep., Computer Network and Communication Graduate Lab, 2018.

[23] A. Salim, "Anomaly detection with auto-encoders: How we used it for cervical cancer detection." https://medium.com/@ally$_2$0818/anomaly-detection-with-auto-encoders-how-

we-used-it-for-cervical-cancer-detection-bdae74cbf05a, 2018. Accessed 2019/03/26.

[24] S. I. García, "L0 Norm, L1 Norm, L2 Norm L-Infinity Norm." https://medium.com/@montjoile/l0-norm-l1-norm-l2-norm-l-infinity-norm-7a7d18a4f40c, 2018. Accessed 2019/04/01.

[25] D. Shiffman, "Vector magnitude normalization." https://www.khanacademy.org/computing/computer-programming/programming-natural-simulations/programming-vectors/a/vector-magnitude-normalization, 2019. Accessed 2019/04/09.

[26] D. Oehm, "PCA vs Autoencoders for Dimensionality Reduction." https://www.r-bloggers.com/pca-vs-autoencoders-for-dimensionality-reduction/, 2018. Accessed 2019/02/27.

[27] Y. Shevchuk, "NeuPy- Neural Networks in Python." http://neupy.com/pages/home.html, 2015-2019. Accessed 2019/02/01.

[28] AltexSoft, "How to Organize Data Labeling for Machine Learning: Approaches and Tools." https://www.altexsoft.com/blog/datascience/how-to-organize-data-labeling-for-machine-learning-approaches-and-tools/, 2018. Accessed 2019/05/03.

[29] D. Powers, "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness Correlation," *Journal of Machine Learning Technologies*, no. 2 (1), p. 37–63, 2011.

[30] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets.," *PLoS One*, vol. 10, no. 3, 2015. ROC Analysis in Pattern Recognition.

[31] AltexSoft, "Machine Learning Project Structure: Stages, Roles, and Tools." https://www.altexsoft.com/blog/datascience/machine-learning-project-structure-stages-roles-and-tools/, 2018. Accessed 2019/05/03.

[32] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[33] F. Chollet *et al.*, "Keras." https://keras.io, 2015.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn:

Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

Appendix A

Tools Used

All code was written in Python 3.7.1. The main libraries used were Tensorflow (32) (version 1.13.1), an open source platform for machine learning, Keras (33) (version 2.2.3) a deep learning library enabling fast experimentation, NeuPy (27) (version 0.8.1), a Python library for prototyping and building neural networks, and Scikit-learn (34) (version 0.20.1), a tool for data analysis. The GPU used was NVIDIA GeForce GTX 1050 Ti.

Master's Theses in Mathematical Sciences 2019:E23
ISSN 1404-6342

LUTFMA-3381-2019

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/