Classification of Prognosis in Breast Cancer Patients from AMCL Analysis using Machine Learning Techniques

Master's Thesis in Mathematical Statistics

Ola Andersson and Johan Ruuskanen

September 6, 2017

Supervised by
Prof. Andreas Jakobsson[1], Prof. Jonas Erjefält[2] and Prof. Kalle Åström[1]

[1]Centre for Mathematical Sciences, Lund University, Sweden.
[2]Faculty of Medicine, Lund University, Sweden and Medetect AB, Lund, Sweden

Acknowledgements

We would first like to thank our supervisor Professor Jonas Erjefält and the rest of the team at Medetect for providing the data and giving us the opportunity to do this thesis. A special thanks to Jimmie Jönsson and Caroline Sandén for preparing the data packages, giving valuable comment and explanations for our progress and answering our different medically related questions. We would also like to thank our head supervisor Professor Andreas Jakobsson at the Department of Mathematical Statistics for preparing the project and guiding us through the entire thesis process, and our third supervisor Professor Kalle Åström at the Department of Mathematics for giving helpful tips and guidance in the area of convolutional neural networks. Lastly we would also like to thank Professor Mårten Fernö at the Department of Oncology and Pathology for giving us valuable feedback and explanations regarding breast cancer.

Abstract

Predicting the development of distant metastasis for breast cancer patients is of high importance for both the patient and the medical staff. The current best method for prediction is the use of handcrafted histopatological features. The aim with this study is to explore how well Additive Multiple Labelling Cytochemistry (AMLC) stained core biopsy images can predict the development of distant metastasis. For this, two cohorts of a total of 488 patients are investigated, each patient having 1-4 images of AMLC stained core biopsies (of size 2 mm) from the tumour area and AMLC features extracted from those images. Each patient is also supplied with the handcrafted histopatolohical features for reference. Both the images and numerical AMLC features extracted from the images contain information on the immunological response of the patient which in turn has been shown to have potential of good predictive ability. The images were analyzed using convolutional neural networks and the AMLC features with a support vector machine, random forest and linear discriminant analysis classifiers. We show that the convolutional neural networks and the numerical classifiers achieve similar performance with an area under the receiver operating characteristic curve (AUC) value of approximately 0.60, which is worse than the result achieved on the histopathological features, which gave an AUC value of approximately 0.74. Further, we show that no difference in prediction can be observed with using AMLC images containing immunological features as compared to AMLC images not containing any immunological features. Finally, we show that an ensemble of the classifiers for the AMLC features and the images gives no significant boost to performance in terms of AUC value. Data from 488 patients were used in the study; the results indicate that the sample size was too small to capture the variance present between patients. Furthermore the core biopsies were most likely too small to capture the behaviour of the tumour. For future studies, it is recommended to increase the number of patients and the size of the core biopsies.

Keywords: Breast Cancer, AMLC, Metastasis Prediction, Convolutional Neural Network, Machine Learning.

Abbreviations

A summary of abbreviations used in the thesis.

AMLC	Additive Multiple Labelling Cytochemistry
PCA	Principal Component Analysis
t-SNE	t-Distributed Stochastic Embedding
SFS	Sequential Feature Selection
SVM	Support Vector Machine
RBF	Radial Basis Function
RF	Random Forest
LDA	Linear Discriminant Analysis
ANN	Artificial Neural Network
MLP	Multi-Layered Perceptron
CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit
TP/TN	True Positive/Negative
FP/FN	False Positive/Negative
ROC	Reciever Operating Characteristic
AUC	Area Under Curve

Notations

A list of notations used in the thesis.

$\boldsymbol{X}$	Data matrix
$\tilde{\boldsymbol{X}}$	Dimension reduced data matrix
$\boldsymbol{Y}$	Label matrix
$\boldsymbol{x}(n)$	Observation n
$y(n)$	Label of observation n
$\hat{y}(n)$	Predicted label of observation n
$\boldsymbol{C}$	Covariance matrix
μ	Mean
σ	Standard deviation
T	Training set
V	Validation set
S	Test set
B	Batch set
$L(\cdot)$	Loss function
$f(\cdot)$	Model function
$\varphi(\cdot)$	Activation function
$g(\cdot)$	Layer function
$\boldsymbol{w}^{(m)}$	Weights of layer m
$\boldsymbol{x}^{(m)}$	Output of layer m
η	Learning rate
α_R	L1/L2 regularization parameter
α_M	Momentum parameter

Contents

1 Introduction

Cancer is a common endemic disease that affects many people world wide, with breast cancer being the most common type for women. The disease represents 25.2% of all diagnosed cancer forms for women and in Sweden alone about 8000 persons are diagnosed with breast cancer each year. Breast cancer has a fairly low fatality rate, that is, a patient often survives the disease after treatment. However, since the disease is so common the mortality rate in the overall population is still high, see [1], [2].

Assessing the prognosis and deciding the type of treatment for a breast cancer patient is far from a trivial matter as it is not a single disease. Instead, breast cancer is highly heterogeneous and can be categorized into a multitude of subgroups depending on a wide range of factors. The different subgroups all have different prognoses and treatments associated with them. Knowledge of the type of cancer is thus required for deciding the correct treatment, see [3]. However, the heterogeneity of the disease is vastly complex and the subtypes of breast cancer are still poorly defined and often overlapping. A specific treatment for a patient with a certain subtype of breast cancer is not always effective, improper treatment will cause unnecessary side effects and economic cost. Due to this, it is important to correctly characterize the tumour. Furthermore, current methods for categorizing cancer subtypes only provide limited information for the prognosis as well. Thus novel methods are called upon to better map the complexity of the disease in order to improve disease prediction and treatment strategies as discussed in [4].

1.1 Background

When categorizing breast cancer, a pathologist typically examines a large biopsy sample that is sectioned and stained with the routine histological stain Hematoxylin and Eosin (H&E) for microscopic examination where a handful of histopathological features are taken in consideration.

To further categorize the breast cancer, the presence of certain molecular biomarkers are examined through immunohistochemistry; biomarkers are in this case proteins associated with certain behaviors of the tumour. Especially four biomarkers are of particular importance for prognosis and the choice of treatment. These four biomarkers are; Estrogen Receptor (ER), if the cancer tumour has receptors for the hormone class Estrogen. Human Epidermal growth factor Receptor-2 (HER-2), if receptors for the specied growth hormone are present. progesterone Receptor (PGR), if receptors for progesterone are present and the protein Ki-67 measuring proliferation (cell division) of the cancer cells. For further information see [1], [3].

A high degree of proliferation and cancer cell expression of hormone receptors is often treated more extensively to reduce the risk of relapse. Altogether, the microscopic (i.e., histological) analysis provides the basis for a histopathological grade of the tumour which describes the malignancy of the cancer, the size of the tumour itself etc. as explained in [1]. Conducting the measurements of the cancer tumour is currently a resource intensive process as it requires the experience of a specialized pathologist.

The analysis does not provide sufficient information for prognosis of cancer outcome, mainly since the outcome vary largely among patients within the same histopathological tumour stage.

The analysis however assumes that the tumour progression is largely a cell-autonomous process, which focuses only on the cancer cells and does not consider the host's immune response and the pattern of tumour-infiltrating immune cells, which might affect the predictability of the disease outcome as shown in [4]. For example, the total number of T cells (a type of white blood cells) has been shown to be related to good prognosis. The total immune response has also been shown to correlate with the prognosis, i.e., a weak response is correlated with a bad prognosis shown in [5]. There are several more studies that show that the development of cancer is affected by the immune system. In the case for colorectal cancer, it has been shown that the immune response in the tumour region is a better prognostic predictor for the staging of the cancer than histopathological methods. The immune response also plays an important role in preventing the tumour from recurring and proliferating in colorectal cancer as shown in [6], [7], [8]. Hence, this type of immune analysis can most likely be further improved if more types of immune cells are analyzed and the complete tumour structure is also included in the analysis and not just the tumour cells.

In order to more accurately measure the immune response in tumours along with the classical biomarkers, new staining methods that simultaneously can stain multiple cell markers are needed (current standard immunohistochemistry is limited to 2-3 markers).

One such method for tissue staining is Additive Multiple Labelling Cytochemistry (AMLC). AMLC is a novel staining approach invented by the company Medetect in Lund. This new technique allows for simultaneous visualization and automatic computerized segmentation and quantification of all major cell populations in a routine tumour section. In practice, this means that several selected immunological and standard biomarkers, smooth muscle tissue and the cancer tumour can be seen simultaneously on high resolution digital images.

Currently, there are no standard procedures for analyzing the complexity of AMLC -stained biopsy samples for prognostic prediction of outcome. The images can however be analyzed from a machine learning perspective, where the prediction of outcome can be viewed as a classification problem. The current leading technique in image classification is called Convolutional Neural Network (CNN) and has been growing in popularity since Alexander Krizhevsky in 2012 competed in the ImageNet Challenge, completely outperforming his opponents. The competition is held annually where teams compete in different image classification related events by submitting a classification algorithm each. During the competition in 2012 the CNN AlexNet as shown in [9], was submitted and it dropped the record error rate from around 25 % down to approximately 16 %. The following years the error rate has dropped even further as more advanced CNNs emerged. As the popularity has increased, so has the application of CNN in different fields of science.

1.2 Related Work

CNNs have been used for estimating the Gleason grading (a microscopic scoring system) in patients with prostate cancer with promising results. In a study by Grummeson et al. [10], 213 Hematosin & Eosin (H&E) stained microscopic prostate images were used to train a network to automatically grade samples. An error rate of 7.3 % was obtained for the grading. It was concluded that a larger data set probably could even further reduce the error rate.

The method has also been used for breast cancer patients with the goal of detecting mitosis (cell division) in histopathological images. In Cireşan et al. [11], H&E stained images were also used with the goal to classify each pixel as being mitosis or not. Two networks were used and yielded an F-score of 0.782, significantly higher than other comparative models.

A similar study when it comes to predicting the outcome, and thus somewhat of a benchmark in terms of result, was conducted on 97 breast cancer patients of which 46 developed distant metastasis within 5 years. In a study by Edén et al. [12], one of the goals was to classify whether or not a patient would develop metastasis. The best result was obtained from classical histopathological markers, e.g. age, histopathological grade, tumour size etc. The area under the receiver operating characteristic (AUC) was 0.78 when using artificial neural networks for the classification.

1.3 Hypothesis

The overarching goal of this Master's Thesis is to investigate the potential benefits of using AMLC analysis in conjunction with machine learning techniques to predict the outcome for breast cancer patients. As of the present date, no studies have been reported with regard to this purpose. Thus, due to the absence of prior research the investigations and findings should be regarded as a pilot study of the potential usage of AMLC analysis in predicting prognostic cancer outcome.

Whether or not a treatment is successful, and thereby the outcome for a breast cancer patient, is not well defined. We will in this study use the following definition for disease outcome; if a patient after treatment does not show any signs of tumour metastases within a five year period, the outcome is considered good. If this is not the case, and thus distant metastasis develops within the 5 year time frame, the outcome is considered bad. This distinction is deemed sufficient and is the same definition used by Edén et al. [12].

Using this definition of outcome, we can make the hypothesis that AMLC stained core biopsies (2 mm) containing both immunological and standard biomarkers can be used to better predict the development of distant metastasis in breast cancer patients. In order to test the hypothesis, the following three questions will be explored:

- How well can numerical features extracted from AMLC stained core biopsies be used to predict the development of metastasis in breast cancer patients?
- How well can images of pre-segmented and color-coded AMLC stained core biopsies be used to predict the development of metastasis for breast cancer patients using convolutional neural networks?
- How important are the immunological biomarkers in the AMLC staining when predicting the development of metastasis in breast cancer patients?

It should be noted that there are several other outcomes and classification problems that can be investigated using the data provided. Due to limitations in hardware and time only one binary classification problem was deemed to be sufficient as initial investigation of the data. Furthermore, there are several software packages that can be used for machine learning and particularly

CNN. Matlab was selected as the main software program since it offers built-in packages and functions that are good enough for the purpose of the thesis. Most packages offer similar functions, however some are more flexible for pre-trained CNN. Both authors are also comfortable using Matlab.

The target audience is the company Medetect, the inventors of the AMLC analysis, that provided the data for this project as well as future students interested in CNN.

1.4 Data

All data used was provided by Medetect, both numerical and color-coded AMLC stained images. The data sets are part of a large Vinnova/Swelife-sponsored research project which apart from Medetect also involves Departments at the Medical Faculty at Lund University (Prof Jonas Erjefält, medical immunology expert and Medetect founder and Prof Mårten Fernö, Dept of experimental Oncology, coordinator of the large clinical study that provides the basis for the clinical characterization and tissue samples). The clinical studies and the present data analysis have been approved by the Lund ethical committee for human medical research.

In total the data received consists of 488 patients diagnosed with breast cancer in two different cohorts, but originates from three independent studies. The first cohort consists of 277 patients, which all had been given an adjuvant treatment, i.e., an additional treatment to lower the risk of relapse after surgery, in this case with the hormone therapy Tamoxifen as seen in [13], [14]. The second cohort consists of 211 premenopausal patients of which 29 patients had been given an adjuvant treatment as seen in [15]. All patients were monitored after diagnosis and had routine check ups in order to detect possible metastasis. It is thus known if a patient developed metastasis of not. A short descriptive statistics about the different cohorts can be seen in table 1.

Table 1: Values of different variables in the two cohorts.

	Cohort 1	Cohort 2	Joint
Type	Pre- and Post-menopausal patients	Premenopausal patients	Pre- and Post-menopausal patients
Number of patients	277	211	488
Mean cores per patients	2.02	1.98	2.00
Mean Age	60.3	46.0	54.1
Met. 5 years	23.1 %	15.6 %	19.9 %
Met. 10 years	29.2 %	22.3 %	26.2 %

The data can be divided in to three different categories. The first type consists of images. Each patient has between one and four images (in most cases two) of an AMLC stained tissue sample obtained from normally two 2 mm core biopsies that were selected from the larger original tumour sample. In total, there are 977 AMLC-images (2 mm cores) available for analysis. Each image contains information about several selected immunological and structural cell markers as well as standard biomarkers in the form of a specific color. It is possible to select which biomarkers to export from the AMLC analysis, this makes it possible to create several packages of images containing different biomarkers. Several image packages was exported for this project as seen in table 2. An example image can be seen in Figures 1 and 2. For further details about the different biomarkers see Appendix A.

Table 2: Different image packages available for analysis. Image package 1 contains all the available biomarkers while the rest of the packages consist of different combinations of those markers. Furthermore image package 6 does not contain any immunological biomarkers and image package 7 only contains immunological biomarkers.

Image package	Content
1	Cytokeratin, Ki67, p63, CD8, CD3, CD11C, CD20, CD31, CD68, CD138, MPO, Tryptase, Langerin, SMA, HTX, FOXP3, Vimentin, PGP
2	Cytokeratin, Ki67, p63, CD8, CD3, CD20
3	Cytokeratin, Ki67, p63, CD8, CD68, CD31
4	Cytokeratin, Ki67, p63, CD8, CD3, MPO
5	Cytokeratin, Ki67, p63, CD8, Tryptase, Langerin
6	Cytokeratin, Ki67, p63, CD31, SMA
7	CD8, CD3, CD11C, CD20, CD68, MPO, Tryptase, Langerin, FOXP3

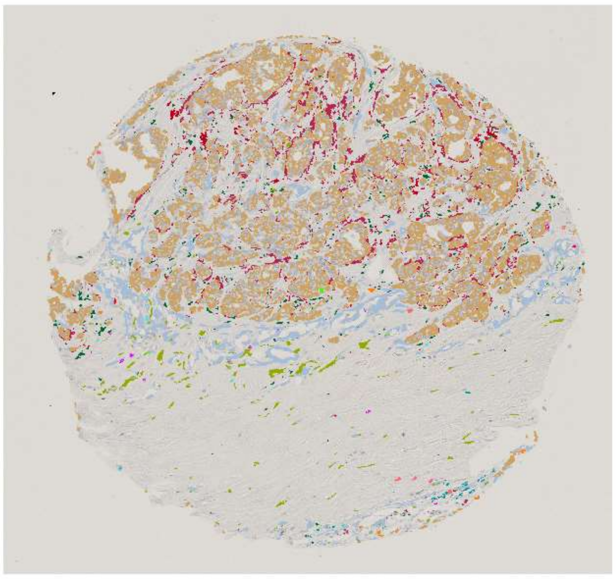


Figure 1: An AMLC stained image of a core sample biopsy from one of the two cohorts in the data set. This particular image has been stained according to Image package 1. The orange area is Cytokeratin, which is the biomarker used for detecting cancer tissue.

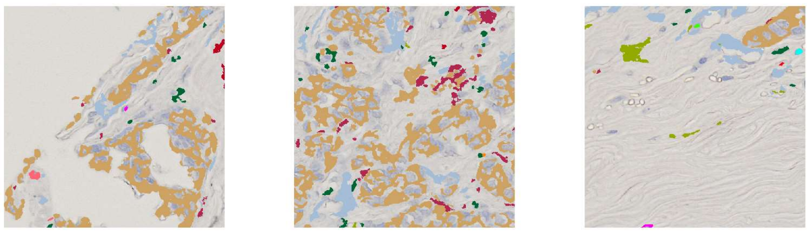


Figure 2: A closer look at three different positions in Figure 1 to discern all the small features.

The second type of data consists of 466 numerical features extracted from the AMLC stained core biopsies. These features consist of different measurements from the 18 available biomarkers, which were used to create image package 1. The measurements used to create the final features include for example the area of the different markers both compared to the whole image and within a defined vicinity of the tumour region, and different area ratios between the biomarkers. The numerical data is thus in a sense a different perspective of the information in the stained images, which has been quantified into numerical values. The quantification implies that most of the spatial structure of the image is lost. To avoid confusion, this category will be referred to as the *AMLC features.*

The last type of data is also numerical and consists of more standard histopathological features extracted from a much larger original tumour biopsy than the available core biopsy images. The data in this category includes the variables patient age, tumour size, lymphocytic infiltration, ER, PGR, HER-2, Ki67 and histopathological grading. These are roughly the same features used by Edén et al [12]. This type of data will be referred to as the *clinical data.*

2 Theory

2.1 Machine Learning

Machine learning is in its simplicity an algorithm that can learn from data. More often than not these data sets can be quite large with many features or dimensions. For a human expert, it becomes a more and more difficult task to draw accurate conclusions or extract meaningful information from a data set as the number of samples and dimensions increases. The idea is then to have the algorithm "learn" to distinguish different patterns in the data either given prior outcome, called *Supervised Learning*, or with no prior outcome, called *Unsupervised Learning.*

In this chapter, we will introduce the theory of the different techniques that will be used in this thesis. First methods for robust prediction error will be presented, then dimensionality reduction techniques, numerical methods for classification and later convolutional neural networks for image classification and lastly how the models will be evaluated. The methods presented will mainly be supervised learning since the outcome of the data is prior knowledge. The outcome used for supervised learning is referred to as *Labels.*

2.2 Training, Testing and k-fold Cross-Validation

In order to learn a model to distinguish different classes, it is common to first split the original data set in three sets called training, validation and test. The training set is used to fit the model, validation for estimate of prediction error and the test for estimate of out-of-sample error for the final selected model. Doing this will generate robust estimates on prediction errors for unknown data. For this project, the training set is defined as T, the validation set as V and the test set as S.

The size of each data set is difficult to select since it is dependent on the specific data set and the size needed for training. A general rule of thumb is to split it into 50 % training and 25 % each for validation and test, as explained in [16]. This method is dependent on the existence of enough data to perform this split. Another approach is to use cross-validation. Regardless of the method, a loss function $L(Y, \hat{f}(\boldsymbol{X}))$ needs to be introduced, where Y is a vector containing the labels for the different classes, $\hat{f}$ is the model fitted on the training data and $\mathbf{X}$ is a input matrix containing the different features. The loss function estimates the prediction error, for both a regular split and for cross-validation.

The K-fold cross-validation splits the data in K sets, :th model is then fitted K times. The kth fitting uses $K-1$ sets to fit the data and uses the K set for validation, which done for all sets, that is for $k = 1 \dots K$. The prediction error is then created as an average of all the K prediction errors, which is defined as

$$CV(\hat{f}) = \frac{1}{K}\sum_{i=1}^{K} L(Y_i, \hat{f}^{-k(i)}(X_i)) \ , \tag{1}$$

where $\hat{f}^{-k}$ is the fitted model without the kth data set. X_i and Y_i are the out-of-sample data sets which are not used for fitting the model. This method is called *leave-one-out* cross-validation.

The number of folds used is a trade-off between bias and variance, where the leave-one-out method usually has low bias but can have high variance.

The number of K is usually set to 5 or 10, see [17]. K-fold cross-validation could also be used for fine-tuning parameters. $f(X, \alpha)$ is a set of models, where the tuning parameter α is indexing the models. Then, $\hat{f}(X, \alpha)^{-k}$ is the αth model fitted without the kth data set. The cross-validation function can then be defined as (see [16])

$$CV(\hat{f}, \alpha) = \frac{1}{K} \sum_{i=1}^{K} L(Y_i, \hat{f}^{-k(i)}(X_i, \alpha)) \ . \tag{2}$$

The tuning parameter that minimizes the error is selected and used as final model. The final model is then fitted to the data as in Eq. 1 in order to obtain an estimate of the prediction error. A visualization of the split for a k-fold cross-validation with $K = 5$ can be seen in Figure 3.

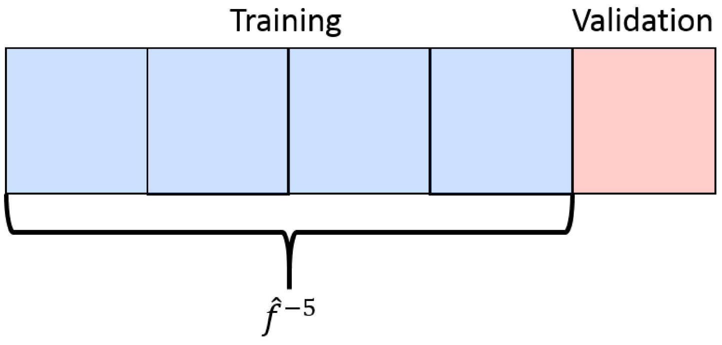


Figure 3: An example of a k-fold cross-validation with $K = 5$. The model is trained using all data except the last set. This is thus the last iteration of the fitting of a model.

2.3 Dimensionality Reduction and Data Visualization

In classification problems, it is not unusual for the data set to contain a huge number of features. It has been shown [18] that given a classifier the predictive accuracy will decrease if the number of dimensions increases beyond a certain threshold. This reoccurring problem is often referred to as *the Curse of Dimensionality.*

In order to utilize data with too many dimensions with respect to the sample size, *dimensionality reduction* should be performed to obtain a better feature subset. Apart from the increased accuracy of a classifier, dimensionality reduction can be used to reduce the data down to 2 or 3 dimensions for visualization which allows for an intuitive inspection of the feature space.

There are two main ways of performing dimensionality reduction, *feature extraction* and *feature selection.* Feature extraction reduces the number of dimensions by transforming the data into a subspace with fewer dimensions whilst feature selection tries to find a subset of features from the already available set of features, see [19].

For this thesis, two feature extraction methods, *principal component analysis* (PCA) and *t-distributed stochastic neighbour embedding* (t-SNE), and one feature selection method, *sequential feature selection* (SFS), are considered.

2.3.1 Principal Component Analysis

PCA was first introduced by Pearsson [20] and later developed and named by Hotteling [21]. It is a very common technique for dimension reduction and visualization of high dimensional data. The goal is to reduce the number of dimensions by projecting the features onto a space with fewer dimensions and still keep the maximum variance of the projected data. This is conducted by an orthogonal linear transformation of a data matrix, $\mathbf{X}$. The transformed data is projected onto a new coordinate system, where the first coordinate contains the largest amount of variance and the second coordinate contains the second largest amount of variance, and so on. This can be defined as

$$\boldsymbol{C} = \mathbf{W}\Lambda\mathbf{W}^T \ , \tag{3}$$

where $\boldsymbol{C}$ is the covariance matrix of $\boldsymbol{X}$ and $\mathbf{W}$ is a matrix containing each eigenvector stored column-wise. Λ is diagonal matrix containing the eigenvalues, $\lambda_1 > \lambda_2 > ... > \lambda_N$. The dimension wise reduced data matrix, $\tilde{\mathbf{X}}$ can thus be described as

$$\tilde{\mathbf{X}} = \mathbf{X}\mathbf{W}_d \ , \tag{4}$$

where $\tilde{\mathbf{X}}$ is a Nxd matrix and $\mathbf{W}_d$ contains d principal components.

2.3.2 t-Distributed Stochastic Neighbour Embedding

Complex datasets with many dimensions and classes might contain nonlinear manifolds in the high dimensional space. These nonlinear manifolds would then not be discovered by using simple linear techniques such as the PCA.

In a paper from Tillburg University [22], comparisons between a multitude of different commonly used non-linear dimensionality reduction techniques and PCA have been performed. These tests compared the different techniques on both artificial data created to include non-linear manifolds and a number of real data sets. Not surprisingly, the non-linear techniques outperformed PCA on the artificial data sets. However, on the different real data sets PCA managed to either outperform or at least perform as well as the best performing non-linear techniques.

However, the algorithm t-SNE presented by van der Maaten et al. was not a part of the non-linear techniques tested. In particular, t-SNE has been shown to outperform other algorithms in data visualization, as seen in [23]. It should also be noted that van det Maaten et al. explains in the paper that it is not really known how well t-SNE will perform on general dimensionality reduction.

There are also other reasons for including t-SNE for comparison. For one, the largest principal components correspond to the largest variance in the data set. Large variance might be an indication on clustering but this is not always the case, e.g. two parallel hyperplanes that are close to one another as we can see in Appendix B. Another reason is that there *might* be nonlinear manifolds in this data that PCA will not be able to capture, even though the tests performed at Tillburg University [22] show that this is unlikely.

In short, the t-SNE algorithm depends on modeling both the data points and the reduction map points as two different conditional distributions in order to obtain a measure on the similarity between the points inside the two sets. For the data points, the Euclidean distance between two points is converted into a conditional probability using the Gaussian distribution to represent similarity. The further the distance, the lower the conditional probability becomes.

$$p_{j|i} = \frac{\exp\left(-||x_i - x_j||^2/2\sigma^2\right)}{\sum_{k\neq i} \exp\left(-||x_i - x_k||^2/2\sigma^2\right)} \quad . \tag{5}$$

In a similar manner, we can compute the similarities between the reduced map points $\tilde{x}_i$ and $\tilde{x}_j$ as conditional distributions but using t-distributions with one degree of freedom instead, which then becomes

$$q_{j|i} = \frac{\left(1 + ||\tilde{x}_i - \tilde{x}_j||^2\right)^{-1}}{\sum_{k\neq i} \left(1 + ||\tilde{x}_i - \tilde{x}_k||^2\right)^{-1}} \quad . \tag{6}$$

In the original SNE, a Gaussian distribution is used for $q_{j|i}$ as well, but to overcome some limitations explained by van der Maaten et al. [23] the t-distribution is used instead.

If the points in the reduced map completely model the similarity between the data points, the conditional distributions $p_{j|i}$ and $q_{j|i}$ will be equal. A measure of how well the reduced map models the similarity between the data points can be obtained via the Kullback-Leibler divergence. Thus in total, the overall ability to model the similarity is formed as the sum of all of the Kullback-Leibler divergences between the data points, calculated as

$$L = \sum_i KL((P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \quad . \tag{7}$$

The smaller the loss function L is, the more similar the reduced map is to the original data points which implies that the reduced map can now be found by minimizing L. This in turn is done using an iterative optimizer, such as gradient descent.

2.3.3 Sequential Feature Selection

Instead of transforming the data to a subset of lower complexity, a subset could be created by selecting a number of features vital for classification. This method is referred to as *Feature Selection*.

In order to perform feature selection, a method is needed to evaluate if a subset of features is good or not. For this, let $\boldsymbol{X}$ be the set of all features for all data, $\boldsymbol{X}_d$ a subset of d features and

L an arbitrary loss function. Finding the best feature subset $\tilde{\boldsymbol{X}}_d$ can then be reduced to the minimization problem

$$L(\tilde{\boldsymbol{X}}_d) = \min_{\boldsymbol{X}_d \subset \boldsymbol{X}} L(\boldsymbol{X}_d) \ . \tag{8}$$

The question now remains how to select and test $\boldsymbol{X}_d$ from $\boldsymbol{X}$ in order to obtain the optimal subset. A straight forward brute force approach would simply be to test all possible combinations of features. This is called a *Complete Search.* Although this method would guarantee that $\tilde{\boldsymbol{X}}_d$ is found, it is computationally unfeasible even for data sets of modest size since the number of possible combinations of features will grow exponentially. In practice, this means that the complete search is almost never used.

Instead, the features could be selected using a greedy approach by sequentially selecting and adding features to $\boldsymbol{X}_d$ that gives the best classification until no further improvements can be made. This technique is called *sequential forward search* [19]. The greedy search is a fast way of generating a good approximation for $\tilde{\boldsymbol{X}}_d$ but it is not guaranteed to find the optimal subset.

The greedy algorithm is executed as follows.

1. Start by considering an empty feature set $\boldsymbol{X}_{i=0}$.
2. For each feature ξ_j evaluate $L_j = L\left(\{\boldsymbol{X}_i, \xi_j\}\right)$.
3. The feature which gives the minimal L_j is added to $\boldsymbol{X}_i$ to create the next subset $X_{i+1} = \{X_i, \xi_j\}$.
4. Steps 2-3 are then repeated until the improvements to the total loss are too small or a maximum number of features has been selected.

2.4 Classification of Numerical Data

2.4.1 Support Vector Machine

The *Support Vector Machine* (SVM) is a useful technique for predictions in high dimensional feature space. The main idea is to separate classes with a half-space with as large margin as possible. There are several different methods and the most basic form assumes that the classes can be separated completely with a linear boundary. This is a very strong assumption and not likely to be true in the most cases. A solution to this is to introduce nonnegative slack variables, $\xi_1, \xi_2, \ldots, \xi_n$. This a less strict assumption and is referred to as soft-SVM, see [16].

If $T = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_1)$ is a training set, where $\mathbf{x}_i \in \mathbb{R}^d$ are the features with their associated classes $y_i \in \{-1, 1\}$ and β_j are parameters to be fitted, then we need to solve the following optimization problem in order to obtain the largest margin between the classes

$$\begin{aligned}
&\underset{\beta_0,\beta_1,\ldots,\beta_d,\xi_1,\ldots,\xi_n}{\text{maximize}} && M \ , \\
&\text{subject to} && \sum_{j=1}^{d} \beta_j^2 = 1 \ , \\
& && y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_d x_{id}) \geq \text{ M}(1 - \xi_i) \quad \forall \ i = 1, \ldots, n \ , \\
& && \xi_i \geq 0, \quad \sum_{i=1}^{n} \xi_i \leq c \ ,
\end{aligned}$$

where M is the margin, i.e., the distance to maximize and c is a cost parameter that describes the size of the margin around the boundary that allows observations to be on the wrong side of the boundary. Missclassification occurs when $\xi_i > 1$, which means that no more than c observations can be missclassified. More specifically, points for which $\xi_i = 0$ are classified correctly, $0 < \xi_i \leq 1$ are points that are inside the margin, but on the correct side of the boundary decision line. The size of c is a trade-off between variance and bias, were a value of c gives low variance but high bias and vice versa. A visualization of the maximization problem for 2 dimension can be seen in Figure 4. The final decision rule can be written as (see [24])

$$\begin{aligned}
\hat{f}' &= sign[\hat{f}(x)] \ , \\
&= sign[x^T\beta + \beta_0] \ .
\end{aligned}$$

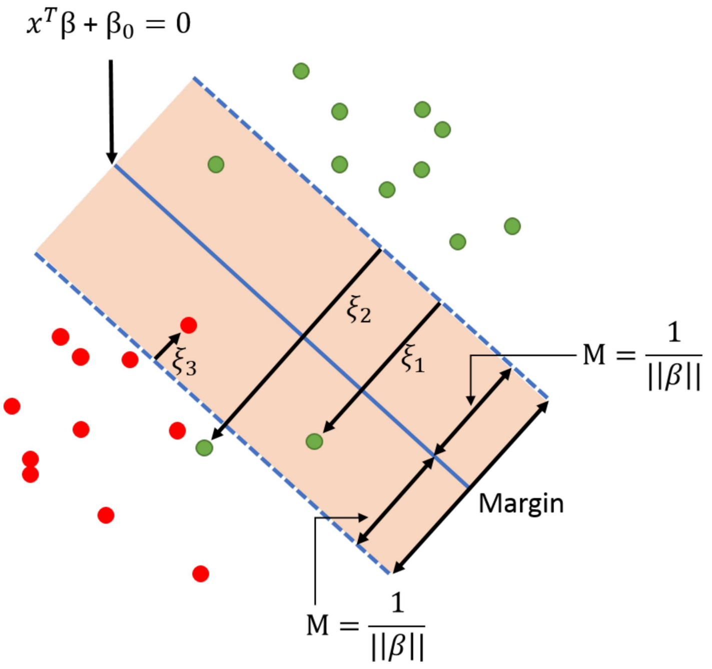


Figure 4: An example of an SVM where the observations are not completely separable. The size of the margin is decided by the tunning parameter c, see [24].

The SVM can also be altered in such a way that non-linear functions can be added in order to get non-linear boundaries. This enables the use of several new functions which can be relevant to use depending on the specific classification problem. It can be shown [24] that the optimization problem and solution can be rewritten with the help of the inner products of the input features, which in turn enable a more general function for the decision rule

$$\begin{aligned} f(x) &= h(x)^T\beta + \beta_0 \ , \\ &= \sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i)\rangle + \beta_0 \ , \\ &= \sum_{i=1}^{N} \alpha_i y_i K(x, x') + \beta_0 \ , \end{aligned}$$

where $h(x)$ is a transformation of the input features and $K(x, x')$ is a symmetric positive (semi-) definite kernel function. There are several different kernel functions that can be selected, in this thesis the *Radial Basis Function* (RBF) was selected, defined as $K(x, x') = exp(-||x - x'||^2)$. Furthermore, α_i and β_0 can be estimated by solving $y_i f(x_i) = 1$ for any (or all) x_i for which $0 < \alpha_i < c$. A large C will lead to a less smooth boundary and possible overfitting, whereas the use of a smaller c will yield smoother boundaries as explained in [16].

2.4.2 Decision Trees and Random Forest

Decisions trees is a method that uses features x_j from a data matrix $\mathbf{X}$ with j features and i observations with the goal of predicting labels from vector $\mathbf{Y}$. The tree starts with a root node and grows in to several additional nodes and leafs. At each node, a feature is used in a binary split of the input data, leading to additional nodes and/or leaves, where each leaf contains a label. Each new subset of the input is disjoint, e.g. at the root node, $\mathbf{X}$ is split into $\mathbf{X}_1$ and $\mathbf{X}_2$ such that $\mathbf{X} = \mathbf{X}_1 \cup \mathbf{X}_2$, see [25]. There are several methods for growing a tree and selecting what feature to use at each split. In this thesis, the Gini index will be used as an impurity function, that is, a measure of missclassification. Below is pseudocode presented for the growing of the tree:

1. Start at the root node.
2. For each x_j, find the split that minimizes the impurity in the new nodes and choose that split.
3. If stopping criteria is reached, stop and turn the node into a leaf.
4. Otherwise repeat from step 2.

The Gini index is defined as

$$I_G(f) = \sum_{i=1}^{J} f_i(1 - f_i) \ , \tag{11}$$

where J is the number of classes, $i \in \{1, 2, ..., J\}$ and f_i is the probability of an element belonging to label i. The function value is zero when all observations in a node belong to the same class. An example of a decision tree can be seen in Figure 5.

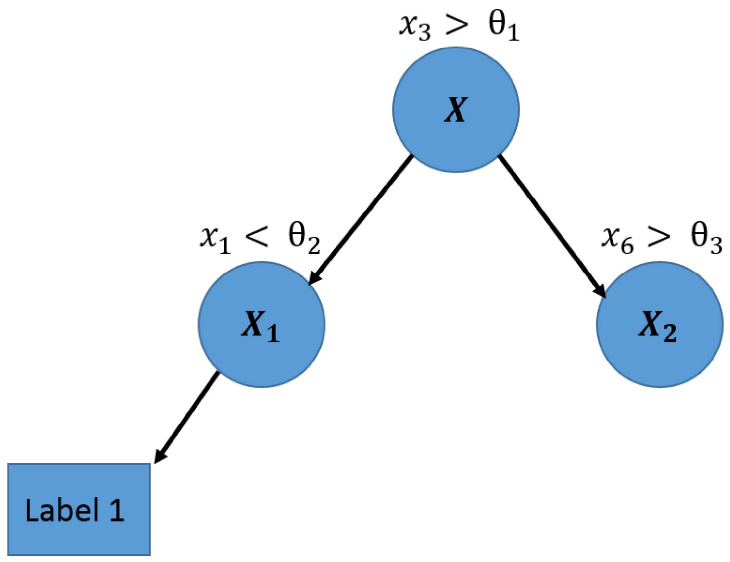


Figure 5: An example of an incomplete decision tree. At the root node the input is split with feature x_3 at a threshold level θ_1, consequently splitting the original input data $\mathbf{X}$ into two disjoint sets $\mathbf{X_1}$ and $\mathbf{X_2}$. The second split also shows when a leaf is created, where all observations are classified as belonging to label 1, thus a stopping criteria is reached.

A more robust method with less risk of overfitting is the *random forest* (RF) method, where an ensemble of trees is used to vote for the most popular class, see [26]. Below is a pseudocode for creating a random forest where B is the set of all trees in the ensemble:

Let b be a single tree in the ensemble and j the number of features. Then, $\forall b \in B$

1. Sample with replacement from $\mathbf{X} and \mathbf{Y}$.
2. Randomly select k features at each node and use for best split according to impurity function, where $k < j$.
3. If $I_G(f) = 0$, stop and turn the node into a leaf. It is also possible to use other stopping conditions, such as the minimum number of observations in a leaf.
4. Otherwise repeat from step 2.

The forest error rate is dependent on the size of randomly selected features k. Increasing the number will lead to higher strength of each individual tree but will also increase the correlation between any two trees. Increasing the correlation will also increase the forest error rate as explained in [26].

2.4.3 Linear Discriminant Analysis

The idea with *Linear Discriminant Analysis* (LDA) is to find weights w for the function

$$y(x) = w^T x \ , \tag{12}$$

that projects the observations x onto a line, see [27]. The observations are then classified as a certain class depending on the sign of the function, e.g. if $y(x) > 0$ then x is classified to belong to class one. There are several variations of LDA. A simple variant is called Fisher's linear discriminant. The proposal for this type is to maximize a function that will give a large separation between the projected class means while also giving a small variance within each class as described in [27]. The proposal is to maximize the expression

$$J(\boldsymbol{w}) = \frac{w^T \boldsymbol{C}_B w}{w^T \boldsymbol{C}_W w} \ , \tag{13}$$

where $\boldsymbol{C}_B$ is the between-class covariance matrix and $\boldsymbol{C}_W$ is the total within-class covariance. The covariance matrices are calculated as

$$\boldsymbol{C}_B = (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T \ , \tag{14}$$

$$\boldsymbol{C}_W = \sum_{n \in \mathcal{C}_1} (x_n - \boldsymbol{\mu}_1)(x_n - \boldsymbol{\mu}_1)^T + \sum_{n \in \mathcal{C}_2} (x_n - \boldsymbol{\mu}_2)(x_n - \boldsymbol{\mu}_2)^T \ , \tag{15}$$

where $\boldsymbol{\mu}_k$ is the mean of each class k. Observations that belong to class one are defined as $\mathcal{C}_1$, thus the total number of observations in class one is $n \in \mathcal{C}_1$ and similar for class two. In order to maximize $J(\boldsymbol{w})$ it is differentiated with respect to $\boldsymbol{w}$ and becomes

$$(w^T \boldsymbol{C}_B w)\boldsymbol{C}_W w = (w^T \boldsymbol{C}_W w)\boldsymbol{C}_B w \quad \Rightarrow \tag{16a}$$

$$w \propto C_W^{-1}(\mu_2 - \mu_1) \ . \tag{16b}$$

Since $w^T \mathbf{C_B} w$ and $w^T \mathbf{C_W} w$ are only scaling factor, they are dropped. This means that w is proportional to the difference of class means as shown in [27].

2.5 Classification of Image Data using Artificial Neural Networks

While the classifiers discussed so far have a long history with providing accurate classifications on numerical data, a problem arises when this is transferred to images. If each pixel value is denoted as a feature, it would leave the algorithm with a very high complexity and also make the algorithm extremely sensitive to translations.

A solution is to manually define different measurements in the images to create hand crafted features. Although hand crafted features do work, crafting them is tedious work and requires some previous knowledge of the data for intuition on which features might be relevant or not. Instead of manually creating features, a *Convolutional Neural Network* (CNN) can be used to automatically find image features.

In order to properly introduce CNN, the notion of an *artificial neural network* (ANN) must be introduced. An ANN seeks to mimic the parallel processing power constructed from small individual processing units as seen in the neurons of a central nervous system. It is worth noting that an ANN is not a simulation of the brain and as Haykin [28] writes, an ANN only resembles the brain in two aspects; that the network knowledge is obtained through learning and that the interconnected strengths between neurons are the storage of the acquired knowledge.

2.5.1 The Multi-Layered Perceptron

First things first, consider the individual processing unit of the central nervous system, the neuron. It takes a number of inputs from other neurons and produces an output according to some rule.

To emulate this behaviour an *artificial neuron* is defined as shown in Figure 6. As shown the artificial neuron takes in a number of input signals x_k weighted with w_k. The weighted inputs are summed and a bias b is added. This sum is then passed through an activation function $\varphi(\cdot)$ and then sent out as z. This gives the artificial neuron the output function (see [29])

$$z = \varphi\left(\sum_{k=1}^{m} w_k x_k + b\right) \quad . \tag{17}$$

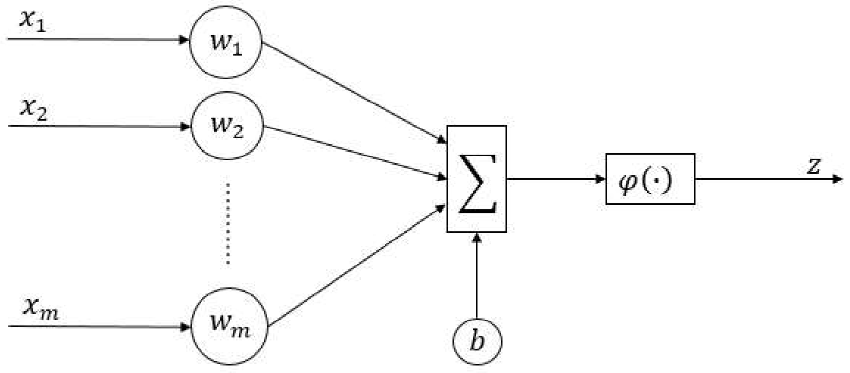


Figure 6: A simple image describing the flow through the artificial neuron.

The next step is to combine artificial neurons into a simple network of two layers as shown in Figure 7. A network of this configuration is referred to as a *perceptron*. The perceptron has one input layer and one output layer. The input layer simply consists of the input values x_k and the output layer is stacked artificial neurons with output z_i. From this, it can be seen that a perceptron with a single output is no different from a single artificial neuron. For an arbitrary perceptron, the output of node i, z_i, is given by (see [29])

$$z_i = \varphi\left(\sum_{k=0}^{m} w_{ik} x_k\right) \quad . \tag{18}$$

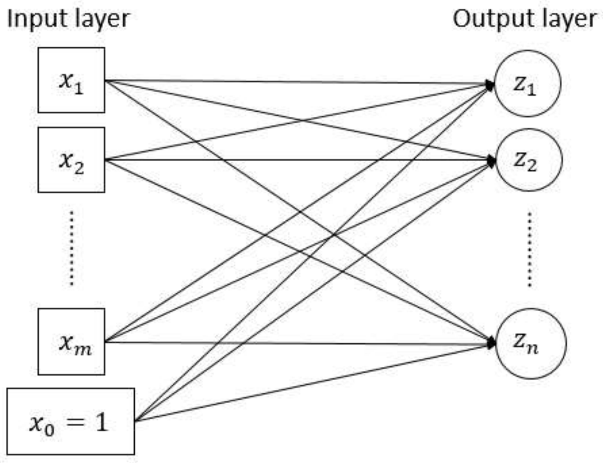


Figure 7: Descriptive overview of a perceptron with m inputs and n outputs. The bias term has been concatenated into x_0 and is now allowed to have weights as any other input.

Further, we can add new layers in between the input and output layers. The perceptron is then transformed into a *multi-layered perceptron* (MLP) as shown in Figure 8. Given the network in Figure 8, we denote the activation function for the hidden layer as $\varphi_h(\cdot)$ and the activation function for the output layer as $\varphi_o(\cdot)$. The output can then be written as

$$h_j = \varphi_h\left(\sum_{k=0} w_{jk}^{(1)} x_k\right) \quad , \tag{19}$$

$$z_i = \varphi_o\left(\sum_{j=0} w_{ij}^{(2)} h_j\right) \quad , \tag{20}$$

$$\Rightarrow \qquad z_i = \varphi_o\left(\sum_{j=0} w_{ij}^{(2)} \varphi_h\left(\sum_{k=0} w_{jk}^{(1)} x_k\right)\right) \quad . \tag{21}$$

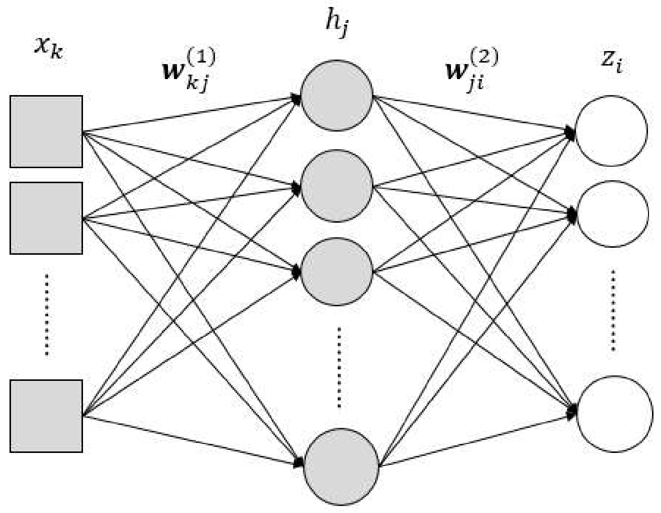


Figure 8: Schematic of the multilayered perceptron, here with one hidden layer.

Adding more layers is a simple task, just replace x_k with a new hidden layer that takes x_k as input.

Finally, the output of the network is often scaled using a *softmax function.* This is done to map the output values into classification probabilities for the different classes. For an output z_i, the softmax function is defined as (see [29])

$$p_i = \frac{e^{z_i}}{\sum_k e^{z_k}} \qquad \text{where} \quad \sum_i p_i = 1 \quad \forall i \ . \tag{22}$$

The final classification $\hat{y}$ is then defined as the output class i which receives the highest probability p_i.

2.5.2 Convolutional Neural Networks

The MLP itself is no better than the classifiers discussed for the numerical data in the area of image classification. A few modifications need to be made to the simple MLP architecture in order to transform it into a CNN, which solves the problems with image translation and high complexity.

Lets introduce a new type of layer called *convolutional layer.* A convolutional layer is defined as a three dimensional convolution with the weights as the different filter kernels. The three dimensions corresponds to the width, height and color channels of an image. Learning the filter kernels would then be an automatic way of extracting features from the images. The mathematical convolution operation is defined as

$$(I * F)[x,y,z] = \sum_{x'=0}^{k-1} \sum_{y'=0}^{k-1} \sum_{z'=0}^{k-1} I(x-x', y-y', z-z')F(x',y',z') \ , \tag{23}$$

where I is a three dimensional input image and F a filter of size $k \times k \times k$. A single filter kernel is often quite small and shared between all the neurons in a layer. Because of this, each neuron is only aware of a small area from the previous layer, called the *receptive field.* This structure greatly reduces the amount of weights between layers and thus reduces the number of parameters in the network, compared with the MLP structure. It also makes the convolutional layer insensitive to spatial translation as the same filter is applied to all of the input image as explained in [30].

The input images will be three dimensional tensors with width, height and depth. The filter kernels have a width and height of k which corresponds to its receptive field, however the depth of the kernels are often set to the same depth as the input image. A kernel applied to a three dimensional input image will thus yield a two dimensional output as discussed in [30].

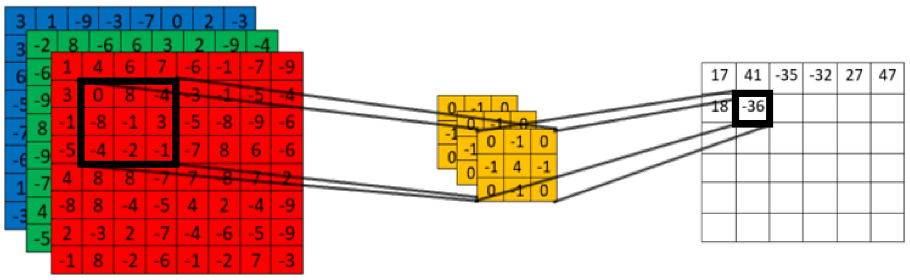


Figure 9: An intuitive explanation of a convolutional layer response. Here, the depth is the three color channels. The figure shows a 8x8x3 image being convolved with a 3x3x3 kernel to create a 6x6x1 output.

As seen in Figure 9 the filter kernel has been shifted so that the center pixel in the image corresponds to the same position in the output. To have an absolute center, the kernel size k should thus be chosen as an odd number. Define the new boundaries as $k_s = \frac{k-1}{2}$, also let k_D define the depth of the image. The convolutional operation for a single kernel as shown in Figure 9 is then defined as

$$z_{i,j} = \varphi_c \left(x_{i,j} * w_{i,j} + b \right) \tag{24}$$

$$= \varphi_c \left(\sum_{i'=-k_s}^{k_s} \sum_{j'=-k_s}^{k_s} \sum_{k'=0}^{k_D} x_{i-i',j-j',-k'} \cdot w_{i',j',k'} + b \right) . \tag{25}$$

This can be compared to the equation of the perceptron output as shown in Eq. 18. The output becomes a two dimensional *feature map* of smaller size than the input image since the mismatched edges are not included, as seen in Figure 9. To include the edges, *Zero Padding* can be performed by expanding the input image with a frame of zeros. In order to detect more than one type of shape in the layer, many independent filter kernels are used for the same input image. The two dimensional output of these kernels are then stacked into a new three dimensional tensor that can be used as an input image in the next convolutional layer as described in [30].

Since the convolution is a linear operation, an activation function needs to be introduced in order to capture non-linearities, shown as $\varphi_c(\cdot)$ in Eq. 24. Normally, the simple activation function *rectified linear unit* (ReLU) is used, this is defined as $f(x) = max(0,\ x)$. Other nonlinear activation functions such as the hyperbolic tangent och sigmoid can be used, but training with ReLU has shown to give faster convergence, see [9].

In order to further reduce the number of parameters in the model a layer called *pooling Layer* is commonly used. A pooling layer reduces the size of the input image by moving a window over the image, and in each position downsampling the pixels contained in the window into a single pixel. The procedure is then repeated for each depth layer in the image. There are different methods for performing the downsample, but the most commonly used strategy is called *max pooling*. Max pooling simply outputs the largest value in each window step as the downsampled pixel as explained in [31].

Both the convolutional and pooling layers utilize sliding windows to calculate the feature maps from the input image. The step size of the windows is called *stride* as explained in [31].

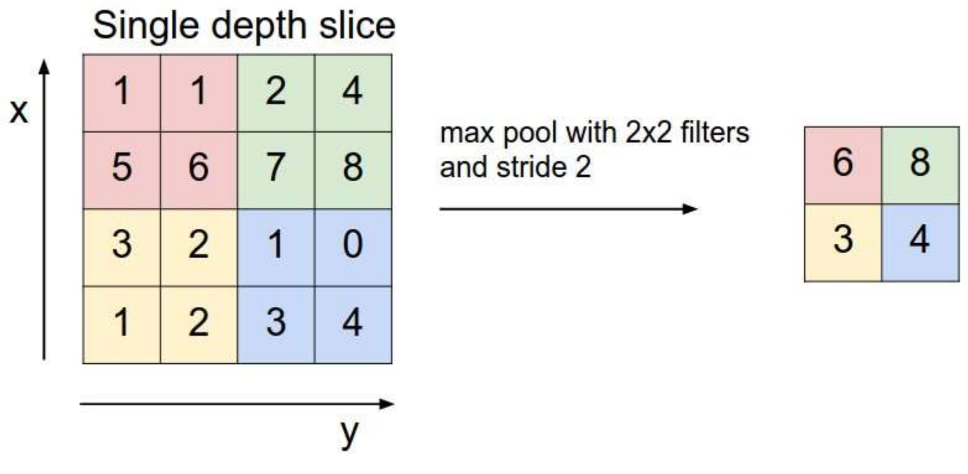


Figure 10: THe figure shows a 4x4 image in a MaxPool layer with stride 2 being reduced to a 2x2 image. Image courtesy of http://cs231n.stanford.edu/.

In convolutional networks such as AlexNet [9], it has been found that performance can be improved by normalizing the layer outputs after the ReLU activations in a process called *local response normalization*. Let $x^d_{i,j}$ be the response of a neuron when the kernel d has been applied to the position i, j and then passed through the ReLU activation function. The response-normalized output $\bar{x}^d_{i,j}$ is then calculated as

$$\bar{x}^d_{i,j} = x^d_{i,j} / \left(k + \alpha \sum_{d'=\max(0,d-n/2)}^{max(N-1,d+n/2)} \left(x^{d'}_{i,j} \right)^2 \right)^\beta \quad . \tag{26}$$

The local response normalization thus normalizes over the n closest kernels in depth for each width/height position i, j. The parameters k, n, α and β are all predefined hyperparameters that need to be set.

In order for a CNN to make a classification, the network is often paired with a couple of fully connected layers in the end that have the same structure as the MLP networks as explained in [30]. Then, by connecting a number of convolutional layers, max pooling layers and finalizing with some fully connected layers, we can create a convolutional neural network as shown in Figure 11.

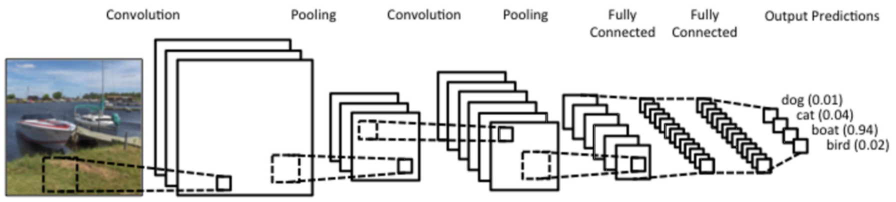


Figure 11: An example of a small CNN with two convolutional layers, two MaxPool layers and two fully connected layers. Image courtesy of www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/.

2.5.3 Learning through Stochastic Gradient Descent

In order to use an ANN for classification, a method needs to be devised to train it on previously labeled data. First, some definitions are needed. Let $\boldsymbol{x}(n)$ be the input data vector of sample n, $\boldsymbol{y}(n)$ its corresponding true class label and $\hat{\boldsymbol{y}}(n)$ the classification given from an artificial neural network that has been fed the data $\boldsymbol{x}(n)$. A set of training data of size N can then be defined as

$$T = \{\boldsymbol{x}(n),\ \boldsymbol{y}(n)\}_{n=1..N} \quad . \tag{27}$$

What is sought is a way to adapt the weights so that $\hat{\boldsymbol{y}}(n)$ becomes similar to the true values $\boldsymbol{y}(n)$. Start by defining a loss function L that measures how much the predicted labels $\hat{\boldsymbol{y}}(n)$ differ from the true labels $\boldsymbol{y}(n)$, the smaller the value the better the accuracy. There are a multitude of ways to construct the loss function but for classification tasks *cross entropy* is commonly used, see [29].

By utilizing the loss function, finding the optimal weights for classification essentially becomes a minimization problem. Let $\boldsymbol{w}$ be the weights of the network and $f(\boldsymbol{w})$ be an ANN classifier, then the optimal weights $\widetilde{\boldsymbol{w}}$ for the data set T can be found by minimizing the loss function

$$L(f(\widetilde{\boldsymbol{w}}) \mid T) = \min_{\boldsymbol{w}} L(f(\boldsymbol{w}) \mid T) \quad , \tag{28}$$

over all possible $\boldsymbol{w}$. To efficiently search the space of $\boldsymbol{w}$, an iterative optimizer is commonly used, often *gradient descent*, which works as follows (see [29]).

1. Initiate the weight block $\boldsymbol{w}$ at time $t = 0$ to be a set of random values.
2. For the next time step, find a small weight step $\Delta\boldsymbol{w}$ according to some rule.
3. Update the weights according to $\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + \Delta\boldsymbol{w}$.
4. Repeat steps 2-3 until the loss function $L(f(\boldsymbol{w}) \mid T)$ is no longer decreasing.

In step 2, a rule needs to be defined in order to update the weights such that L decreases in value. This can be done by taking the derivative of the loss function with respect to the weights and then moving a small step the opposite direction, calculated as

$$\Delta\boldsymbol{w} = -\eta\frac{\partial L}{\partial\boldsymbol{w}} \quad . \tag{29}$$

Here, η decides the step length and is referred to as the *learning rate.* To increase convergence, the learning rate is often decreased during training. One such method is *step decay* in which the learning rate is reduced by a factor after a set number of iterations as explained in [30].

In practice, it is desirable to perform each step with the gradient descent on a subset of the training set, called *batch updating.* In each iteration of a new batch B will be drawn from T and then the gradient step will be calculated from B. Performing the update on the entire training set at once is inefficient when dealing with large data sets. On the other hand, performing the update using a single data point is numerically unstable as explained in [29]. A batch updating block of size N_B is then defined as

$$B = \{\boldsymbol{x}(\mu),\ \boldsymbol{y}(\mu)\}_{\mu=1..N_b} \qquad B \subset T \quad . \tag{30}$$

Drawing a new batch from the training set is a semi-random process were each data point will be used once before a data point can be included in a batch for a second time. The number of iterations that is required for all training data points to have been used once is referred to as an *epoch.*

The numerical instability that arises with single or very few data points comes from the fact that the loss function L is evaluated over the inputs and outputs of those data points. Using few samples in each iteration will make the overall structure of L vary between batches which in turn complicates the convergence of the weights. A batch updating thus seeks to approximate the entire behaviour of the data set T without needing to calculate the derivative for all training data in each step. The size of the batch depends on the number of classes and the general heterogeneity of the data. Using gradient descent with batch updating is sometimes referred to as the *stochastic gradient descent* method as explained in [30].

Often the stochastic gradient descent method is enhanced with *momentum* in order to make it more robust. The enhancement modifies the gradient descent updating rule shown in Equation 29 by adding a part of the previous gradient update. Intuitively the momentum can be thought of as a kind of velocity for the algorithm which will hopefully speed up the convergence and keep the algorithm clear of most local minimas. The updating rule for gradient descent with momentum can be written as (see [29])

$$\Delta \boldsymbol{w}^{(m)}(t+1) = -\eta \frac{\partial L}{\partial \boldsymbol{w}^{(m)}} + \alpha_M \Delta \boldsymbol{w}^{(m)}(t) \ , \tag{31}$$

where α_M is a hyperparameter deciding the amount of momentum.

2.5.4 Training a Feed-Forward Network through Back-Propagation

The MLP and the CNN structures discussed are part of a type of networks called *feed-forward networks*. For these types of networks, each layer only has a single input, the output of the previous layer. This structure can be used to simplify the calculations for the derivatives.

Computing the derivatives of the loss function with respect to the weights is not a trivial task. To succeed the feed-forward structure will be used in an algorithm called *back propagation.* Let $\boldsymbol{w}^{(m)}$, $\boldsymbol{x}^{(m)}$ and $g(\cdot)^{(m)}$ be the weight block, input and layer function respectively for layer m and consider the arbitrary feed forward network as seen in Figure 12.

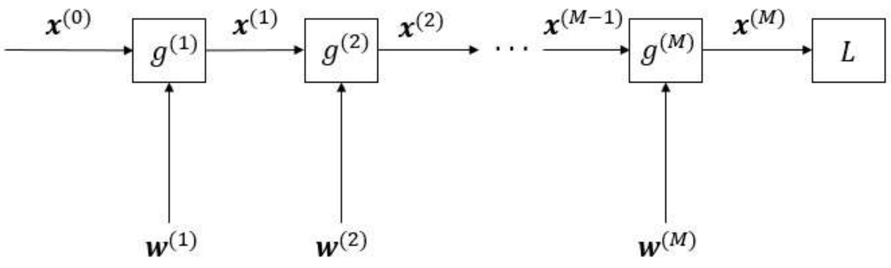


Figure 12: The flow of an arbitrary feed-forward neural network. Here, $\boldsymbol{x}^{(0)}$ is the input image while $\boldsymbol{x}^{(i)}, \quad i \in [1, M]$, is the output from hidden layer M, $g^{(m)}$ is the layer functions and L an arbitrary loss function.

Since the loss function L depends on many nested layer functions, deriving it with respect to the weights on layer m can be simplified using the chain rule with one derivative per layer as shown in [32]. The derivative becomes

$$\frac{\partial L}{\partial \boldsymbol{w}^{(m)}} = \frac{\partial L}{\partial \boldsymbol{x}^{(M)}} \cdot \frac{\partial g^{(M)}\left(\boldsymbol{x}^{(M-1)}|\boldsymbol{w}^{(M)}\right)}{\partial \boldsymbol{x}^{(M-1)}} \cdot \ldots \cdot \frac{\partial g^{(m)}\left(\boldsymbol{x}^{(m-1)}|\boldsymbol{w}^{(m)}\right)}{\partial \boldsymbol{w}^{(m)}} \ . \tag{32}$$

Now consider an arbitrary layer function with input $\boldsymbol{x}$, layer function $g(\cdot)$ and output $\boldsymbol{z} = g(\boldsymbol{x})$. For a convolutional layer especially both the input and the output are three dimensional tensors, $\boldsymbol{x} \in \mathbb{R}^{H \times W \times D}$ and $\boldsymbol{z} = \mathbb{R}^{H' \times W' \times D'}$. A derivative of the layer function $g(\cdot)$ with respect to the input would result in a 6 dimensional tensor. For simplicity, we want to express the derivation in matrix notation, which can be done by stacking both the input and output in lexicographical ordered vectors and then performing the derivation. Define $\boldsymbol{x}_v$ to be the vector stacked tensor $\boldsymbol{x}$

$$\boldsymbol{x}_v = \begin{bmatrix} x_{111} \\ x_{211} \\ \vdots \\ x_{H11} \\ x_{121} \\ \vdots \\ x_{HWD} \end{bmatrix} . \tag{33}$$

The derivation of the stacked output of the layer function $\boldsymbol{z}_v = g(\boldsymbol{x})$ with the stacked input $\boldsymbol{x}_v$ would now result in the *Jacobian matrix*

$$\frac{d\boldsymbol{z}_v}{d\boldsymbol{x}_v^T} = \begin{bmatrix} \frac{\partial z_{111}}{\partial x_{111}} & \frac{\partial z_{111}}{\partial x_{211}} & \cdots & \frac{\partial z_{111}}{\partial x_{H11}} & \frac{\partial z_{111}}{\partial x_{121}} & \cdots & \frac{\partial z_{111}}{\partial x_{HWD}} \\ \frac{\partial z_{211}}{\partial x_{111}} & \frac{\partial z_{211}}{\partial x_{211}} & \cdots & \frac{\partial z_{211}}{\partial x_{H'11}} & \frac{\partial z_{211}}{\partial x_{121}} & \cdots & \frac{\partial z_{211}}{\partial x_{HWD}} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial z_{H'11}}{\partial x_{111}} & \frac{\partial z_{H'11}}{\partial x_{211}} & \cdots & \frac{\partial z_{H'11}}{\partial x_{H11}} & \frac{\partial z_{H'11}}{\partial x_{121}} & \cdots & \frac{\partial z_{H'11}}{\partial x_{HWD}} \\ \frac{\partial z_{121}}{\partial x_{111}} & \frac{\partial z_{121}}{\partial x_{211}} & \cdots & \frac{\partial z_{121}}{\partial x_{H11}} & \frac{\partial z_{121}}{\partial x_{121}} & \cdots & \frac{\partial z_{121}}{\partial x_{HWD}} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial z_{H'W'D'}}{\partial x_{111}} & \frac{\partial z_{H'W'D'}}{\partial x_{211}} & \cdots & \frac{\partial z_{H'W'D'}}{\partial x_{H11}} & \frac{\partial z_{H'W'D'}}{\partial x_{121}} & \cdots & \frac{\partial z_{H'W'D'}}{\partial x_{HWD}} \end{bmatrix} . \tag{34}$$

However, the Jacobian matrix quickly becomes extremely large and it is unfeasible to explicitly calculate it. As an example from [32], using only small input and output images of size $32 \times 32 \times 128$, we would need roughly 68 GB to store the entire matrix in single precision! Fortunately the back propagation technique allows us to calculate the derivatives without incurring this huge memory cost. First rewrite the chain rule derivative of L in Eq. 32 with the stacked vector notation as

$$\frac{\partial L}{\partial \left(\boldsymbol{w}_v^{(m)}\right)^T} = \frac{\partial L}{\partial \left(\boldsymbol{x}_v^{(M)}\right)^T} \cdot \frac{\partial g^{(M)}\left(\boldsymbol{x}_v^{(M-1)} | \boldsymbol{w}_v^{(M)}\right)}{\partial \left(\boldsymbol{x}_v^{(M-1)}\right)^T} \cdot \ldots \cdot \frac{\partial g^{(m)}\left(\boldsymbol{x}_v^{(m-1)} | \boldsymbol{w}_v^{(m)}\right)}{\partial \left(\boldsymbol{w}_v^{(m)}\right)^T} . \tag{35}$$

Here the first derivative, L with respect to its input in transpose, is simply a row vector of size $\left(\boldsymbol{x}_v^{(M)}\right)^T$ since L is scalar. The rest of the terms become Jacobian matrices as shown in Eq. 34. Denote this first row vector in the chain rule as

$$\left(\delta_v^{(M)}\right)^T = \frac{\partial L}{\partial \left(\boldsymbol{x}_v^{(M)}\right)^T} . \tag{36}$$

By taking a step backwards $\left(\delta_v^{(M-1)}\right)^T$ could possibly be constructed by using the product of the previous δ and the next derivative in line as

$$\left(\delta_v^{(M-1)}\right)^T = \left(\delta_v^{(M)}\right)^T \cdot \frac{\partial g^{(M)}\left(\boldsymbol{x}_v^{(M-1)}|\boldsymbol{w}_v^{(M)}\right)}{\partial\left(\boldsymbol{x}_v^{(M-1)}\right)^T} \quad . \tag{37}$$

Eq. 37 is a vector matrix multiplication and will yield a new row vector of size $\left(\boldsymbol{x}_v^{(M-1)}\right)^T$, but since the matrix is the huge Jacobian, it cannot explicitly be stored in memory. However, because of the feed-forward structure, the $\left(\delta_v^{(M)}\right)^T$ is only dependent on the output of the previous layer $\left(\boldsymbol{x}_v^{(M)}\right)^T$ and is thus seen as constant with respect to $\left(\boldsymbol{x}_v^{(M-1)}\right)^T$. Since the derivative is a linear operator, the constant can be moved inside the operation as

$$\left(\delta_v^{(M-1)}\right)^T = \frac{\partial}{\partial\left(\boldsymbol{x}_v^{(M-1)}\right)^T}\left(\left(\delta_v^{(M)}\right)^T \cdot g^{(M)}\left(\boldsymbol{x}_v^{(M-1)}|\boldsymbol{w}_v^{(M)}\right)\right) \quad . \tag{38}$$

Now, the inner δ is a row vector of the same size as $\left(\boldsymbol{x}_v^{(M)}\right)^T$ and the function $g^{(M)}$ is simply the function output of layer M which is $\left(\boldsymbol{x}_v^{(M)}\right)^T$ itself. This multiplication thus becomes the standard inner product and leaves us with a scalar derived with a row vector, which in turn becomes a new row vector of size $\left(\boldsymbol{x}_v^{(M-1)}\right)^T$. Let $\langle\cdot,\cdot\rangle$ denote the inner product. Eq. 38 can now be reformulated to the the calculation of the back-propagation on step m as

$$\delta^{(m)} = \frac{\partial\left\langle \delta_v^{(m+1)}, g^{(m+1)}\left(\boldsymbol{x}_v^{(m)}|\boldsymbol{w}_v^{(m+1)}\right)\right\rangle}{\partial\left(\boldsymbol{x}^{(m)}\right)^T} \quad . \tag{39}$$

In order to then update all the weight blocks, simply start by computing $\left(\delta_v^{(M)}\right)^T$ and $\Delta\boldsymbol{w}^{(M)}$. Then, take a step backwards, compute $\left(\delta_v^{(M-1)}\right)^T$ using $\left(\delta_v^{(M)}\right)^T$ and use the results to compute $\Delta\boldsymbol{w}^{(M-1)}$. Continue to step backwards until all weight blocks are updated. For an arbitrary layer m the gradient descent update becomes

$$\frac{\partial L}{\partial \boldsymbol{w}^{(m)}} = \frac{\partial\left\langle \delta_v^{(m)}, g^{(m)}\left(\boldsymbol{x}_v^{(m-1)}|\boldsymbol{w}_v^{(m)}\right)\right\rangle}{\partial \boldsymbol{w}^{(m)}} \quad , \tag{40}$$

$$\Delta\boldsymbol{w}^{(m)} = -\eta\frac{\partial L}{\partial \boldsymbol{w}^{(m)}} \quad , \tag{41}$$

$$\boldsymbol{w}^{(m)}(t+1) = \boldsymbol{w}^{(m)}(t) + \Delta\boldsymbol{w}^{(m)} \quad . \tag{42}$$

What remains is to derive the loss function and the layer functions with respect to the inputs and weights, but we will not go in to further details. The general idea of the back propagation has been introduced and going further is not necessary for additional comprehension.

2.5.5 Overfitting and Network Regularization

Neural networks often contains many millions parameters. Here, a large problem arises since more parameters means that the model has more degrees of freedom to perform the model fitting. The more freedom a model has with its parameters, the more the model tends to capture variations in each individual data point and not the entirety of the data set as explained in [31]. An example of this is seen in Figure 13 below.

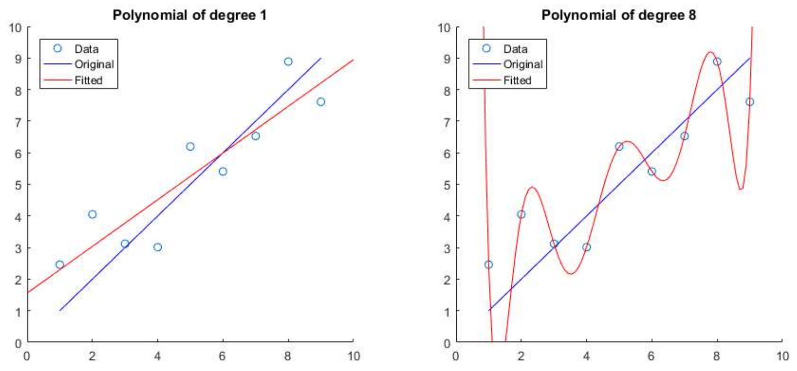


Figure 13: The figure shows the dangers of overfitting. A set of data points from a polynomial of degree 1 has been generated and distorted with a gaussian variable of standard deviation 1. The data points are then fitted to two new polynomials, one of degree 1 and one of degree 8. The one with higher degree captures the individual variations much better than the polynomial of lower degree.

A robust model with good *generalization* can be transferred on to a new data set and still give good classification. On the other hand, a model that is *overfitted* will not perform well on a new data set. This is why the initial data set is split into multiple smaller partitions. By training the model on the training set T, performance on previously unseen data can be measured on the validation set V. A good performance on T but bad on V exposes poor generalization. A simple way to make a model more robust to overfitting is simply by using more data in T, but this is not always possible as explained in[31].

If no more data is available, a good option is instead to simply expand the data set through *data augmentation.* This technique allows for the creation of more data points by simple transformations of the available training data, such as translation or rotation. It is important to perform this step of data augmentations after the original data set has been split into its sub parts. Otherwise, it is very likely that an augmented image will end up in the validation set V while the original image is in the training set T.

Beyond data augmentation there is a large group of methods for combating poor generalization in networks called *regularization.* The first regularization technique that we will discuss is called *weight decay.* Weight decay modifies the loss function as (see [29])

$$L_R(\boldsymbol{w}) = L(\boldsymbol{w}) + \alpha_R \Omega(\boldsymbol{w}) \ . \tag{43}$$

Here, Ω is the regularization term and $\alpha_R > 0$ is a hyper-parameter controlling the amount of regularization. There are many different ways of determining $\Omega(\boldsymbol{w})$ but most commonly used are two techniques called L2 and L1 regularization defined as (see [31])

$$\Omega(\boldsymbol{w}) = \frac{1}{2}\sum_i w_i^2 \qquad \text{L2 regularization} \;, \tag{44}$$

$$\Omega(\boldsymbol{w}) = \sum_i |w_i| \qquad \text{L1 regularization} \;. \tag{45}$$

In theory, L2 and L1 regularization will force any excess weight that is not needed for classification to zero while keeping the necessary weights small. These regularizations often decrease overfitting and increase generalization performance, but in practice the hyper-parameter α_R can be hard to tune and sometimes large weights are necessary which are penalized using L2 or L1 as explained in [29].

A different regularization technique is a method called *drop-out.* Instead of modifying the loss function the network itself is changed. Define a drop-out factor d as a number between 0 and 1. In each iteration, we will temporarily remove a factor d of all artificial neurons from the hidden layers and then train the network. After the iteration, the removed neurons are reinserted as explained in [31].

Drop-out implies that an artificial neuron cannot rely on any other particular neuron to be present in the network when performing the weight update. This results in the neuron trying to learn more robust features that work in conjunction with any random subset of the neurons in the network. Although not entirely intuitive, the drop-out method has shown [33] to successfully increase the generalization and decrease overfitting in a large variety of neural network implementations.

2.5.6 Transfer Learning

Training a completely new CNN is a resource demanding process, both concerning time, GPU-programning skills and expensive equipment. Furthermore, one could be in a situation where there is not enough labeled data in order to train a network. Due to these high thresholds other alternatives than training a CNN from scratch have evolved. The main idea is to use an already trained network and apply it to a new classification problem. This process is called transfer learning. It has been shown that this is a very successful path to get good results as seen in [34], [35], [36], [37].

Intuitively, the first layers of a network capture more general features while the last layers capture more specific features depending on what training set and classes were used during the original training. It therefore becomes an interesting problem to find which layers are general enough to transfer and which are too specific. In transfer learning the first n layers are copied from the original network while the remaining layers are randomly re-initialized. It has been shown that the generalization of the network increases if it is *fine-tuned*, i.e., if the original weights are included and updated during the back-propagation. Another alternative is to freeze the original layers, which does not increase generalization as much as fine-tuning, see [34].

Which layers should be copied from the original network and which layers should be retrained is a question with only general guidelines. It is mostly dependent on how similar the original data set is to the new data set. If the data sets are similar, then more layers from the original network can be used and vice versa. The number of parameters of a network does also influence the performance when transfer learning. Overall it is better to have more parameters, especially when the original and new data sets are similar. Smaller networks are not able to generalize in the same manner as a larger networks. On the other hand, a large network with many parameters can have problems with overfitting if the original and new data sets are not similar enough as shown in [35]. It should also be noted that all results are dependent on the network that was used and the new data set that it was tested on. With this into account, all results can only be seen as general heuristics and it is necessary to find exact settings for best results for each specific problem that has not been tested by previous studies.

2.6 Ensemble Techniques

To increase the accuracy of the predictions, different algorithms can be combined into a single classifier called an *ensemble*. The idea behind the ensemble is that by using multiple classifiers a reduction in variance can be achieved and thus a higher accuracy can be obtained as explained in [38]. An ensemble also gives a natural way of combining results from different classifiers using different types of data.

Kantardzic [39] has an interesting example demonstrating how an ensemble learner outperforms the individual classifiers. Consider an ensemble of 15 classifiers for a binary classification task. Let the final ensemble classification be a simple majority vote between the classifiers. If we assume that the different classifiers are independent with the identical error rate ϵ, the probability of k classifiers being wrong will follow the binomial distribution

$$p(k) = \binom{15}{k} \epsilon^k (1-\epsilon)^{15-k} \quad . \tag{46}$$

Since an ensemble miss-classification will occur if at least half of the classifiers are wrong the total ensemble error can be calculated as the sum of all probabilities when $k \geq 8$

$$\epsilon_{tot} = \sum_{k=8}^{15} p(k) = \sum_{k=8}^{15} \binom{15}{k} \epsilon^k (1-\epsilon)^{15-k} \quad . \tag{47}$$

If the identical error rate is set to $\epsilon = 0.3$, the total ensemble error rate would reduce to a staggering $\epsilon_{ens} = 0.05$. In reality, it is safe to assume that this gain is a bit optimistic, because it assumes that the classifiers are completely independent which is a major simplification. Instead, as a rule of thumb it can be said that the more independent the different classifiers are, the more the ensemble will boost the overall accuracy as explained in [38].

We proceed to define a general ensemble model. Let $\boldsymbol{f}(\cdot)$ denote a set of n classifiers and $\boldsymbol{X}$ the set of input data. The set of class probabilities returned from each classifier and each data point can then be written as

$$\boldsymbol{P} = \boldsymbol{f}(\boldsymbol{X}) \quad . \tag{48}$$

Now let $f_{ens}(\cdot)$ denote an ensemble model function. The ensemble classification can then be written as

$$\hat{\boldsymbol{Y}} = f_{ens}(\boldsymbol{P}) = f_{ens}(\boldsymbol{f}\,(\boldsymbol{X})) \ . \tag{49}$$

The simplest way of defining f_{ens} is by *voting*. Denote $p_{ij}^{(c)}$ as the class probability for class c returned from the classifier j with the input data point i and let w_j be a weight associated with the classifier j. Also let the output for input data i and class c be written as $y_i^{(c)}$. The voting system thus becomes

$$y_i^{(c)} = \sum_{j=1}^{n} w_j p_{ij}^{(c)} \quad \text{where} \quad \sum_{j=1}^{n} w_j = 1 \ . \tag{50}$$

Another ensemble technique is *stacking* which considers the outputs $\boldsymbol{P}$ from the different classifiers as features of their own. These features can then be learned by a meta-classifier on top of the original classifiers.

2.7 Evaluation of Models

In order to evaluate models one can use several measurements and methods, all with their own advantages and drawbacks. A common method is to present a confusion matrix, i.e., a matrix with the percentage of correctly and incorrectly predicted labels from a certain model. It is straightforward and can easily be interpreted, especially if there are only two classes.

Another method for evaluation is the *receiver operating characteristic* (ROC), a graph that plots the true positive rate and the false positive rate. It is also a common measurement for evaluation of a classifier model and is commonly used in the medical field. It is possible to obtain the area under the curve (AUC) from the ROC. The AUC represents the probability for a classifier to classify a randomly selected positive input higher than a randomly selected negative input as explained in [40]. That is, $P(X_1 > X_0)$, where X_1 is the score for a positive input obtained from a classifier and X_0 is the score from a negative input. An example of this can be seen in Figure 14. A random classifier will receive an AUC value of 0.5 and a perfect classifier will have an AUC value of 1. The AUC has however been criticized for being misleading as discussed in [41]. It will still be used in this thesis since it provides a baseline for comparison with other studies.

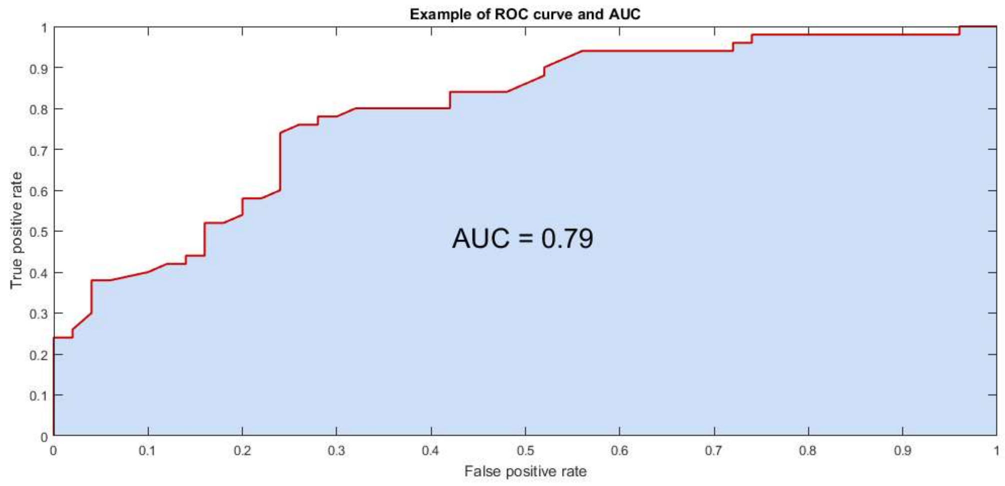


Figure 14: An example of a ROC curve in red. The blue coloured area respresents the AUC.

In Table 3, we can see the confusion matrix in the blue cells. For this thesis however, the position of the negative and positive predictions have been switched when presenting a confusion matrix. The accuracy can also be seen which is used for model evaluation. It is a good measurement for total correctly predicted labels but does not show the distribution of correctly classified labels as can be seen in the confusion matrix. Instead a modified version of the accuracy will be used which will be defined as the mean of the per class accuracy as can be seen below.

$$
\begin{aligned}
a &= \frac{\sum TP + \sum TN}{n} && \text{normal accuracy} \ , \\
a_m &= \frac{\frac{\sum TP}{n_{TP}} + \frac{\sum TN}{n_{TN}}}{2} && \text{modified accuracy} \ .
\end{aligned}
$$

Table 3: Prediction and true condition of a label. From this it is possible to obtain the confusion matrix, in blue cells, which is used for evaluation of a model. The accuracy can be seen in the bottom left corner, which indicates the total correctly predicted labels.

Total Population	Positive Prediction	Negative Prediction	Rate
Positive condition	True Positive (TP)	False Negative (FN)	$TPR = \frac{\sum TP}{\sum_{n \in \mathcal{C}_1} n}$
Negative condition	False Positive (FP)	True Negative (TN)	$FPR = \frac{\sum FP}{\sum_{n \in \mathcal{C}_1} n}$
$Accuracy = \frac{\sum TP + \sum TN}{n}$	$\frac{\sum TP}{\sum \text{Positive Prediction}}$	$\frac{\sum FN}{\sum \text{Negative Prediction}}$	
	$\frac{\sum FP}{\sum \text{Positive Prediction}}$	$\frac{\sum TN}{\sum \text{Negative Prediction}}$	

3 Initial Data Remarks

The following chapters will revolve around training different classifiers on the available numerical and image data. Each classifier will be evaluated in terms of performance, that is their accuracy and AUC score. Later, some of the classifiers will be used in order to determine the eventual relevance of the immunological features in the data and finally the classifiers will be used to create an ensemble system. A simplified flow chart has been created in Figure 15 to make it easier to follow the different steps in the process.

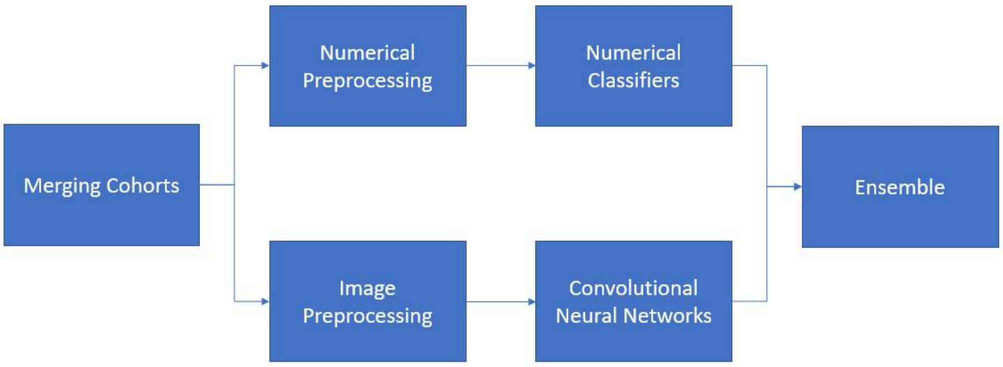


Figure 15: The flow chart of the final ensemble learner. References to the sub-parts of this image will be made further on in the method for a more intuitive understanding.

3.1 Merging Cohorts

The data can be categorized into two classes depending on whether or not the patient develops metastasis within a 5-year period. We refer to all patients that do not develop metastasis within the given time period as class 1 and all patients that do as class 2. The cohorts were merged in order to create a single data set. This is conducted since about 20 % of the patients belong to class 2. If the two cohorts are investigated individually, the estimate of the prediction error will have a too high variance. The validation set would only contain approximately 11 respectively 8 patients from class 2 for each cohort. Henceforth, the referred data set consists of both cohorts.

3.2 Estimating Label Errors

It is possible that some of the labels in the data are incorrect. Some patients have passed away or were not followed up in the study. Thus, it is not known whether or not they did develop metastasis within 5 years. These patients have been labeled as belonging to class 1, not developing metastasis, but it might not have been the case. These patients will either have to be removed from the data set or kept knowing that they will affect the results.

A patient was categorized as a *potential error* if the person was marked as class 1 and had an overall survival of less than 5 years or if the latest metastasis-free checkup date was less than 5 years from initial diagnosis. Using this distinction, $N_{err} \approx 12.7$ % of the set of all patients were found to have a potential error.

By excluding the patients with potential label errors, we can estimate the true risk of developing metastasis within a 5 year period. From our data set, this becomes $P(metastasis) = 22.8\%$. If we assume that the patients with potential label errors have the same risk of developing metastasis as the rest of the group, the percentage of errors ϵ in the labels can then be estimated as

$$\hat{\epsilon} = P(metastasis) \cdot N_{err} \approx 2.89 \text{ \%} \ . \tag{51}$$

Since an error rate of 2.89 % is more acceptable than removing 12.7 % of the data, the patients with potential label errors will be kept in the data set.

3.3 Creating Training, Validation and Test Sets

The available data set is split using a 5-fold cross-validation scheme to create 5 training sets T and 5 disjoint test sets S. These 5 folds will be the same throughout all of the tests. In some of the algorithms used, the training set T will be split using yet another k-fold to encompass a validation set V in order to estimate parameters. By splitting the data set on the patients, we can use the same folds in all three data types which is important for correct comparisons.

4 Classifying the Numerical Data

In this chapter, the usability of the numerical data in predicting prognosis is to be investigated using a number of numerical classifiers. Initially, we will conduct numerical preprocessing, to later fit different classifiers and present the results. The numerical data consists both of the AMLC feature data type and the clinical data type. Mainly, the AMLC features will be investigated and the clinical data will be used as reference.

We begin by structuring the data from the AMLC features and the clinical data into columns, representing the different features. The rows will then represent the different patients. Denote a data matrix of this type as $\boldsymbol{X}$, this could be either the AMLC features or the clinical data. Further, let $\boldsymbol{X}_T$ be the data matrix from all the patients in the training set and $\boldsymbol{X}_S$ the data matrix from all the patients in the test set.

4.1 Numerical Preprocessing

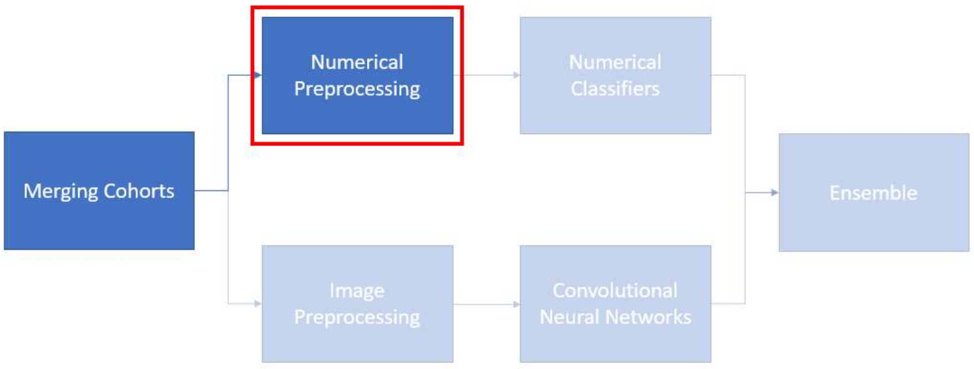


Figure 16: The red rectangle indicates what part of the ensemble structure that we currently are presenting.

Before any classification can be made with the numerical data, it first needs to be preprocessed. There exists not-a-number (NaN) values in the numerical data that needs attention and the data need to be normalized in order to work properly. The huge amount of variables in the AMLC features also need to be reduced. Here, we will examine and present the feature reduction using PCA, t-SNE and SFS.

4.1.1 Dealing with NaN Values

In the numerical part of the data set there are some features that have observations with NaN values. For AMLC features, some values are denoted as the fraction between different features. In some cases, there does not exist any value in the denominator, i.e., a specific biomarker does not exist in the sample. The value will then be division by 0. For the clinical data NaN values exists in some features in the form of missing values.

Furthermore each patient is associated with 1 to 4 core sample images each, which in turn means that each patient has 1 to 4 different values for each feature in the AMLC features. Since the goal is to predict the outcome for each patient, a mean value of each feature is created. The mean will represent the patient more accurately. Another advantage is that most of the NaN values in the AMLC features can be removed in this step by simply omitting them when calculating the mean values.

However, not all NaN values can be removed and the missing values in the clinical data still remain. For the clinical data, the NaN values occur in features with only binary values. These NaN values have simply been replaced with the average between the two available binary values in order to in a fast and simple way minimize their impact. For the AMLC features, we instead calculate a feature mean over all available patients for each feature, and then replace the NaN values with the corresponding mean.

4.1.2 Data Normalization

The range of the values in the AMLC features or clinical data differs much from feature to feature. The solution to this is to normalize each feature vector to zero mean and standard deviation 1. This is performed by removing the estimated mean and dividing by the estimated standard deviation. To keep the test and training sets T and S separated, the mean and standard deviations are estimated only from T. Let $\boldsymbol{x}_i$ be feature vector i for all the patients in the the training set data matrix $\boldsymbol{X}_T$. The mean and standard deviation can then be estimated as

$$\hat{\mu}_i = \frac{1}{N}\sum_{j=1}^{N} x_{ij} \qquad \boldsymbol{x}_i \in \boldsymbol{X}_T \ , \tag{52}$$

$$\hat{\sigma}_i = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N} (x_{ij} - \hat{\mu}_i)^2} \qquad \boldsymbol{x}_i \in \boldsymbol{X}_T \ . \tag{53}$$

After calculating a mean and standard deviation estimation for each feature vector i the normalization can be computed as

$$\boldsymbol{x}_i^* = \frac{\boldsymbol{x}_i - \hat{\mu}_i}{\hat{\sigma}_i} \qquad \forall \boldsymbol{x}_i \in \boldsymbol{X}_T \ , \tag{54}$$

$$\boldsymbol{x}_i'^* = \frac{\boldsymbol{x}_i' - \hat{\mu}_i}{\hat{\sigma}_i} \qquad \forall \boldsymbol{x}_i' \in \boldsymbol{X}_S \ . \tag{55}$$

This is done for both the AMLC feature data matrix and the clinical data matrix.

4.1.3 Extracting Relevant Features

The AMLC features consists of a large cascade of 466 different features and no or only minor knowledge of their relevance exists. It is therefore very important that the feature space is reduced into a manageable size. Three methods of dimensionality reduction will be used for finding the relevant features; PCA, t-SNE and SFS. The clinical data however only consists of a collection of 8 features that all are hand picked for determining the overall survival of a patient. No reduction will be used on this data type.

In order to compute PCA and t-SNE, we will use a third party Matlab toolbox called the *Matlab Toolbox for Dimensionality Reduction* written by Laurens van der Maaten [42], one of the inventors of the t-SNE algorithm. The reduction needs to be calculated simultaneously for both the training and test sets, otherwise the new features for the two sets might represent completely different things. This is done by stacking $\boldsymbol{X}_T$ on top of $\boldsymbol{X}_S$ into a single data matrix, reducing the number of columns using either PCA or t-SNE and then extracting the reduced $\tilde{\boldsymbol{X}}_T$ and $\tilde{\boldsymbol{X}}_S$ from the stacked matrix. This is simple because the methods do not shuffle the patient indexes.

Using the training data set T from the first fold, we can examine how the features behave. We begin by reducing the data matrix $\boldsymbol{X}_T$ for both the AMLC features and the clinical data down to two dimensions using PCA and t-SNE to see if we can spot any clustering. It is pretty clear from the plots in Figure 17 that there are no real clustering to be observed. But if we instead look at the clinical data in Figure 18, we can see that it is more clustered but still has no clear and distinct patterns.

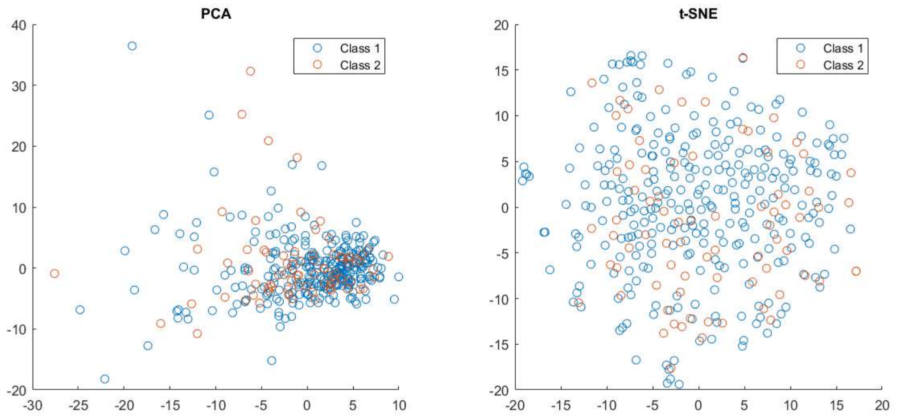


Figure 17: The AMLC features reduced from 466 dimensions down to 2. Each circle represents a patient in the training set T.

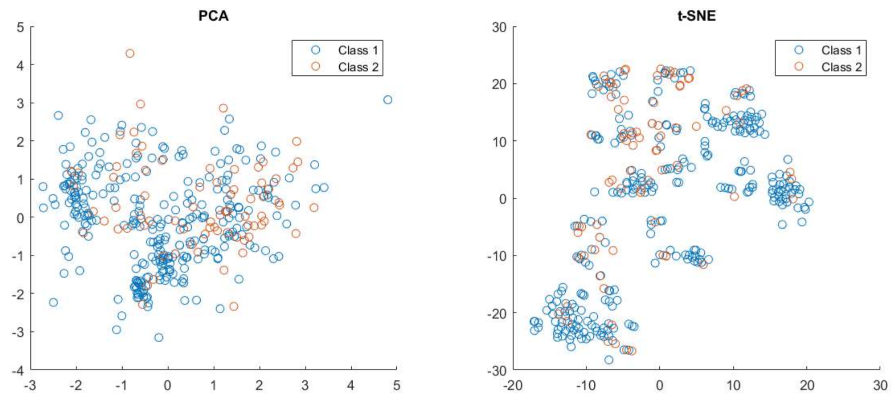


Figure 18: The clinical data reduced from 466 dimensions down to 2. Each circle represents a patient in the training set T.

The strengths of the different principal components are shown in Figure 19. As can be seen, the principal components over approximately 30 has little influence while the components over approximately 100 has practically none. For this reason, we will test dimensionality reduction using PCA on the AMLC features with both 30 and 100 principal components.

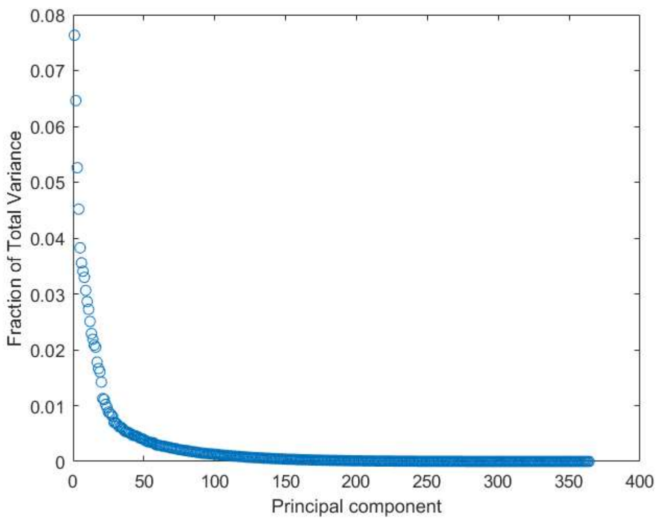


Figure 19: The strengths of the different principal components from the PCA reduction on the AMLC features of the training set data matrix $\boldsymbol{X}_T$. A maximum of 364 principal components have been evaluated since we only have 364 patients in the training data set.

Since the t-SNE features are highly dependent on the number of features used when performing the reduction, there is no easy way of doing the same type of strength comparison as with the PCA. Instead, we will use a t-SNE reduction with the same number of dimensions as the PCA for comparison.

The SFS will be computed using the built in Matlab function *sequentialfs*. This function takes a loss function L, a data matrix $\boldsymbol{X}_T$ and the correct labels for each patient and sequentially selects the different columns until there is no more improvement to L. The loss function L consists of a classifier to generate predictions from $\boldsymbol{X}_T$ to test against the true labels. The loss is then measured based on the prediction error. In our case, we will use SVM with two kinds of kernels, linear and RBF. For measuring the loss, we will also consider using both the accuracy and the AUC.

The selection itself in *sequentialfs* is performed using a k-fold to split the input data into k new training sets and k disjunct validation sets. For each fold the classifier in L is trained on the training set and evaluated on the validation set for each feature suggestion. Each feature suggestion will then have k loss values associated with it, and a final suggestion L_j can then be computed using the average. The feature suggestion with the best average loss value will be selected. For this thesis, we select k to be 5.

4.2 Training the Numerical Classifiers

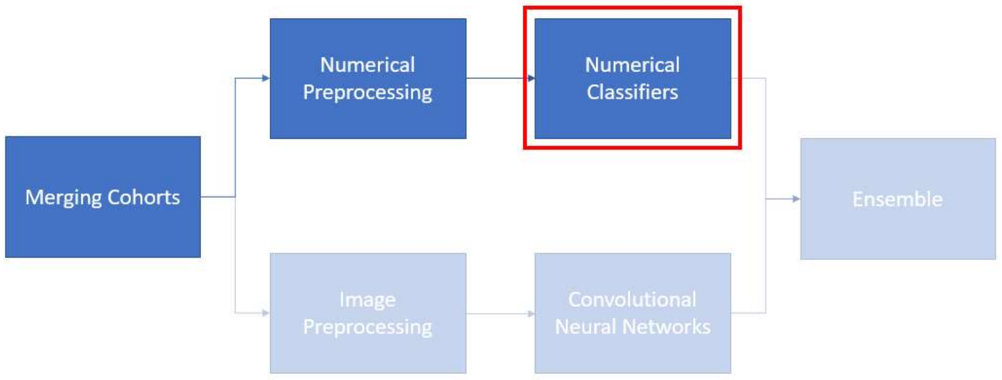


Figure 20: The red rectangle indicates what part of the ensemble structure that we currently are presenting.

After normalizing the features for both the AMLC features and the clinical data and reducing the dimensionality of the AMLC features the classifiers can be trained. We are going to consider three types of classifiers for the numerical data, namely SVM, RF and LDA. All are commonly used methods and could potentially yield different prediction scores, which is beneficial for an ensemble classifier.

The first classifier to be used is the SVM and for this we will consider two types of kernels, the linear kernel and the RBF. The SVM will be trained using the built in Matlab function *fitcsvm*, which takes a data matrix $\boldsymbol{X}$ with the true labels and fits a support vector machine to the data. The Matlab function can also optimize hyperparameters which will be done with the parameters *BoxConstraint*, which is related to the cost c, and *KernelScale*, a scaling parameter for the input features in $\boldsymbol{X}$. The optimization is performed on a 5-fold cross-validation of $\boldsymbol{X}$.

The second type of classifier is the RF algorithm. Here, the built in Matlab function *TreeBagger* will be used to create the decision tree ensemble. The function takes a data matrix $\boldsymbol{X}$, the true labels and a third parameter that decides the number of trees in the ensemble. We will test two different bag sizes of 300 and 600 trees but we will not perform a complete optimization on this parameter.

The last classifier we will consider is the LDA. The built in Matlab function used for fitting the LDA is called *fitcdiscr*. For the LDA as with the SVM it is also possible to optimize over hyper-parameters which we also will do for this function. The parameters that will be optimized are *Delta* which is a threshold for the feature weights (a feature with a weight lower than the threshold will be omitted) and *Gamma* which decides the amount of regularization in the covariance matrix. Matlab defines the regularization as follows. Let $\boldsymbol{S}$ be a covariance matrix, the regularized matrix then becomes

$$\hat{\boldsymbol{S}} = (1-\gamma)\boldsymbol{S} + \gamma \mathrm{diag}(\boldsymbol{S}) \ . \tag{56}$$

The optimization for the LDA is also calculated using a 5-fold cross-validation of the input data matrix.

Before training the classifiers the over-representation of class 1 needs to be dealt with. Otherwise a classifier with an accuracy of ≈ 75 % can be achieved by simply guessing at class 1 repeatedly. This is handled by the built in Matlab functions which can be fed with a prior probability, by setting the prior to uniform we ensure that the classifiers see the data set as balanced.

4.3 Evaluating the Classifiers

The results from the evaluated classification performance are presented below in the Table 4. The results are presented as the average AUC score from the 5 folds together with the standard deviation. In the top row, the results for the original AMLC features are shown while the bottom most row shows the performance on the 8 features in the clinical data.

Table 4: Results for AMLC features using different classifiers and dimensionality reduction techniques. The values are presented as mean AUC ± standard deviation, for a 5-fold cross-validation for both AMLC features and clinical data. The best dimensionality reduction technique for each classifiers are shown in bold text. It can easily be seen that the clinical data outperforms the classifiers using AMLC features.

Reduction technique	SVM-Linear	SVM-RBF	RF-300	RF-600	LDA
All features	0.57 ± 0.07	$\mathbf{0.58} \pm 0.06$	0.55 ± 0.05	0.56 ± 0.04	$\mathbf{0.60} \pm 0.03$
PCA (30)	0.55 ± 0.08	$\mathbf{0.58} \pm 0.06$	$\mathbf{0.60} \pm 0.07$	$\mathbf{0.60} \pm 0.09$	0.54 ± 0.05
PCA (100)	0.53 ± 0.12	0.54 ± 0.05	0.58 ± 0.06	0.58 ± 0.06	0.53 ± 0.11
t-SNE (30)	0.55 ± 0.09	0.56 ± 0.08	0.49 ± 0.06	0.49 ± 0.06	0.56 ± 0.09
t-SNE (100)	0.57 ± 0.05	0.56 ± 0.08	0.51 ± 0.07	0.51 ± 0.08	0.56 ± 0.07
SFS - Lin (Acc)	$\mathbf{0.58} \pm 0.07$	0.57 ± 0.06	0.56 ± 0.04	0.54 ± 0.03	0.56 ± 0.08
SFS - Lin (AUC)	0.57 ± 0.08	0.55 ± 0.07	0.57 ± 0.05	0.55 ± 0.04	0.59 ± 0.04
SFS - RBF (Acc)	0.58 ± 0.09	0.56 ± 0.09	0.57 ± 0.04	0.55 ± 0.04	0.54 ± 0.08
SFS - RBF (AUC)	0.56 ± 0.08	$\mathbf{0.58} \pm 0.06$	0.54 ± 0.04	0.55 ± 0.04	0.57 ± 0.06
Clinical data	0.73 ± 0.07	0.74 ± 0.04	0.74 ± 0.09	0.74 ± 0.09	0.74 ± 0.06

All of the classifiers on all of the reduction techniques perform very poorly and are as bad or worse as not using any dimensionality reduction at all. The clinical data however performs as expected, a bit lower than the results obtained by Edén et al. [12] but still within a single standard deviation for all classifiers. We can go further and examine the confusion matrices of the clinical data and the AMLC features without any dimensionality reduction to see how the classifiers manage to capture each of the two classes. This can be seen in Table 5 and Table 6.

Table 5: Confusion matrices for the clinical data, estimated over a 5-fold cross-validation.

SVM - Linear

	Pred. no met.	Pred. met.
No met.	0.68 ± 0.05	0.32 ± 0.05
Met.	0.39 ± 0.17	0.61 ± 0.17

SVM - RBF

	Pred. no met.	Pred. met.
No met.	0.64 ± 0.10	0.36 ± 0.10
Met.	0.31 ± 0.19	0.69 ± 0.19

RF - 300

	Pred. no met.	Pred. met.
No met.	0.83 ± 0.02	0.17 ± 0.02
Met.	0.49 ± 0.22	0.51 ± 0.22

RF - 600

	Pred. no met.	Pred. met.
No met.	0.83 ± 0.03	0.17 ± 0.03
Met.	0.50 ± 0.23	0.50 ± 0.23

LDA

	Pred. no met.	Pred. met.
No met.	0.70 ± 0.03	0.30 ± 0.03
Met.	0.34 ± 0.07	0.66 ± 0.07

Table 6: Confusion matrices for the AMLC features with no dimensionality reduction, estimated over a 5-fold cross-validation.

SVM - Linear

	Pred. no met.	Pred. met.
No met.	0.82 ± 0.09	0.18 ± 0.09
Met.	0.74 ± 0.09	0.26 ± 0.09

SVM - RBF

	Pred. no met.	Pred. met.
No met.	0.62 ± 0.23	0.38 ± 0.23
Met.	0.58 ± 0.29	0.42 ± 0.29

RF - 300

	Pred. no met.	Pred. met.
No met.	0.95 ± 0.02	0.05 ± 0.02
Met.	0.91 ± 0.06	0.09 ± 0.06

RF - 600

	Pred. no met.	Pred. met.
No met.	0.94 ± 0.03	0.06 ± 0.03
Met.	0.91 ± 0.06	0.09 ± 0.06

LDA

	Pred. no met.	Pred. met.
No met.	0.73 ± 0.05	0.27 ± 0.05
Met.	0.61 ± 0.09	0.39 ± 0.09

The confusion matrices for the clinical data shown in Table 5 are well-balanced for both the SVM classifiers and the LDA classifiers. It is however equally unbalanced for both the RF classifiers.
It seems that the two sets of trees in the random forest do not have any impact on the actual results for this data set. The different SVM kernels perform equally well and yield similar results, but the standard deviation for class 1 with the RBF kernel is twice as high. The LDA seems to

have the strongest performance, well balanced with a high accuracy and low standard deviation in both class 1 and class 2. The standard deviation for class 2 where there are very few values in the validation data becomes very large for all of the remaining classifiers.

In comparison, the confusion matrices for the unreduced AMLC features shown in Table 6 are always skewed. The only exception is the SVM with the RBF kernel, although it instead has a large variance in both classes which indicates that some of the k-fold evaluations where skewed as well.

5 Classifying the Image Data

In this chapter, the relevance of using CNNs on the core biopsy images for predicting distant metastasis development will be tested. Both custom made small networks and different fine-tuning settings on the classical CNN AlexNet will be used to measure if and how well the prognosis can be predicted. Before any testing can begin the images will be cropped into a set of smaller subimages and further preprocessed by normalization and data augmentation.

For comparing, the different neural network configurations only the images belonging to the patients in the first fold will be used. This is done in order to save time as the networks require a very long time to train and evaluate. By not training and evaluating on every fold, we will have to forgo estimating the network performance variance for the available data. However, in our case we want only to compare the relative difference between network settings and since the T and S sets will not change between runs the scheme is deemed suboptimal but adequate.

5.1 Image Preprocessing

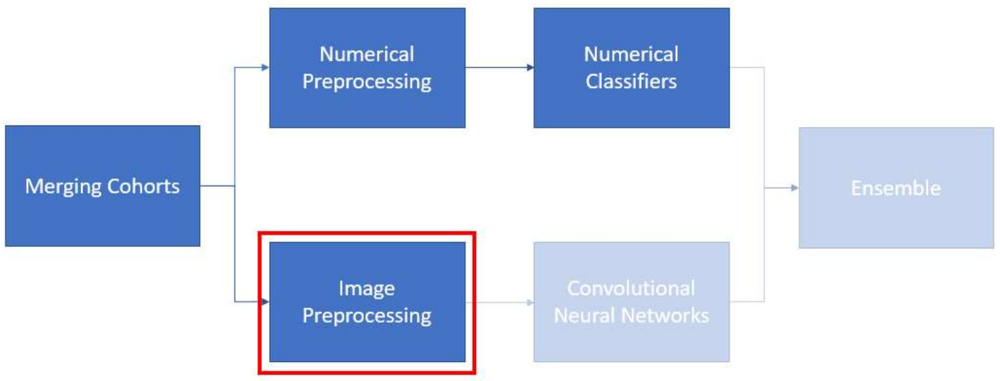


Figure 21: The red rectangle indicates what part of the ensemble structure we currently are presenting.

5.1.1 Extracting Cropped Images from the Core Biopsies

The core biopsy images in the data set are all very large, approximately 3500 × 3500 pixels each, and cover many small details. A typical CNN takes input images of a much smaller size and thus the large biopsies need to be preprocessed in some way. The original implementation of the CNN AlexNet [9] took images of 256 × 256 to generate an input image to the network of 224 × 224 by data augmentations. Since we want to use similar types of data augmentations our aim is to preprocess the core biospy images down to images of 256 × 256 pixels. A comparison between the size of the core biopsy and the sought input image is shown in Figure 22.

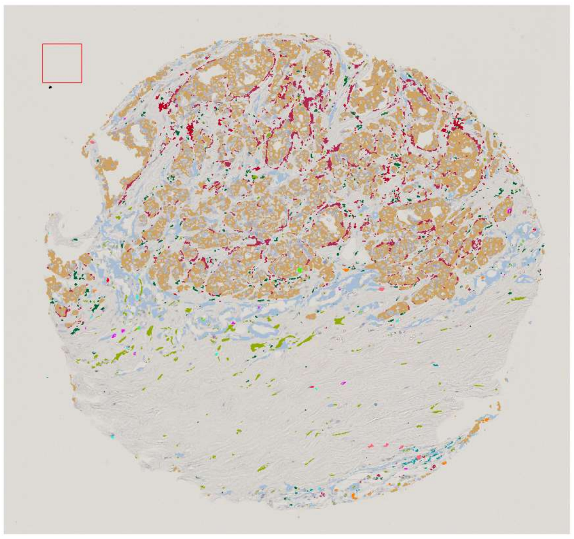


Figure 22: An image of a core biopsy from the data set. The orange areas are regions of cancerous cells. The small red rectangle is a 256×256 crop for scale.

The small details in the large image can become problematic if a simple down scale is conducted as it will not keep the finer structure, such as biomarkers, which might be important for classification. Another approach is to crop the image into smaller subimages of size 256×256. However, cropping also has its own problems. Cropping a large image will not keep larger structures that might be present in the image. A middle ground solution is instead to first scale down the original image by a fraction and then cropping the image.

Before defining a cropping method, we must address the next problem; how can we know that it is even possible to predict distant metastasis from a crop? A feasible assumption is that *if* it is possible, then it is only possible from parts of the image that contain enough cancer cells. The rest of the crops containing very few cancer cells will thus not contribute to classification. Since there is only a single true value for the entire image, a crop from which classification is impossible will introduce noise in the data set which will interfere with the entire classification process.

The color of the cancer cell biomarkers from the AMLC analysis is known and by comparing each pixel with this color a binary image can be created. However, since the images have been scaled down the borders of the cancer areas will be contaminated with non-cancer colored pixels. This is because on the border both pixels with and without this color will have been averaged to create a new pixel in the down-scaling. The measured cancer area will thus be somewhat smaller than the true area if only the exact color is considered. This is especially true if we have a sample with lots of very small cancerous regions. To combat this all pixel values within a small neighbourhood of the sought color is considered to be a pixel of a cancerous region.

After the binary image of cancerous and non-cancerous regions has been created, the number of true pixels relative to the false pixels can be counted and used to define a threshold for inclusion. To keep the vital regions centered in the crops we will only search for cancerous regions in a central square of half the size of the crop as seen in Figure 23.

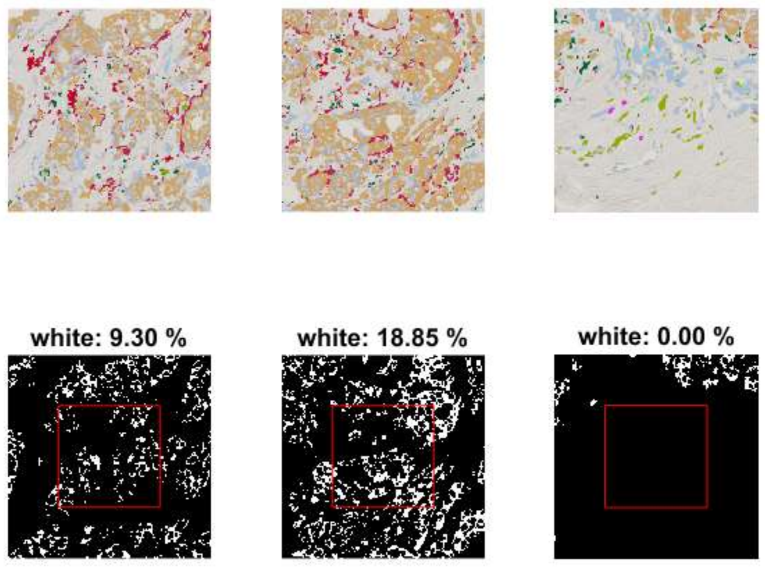


Figure 23: Three different crops of size 256x256 from Figure 22. The image has first been scaled down to 30 % of the original size. The lower row contains the same crops but as binary images where the white pixels represent the areas with cancer cells. The red square denotes the area in which we search for cancer cells and the percentage in the titles show how many pixels marked as cancer are found within the square.

After rescaling the input image into a more suitable size and defining an algorithm for selecting crop suggestions we need to define a way to perform the cropping itself. For this task, a sliding window method was used to generate crop suggestions. A window will slide over the image from left to right in a deterministic fashion, in each step a crop suggestion will be generated and tested for inclusion. The step length is set to half the crop size, i.e., 128 pixels, as this will make the area in which we search for cancer cells unique for each crop. Finally, each included crop is given the same label as the original image.

5.1.2 Image Normalization

Before we can train a network the image data needs to be normalized. For the network AlexNet [9], the normalization was performed using zero centering, i.e., from each image substracting the image mean calculated from the entire training data. This approach is used in the Matlab implementation of AlexNet as well, and it is the normalization we will use in this thesis for both our fine-tuning and the three custom made networks.

5.1.3 Data Augmentation

To increase robustness and decrease overfitting, data augmentation should be implemented to expand the training set. For inspiration, we can yet again look at the original AlexNet implementation [9] which used the following types of data augmentations; *mirroring*, *translation* and *color channel transformation*. Mirroring creates a new image by horizontal reflection of the original image, translation extracts a random crop of the correct input size from the 256x256 input image whilst color transformation randomly alters the intensities of the different color channels.

Both mirroring and translation will be used as data augmentations for our networks. Color channel transformation will however be discarded. This is due to the fact that the image data are all taken during controlled conditions and the color staining are always the same for the same type of markers, i.e., the cancer cells will always have the same orange color. Altering the color channels will thus not decrease any overfitting since there is little to no natural color variation in the images.

Apart from the two augmentations kept from AlexNet, a third will be added, namely *rotation*. By randomly rotating the image either 0, 90, 180 or 270 degrees, we can expand a single image to 4 different variations. This is possible because compared with the images used to train the original AlexNet, our data set has no natural orientation of up and down.

5.2 Training the Neural Networks

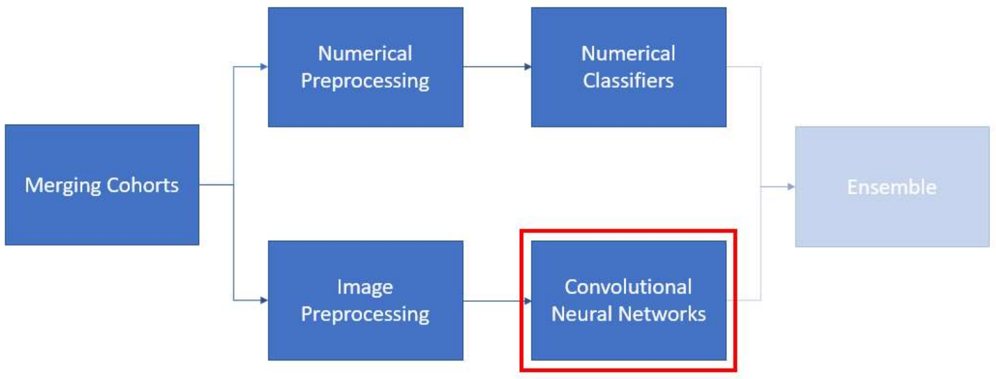


Figure 24: The red rectangle indicates what part of the ensemble structure we currently are presenting.

For testing the predictions, three types of custom made network structures trained from scratch and four fine-tuned settings of AlexNet will be used. Since it is not known how well or if at all it is possible to predict the development of metastasis from our data set or which biomarkers present in the packages that are relevant, we will evaluate the networks on the image packages 1-5 presented in the introduction. The best image packages and the best networks will lastly be evaluated in a 5-fold cross-validation in order to estimate a robust prediction error.

The three custom made networks are all loosely based on AlexNet but smaller and referred to as SmallNet-4, SmallNet-5 and SmallNet-6, the numerical suffix indicates how many learnable layers the networks contain. The main difference between the networks is the number of convolutional layers they contain while the number of fully connected layers remains the same. The custom made networks are tested for evaluating the performance of different structures. The small networks do also have fewer parameters compared to AlexNet, the large amount of parameters might limit the performance on our small data set. The structures of the three networks and AlexNet are shown in Appendix C.

Before fine-tuning AlexNet, we will first have to replace the final fully connected layer which has 1000 output nodes for the 1000 original classes that AlexNet uses, with a new layer containing only 2 output nodes for our binary classification. Later, four different fine-tuning settings will be tested. Since there only exists general guidelines on how to fine-tune a network, it was deemed necessary to test many different approaches. The settings to be used are shown below.

1. Ordinary fine-tuning, no further settings apart from replacing the final layer.
2. Replacing all the fully connected layers with layers of smaller sizes. From (4096, 4096, 1000) to (4096, 1024, 2).
3. Freezing the first two convolutional layers.
4. Freezing the first three convolutional layers.

All of our networks will be trained using the built in Matlab library for deep learning, more detail on the actual implementation can be found in [43]. The following values will be used for the hyperparameters seen below.

$N_B = 256$	Size of batch
$\eta = 0.001$	Learning Rate
$\alpha_M = 0.9$	Momentum
$\alpha_R = 0.0001$	L2 Regularization

To increase convergence and reduce the risk of overfitting the learning rate will be decayed using step decay. The learning rate will be reduced by a factor of 0.9 every 2 epochs. Further, no other configurations of the hyperparameters will be used when training the networks we present.

Since our data is skewed, we want to devise a method to avoid training biased networks. The Matlab library offers no way of defining priors for convolutional neural networks, instead our solution is to subsample the crops from the training set into a new training set where the classes are balanced. The subsampling will be performed once before each epoch to create new training sets, by resampling we can utilize more of the training data from class 1 even though most of it is discarded inside an epoch.

5.3 Evaluating the Different Networks

5.3.1 Image Voting

From our trained neural networks, we will obtain prediction scores for each crop in an image. In order to obtain a single prediction for each patient, we will construct a small voting system that takes the average of each prediction score, i.e., the softmax output, for class 2 for all the crops in an image and denote the resulting value as the total image prediction score. For patients with multiple images, the average of all of the image prediction scores will be calculated to form a per patient prediction score.

A benefit of the voting ensemble is that if we assume averaging, i.e., $w_i = 1/n \quad \forall i \in [1, ..., n]$ then the method is independent of the number of classifiers used. This is to our great advantage when calculating the image prediction scores. Different images will have different amount of crops because of our crop suggestion selection method.

5.3.2 Setting Preprocessing Hyperparameters

The first thing that needs to be tested is the relevance of the two hyperparameters; scaling and cancer threshold. Initial testing has shown that a scaling of 0.3 and a cancer threshold of 0.01 seems to yield adequate results but a more thorough evaluation needs to be made. To do this, the two hyperparameters will be evaluated individually by cropping images from image package 1 which will be used to fine-tune AlexNet with ordinary settings. When testing the scaling the cancer threshold will be kept at 0.01. When evaluating the threshold the best performing scaling factor will be used. We will also compare the scaling with simply down sampling the core biopsy image down to 256x256 pixels. The results for the different scaling of the images can be seen in Table 7 while the results for the different thresholds can be seen in Table 8.

Some permuations of different hyperparameters were tested briefly, but no significant change of result was detected and because of this not tested more extensivly. Some of the different levels of scaling were also tested on other image packages, however with similar results, hence it was not further evaluated.

Table 7: Results for different scaling of the image. All results were evaluated on AlexNet and on image package 1.

Scale	Accuracy test	Accuracy vote	AUC test	AUC vote
0.1	0.54	0.50	0.58	0.58
0.2	0.56	0.61	0.59	0.65
0.3	**0.57**	**0.64**	**0.64**	**0.72**
0.4	0.56	0.54	0.61	0.69
0.5	0.56	0.52	0.59	0.64
256x256	0.54	0.57	0.52	0.55

Table 8: Results for different cancer thresholds for accepting a crop. All results were evaluated on AlexNet and on image package 1.

Cancer threshold	Accuracy test	Accuracy vote	AUC test	AUC vote
0.001	0.57	0.645	0.61	0.69
0.005	**0.59**	**0.66**	**0.63**	**0.71**
0.01	0.57	0.64	0.62	0.70
0.05	0.58	0.65	0.62	0.68
0.1	0.57	0.58	0.60	0.60

As can be seen in Table 7, the scale of 0.3 yielded the best results in all four categories while a simple down scaling gave the worst results. Scaling 0.3 will thus be selected to be used henceforth. The improvement for using the cropping method can be attributed to two things. First, the cropping method will expand the data set a lot by creating multiple images per core biopsy and thus decreasing overfitting. Secondly, as argued earlier, there might also be smaller features relevant to prediction that disappear when performing a simple down scaling.

For the different cancer thresholds seen in Table 8, the results are much more inconclusive. The best performing threshold of 0.005 is only best by a very small margin. We instead choose to use threshold 0.01 since it is almost as good. Further, time consumption is relevant for the evaluation and thresholds lower than 0.01 will yield more images which in turn will make the training process slower. In practice, we can save up to 25 hours of computing time for a full 5-fold cross-validation on five image packages by choosing 0.01 over 0.005.

5.3.3 SmallNet-4

The results for our smallest custom built network SmallNet-4 for the different image packages can be seen in Table 9. The results are all similar and the best was obtained from image package 4. To investigate the behaviour in more detail, the confusion matrix for image package 4 can be seen in Table 10.

Table 9: Results for SmallNet-4 on all five image packages, including accuracy for testing and voting as well as AUC for testing and voting.

Image package	Accuracy test	Accuracy vote	AUC test	AUC vote
1	0.60	0.64	0.61	0.65
2	0.58	0.62	0.59	0.63
3	**0.61**	0.63	0.61	0.65
4	0.61	**0.65**	**0.63**	**0.69**
5	0.56	0.64	0.56	0.63
Mean	0.59	0.63	0.60	0.65

Table 10: Confusion matrices for SmallNet-4 on image package 4.

	Train		Test		Vote	
	Pred. no Met	Pred. Met	Pred. no Met	Pred. Met	Pred. no Met	Pred. Met
No Met.	0.6487	0.3513	0.63	0.37	0.65	0.35
Met.	0.51	0.49	0.41	0.59	0.35	0.65

We can see in the confusion matrices shown in Table 10 that the accuracy for training, testing and voting for class 1 seems stable. It is to be expected that both the training and voting accuracies are higher than the testing accuracy which can be seen. For class 2 however the test accuracy is higher than the training accuracy which is unexpected.

5.3.4 SmallNet-5

The results for the second custom built network SmallNet-5 in terms of accuracy and AUC can be seen in Table 11. As with SmallNet-4, the highest AUC value was obtained for image package 4. The results are however a bit less diverse, as image packages 3 and 5 also obtain good results. The confusion matrices for image package 4 can be seen in Table 12

Table 11: Results for SmallNet-5 on all image packages, including accuracy for testing and voting as well as AUC for testing and voting.

Image package	Accuracy test	Accuracy vote	AUC test	AUC vote
1	0.61	0.67	0.63	0.68
2	0.60	0.63	0.60	0.65
3	**0.64**	0.68	**0.65**	0.69
4	0.61	0.65	0.63	**0.71**
5	0.61	**0.70**	0.63	0.70
Mean	0.61	0.67	0.63	0.69

Table 12: Confusion matrices for SmallNet-5 on image package 4

	Train		Test		Vote	
	Pred. no Met	Pred. Met	Pred. no Met	Pred. Met	Pred. no Met	Pred. Met
No Met.	0.65	0.35	0.63	0.37	0.69	0.31
Met.	0.55	0.45	0.40	0.60	0.39	0.61

Similarly to SmallNet-4, the results for class 2 shown in Table 12 are higher for the test set than the training set which is unexpected. Similarly, the results for class 1 are still normal.

5.3.5 SmallNet-6

The results for our final custom made network SmallNet-6 can be seen in Table 13. As can be seen the results are all very similar, with image package 2 giving sightly better results. The confusion matrices for SmallNet-6 on image package 2 is shown in Table 14.

Table 13: Results for SmallNet-6 on all 5 image packages, including accuracy for testing and voting as well as AUC for testing and voting.

Image package	Accuracy test	Accuracy vote	AUC test	AUC vote
1	0.58	0.61	**0.62**	0.66
2	**0.59**	**0.64**	**0.62**	**0.67**
3	0.58	0.61	**0.62**	0.65
4	0.56	0.62	0.58	0.66
5	0.55	0.62	0.55	0.63
Mean	0.57	0.62	0.60	0.65

Table 14: Confusion matrices for SmallNet-6 on image package 2

	Train		Test		Vote	
	Pred. no Met	Pred. Met	Pred. no Met	Pred. Met	Pred. no Met	Pred. Met
No Met.	0.39	0.61	0.39	0.61	0.37	0.63
Met.	0.36	0.64	0.22	0.78	0.10	0.90

As can be seen in Table 14 the results are skewed. The prediction errors for class 1 are significantly lower than 50 % while the results for class 2 are instead much higher. We can also see that the test accuracy for class 2 is higher than the training accuracy similarly to SmallNet-4 and SmallNet-5. For the voting accuracy, it seems as though the voting method enhances the skewness of the testing accuracy.

5.3.6 Fine-tuning of AlexNet

Fine-tuning AlexNet by replacing the fully connected layers with layers containing fewer artificial neurons yielded the best results in our fine-tuning testing ground. The results can be seen in Table 15. The corresponding confusion matrix for the best performing image package is shown in Table 16. No matter how the network was fine-tuned, it was prone to be skewed in class 2 for the test data and the vote of the test data.

Table 15: Results for AlexNet when changing the size of the fully connected layers for all image packages. Including accuracy for testing and voting as well as AUC for testing and voting. The last fully connected layer was randomized.

Image package	Accuracy test	Accuracy vote	AUC test	AUC vote
1	**0.61**	**0.69**	**0.64**	**0.69**
2	0.55	0.59	0.56	0.60
3	0.57	0.60	0.59	0.66
4	0.55	0.63	0.56	0.62
5	0.54	0.64	0.56	0.61
Mean	0.56	0.63	0.58	0.63

Table 16: Confusion matrices for the fine-tuned AlexNet when replacing the fully connected layers with layers of fewer nodes, evaluated on image package 1.

	Train		Test		Vote	
	Pred. no Met	Pred. Met	Pred. no Met	Pred. Met	Pred. no Met	Pred. Met
No Met.	0.63	0.37	0.57	0.43	0.61	0.39
Met.	0.47	0.53	0.35	0.65	0.23	0.77

As can be seen for the confusion matrices in Table 16 the same problem has occured for the fine-tuned AlexNet as with SmallNet-4/5/6. The class 2 results are much higher in the test data than in the training data while the results of class 1 has a more expected distribution.

Below in Tables 17, 18 and 19 the results can be seen for fine-tuning AlexNet by only replacing the final fully connected layer, freezing the first two convolutional layers and finally freezing the first three convolutional layers.

Table 17: Results for AlexNet on all image packages, including accuracy for testing and voting as well as AUC for testing and voting. The last fully connected layer was randomized.

Image package	Accuracy test	Accuracy vote	AUC test	AUC vote
1	**0.58**	**0.63**	**0.61**	0.66
2	0.53	0.51	0.55	0.60
3	0.57	0.59	0.58	0.63
4	0.55	0.56	0.56	0.63
5	0.56	0.61	0.58	**0.67**
Mean	0.56	0.58	0.58	0.64

Table 18: Results for AlexNet when freezing the first two convolutional layers for all image packages. Including accuracy for testing and voting as well as AUC for testing and voting. The last fully connected layer was also replaced.

Image package	Accuracy test	Accuracy vote	AUC test	AUC vote
1	**0.58**	0.61	**0.60**	**0.66**
2	0.56	0.57	0.57	0.63
3	0.57	0.61	0.57	0.63
4	0.54	0.57	0.56	0.52
5	0.54	**0.62**	0.56	0.63
Mean	0.56	0.59	0.57	0.61

Table 19: Results for AlexNet when freezing the first three convolutional layers for all image packages. Including accuracy for testing and voting as well as AUC for testing and voting. The last fully connected layer was also replaced.

Image package	Accuracy test	Accuracy vote	AUC test	AUC vote
1	**0.57**	0.62	0.58	**0.65**
2	0.54	0.57	0.56	0.61
3	**0.57**	**0.62**	**0.59**	**0.65**
4	0.53	0.52	0.55	0.59
5	0.54	0.55	0.55	0.61
Mean	0.55	0.58	0.57	0.62

5.3.7 Mean of each Image Package

From the previously presented results of SmallNet-4/5/6 and the different fine-tuning settings for AlexNet, it is pretty clear that the prediction does not depend that much on the choice of network structures. Some are slightly better than others but keep in mind that the results have high variance due to the small data set and lack of cross-validation. If we, for example, compare the result of the AlexNet fine-tuned with only replacing the final fully connected layers, three different results of the same setting can be seen in Table 7, Table 8 and Table 17. The three tables have different prediction errors which shows that when training a network there exists variance in the results, even when using the same settings.

We can also compare the results of the five different image packages by calculating the mean and standard deviation for each evaluation on each image package. The results can be seen in Table 20. It is clear that image package 1, containing all biomarkers available, is performing the best across all network structures. Performing a t-test with a 5% significance level between the different vectors of AUC vote scores for the image packages, reveals that image package 2 is significantly worse compared to image package 1 for the presented measurements. No other image package was shown to be worse than image package 1 using this significance level.

Table 20: Mean and standard deviation for test and vote both in accuracy and AUC, evaluated over each included image package.

Image package	Accuracy test	Accuracy vote	AUC test	AUC vote
1	**0.59** ± 0.02	**0.64** ± 0.03	**0.61** ± 0.02	**0.66** ± 0.02
2	0.56 ± 0.02	0.59 ± 0.04	0.58 ± 0.03	0.63 ± 0.03
3	0.59 ± 0.03	0.62 ± 0.03	0.60 ± 0.03	0.65 ± 0.02
4	0.57 ± 0.03	0.60 ± 0.05	0.58 ± 0.03	0.63 ± 0.06
5	0.56 ± 0.04	0.63 ± 0.04	0.57 ± 0.03	0.64 ± 0.03

5.3.8 Robust Estimations

From our tests, we can see that SmallNet-5 performed best among the custom made networks and that the fine-tuning by replacing the fully connected layers with layers of smaller size gave the best performance for AlexNet. These two networks will now be evaluated with a 5-fold cross-validation to get a more robust estimation on the actual performance of the networks on the image packages.

The results for the robust estimation of SmallNet-5 and AlexNet can be seen below in Table 21. As can be seen the results are worse than the preliminary results presented in Table 11 and only as good as the best AMLC feature results presented in Table 4. All of the image packages have similar results, slightly better than guessing. The best performing image package was image package 1 and its confusion matrices can be seen in Table 22.

Table 21: AUC results for AlexNet and SmallNet-5 with a 5-fold cross-validation.

CNN	Image Pack 1	Image Pack 2	Image Pack 3	Image Pack 4	Image Pack 5
SmallNet-5	**0.60** ± 0.05	0.59 ± 0.07	0.60 ± 0.06	0.59 ± 0.07	0.56 ± 0.04
AlexNet	**0.62** ± 0.04	**0.62** ± 0.04	0.60 ± 0.03	0.60 ± 0.04	0.60 ± 0.03

Table 22: Confusion matrices for image package 1 on SmallNet-5 evaluated over 5-fold cross-validation.

	Train		Test		Vote	
	Pred. no Met	Pred. Met	Pred. no Met	Pred. Met	Pred. no Met	Pred. Met
No Met.	0.69 ± 0.01	0.31 ± 0.01	0.68 ± 0.05	0.32 ± 0.05	0.66 ± 0.06	0.34 ± 0.06
Met.	0.51 ± 0.02	0.49 ± 0.02	0.54 ± 0.11	0.46 ± 0.11	0.48 ± 0.15	0.52 ± 0.15

In Table 22, it can clearly be seen that it is class 2 that fails to be properly classified. Class 1 manages to get continuously high scores for both training, test and vote. Class 2 also has a higher standard deviation. This behaviour was seen on all the five image packages and not just the one presented here.

Since we have estimated SmallNet-5 using 5-fold cross validation, we have obtained one out-of-sample prediction score per patient. Each test set in the folds are disjunct and together make up the entire data set. Using this we can construct a total ROC curve and AUC value for the entire data set. The plot can be seen in Figure 25.

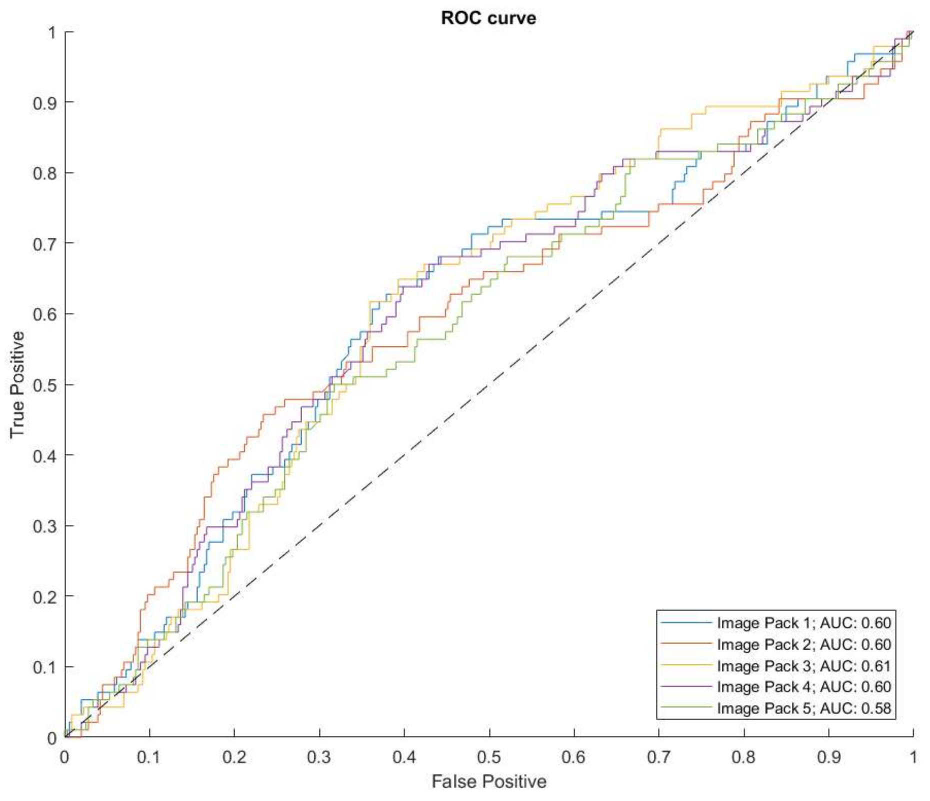


Figure 25: The ROC curve for the validation scores calculated on SmallNet-5 for all the patients. The AUC score calculated this way is somewhat higher than the estimated scores presented in Table 21 but well within one standard deviation.

6 Relevance of Immunological Biomarkers

Using our networks, we can investigate if the immunological biomarkers seem to have any effect on the prediction of metastasis development. For this, we will use our network SmallNet-5 trained with the same parameters as previous networks but on the three image packages 1, 6 and 7. Image package 1 contains all markers, image package 6 contains only non-immunological markers while image package 7 does only contain immunological markers. Since image package 7 does not contain any colored cancer cells the cropping method devised will not work, instead the positions of the crops from image package 1 are saved and used for image package 7. Three crops from the three image packages taken from the same area are shown in Figure 26.

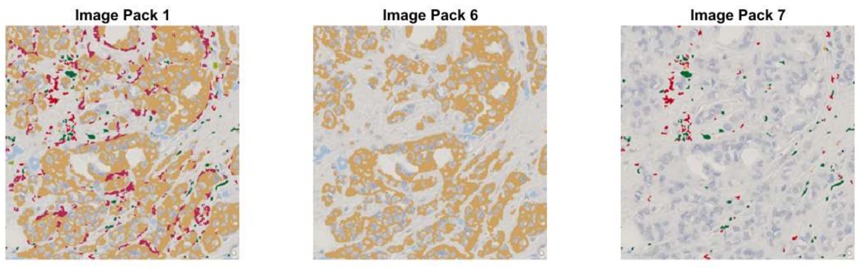


Figure 26: Three crops from the three image packages; image package 1, image package 6 and image package 7. The three crops are taken from the same position.

The results for running SmallNet-5 on the three image packages can be seen in Table 23. As can be seen no real difference between the image packages can be observed. As with the robust estimations displayed in Table 21, the AUC scores are slightly better than guessing but not more. In Table 24, the confusion matrices for the three image packages are shown. Here, the results are also similar between the image packages, no real difference can be observed between them. As with the confusion matrices shown for the robust estimation of SmallNet-5 shown in Table 22, the AUC value for class 2 is close to 0.5 and its variance is generally twice as high as for class 1. This indicates that the networks are not able to meaningfully predict this class.

	Image Pack 1	Image Pack 6	Image Pack 7
AUC	**0.61** $\pm$ 0.05	0.59 $\pm$ 0.05	0.59 $\pm$ 0.07

Table 23: Results SmallNet-5 with a 5-fold cross-validation.

Table 24: Confusion matrices of the AUC vote scores for the three image packages.

	Image Pack 1		Image Pack 6		Image Pack 7	
	Pred. no Met	Pred. Met	Pred. no Met	Pred. Met	Pred. no Met	Pred. Met
No Met.	0.66 ± 0.06	0.34 ± 0.06	0.65 ± 0.05	0.35 ± 0.05	0.61 ± 0.09	0.39 ± 0.09
Met.	0.46 ± 0.14	0.54 ± 0.14	0.51 ± 0.09	0.49 ± 0.09	0.45 ± 0.16	0.55 ± 0.16

As with the networks in Section 5.3.8, there exists one out-of-sample prediction score from the test set for each patient in the data set. These prediction scores can yet again be used to create a total out-of-sample ROC curve of the data set, this time for the three image packages 1, 6 and 7. The ROC curve is shown in Figure 27. No difference between the three image packages can be observed in the ROC curve.

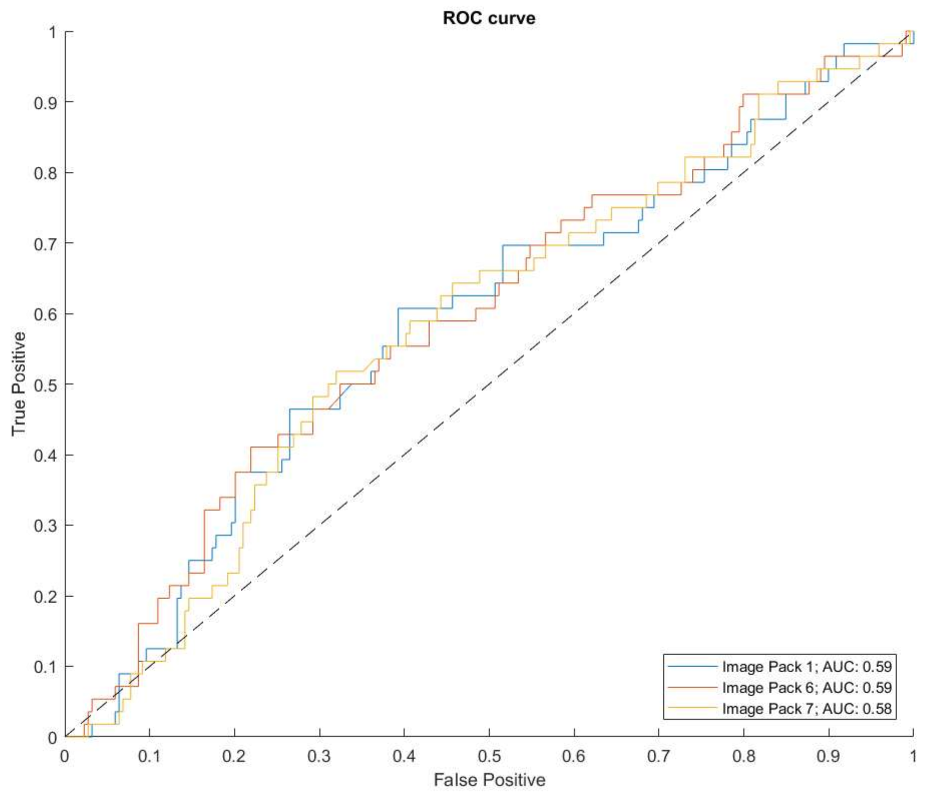


Figure 27: The ROC curve for the three image packages calculated from the out-of-sample prediction scores of the entire data set.

7 Fitting the Ensemble

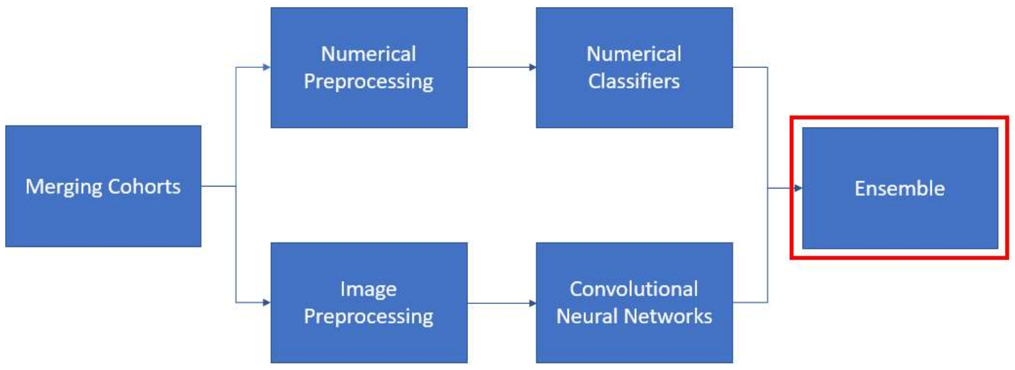


Figure 28: The red rectangle indicates what part of the ensemble structure we currently are presenting.

In this final part, we are aiming to combine the values generated from CNNs with the values from both the AMLC features and the clinical data to test if the AUC values can be boosted through an ensemble system. The CNNs to be considered are the same SmallNet-5 classifiers evaluated in Subsection 5.3.8 with the different image packages 1 to 5. We will consider the same numerical classifiers used in Section 4 with the exception that we will only use a single RF with 500 trees.

The prediction scores will be generated using a 5-fold cross-validation for each classifier for robust estimations. As before the same patient partitioning will be used for each classifier. This is vital since after the prediction scores are obtained, the different classifiers will be paired up for each patient. For the numerical classifiers, both the AMLC features and the clinical data will be considered in the ensemble, no dimensionality reduction will be performed on the AMLC features.

7.1 Voting and Stacking

Both voting and stacking will be considered in this ensemble system. For voting, we will use a simple average voting system that takes the scores from each classifier and calculates the mean of their results. For stacking, we will consider two meta-classifiers, the SVM with a linear kernel and the RF. The SVM will utilize optimization for its hyperparameters and the RF will use a bag of 500 trees.

Training of the meta-classifiers will be performed on the training set. This approach might yield a higher overfitting, since the input features are scores from classifiers trained on the same training data. However, since our data set is limited this approach was deemed necessary.

7.2 Ensemble Results

The results from the different ensembles can be seen in Table 25 were it seems that a simple vote is in terms of AUC value the general best approach. It should however be noted that the confusion matrices that can be seen in Table 26 reveals that the accuracy is skewed. This is especially true if only conducting voting on the networks. The best AUC value was obtained from RF using both the results from the networks and the clinical data. The confusion matrix from this ensemble was also fairly balanced even though class 2 as usual has the highest variance. This was achieved by limiting the leaf size to 20 of each tree that is grown in RF and introducing a miss-classification cost.

Table 25: The result of the ensemble system using voting and stacking. Presented are different combinations of the CNN scores with the other classifiers.

Ensemble	CNN	CNN + AMLC features	CNN + Clinical	CNN+AMLC features+Clinical
Voting	**0.61** $\pm$ 0.07	**0.61** $\pm$ 0.06	0.71 $\pm$ 0.05	**0.69** $\pm$ 0.05
SVM-linear	0.59 $\pm$ 0.06	0.54 $\pm$ 0.07	0.73 $\pm$ 0.11	0.67 $\pm$ 0.09
RF-500	0.61 $\pm$ 0.07	0.57 $\pm$ 0.02	**0.75** $\pm$ 0.08	0.69 $\pm$ 0.03

In total, almost no benefit can be seen with using an ensemble system. Voting with the different image packages yields an AUC score only slightly higher than the best performing image package and using CNN in conjunction with the clinical data yields very similar results compared to using the clinical data by itself. By omitting image package 5 from *CNN+clinical* it was possible to achieve an AUC score of 0.76.6 $\pm$ 0.06, which is the best result from the ensemble. Other permutations were also explored, but did not yield additional increase in AUC, accuracy or decrease in standard deviation. The confusion matrix for the best ensemble can be seen in Table 27.

Table 26: Confusion matrices for ensemble systems. The confusion matrix for the method with the highest value, marked with bold numbers in Table 25, is presented for each combination of input features.

CNN

	Pred. no met.	Pred. met.
No met.	0.79 $\pm$ 0.08	0.21 $\pm$ 0.08
Met.	0.68 $\pm$ 0.08	0.32 $\pm$ 0.08

CNN + AMLC

	Pred. no met.	Pred. met.
No met.	0.63 $\pm$ 0.07	0.37 $\pm$ 0.07
Met.	0.41 $\pm$ 0.10	0.59 $\pm$ 0.10

CNN + Clinical

	Pred. no met.	Pred. met.
No met.	0.70 $\pm$ 0.03	0.30 $\pm$ 0.03
Met.	0.33 $\pm$ 0.08	0.67 $\pm$ 0.08

CNN + AMLC + Clinical

	Pred. no met.	Pred. met.
No met.	0.85 $\pm$ 0.08	0.15 $\pm$ 0.08
Met.	0.63 $\pm$ 0.14	0.37 $\pm$ 0.14

Table 27: Confusion matrix for best performing ensemble using all clinical classifiers and all image packages except image package 5.

	Pred. no met.	Pred. met.
No met.	0.78 ± 0.03	0.22 ± 0.03
Met.	0.40 ± 0.08	0.60 ± 0.08

8 Conclusions

The first question to be answered was to what extent numerical features extracted from the AMLC stained core biopsies could be used to predict the development of metastasis. The results from AMLC features presented in Table 4 indicate that it is possible from the available data but not particularly well. Only slight differences between the classifiers and the reduction techniques can be seen and all the dimensionality reduction techniques perform as bad or worse than using all features. The classifier with the highest average AUC value and lowest standard deviation was LDA using all features, which obtained an average AUC value of 0.60 for a 5-fold cross-validation. This can be compared to the best classifier for clinical data, which also was obtained with LDA with an AUC value of 0.74. Further, the classifiers used on the AMLC features are very skewed and prone to predict class 1 more often than the classifiers that use the clinical data.

The second question to be answered was how well the AMLC images could be used to predict metastasis development using CNN. The results from the robust estimation using 5-fold cross-validation of SmallNet-5 presented in Table 21 show that it is possible, but yet again does not work particularly well. Further, the preliminary comparisons of the networks indicate that there are only minor differences on the first fold between the different structures. The estimated mean AUC value of the image packages were no better than the results obtained from the AMLC features presented in Table 4. All of the image packages in the robust estimation perform similarly and the image packages for the preliminary results shown in Table 20 also perform equally, although image package 1 was shown to be slightly better than image package 2. The confusion matrices for the robust estimation of SmallNet-5 shown in Table 22 and later in Table 24 show that the network does not manage to predict class 2 as the AUC value is always close to 0.50 and class 2 also obtains a higher standard deviation than class 1.

For the third and final question, the relevance of immunological biomarkers in the AMLC staining for predicting the development of metastasis was to be investigated. To answer the question the CNN SmallNet-5 was used in Section 6 on the three image packages 1, 6 and 7. As can be seen in Table 23, no relevance of immunological biomarkers in predicting the development of metastasis can be found. Surprisingly, the three image packages manage to perform equally well, even though completely different biomarkers were present.

The results from the different numerical and CNN classifiers were further combined in order to create an ensemble. The results from the different ensemble configurations evaluated are displayed in Table 25 which indicates that there are only marginal gains to be made. When using a voting system the confusion matrix for the ensemble showed the similar skewness as both the numerical classifiers and the CNNs. However, it was possible to obtain less skewed results in the ensemble. Using the scores from both the CNN and clinical data and training a random forest classifier it was possible get almost equal accuracy in both classes.

Finally, the aswers to the questions give reason for rejecting the hypothesis. The results show that it is possible to make predictions from the AMLC stained core biopsies of 2 mm but the methods are worse than the current standard method.

9 Discussion

The results for the classifiers on the numerical classifiers and the CNNs are considered fairly equal. However, by looking at the different confusion matrices it can be seen that the skewness is less prevalent when using the CNNs on the AMLC images. If this is due to the use of different classification methods or to the images containing structures relevant for less skewness can not be determined. Further the CNN classifiers as well as the classifiers using the clinical data showed a similar pattern in terms of the variance in prediction accuracy in class 2. This likely occurs for mainly two reasons. Firstly, as class 2 only contains about 20 patients, a single missclassification will thus have a bigger impact on the accuracy than it would on class 1, which contains about 80 patients. The second possible reason is that the number of patients in class 2 are too few, so that the variance present in the population can not be presented accurately with the current sample size.

The results for investigating the relevance of immunological biomarkers show that there is no difference between the image packages. In fact, all the image packages, both from the robust estimation of SmallNet-5 and the investigation of immunological relevance performed fairly equal. One possible explanation is that since the images are so heterogeneous only the cancer tissue are of any significant importance for classification. We then suspect that since the presence of cancer tissue is still prevalent in the images for image package 7, although not colored, the CNN manages to capture enough of the cancerous regions for a similar classification as the rest of the image packages.

The ensemble classifiers were only able to boost the results marginally. This is most likely due to the fact that the classifiers used behaved similarly and had high correlation. It is not surprising that a simple vote method was not beneficial since all classifiers were similarly skewed, the voting system mainly solidified the skewness. It should however be noted that we used the same network structure for the entire ensemble as we assumed that there would be larger difference between the image packages. If instead also using different networks a significant boost might have been achieved.

All in all, we suspect that there are two main limitations for obtaining better results; sample size of the patients and how representative the small 2 mm core biopsies are for the entire tumour area.

The numbers of patients together with the distribution of the classes have limited the analysis in several aspects. The results show that the accuracy between the training and test sets in class 1 is as expected. On the other hand, class 2 has a very large difference between the training and test sets. Further, when performing the robust estimation class 1 obtains a robust value with little standard deviation, while class 2 gets a score of about 0.50 with a much higher standard deviation. If we assume that this is due to the sample size needed for SmallNet-5 to obtain robust results, it is possible to estimate the total sample size needed. If we further assume that the variance within each class is equal and that we want the same level of difference between train and test sets, it would require a sample size of approximately 2000 patients with the same class distribution. However, if the distribution between the classes would be more equal it would require less patients.

If data from more patients were available, it would have been possible to examine the prediction results in different breast cancer subgroups. This would enable tests to better determine the relevance of immunological features in the different subgroups, e.g. patients with ER+.

In the method, it is assumed that the small 2 mm core biopsies are representative of the behaviour for the immune markers in the entire tumour, which might very well not be the case. The large amount of heterogeneity between the different cores within the same patient indicates that the characteristics of the entire tumour are not captured. The original biopsy is on the scale of 1 cm^2 and thus a lot of the available information about the patient is lost. There is a risk that potentially important biomarkers for classification will not be discovered since the presence of biomarkers varies a lot, both within the same patient and between patients. It is very possible that the limited results are more of an indicator of how non-representative the data are rather than the method and potential of using immunological features itself. Further, it should be noted that the clinical data and the AMLC images are not completely equal for comparison, since the clinical data use more information from the patient than was available from the AMLC images used in this study.

A legitimate critique to our method is the decision to not use k-fold when exploring hyperparameters and deciding the best performing networks. This decision was made since insufficient computational resources were available. Initially, the Lund University data cluster LUNARC was to be used for training and evaluation. Even though this worked initially, external circumstances beyond our control stopped the continuation of the resource. Instead, the authors had to resort to their own personal computers, which in practice meant that a 5-fold cross-validation took up to 7 days to compute. Because of the decision not to use k-fold, the initial results became higher than the final evaluation results as the different folds are very different, as can be seen in the standard deviation of the results.

In the method, it has been assumed that predicting distant metastasis is related to the cancer regions. Crops of original images were only accepted if they contained cancer cells above a certain threshold. This is an assumption that not necessarily need to be true. It is likely that cancer regions are important but other features might also play a vital part. Another approach is to simply skip this condition when cropping but then the algorithm will end up with many crops not containing any biomarkers. This will introduce a lot of noise in the form of crops that can not be classified. A suggestion for future work is to instead use other biomarkers as co-features in determining the acceptance of crop suggestions.

The focus in regard to using different pre-trained CNNs has been to explore few networks more thoroughly rather than to test several networks more briefly. Initially, it was not known if the results were connected to preprocessing, network, or quantity of data. In order to lay a foundation for future work, it was decided that it is better to more thoroughly explore the parameters and settings of AlexNet. The custom made SmallNet was used to compare against AlexNet to give indications on what was most influential on the performance. In hindsight the difference was not that significant, but SmallNet was less prone to overfit. This is most likely due to fewer parameters. However, there are many other networks that have shown great performance that were not tested. It can thus not be ignored that the use of AlexNet and SmallNet might have limited the results. Due to this, it is recommended for future work to use the best performing network available.

However, defining what is a good network is not a trivial matter. A network having a high score in image recognition is very dependent on the type of classes and the data available, and thus it does not necessarily mean that the network is good at a completely different recognition task. This can also be a problem with using AlexNet, since it is not trained on histopathological data but on standard image recognition. The best solution would be if there was a network available that had been trained using a large set of biopsy images to remove the large gap in data types. At current time, no such network was easily available.

It is possible to obtain an image channel for each biomarker in the AMLC stained images. In the exported images, the standard RGB channels were however the only ones available. Theoretically, the image contents are identical since the biomarker values are binary. However, when scaling down the images the RGB values for the biomarkers might be blurred which will not be the case if they have their separated channels. It is also feasible that it is easier for a network to be trained on images that contain separated channels for each biomarker. Using different channels for each biomarker would also remove any underlying tissue from the image, which would remove the possibility that a CNN captures uncolored cancer tissue which was probably the case for image package 7. For future, work this approach is recommended.

Finally, for any future studies we strongly suggest that a larger sample size is used. This will most likely solve the issue of skewed class accuracies and the high variance for the patients in class 2 and between folds.

References

[1] Bröstcancerprocessgruppen. *Rosa Boken.* Västra Götlandsregionen, Region Halland och Regonalt cancercentrum väst, Halmstad, Sweden, 2016.

[2] Stewart B. W. and Wild C. P. *World Cancer Report 2014.* International Agency for Research on Cancer, Lyon, France, 2014.

[3] Breast Cancer Prevention Partners. Breast cancer subtypes, 2016. URL: `https://www.bcpp.org/resource/breast-cancer-subtypes/#_edn7`.

[4] Galon J., Pagès F., Marincola F. M., Thurin M., Trinchieri G., Fox B. A., and Ascierto P. A. The immune score as a new possible approach for the classification of cancer. *Journal of Translational Medicine*, 10(1), 2012.

[5] Bindea G., Mlecnik B, Fridman W. H., Pagès F., and Galon J. Natural immunity to cancer in humans. *Current Opinion in Immunology*, 22(2):215–222, 2010.

[6] Galon J., Costes A., Sanchez-Cabo F., Kirilovsky A., Mlecnik B., Lagorce-Pagès C., and Zinzindohoué F. Type, density, and location of immune cells within human colorectal tumors predict clinical outcome. *Science*, 313(5795):1960–1964, 2006.

[7] Mlecnik B., Tosolini M., Kirilovsky A., Berger A., Bindea G., Meatchi T., and Galon J. Histopathologic-based prognostic factors of colorectal cancers are associated with the state of the local immune reaction. *Journal of Clinical Oncology*, 29(6):610–618, 2011.

[8] Pagès F., Berger A., Camus M., Sanchez-Cabo F., Costes A., Molidor R., and Meatchi T. Effector memory t cells, early metastasis, and survival in colorectal cancer. *New England Journal of Medicine*, 353(25):2654–2666, 2005.

[9] Krizhevsky A., Sutskever I., and Hinton G. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25.* Neural Information Processing Systems, 2012.

[10] Gummeson A., Arvidsson I., Ohlsson M., Overgaard N. C., Krzyzanowska A., Heyden A., and Aström K. Automatic gleason grading of he stained microscopic prostate images using deep convolutional neural networks. In *In SPIE Medical Imaging*, pages 101400S–101400S, 2012.

[11] Cireşan D.C., Giusti A., Gambardella L.M., and Schmidhuber J. Mitosis detection in breast cancer histology images with deep neural networks. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2013*, 2013.

[12] Edén P., Ritz C., Rose C., M. Fernö, and Peterson C. Good old clinical markers have similar power in breast cancer prognosis as microarray gene expression profilers. *European journal of cancer*, 40(12):1837–1841, 2004.

[13] Rydén L., Jönsson P. E., Chebil G., Dufmats M., Fernö M., Jirström K., and Nordenskjöld B. Two years of adjuvant tamoxifen in premenopausal patients with breast cancer: a randomised, controlled trial with long-term follow-up. *European journal of cancer*, 41(2):256–264, 2005.

[14] Rutqvist L. E., Hatschek T., Ryden S., Bergh J., Bengtsson N. O., Carstensen J., and Wallgren A. Randomized trial of two versus five years of adjuvant tamoxifen for postmenopausal early stage breast cancer. *J Natl Cancer Inst*, 88:1543, 1996.

[15] Malmstrom P., Bendahl P. O., Boiesen P., Brunner N., Idvall I., and Fernö M. S-phase fraction and urokinase plasminogen activator are better markers for distant recurrences than nottingham prognostic index and histologic grade in a prospective study of premenopausal lymph node–negative breast cancer. *Journal of clinical oncology*, 19(7):2010–2019, 2001.

[16] Shalev-Swartz S. and Ben-David S. *Understanding machine learning: From theory to algorithms.* Cambridge University Press, New York, USA, 2014.

[17] Kohavi R. A study of cross-validation and bootstrap for accuracy estimation and model selection. *International joint conference on Artificial intelligence*, 2(14):1137–1145, 1995.

[18] Hughes G. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14(1), 1968.

[19] Cawley G. Webb A. R., Copsey K. *Statistical Pattern Recognition.* Wiley-Blackwell, Oxford, 2011.

[20] Pearson K. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(11):559–572, 1901.

[21] Hotelling H. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933.

[22] van der Maaten L. and Postma E. amd van den Herik J. Dimensionality reduction: A comparative review. Technical report, Tilburg University, Tilburg Center for Creative Computing, 2009.

[23] van der Maaten and Hinton G. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11):2579–2605, 2008.

[24] Hastie T., Tibshirani R, and Friedman J. *The Elements of Statistical Learning Data Mining, Inference, and Prediction.* Springer, New York, USA, 2009.

[25] Breiman L., Friedman J. H., Olshen R. A., and Stone C. J. *Classification and Regression Trees.* Wadsworth, Belmont, California, 1984.

[26] Breiman L. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[27] Bishop C.M. *Pattern Recognition and Machine Learning.* Springer Science+Business Media, LLC, Singapore, 2006.

[28] Haykin S. *Neural Networks, A Comprehensive Foundation.* Pearson Education, Delhi, India, 1999.

[29] Ohlsson M. Lecture notes on artificial neural networks (FYTN06). Lund University, Department of Astronomy and Theoretical Physics, Fall 2015.

[30] Li F.F., Karpathy A., and Johnson J. Convolutional neural networks for visual recognition (CS231n). Stanford University, Stanford Vision Lab, 2016. URL: `http://cs231n.github.io/convolutional-networks/`.

[31] Nielsen M. Neural networks and deep learning. Determination Press, 2015. URL: `http://www.neuralnetworksanddeeplearning.com`.

[32] Vedaldi A., Lenc K., and Gupta A. Matconvnet - convolutional neural networks for MATLAB. *CoRR*, 2014. URL: `http://arxiv.org/abs/1412.4564`.

[33] Hinton G., Srivastava N., Krizhevsky A., Sutskever I., and Salakhutdinov R. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[34] Yosinski J., Clune J., Bengio Y., and Lipson H. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems 27*, pages 3320–3328. Curran Associates, Inc., 2014. URL: `http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf`.

[35] Azizpour H., Sharif Razavian A., Sullivan J., Maki A., and Carlsson S. From generic to specific deep representations for visual recognition. In *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 36–45, 2015.

[36] Sharif Razavian A., Azizpour H., J. Sullivan, and Carlsson S. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813, 2014.

[37] Donahue J., Jia Y., Vinyals O., Hoffman J., N. Zhang, Tzeng E., and Darrell T. Decaf: A deep convolutional activation feature for generic visual recognition. *Icml*, 32:647–655, 2014.

[38] Polikar R. *Ensemble Machine Learning Methods and Applications.* Springer, Boston, MA, 2012.

[39] Kantardziv M. *Data Mining: Concepts, Models, Methods and Algorithms.* Wiley, Hoboken, NJ, 2011.

[40] Fawcett T. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[41] Lobo J. M., JiménezValverde A., and Real R. Auc: a misleading measure of the performance of predictive distribution models. *Global ecology and Biogeography*, 17(2):145–151, 2008.

[42] van der Maaten L. Matlab toolbox for dimensionality reduction, 2017. URL: `https://lvdmaaten.github.io/drtoolbox/`.

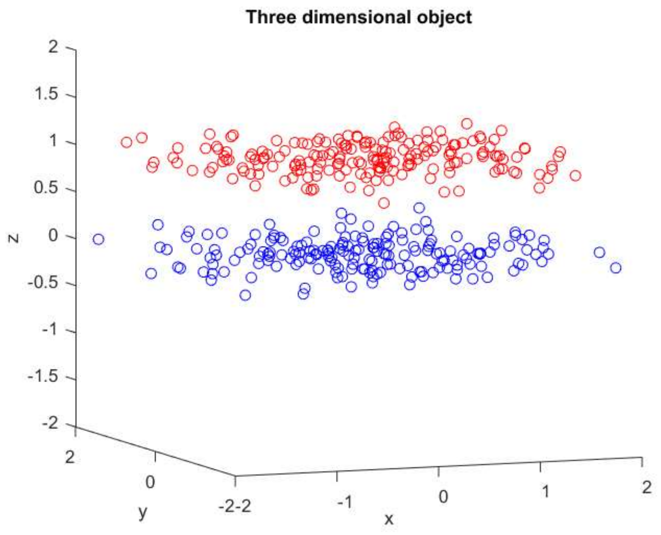
[43] Andersson O. and Ruuskanen J. classification thesis, 2017. URL: `https://github.com/OlaJohan/classification_thesis`.

A Biomarkers in the AMLC stained biopsy samples

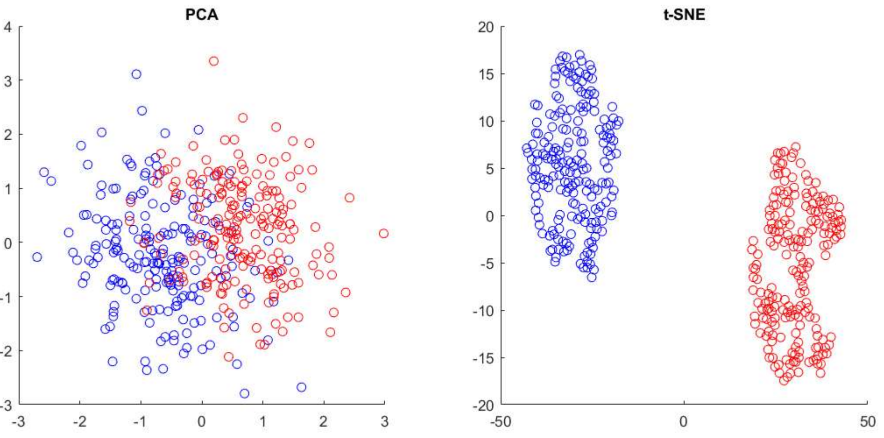
Table 28: All available biomarkers in AMLC stained biopsy samples that was provided by Medetect.

Biomarker	Description
Cytokeratin	Epithelial marker, in this case tumour tissue
Ki67	Protein, proliferation marker
p63	Basal-cell, part of the epithelial
CD8	T cells (cytotoxic)
CD3	CD4+ T cells (T helper cells)
CD11C	Myeloid dendritic cells
CD20	B-lymphocyte
CD31	Endothelium (blood vessels)
CD68	Macrophage
CD138	Plasma cell (white blood cell)
MPO	Neutrophil
Tryptase	Mast cell (white blood cell)
Langerin	Dendritic cell
SMA	Smooth muscle tissue
HTX	Cell nucleus
FOXP3	Protein expressed among others as suppressor T cells
Vimentin	Intermediate filaments
PGP	Nerves and soe tumour tissue

B Comparing PCA to t-SNE



(a)



(b)

Figure 29: Visualization of two parallell planes in 3 dimensions. (a) Two classes of data generated as 3 dimensional normal distributions. Each dimension has a standard deviation of 1 except for the z direction which has a standard deviation of 0.1. Class one has a expected value vector of $\mu_1 = (0, 0, 0)$ while class two has a expected value vector of $\mu_2 = (0, 0, 1)$. (b) As we can see in this small example, the PCA does not manage to capture the two clusters that well when those are not separated in the maximum variance, whereas t-SNE manages to do so.

C The Network Structure of the CNNs Used.

All convolution and fully connected layers uses ReLU activation functions. The suffix of the Small-Nets refer to the number of learnable layers, i.e. convolutional layers and fully connected layers.

Structure of SmallNet-4

Layer	Type	Description
1	Input	Input layer of 227x227x3.
2	Conv	32 5x5x3 convolutions with stride 3 and padding 1.
3	Conv	64 3x3x32 convolutions with stride 2 and padding 1.
4	FC	256 neuron fully connected layer with 50% dropout.
5	FC	2 neuron fully connected layer.
6	Output	2 class probabilities from SoftMax normalization.

Structure of SmallNet-5

Layer	Type	Description
1	Input	Input layer of 227x227x3.
2	Conv	32 5x5x3 convolutions with stride 3 and padding 1.
3	Conv	64 3x3x32 convolutions with stride 1 and padding 1.
4	Conv	64 3x3x32 convolutions with stride 2 and padding 1.
5	FC	256 neuron fully connected layer with 50% dropout.
6	FC	2 neuron fully connected layer.
7	Output	2 class probabilities from SoftMax normalization.

Structure of SmallNet-6

Layer	Type	Description
1	Input	Input layer of 227x227x3.
2	Conv	32 5x5x3 convolutions with stride 3 and padding 1.
3	Conv	128 3x3x32 convolutions with stride 2 and padding 1.
4	Pool	2x2 max pooling layer stride 2 and padding 0.
5	Conv	128 3x3x32 convolutions with stride 1 and padding 1.
6	Conv	128 3x3x32 convolutions with stride 1 and padding 1.
7	Pool	2x2 max pooling layer stride 2 and padding 0.
8	FC	256 neuron fully connected layer with 50% dropout.
9	FC	2 neuron fully connected layer.
10	Output	2 class probabilities from SoftMax normalization.

Structure of AlexNet in the Matlab 2017a implementation

Layer	Type	Description
1	Input	Input layer of 227x227x3.
2	Conv	96 11x11x3 convolutions with stride 4 and padding 0 with Local Response Normalization (LRN).
3	Pool	3x3 max pooling later with stride 2 and padding 0.
4	Conv	256 5x5x48 convolutions with stride 1 and padding 2 with LRN.
5	Pool	3x3 max pooling later with stride 2 and padding 0.
6	Conv	384 3x3x256 convolutions with stride 1 and padding 1.
7	Conv	384 3x3x192 convolutions with stride 1 and padding 1.
8	Conv	256 3x3x192 convolutions with stride 1 and padding 1.
9	Pool	3x3 max pooling later with stride 2 and padding 0.
10	FC	4096 neuron fully connected layer with 50% dropout.
11	FC	4096 neuron fully connected layer with 50% dropout.
12	FC	1000 neuron fully connected layer.
13	Output	1000 class probabilities from SoftMax normalization.