LU TP 19-16
May 2019

Microarray Quality Control using Artificial Neural Networks

Emil Andersson

Department of Astronomy and Theoretical Physics, Lund University

Master thesis supervised by Patrik Edén

LUND UNIVERSITY

Abstract

When using antibody microarrays to diagnose diseases, the process of quality control of the microarray data is an important step. Currently, a part of this process is performed manually by visual inspection. In this master project, we aim to automate the quality control in order to make it reproducible, as well as to find out which properties of the data that have the most to do with its quality. The tools we use for automation are artificial neural networks. From the microarray data, we construct variables based on the spot and background signal, as well as their spatial variations. We find that it is possible to reproduce the visual quality assessment with small networks using the mean and standard deviation of the microarrays' background as inputs. Finally, we introduce a new, calculable measure of quality and compare it to the visual quality control classification.

Populärvetenskaplig sammanfattning

Det finns många komplexa sjukdomar som i dagsläget är mycket svåra att diagnostisera. Det är inte ovanligt att när dessa sjukdomar slutligen diagnostiseras så är det för sent för att kunna göra någonting åt det. Men med en ny teknik kan det bli möjligt att ställa diagnoser i tid. Tekniken grundar sig i att, från ett vanligt blodprov, identifiera vilka proteiner som finns i blodet varpå sedan låta komplexa datoralgoritmer analysera proteindatan och ställa en diagnos. Dock, för att datorn ska kunna ställa en korrekt diagnos är det viktigt att proteindatan är av hög kvalitet, vilket inte alltid är fallet. I dagsläget utförs kvalitetskontrollen genom att granska datan visuellt; det här masterprojektet går ut på undersöka ifall det går att automatisera, och eventuellt förbättra, kvalitetskontrollen.

För att automatisera kvalitetskontrollen provar vi att använda så kallade artificiella neuronnätverk (ANN): en speciell inriktning inom artificiell intelligens (AI) som är löst baserat på—men under inga omständigheter ett försök att simulera—den mänskliga hjärnan. ANN låter kanske avancerat, men det är egentligen bara relativt simpla datoralgoritmer som används för att lösa problem på ett lite annorlunda sätt. Om vi med konventionell programmering försöker lösa ett klassificeringsproblem, exempelvis att avgöra om proteindata är av hög eller låg kvalitet, måste de olika klasserna definieras väl i förhand. Därmed fungerar det inte alls om definitionen av klasserna inte är kända. ANN fungerar däremot på ett annorlunda sätt: de fungerar ungefär på samma sätt som små barn. Barn lär sig inte veta ifall frukten de pekar på är ett äpple eller en banan genom att deras föräldrar beskriver de olika frukternas definition för dem. De lär sig istället genom att vid en mängd upprepade tillfällen få det berättat för sig om frukten de pekar på är ett äpple eller en banan, så får de själva lära sig vad som är skillnaden. Samma princip gäller för ANN. Istället för att programmera in hur vi definierar bra respektive dålig kvalitet på proteindata, så berättar vi för dem om proteindatan som de bearbetar för tillfället är av hög eller låg kvalitet, så får de själva lära sig vad det innebär.

En automatiserad kvalitetskontroll har två stora fördelar jämfört med en manuell sådan. För det första blir processen reproducerbar då den mänskliga faktorn tas bort. För det andra får vetenskapspersonerna som för tillfället utför kvalitetskontrollen mer tid att lägga på mer avancerat arbete som en dator inte kan klara av. Dessutom skulle en förbättrat kvalitetskontroll öka datorns precision när den ställer diagnoser.

Abbreviations

ANN Artificial Neural Network

CNN Convolutional Neural Network

MAE Mean Absolute Error

MSE Mean Square Error

QC Quality Control

Acknowledgements

First and foremost, I would like to thank my supervisor Patrik Edén for his great supervising and for always having an open door and time for my questions. I would also like to seize this opportunity to thank all those people whom I have encountered during my education who have made my studies both easier and funnier. Finally, I would like to thank Markus Ernstsson, with whom I have shared an office during the last nine month, for many great discussions and for being an excellent rubber duck.

Contents

1 Introduction

With the technology of biological microarrays, it is possible to diagnose a multitude of complex diseases from a blood sample [1]. There exist many different kinds of microarrays, where DNA- and RNA microarrays are the most common [1]. A few applications of these microarrays are to find sequence aberrations in the genome, and analysis of gene expression related to diseases [2]. There is also a newer type of microarrays which map the proteome directly by using antibodies or fragments of them [2]. By using machine learning algorithms, it is possible to recognise proteins which indicate the targeted disease [3].

An antibody microarray is a platform which uses antibodies and fluorescent markers to measure the protein content in the blood [4]. The protein data are extracted from images of scanned microarrays. However, in order to obtain reliable data, and in extension a reliable diagnosis, the data must first undergo quality control (QC). This process is currently done in part manually by visual inspection of images of scanned microarrays, which is suboptimal since it is neither reproducible nor time efficient.

This master project aims to investigate if it is possible to automate the process of QC by using artificial neural networks (ANNs). The main focus of the project is to find which features of the data that correlate most with the quality and how to extract those from the microarrays. From the microarray data, we construct variables that contain information about the protein signals, the background, and both their spatial variations by using two-dimensional discrete Fourier transforms (DFTs). We use the different variables in various combinations as ANN-input and train the networks in order to classify the microarrays into different quality categories.

The paper is structured as follows: First, in Section 2, we give a background on how microarrays work, how the data are extracted and what kind of datasets we work with. Next, a description of the variables that were constructed and what they represent follows in Section 3. Section 4 describes the networks that we used when reproducing the visual QC and how well they performed. Finally, we introduce a new, calculable measure of quality in Section 5 and compare it to the visual QC-classifications.

2 Background

2.1 Microarrays

One way of diagnosing a variety of complex diseases from a blood sample, is by using antibody microarrays [4]. The antibodies are fixed at specific locations on a plate and bind to their respective proteins when the blood sample is applied. There are means of detecting bound proteins, which means that it is possible to deduce the protein content in the sample.

There is a multitude of papers and reviews on the development and workings of antibody microarrays by Borrebaeck and Wingren, see for instance [4–8], but we will give a short introduction here as well. On a *microarray*, several *spots* of *antibodies* (antibodies will often be referred to as *reporters* in this thesis) are fixed, ordered in a matrix formation. The size of a microarray is less than $1\,\mathrm{cm}^2$ but contains several hundreds of spots, see Figure 1. A *slide*, in turn, is a plate with a solid surface containing several microarrays ordered in two columns. The antibodies that are used bind only to specific proteins, where it is a one-to-one mapping between antibodies and proteins.

There are several ways of making the proteins detectable [7, 8]. The main idea is that the proteins are marked with a fluorescent molecule. After the sample has been applied to the microarray, and the proteins have had time to bind to the antibodies, the unbound residues are washed away. Hence, it is possible to detect the fluorescent molecules by scanning the microarray with light of a specific wavelength. Given that the number of fluorescent molecules bound to the proteins is proportional to the protein level, the scanned intensity of a spot will also be proportional to the amount of that specific protein.

A few examples of how a scanned microarray may look are shown in Figure 1. The figure furthest to the right shows a scan with high quality: the spots stand out clearly from the background and there is a clear difference in intensity between the

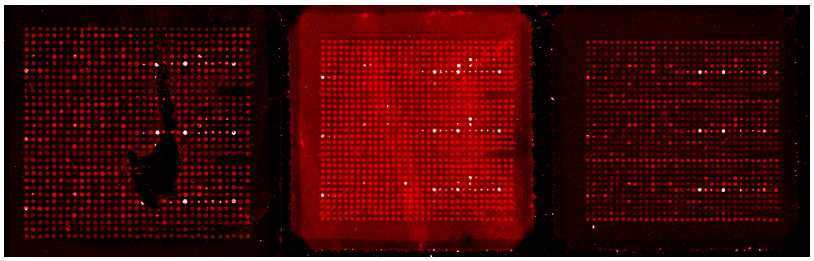


Figure 1: Examples of scanned microarrays. The left and the central image are both examples of scanned microarrays with low quality. The left microarray is damaged which give rise to the dark area while the central scan is overexposed. The image to the right is an example of a scanned microarray with high quality. Image courtesy of Department of Immunotechnology, LTH.

spots. The middle figure shows an example of how a scan with low quality may look: it has very high background intensity which could be due to overexposure or that some part of the sample preparation failed. From an image like this, it is hard to say anything accurate about the protein data unless the effect of overall high intensity is understood. The image furthest to the left shows how a scanned microarray may be of low quality in another way: the scan itself is good, but there is a large dark area where the array has been damaged. By considering these examples of scans, it can be understood why the quality control of microarray data is of such importance. There are many ways in which the scans may get corrupted which, if used, leads to inaccurate protein data and in extension also unreliable diagnoses. As it is today, the process of array quality control is performed manually where a visual assessment is made for each scanned microarray. We have used data where the microarrays were classified into one of three classes: *pass* if it is considered to have high quality, *discard* if it is considered to have low quality, and *note* if it is not a clear case and further assessment is needed.

2.2 The datasets

The data used in this project consist of three datasets A, B and C. Datasets A and B are rather similar to each other where their main difference is that they were produced at different times. These two sets are suitable for this study since, in addition to the arrays being of high quality, also arrays of low quality are included in the datasets. Dataset C, on the other hand, differs quite substantially from the other two and do not contain any labels from visual QC. Thus, we will only use dataset C for the last part of the study where this feature is not required.

From the images of the scanned microarrays, a set of pixels with intensities are provided. During *segmentation*, it is identified which pixels that belong to spots and which belong to the background. The segmentation yields two values for each spot: a *signal* and a *background*. There are many different ways in how the signal and background may be calculated, see e.g. review by Amaratunga and Cabrera [9]. We focused on performing array quality assessment based on the "state-of-the-art" segmentation already performed on the datasets and did not consider changing the segmentation procedure .

Dataset A contains data from 1120 microarrays while dataset B contains data from 673 microarrays. Out of the 1120 microarrays in dataset A, 805 were labelled as *pass*, 104 as *discard* and 211 as *note*. In dataset B, 472 microarrays were labelled as *pass*, 61 as *discard* and 140 as *note*. All the microarrays in these datasets are of the same kind: they all contain the same reporters in the same pattern. In total, there are 63 different reporters and 21×21 spots. On these microarrays, the pattern containing a number of spots of each reporter is repeated three times so that it is possible to notice spacial deviations in the signal. A sample from a single

patient can occur in many different microarrays. Dataset A contains data from 715 different samples, where 17 of these samples are replicated on three or more microarrays distributed on 318 different microarrays. Dataset B contains fewer samples but more replicates: there are 216 different samples where 63 of these occurs on three or more microarrays distributed on 367 microarrays.

The microarrays in dataset C are shaped differently: the spots are arranged in an 18×14-pattern. Next, the number of different reporters are 40 instead of 63. Further, there are no labels from visual QC since the original study had a different purpose than the one datasets A and B were used in. Finally, the dataset contains 1485 microarrays with 20 different samples where all of the samples occur on at least 53 microarrays each.

3 Preparation of the Data

3.1 Construction of the QC-variables

In order to be able to use ANNs, we need suitable variables that can be used as input. Since the goal is to assess the quality of arrays, we need variables that say something about an array as a whole. It would, therefore, not be suitable to use the individual spot-data directly since that data only say something about the spots themselves. Further, there would be an unreasonable amount of input. Thus, we need to construct new variables from the spot and background data.

As mentioned in Section 2.1, low quality could be that the whole array has a very high intensity or that there are dark areas. Hence, we have constructed four variables which may capture these effects. These four, which will be referred to as the QC-variables, are the mean background intensity of an array, the standard deviation of the background intensity, the standard deviation of the relative expression (see definition in Section 3.1.2), and the Pearson correlation between the background intensity and relative expression. In the following sections, we will motivate why these variables are used and how they are constructed.

3.1.1 Background mean and standard deviation

Ideally, a scanned microarray should look like the array to the right in Figure 1 where the background intensity is low and fairly constant. Thus, if the background intensity is high, and if it varies a lot over the array, that should imply low quality. These effects can be captured by the mean and the standard deviation of the background intensity.

The data contained a measure for the background intensity of each separate spot. However, before we calculate the mean and standard deviation, we take the

2-logarithm:

$$\text{log\_bg}_{a,i} = \frac{1}{\ln(2)} \ln(\text{background}_{a,i}), \tag{3.1}$$

for spot i on array a. From this, it is straight forward to define the mean and standard deviation:

$$\text{log\_bg\_mean}_a = \frac{1}{N_{\text{spots}}} \sum_{i=1}^{N_{\text{spots}}} \text{log\_bg}_{a,i} \tag{3.2}$$

and

$$\text{log\_bg\_stddev}_a = \sqrt{\frac{1}{N_{\text{spots}} - 1} \sum_{i=1}^{N_{\text{spots}}} \left(\text{log\_bg}_{a,i} - \text{log\_bg\_mean}_a\right)^2}. \tag{3.3}$$

One reason to take the logarithm is that log_bg_stddev and log_bg_mean are fairly independent, as seen in Figure 3 (a).

It might happen that spots are missing in the data file. This could be due to that the spot was deformed, had not been printed correctly on the microarray, or that there was some issue with the array segmentation. In the case when a spot is missing, it is replaced with log_bg_mean_a calculated from the existing spots on the array.

3.1.2 Relative expression

The variable relative expression is based on the spot intensity and is a way to utilise the fact that there are many replicates of each reporter on a single microarray. The variable, therefore, says something about if the spot intensity varies at different spatial locations of the array, i.e. it could perhaps capture if there are some dark areas or an overall intensity gradient.

In order to calculate the relative expression and its standard deviation, first, the expression, x, must be defined. We define the expression from the difference between the spot signal and the background around the spot according to

$$x_{a,i} = \frac{1}{\ln(2)} \ln(\max\{1, \text{signal}_{a,i} - \text{background}_{a,i}\}), \tag{3.4}$$

where, as earlier, i denotes the specific spot on array a. The relative expression compares how one spot deviates from the mean of the other spots of the reporter. Thus, also the mean for each type of reporter on an array is required:

$$\langle x \rangle_{a,r} = \frac{1}{N_{a,r}} \sum_i x_{a,r,i}, \tag{3.5}$$

where r denotes the reporter. It should be clarified that $x_{a,i}$ in Equation (3.4) is the same as $x_{a,r,i}$ in Equation (3.5), where we with the new notation emphasise that we only sum over the spots with reporter r.

Now, the relative expression can be defined as

$$\text{rel\_x}_{a,i} = x_{a,r,i} - \langle x \rangle_{a,r} \,. \tag{3.6}$$

Finally, the standard deviation of the relative expression is straight forward to obtain:

$$\text{rel\_x\_stddev}_a = \sqrt{\frac{1}{N_{\text{spots}} - 1} \sum_{i=1}^{N_{\text{spots}}} (\text{rel\_x}_{a,i})^2}. \tag{3.7}$$

If there exist missing spots, they are replaced with 0 since the mean of the relative expression is

$$\langle \text{rel\_x}_{a,i} \rangle = \left\langle x_{a,r,i} - \langle x \rangle_{a,r} \right\rangle = \langle x \rangle_{a,r} - \langle x \rangle_{a,r} = 0. \tag{3.8}$$

3.1.3 Pearson Correlation

The last of the four QC-variables is the Pearson correlation between the variables log_bg and rel_x. The usage of this variable is motivated by that it could be able to capture if the pattern in the background signal and the spot signal follow each other. The Pearson correlation is defined as follows [10]:

$$\text{pearson\_correlation}_a = \frac{\left\langle \left[\text{rel\_x}_{a,i} - \langle \text{rel\_x}_{a,i} \rangle \right] \left[\text{log\_bg}_{a,i} - \left\langle \text{log\_bg}_{a,i} \right\rangle \right] \right\rangle}{\text{rel\_x\_stddev}_a \cdot \text{log\_bg\_stddev}_a}. \tag{3.9}$$

3.2 Discrete Fourier Transforms

In addition to the four constructed QC-variables, there could be other means to extract useful information from the data. One way to access the spatial structures of the background signal could be by calculating the two-dimensional discrete Fourier transform (DFT) for the arrays. Then, Fourier components with small wavenumbers correspond to large spatial patterns while components with large wavenumbers correspond to small noise.

The two-dimensional DFT is conveniently enough performed on rectangular structures of discrete data. The spots on the microarrays compose such a structure and we can make the transformation separately for both the background and the relative expression. With n_1 and n_2 being the coordinates for the spot on the array, where the array consists of $N_1 \times N_2$ spots, the DFT of spot values $X_{n_1 n_2}$(either

background or relative expression) is by definition [11]

$$C_{k_1k_2} = \frac{1}{N_1N_2}\sum_{n_0=1}^{N_1-1}\sum_{n_2=0}^{N_2-1}\exp\left(-2\pi i\left[\frac{k_1n_1}{N_1}+\frac{k_2n_2}{N_2}\right]\right)X_{n_1n_2}, \tag{3.10}$$

with the inverse transform

$$X_{n_1n_2} = \sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1}\exp\left(2\pi i\left[\frac{n_1k_1}{N_1}+\frac{n_2k_2}{N_2}\right]\right)C_{k_1k_2}. \tag{3.11}$$

The normalisation is chosen so that the C_{00}-component becomes the mean:

$$C_{00} = \frac{1}{N_1N_2}\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}X_{n_1n_2} = \mu. \tag{3.12}$$

The variance can also be expressed in the terms of the Fourier components. Starting from the definition of the sample variance σ^2, we have that

$$\begin{aligned}\frac{N_1N_2-1}{N_1N_2}\cdot\sigma^2 =& \frac{1}{N_1N_2}\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}(X_{n_1n_2}-\mu)^2\\ =& \frac{1}{N_1N_2}\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}\left(X_{n_1n_2}^2-2X_{n_1n_2}\mu+\mu^2\right)\\ =& \frac{1}{N_1N_2}\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}X_{n_1n_2}^2-2\mu\underbrace{\frac{1}{N_1N_2}\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}X_{n_1n_2}}_{=\mu}+\mu^2\\ =& \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}|C_{k_1k_2}|^2-\mu^2 = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}|C_{k_1k_2}|^2-|C_{00}|^2,\end{aligned} \tag{3.13}$$

where Parseval's theorem [12] is used at the fourth equality.

By straightforward calculations, it is easy to show that the two-dimensional DFT possesses the symmetry

$$|C_{k_1k_2}|^2 = |C_{N_1-k_1N_2-k_2}|^2. \tag{3.14}$$

Thus, for the price of throwing away the information about the complex phase, the number of unique components get reduced by almost a factor of two as is shown in Figure 2.

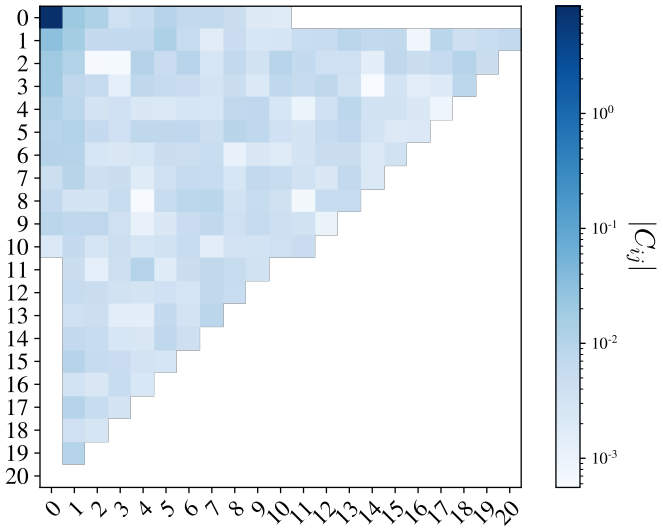


Figure 2: Example of the Fourier space with the symmetry-redundant components removed.

Finally, we divide the square of the absolute values of the components with the variance

$$Y_{k_1 k_2} = \frac{|C_{k_1 k_2}|^2}{\sigma^2}, \tag{3.15}$$

which are the variables that will be used as extra inputs to the networks. Since the sum of all the Fourier components (excluding the C_{00} component) is the variance, this measure is the proportion of the total variance which each component contains. The interpretation of the Y_{00}-component is how many times larger the mean intensity is than the sizes of the fluctuations. These quantities will in the future most often be referred to as the scaled Fourier intensities, or just the Fourier data.

3.3 Visualising the datasets with respect to the visual QC

The datasets were provided with the three quality classes *pass*, *discard* and *note* based on visual inspection. By plotting two of the four variables against each other, it is possible to see both if the two variables seem independent of each other and how well the variables separate the different classes. This have been done for the three datasets in Figure 3, and Figures 21 and 22 in Appendix A.

For most of the variable pairings, there is no clear correlation, which means that they are more or less independent. However, plot (a) in all of Figures 3, 21 and 22, i.e. log_bg_mean against log_bg_stddev, show some periodic correlation. Though, this is due to technical details of the scanner. It measures the intensity in discrete

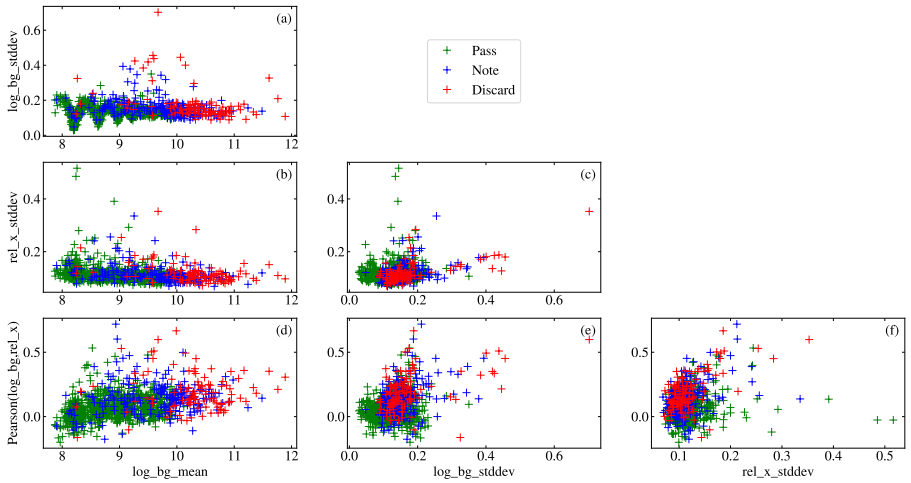


Figure 3: Two out of the four QC-variables are plotted against each other in each sub figure for dataset *A*. Each marker denotes the position for each array and the colours indicate the array's visual classification.

steps which, apparently, affects the standard deviation of the median values as well. When it comes to how well the variables partition the three classification classes, log_bg_mean seems to be the one which does that the best, especially together with log_bg_stddev as can be seen in Figure 3 (a) and 21 (a).

One interesting point to note is that in Figure 21, there are six microarrays classified as *discard* which stand out in every plot. These arrays have lower log_bg_mean than any array classified as *pass* at the same time as rel_x_stddev is high. This could correspond to that these are arrays where a large area, but not the whole array, are dark. This clearly shows that it is not that simple as to say that low log_bg_mean with low log_bg_stddev corresponds to high quality, but if it is too low, so is the quality.

A similar analysis can be made for the Fourier data. In Figure 4, the mean over all arrays for each scaled Fourier intensity-component are plotted for dataset *A*. This is done both for the Fourier data from the background and from the relative expression. For the case of the background, in subfigure (a), there is a clear separation between the three classes of the visual classification, at least for all the components where none of the wavenumbers are zero. Since there is such a clear separation, a network might be able to pick up this, even though this is just the means of all the arrays' Fourier intensities. However, there is quite a substantial spread between the Fourier intensities of the different arrays. This is illustrated in Figure 5 which shows a boxen plot [13, 14] of all the $(3, k_2)$-components of the

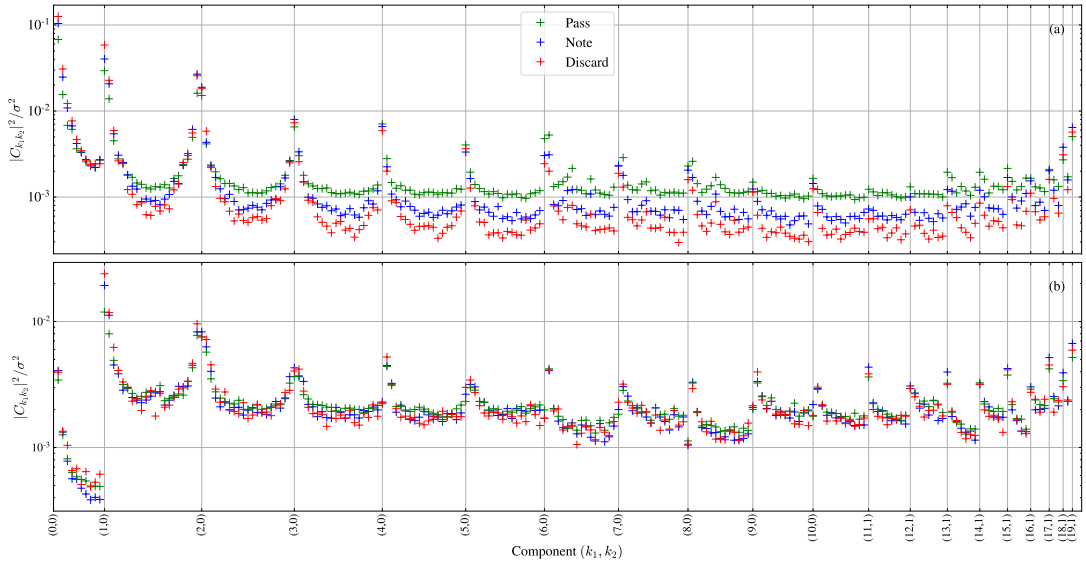


Figure 4: The mean of all the arrays in dataset A for each scaled Fourier intensity-component, where (a) shows the Fourier data for the background and (b) for the relative expression. A seperate mean is calculated for each QC-classification. The components are ordered so that consecutive rows of the Fourier space (see Figure 2) comes immediately after each other.

Fourier data for the background. In this representation of the Fourier data, the separation is not as clear as in Figure 4 and the distributions overlap, but it is still possible to see the tendency to a separation. The mean of the Fourier data of the relative expression in Figure 4(b), on the other hand, do not show a clear separation between the three classes and will therefore not be used further.

Another way to visualise the Fourier data is by plotting the Fourier space for the mean of the different classes, as done in Figure 6. In this figure, it is shown in a more direct way that it is the central region of the Fourier space that differs most between the classifications. From both Figures 4 and 6, it appears that the scaled Fourier intensities are smaller in the central region of the Fourier space while they are larger in the upper-left corner for *discard* compared to what they are for *pass*. Thus, it seems to be worse for the quality with large scale fluctuations than with small noise.

The large fluctuations between the arrays' scaled Fourier intensities could make it hard for a network when training on the Fourier data. In an attempt to reduce the fluctuations, the scaled intensity-components were averaged together in 3×3-bins. The components which had been put to zero due to symmetry reasons were not included when calculating the average of the bin, i.e. if two reduced components were in a bin, the average was calculated for the other seven components. In order to check if this way of averaging together neighbouring components is reasonable,

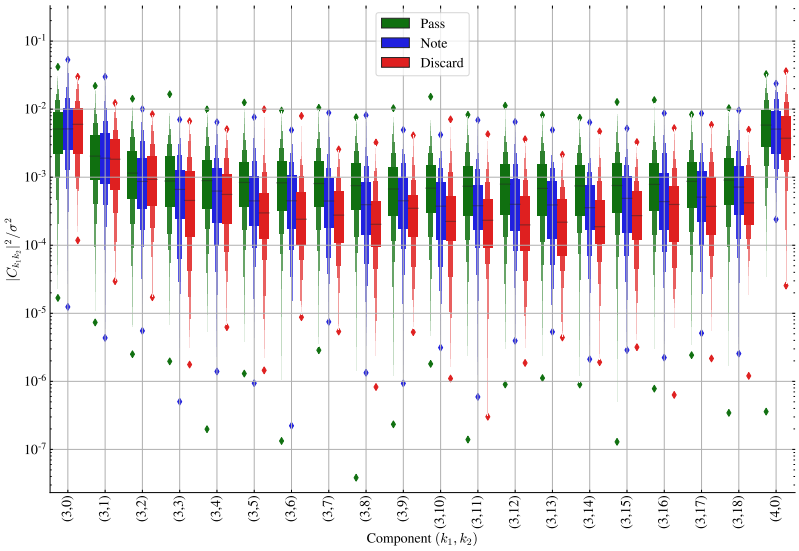


Figure 5: A boxen plot over all the $(3, k_2)$-components of the scaled Fourier intensities for the background. The arrays are separated into the three classes of visual QC: *pass*, *note* and *discard*.

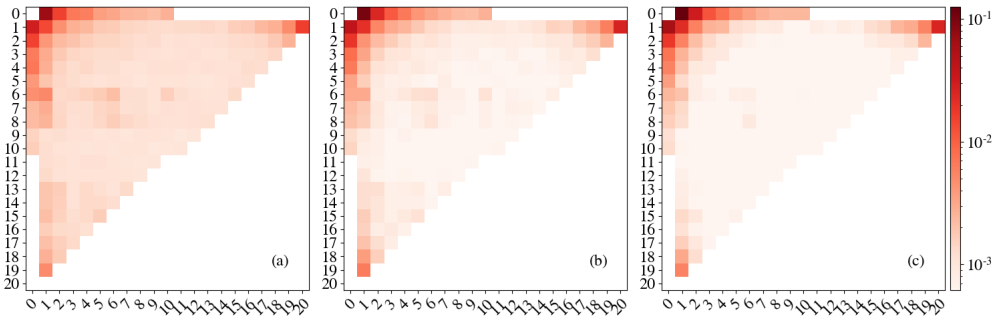


Figure 6: The mean of the scaled Fourier intensities, for each QC-category, represented in the Fourier space (*pass* in (a), *note* in (b) and *discard* in (c)). The (0,0)-components are excluded from the figure since they are about 5-6 orders of magnitude larger than the other components and would, thus, sabotage the colour scale.

the values of the binned components were plotted in the same manner as in Figure 4, see figure Figure 7. It is clear that this way of binning the Fourier data manages to keep the information that separates the different classes. Furthermore, the number of Fourier-based inputs to a network has been reduced from 221 to 31. This is of course only one possible way to bin together the intensity components: other ways of binning together the components were also used, as described in Section 4.2.3.

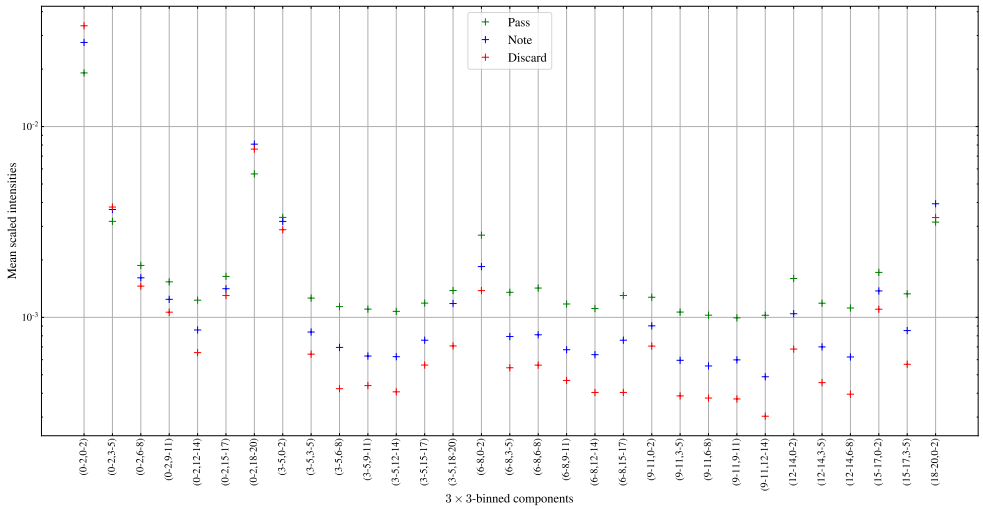


Figure 7: The scaled Fourier intensities averaged together in 3×3-bins.

4 Reproducing visual quality control

In order to reproduce the visual quality control, artificial neural networks (ANNs) were used. However, the main goal of this project is not to find the perfect network for the task, but to find out if the data contain enough information about the quality of the arrays and if it is possible to extract that information using ANNs. Therefore, we allowed ourselves to not go into the deep in trying out advanced architectures and regularisation methods, but, instead, focused on what could be done with the input data.

4.1 Method

Our approach to this problem was to build rather simple networks, using various combinations of the QC-variables and the scaled Fourier intensities discussed in Section 3, and then validating them. To keep the networks simple, we used the same number of nodes for all the hidden layers in a given network. Some larger networks were evaluated as well. For those, we used L2-norm regularisation and in some cases early stopping to reduce overtraining [15]. All the neural networks were constructed in Python3 using Keras with Tensorflow as backend [16, 17].

The datasets A and B both contain three classification categories: *pass*, *note* and *discard*. However, given that the microarrays labelled with *note* are on the border between *pass* and *discard* in the visual quality control, some of these arrays may have qualities that agree more with *pass* while others agree better with *discard*. Thus, these arrays would most likely decrease the performance of the networks.

Therefore, we structured the classification task as two different problems: one where the networks trained on complete datasets and made classifications into all three categories, and one simplified problem where all the microarrays labelled with *note* were removed from the datasets and, instead, the networks made a binary classification into either *pass* or *discard*.

Many different network architectures were used, but the mathematical functions were the same for all networks. As activation function for the hidden layers, $\tanh(x)$ was used. However, which output- and error functions that were used, of course, depended on the type of problem. For the two-classes classification problem, the binary cross-entropy was used as the error function and the sigmoid function was used as the output function with one output node [18]. For the three-classes classification problem, the categorical cross-entropy was used as the error function and the softmax function was used as the output function with three output nodes [18]. Further, Adam was used as the optimiser [19]. Finally, the networks were scored both with the accuracy, i.e. the fraction of correctly classified arrays, and with the receiver operating characteristic (ROC) area under the curve (AUC)-score [20].

Our ambition is to find out which input variables that contain the most information about the quality, and, also, if there are some network architectures that are better at extracting that information than others. Thus, we trained and validated each network separately on the two datasets A and B so that it was possible to see if the same network setup worked equally well on different datasets. Another feature that was tested was how well a networked trained on one of the datasets could predict the labels on the other dataset.

When validating one network setup, repeated K-fold validation was used [21]. We used K-fold loops with 10 folds where the dataset was divided into 10 parts by random. The K-fold loop was repeated 20 times, with different partitions for each iteration, where the mean and standard deviations of all the scores were calculated. If nothing else is stated, the networks were trained for 100 epochs. The datasets were normalised to zero mean and unit variance.

4.2 Binary networks: Classification into *pass* and *discard*

4.2.1 QC-variables as input

The first set of networks that were trained for the binary classification problem took two out of the four, or all, QC-variables in the different combinations as input at a time. In other words, we attempted to find the decision boundaries to Figures 3 and 21 with all the arrays labelled with *note* removed. A set of rather small networks were used: either one or two hidden layers with two, four or ten nodes per hidden layer. We trained networks both with L2-regularisation and

without. In the cases where L2-regularisation were used, the regularisation control parameter α was either $\alpha = 10^{-4}$ or $\alpha = 10^{-3}$. Since no tendency of overtraining was seen for these small networks in the trial runs, all the networks were trained for 100 epochs without early stopping being employed. The result for a selection of the configurations used can be seen in Figure 8 for dataset A and Figure 9 for dataset B. In the figures, bullets mark which input variables that were trained on. The shorthand notations for the variables are: M for log_bg_mean, Sbg for log_bg_stddev, Sx for rel_x_stddev and P for the Pearson correlation.

It is clear from Figure 8 that neither the L2-regularisation nor the architecture of the networks, cause much of a difference for the performance of the models; hence, not all setups are displayed in these figures nor in any future figures. The main cause of the difference is the selection of input variables. It is clear that the visual quality control is mostly based on the background mean with some help of the fluctuations in the background signal, something which could be suspected from Figures 3 and 21. The best accuracy achieved was 0.974 for the network with two layers with ten hidden nodes per layers and $\alpha = 10^{-4}$ with background mean and standard deviation as input, while the best AUC was 0.981 for the same network architecture but with all the QC-variables as input and without L2-regularisation. None of the networks that were trained on dataset B reached as high numbers; however, the same network setups on the same input were among those that scored the highest. The fact that the threshold independent AUC-scores are affected more by the less successful input combinations than the threshold dependent accuracy is could be due to the unbalanced distributions of *pass* and *discard* in the datasets.

4.2.2 The scaled Fourier intensities as input

Next, the scaled Fourier intensities were used as input. They were used both together with all the four QC-variables as well as alone without giving the networks any further information. First, the same, rather small network configurations as before were used. However, since the number of input variables was much larger now, overtraining was observed during the trial runs, and, therefore, we employed L2-regularisation and some coarse early stopping in an attempt to avoid this. When we only used the Fourier data, we, in addition to the rather small network, also used larger networks. For those, we used one to five hidden layers with as many nodes per hidden layer as the number of inputs, which in this case was 221. The results are displayed in Figure 10 for dataset A.

The first detail to notice in Figure 10 is that none of the networks outperform the best ones that took only QC-variables as input. The highest accuracy reached here was 0.924 and the highest AUC was 0.949, both for networks with one layer, two hidden nodes and both QC-variables and Fourier data as input. It seems that the additional information from the Fourier data obscures some of the information

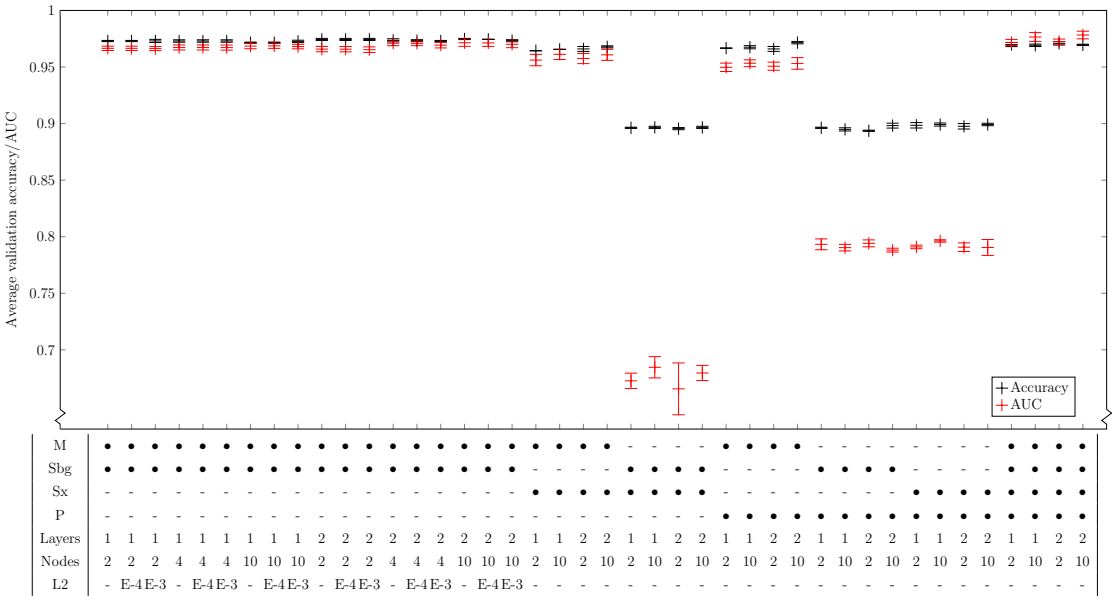


Figure 8: The performance of a selection of different network setups for the binary classification problem for dataset A with QC-variables as input. The vertical axis is the accuracy or AUC-score. The table denotes the network architecture, which inputs were trained on, and the strength of the L2-regularisation.

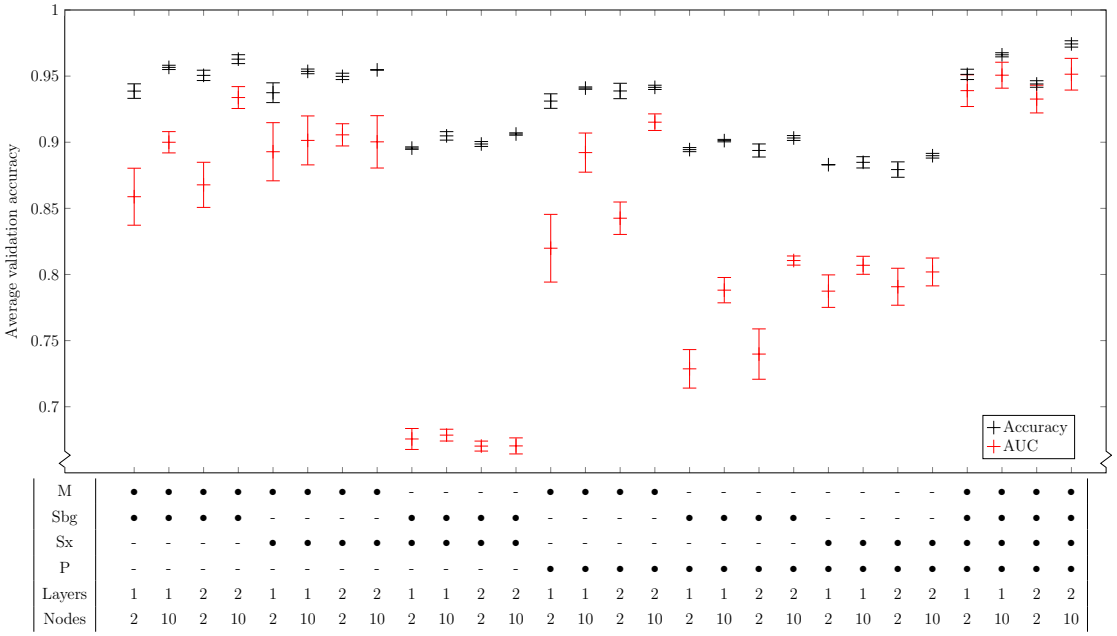
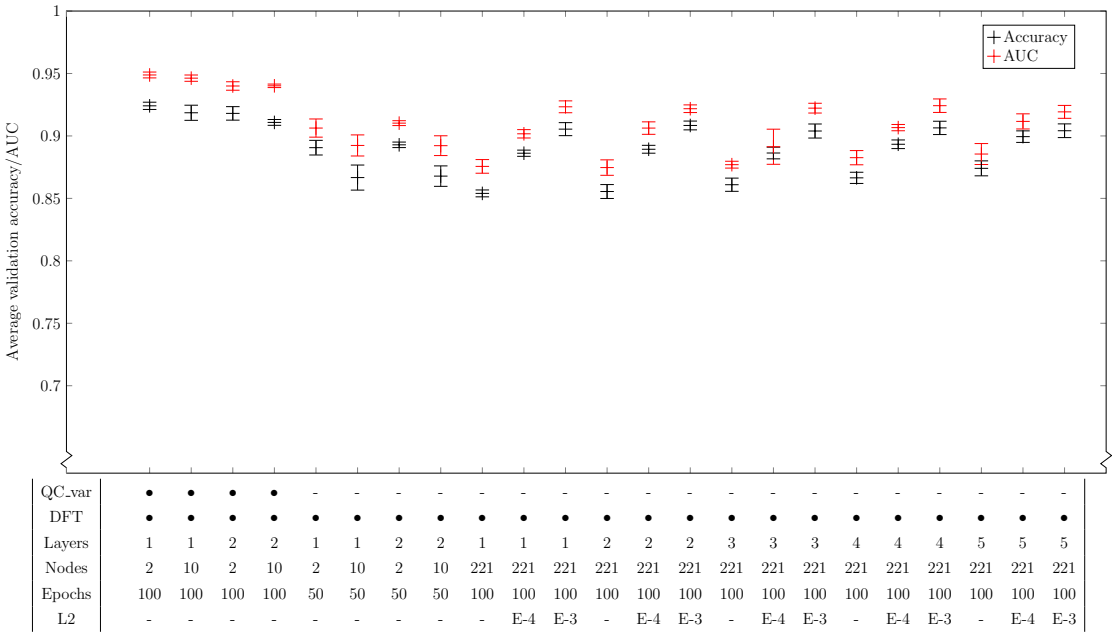


Figure 9: The performance of a selection of different network setups for the binary classification problem for dataset B with QC-variables as input. The vertical axis is the accuracy or AUC-score. The table denotes the network architecture and which inputs were trained on.



QC_var	•	•	•	•	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DFT	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
Layers	1	1	2	2	1	1	2	2	1	1	1	2	2	2	3	3	3	4	4	4	5	5	5	
Nodes	2	10	2	10	2	10	2	10	221	221	221	221	221	221	221	221	221	221	221	221	221	221	221	
Epochs	100	100	100	100	50	50	50	50	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	
L2	-	-	-	-	-	-	-	-	-	E-4	E-3	-	E-4	E-3	-	E-4	E-3	-	E-4	E-3	-	E-4	E-3	

Figure 10: The performance of a selection of different network setups for the binary classification problem for dataset A with Fourier data and QC-variables as input. The vertical axis is the accuracy or AUC-score. The table denotes the network architecture, which inputs were trained on, the number of epochs of training, and the strength of the L2-regularisation.

contained in the QC-variables. However, it is interesting that some networks still perform rather well (best accuracy 0.908 and best AUC 0.923) when only using the Fourier data, which means that this input also seems to contain information about the quality.

4.2.3 DFT reduction and CNN

In order to reduce the fluctuations in the Fourier data, as well as the number of input variables, thus, decreasing the sizes of the networks, we averaged together the scaled Fourier intensities in 3×3-bins as described in Section 3.3. To explore other binning options, we applied convolutional neural networks (CNNs).

A CNN is a special kind of network which has been used and developed since the 1980s, but has had some major breakthroughs in the last decade in the field of computer vision [22]. These networks have been used to, with remarkable accuracy, recognise and classify objects in images. In short, the different layers of the networks extract different features, by letting a kernel pass over the image performing a convolution operation, until all the information have been boiled down to classifiable signal.

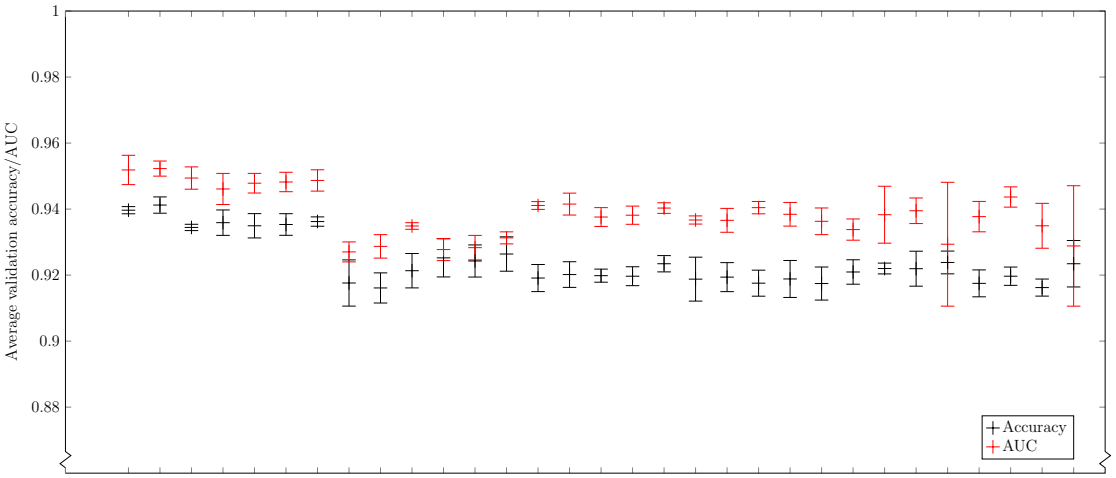
As with the ANNs, we used CNNs with simple architectures. We had one to

three layers with quadratic kernels. For a specific network, the kernels always had the same sizes. The sizes we used were 2×2, 3×3 and 5×5, and we had either one, eight or thirty-two channels, i.e. the number of kernels per layer. The kernels' stride lengths were, in both directions, either one, two or three. Zero-padding was used so that the Fourier space kept its dimensions through every layer of processing. One interesting comparison to make is that our manual 3×3-binning is basically a CNN with one layer, one channel, a 3×3 kernel with stride length 3 and with all the weights being one-ninth (except for the fact that we adapt the weights if the bin contains symmetry reduced components).

We did not use the commonly used max-pooling operation [23] in our CNNs. The main point with max-pooling is to downsample the input, but as an additional effect, it also increases the translational invariance of the feature detection. When applying CNNs to images, it should not matter if e.g. a dark spot is found in the upper left corner of the image or in the centre, it is still a dark spot that is found. However, it makes a crucial difference where the dark spot is found in the Fourier space. Thus, we want to avoid translational invariance and, therefore, max-pooling was not used.

The CNNs were applied to a partly symmetry reduced Fourier space: all of the $(0, k_2)$ and $(k_1, 0)$ components were kept. For a selection of the network setups that performed best, early stopping was employed in order to improve the performance additionally. The results for the manual 3×3 binning and the CNNs for dataset A can be seen in Figure 11. This plot illustrates two notable points. The first point is that the networks with the manually binned intensity components as inputs, as well as most of the CNNs, perform better than the networks with all the Fourier data. The best performing network, the one with one layer and ten hidden nodes, scored an accuracy of 0.941 and AUC of 0.952. The second point is that the manual binning performed better than the CNNs. Quite remarkably though, is that one of the best performing CNN was the one with one layer, 3×3 kernel and a stride length of three, i.e. the one that up to the summing weights is equivalent to the manual binning. Further, the binning of Fourier data helped the networks to extract the information, but the manual binning was better than the CNN. Hence, we did not use CNN any further.

With the different possibilities for usage of the Fourier data explored, a summarised version of the results for dataset B for the same network setups as used for dataset A, are presented in figure Figure 12. Here, the networks which trained on the full Fourier data performed rather badly compared to what they did on dataset A. The accuracy of the best setup is roughly the same as the best for dataset A, but all the AUC-scores are much worse. Once again, the networks that took the binned intensities as input performed better than the ones with all the components. However, they are still not at the same level as when only the QC-variables are



ANN/CNN	A	A	A	A	A	A	A	A	A	A	A	A	A	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C
Layers	1	1	2	2	1	1	1	3	3	3	5	5	5	1	1	1	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
Nodes	2	10	2	10	31	31	31	31	31	31	31	31	31	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Sizes	-	-	-	-	-	-	-	-	-	-	-	-	-	2	3	3	3	3	5	5	5	2	3	3	3	5	2	2	2	3	5
Channels	-	-	-	-	-	-	-	-	-	-	-	-	-	32	8	8	32	32	8	8	32	8	8	8	32	8	8	32	32	32	8
Strides	-	-	-	-	-	-	-	-	-	-	-	-	-	2	1	3	1	3	2	3	1	1	1	2	3	1	1	1	2	3	1
Epochs	50	50	50	50	100	100	100	100	100	100	100	100	100	10	10	20	5	10	10	15	5	10	10	20	10	10	10	5	10	10	10
L2	-	-	-	-	-	E-4	E-3	-	E-4	E-3	-	E-4	E-3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 11: The performance of a selection of different network setups for the binary classification problem for dataset A. The ANNs took the manually binned Fourier data is input while the CNNs took all of the Fourier data as input. The vertical axis is the accuracy or AUC-score. The table denotes whether an ANN or a CNN was used, the network architecture, the number of epochs of training, and the strength of the L2-regularisation.

used as in Figure 9.

4.2.4 Predictions

In order to investigate the transferability of networks between datasets, we trained a network on a complete dataset (i.e. no part of the dataset was excluded for validation) and used it to classify the arrays from the other dataset. We chose to do this for the network setup that seemed to perform best overall, which is the one with two hidden layers with ten nodes per layer when it trains on the background mean and standard deviation. The number of arrays that had been correctly and falsely classified as *pass* (TP and FP), and correspondingly those that had been correctly and falsely classified as *discard* (TD and FD) are presented as confusion matrices in Table 1.

With these measures, it is possible to calculate the accuracy, sensitivity and specificity, which are also presented in Table 1 [20]. The network which trained on A gave about the same accuracy when predicting on B as it got during validation, and the same is also true for the other way around. For both cases, the sensitivity

was very high while the specificity was not as good. This means that we can be almost completely certain that an array classified as *discard* actually is that while, on the other hand, we cannot be as certain that the arrays classified as *pass* actually are that.

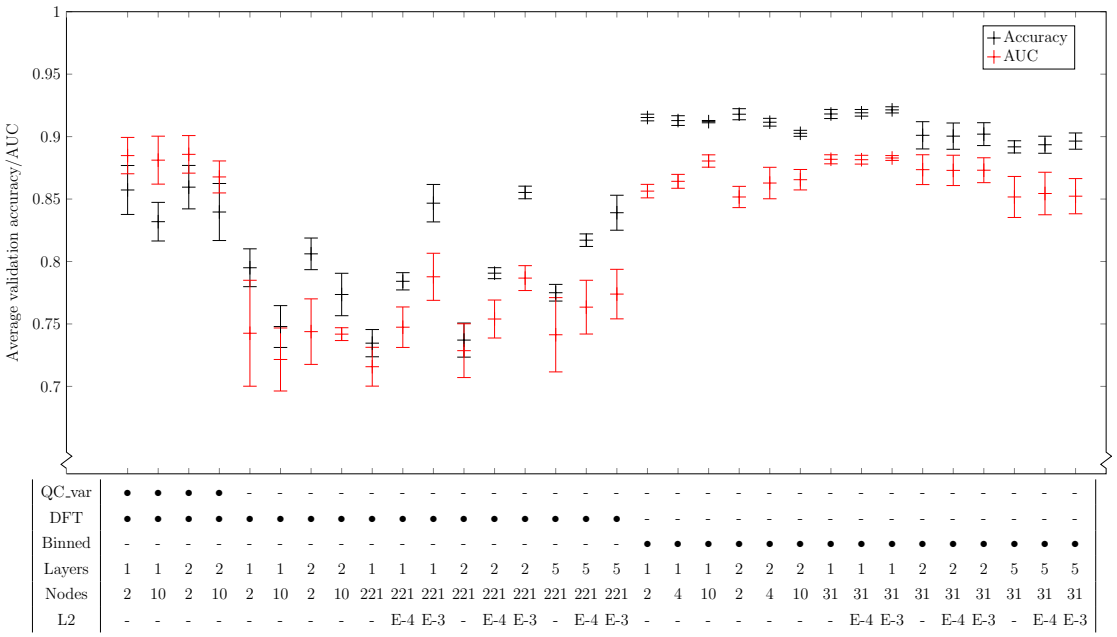


Figure 12: The performance of a selection of different network setups for the binary classification problem for dataset B with Fourier data (complete or binned) and QC-variables as input. The vertical axis is the accuracy or AUC-score. The table denotes the network architecture, which inputs were trained on, and the strength of the L2-regularisation.

Table 1: Example of confusion matrices for binary predictions of a network with two hidden layers with ten nodes per layers that trained on background mean and standard deviation. The left matrix is for when the network trained on dataset A and predicted on B and the central matrix is for the vice versa. The table to the right is a compilation of the accuracies, sensitivities and specificities obtained for the predictions.

A on B

	T	F
P	464	18
D	43	8

B on A

	T	F
P	800	22
D	82	5

Dataset	Accuracy	Sensitivity	Specificity
A on B	0.95	0.98	0.70
B on A	0.97	0.99	0.79

4.3 Ternary networks: Classification into *pass*, *note* and *discard*

4.3.1 Training and validation

For the problem of classifying the data into all three classes, the same network setups were used as for the binary classification task, but, with the adaptation to three classes as described in Section 4.1. However, since AUC is only valid for binary classification, it was not used. The results for dataset A are shown in Figures 13 and 14 and the results for B are showed in Figures 23 and 24 in Appendix A. As expected, the performance is in general much lower than for the binary classification problem. As for the result in Section 4.2, it is quite clear that the networks trained on the QC-variables best reproduce the visual quality control, and that the background mean is the most important variable. The accuracy now varies between 0.827 (the best network setup for dataset A) and 0.580 (the worst network setup for dataset B), where most of the networks land somewhere between 0.7 and 0.75.

This range is approximately the proportion of microarrays labelled with *pass*. Hence, there is a risk that the networks learned to always "majority guess" on *pass* and that this would possibly yield the best accuracy. In order to make sure that this was not the case, we also measured the performance with a modified accuracy

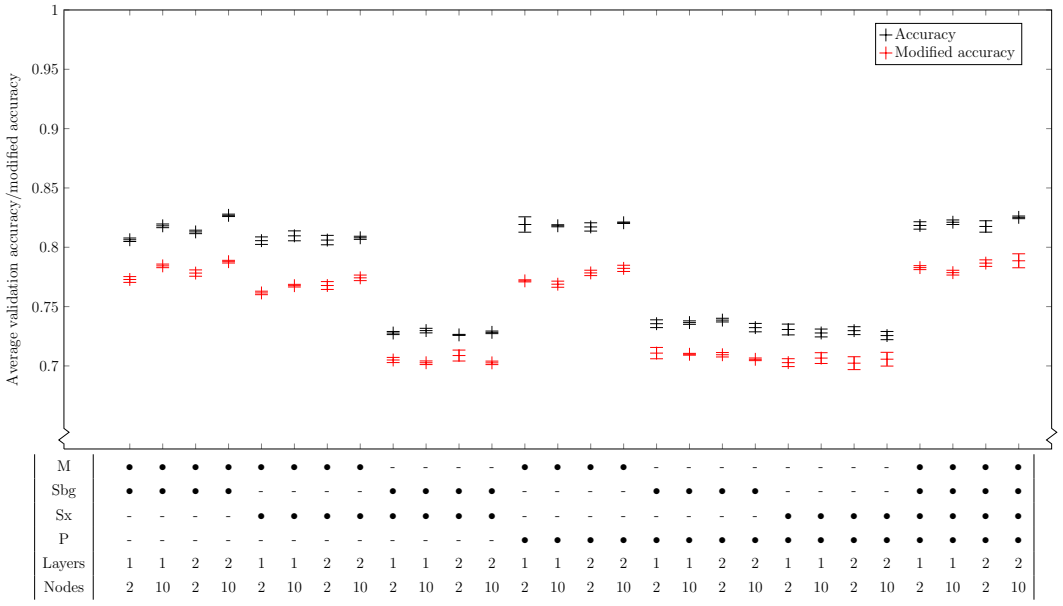


Figure 13: The performance of a selection of different network setups for the ternary classification problem for dataset A with QC-variables as input. The vertical axis is the accuracy or modified accuracy. The table denotes the network architecture and which inputs were trained on.

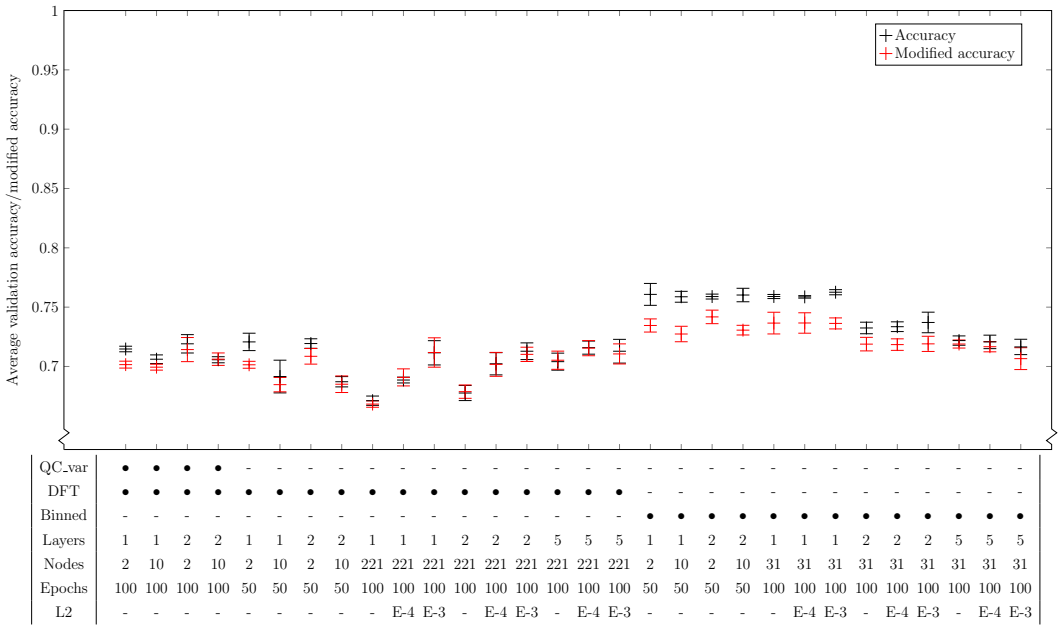


Figure 14: The performance of a selection of different network setups for the ternary classification problem for dataset A with Fourier data and QC-variables as input. The vertical axis is the accuracy or modified accuracy. The table denotes the network architecture, which inputs were trained on, the number of epochs of training, and the strength of the L2-regularisation.

which takes the category sizes into account. If the signal from each of the three output nodes are denoted y_1, y_2 and y_3, where their sum is 1 (as is the property of the softmax output function), then the standard accuracy classifies an array into class 1 if $y_1 > \max\{y_2, y_3\}$ and correspondingly for the other classes. The modified accuracy that we used weights the signal with the sizes of each class so that an array, instead, is classified into class 1 if $y_1 > \max\left\{\frac{N_1}{N_2}y_2, \frac{N_1}{N_3}y_3\right\}$, and similarly for the other two classes. Since the modified accuracy yielded roughly similar results as the standard accuracy, we can be fairly certain that majority guessing was not the issue.

4.3.2 Predictions

The predicting procedure described for the binary networks in Section 4.2.4 was repeated for the ternary versions of the network. Just as for the binary predictions, we used the network with two hidden layers with ten hidden nodes each which trained on the background mean and standard deviation. The results are presented as two 3×3 confusion matrices in Table 2. From the confusion matrices, we can calculate that for the case "A on B", the accuracy is 0.77, and the case "B on A" has accuracy = 0.81. Just as for the binary networks, we observe that the

Table 2: Example of confusion matrices for ternary predictions of a network with two hidden layers with ten nodes per layers that trained on background mean and standard deviation. The left matrix is for when the network trained on dataset A and predicted on B and the right matrix is for the vice versa.

A on B

Predicted class	Actual class			
		P	N	D
	P	**450**	72	13
	N	21	**41**	18
	D	1	27	**30**

B on A

Predicted class	Actual class			
		P	N	D
	P	**784**	106	11
	N	19	**93**	68
	D	2	12	**25**

accuracies are the same when predicting and cross-validating on a dataset, i.e. the networks perform equally well on both datasets independently of which dataset it trained on.

5 Array deviation

In the ideal situation, the scans of two different microarrays containing the same sample should look exactly the same. However, this is not always the case which is why quality control is needed. Based on this concept, we propose an alternative measure of quality which we call *array deviation*. It is a quantity which measures how much one array differs from another array containing the same sample.

5.1 Definition of array deviation

Each array contains the same set of 63 or 40 different reporters.[1] By taking the mean of the replicates of each reporter's expression on an array, one single value per reporter is obtained. This can be thought of as the arrays are located at a specific point in the 63- or 40-dimensional reporter space. All array replicates of the same sample can, hence, be placed in the reporter space according to the same principle. Once all replicates are placed in the reporter space, their mean point can be calculated. Then, it is straight forward to calculate the Euclidean distance between each array and the sample's mean point. The distance divided by the number of reporters, i.e. the mean distance each reporter differs between an array and the sample mean, is what we define as the array deviation, see Figure 15 for a

[1]Datasets A and B contain 63 different reporters and dataset C contains 40.

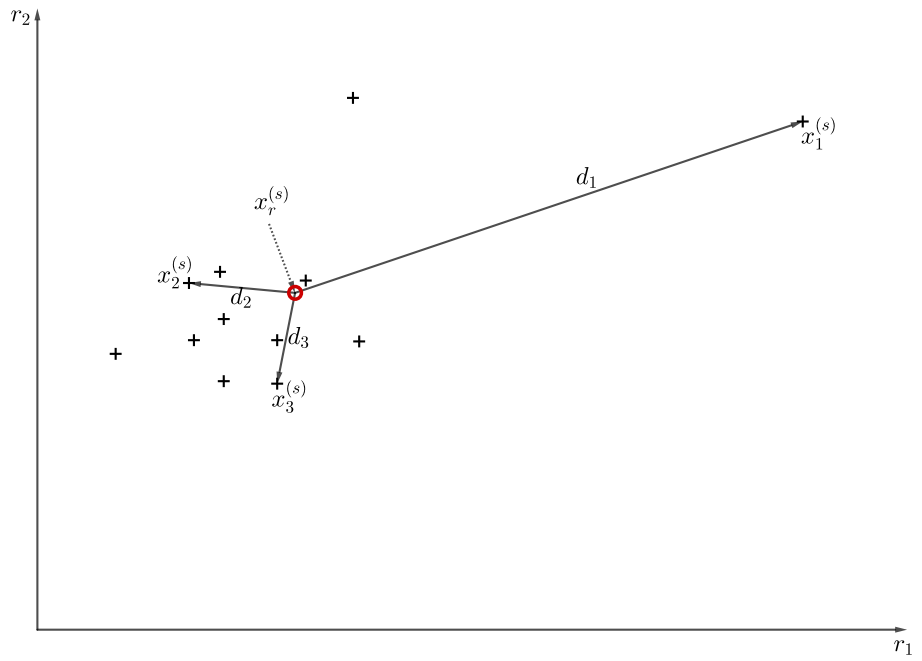


Figure 15: A simplified illustration of the array deviation. Each marker represents an array's position in the (in this case) 2-dimensional reporter space. The red circle is the mean position of all the arrays, and the distance d_i between the mean point and array i is the array deviation.

simplified illustration. With this measure of quality, a high value means low quality and vice versa.

Put in mathematical terms, first, the mean expression for each reporter on an array, i.e. the r-coordinate for each array's position in reporter space, is calculated by

$$x_{a,r} = \frac{1}{N_r} \sum_{i=1}^{N_r} x_{a,r,i}, \tag{5.1}$$

where a as usual denotes the identity of the microarray, r is the type of reporter, N_r is the number of spots with reporter r and i is the spot index. It is understood that the sum is only over the spots i with reporter r. This is exactly the same quantity as in Equation (3.5) but where we reserve the $\langle\cdot\rangle$-notation for the sample mean. Next, we can find the r-coordinate of the mean position of all arrays of sample s in the reporter space:

$$\langle x \rangle_r^{(s)} = \frac{1}{N} \sum_a x_{a,r}^{(s)}, \tag{5.2}$$

where N are the number of different reporters. The superscript (s) explicitly indicates the array's sample and implicitly restricts the sum to only include arrays

with that sample. Finally, the definition of the array deviation d_a is

$$d_a = \sqrt{\frac{1}{N}\sum_{r=1}^{N}\left(x_{a,r}^{(s)} - \langle x \rangle_r^{(s)}\right)^2}. \tag{5.3}$$

In order to be able to say anything from the array deviation, at least three replicates are necessary. If only two exist, it is not possible to distinguish which array that deviates from which. With the demand of at least three replicates of each sample, dataset A is reduced to 318 instead of 1120 arrays and dataset B is reduced to 367 instead of 673. This may, of course, affect the networks possibility to learn since the sizes of the datasets have been reduced quite considerably. However, this is not an issue for dataset C since all its 1485 microarrays are part of samples with more than three replicates.

5.2 Agreement between array deviation and visual QC

To illustrate if the array deviation agrees with the visual QC, each array's array deviation has been plotted in Figure 16 with a division between the visual classifications. This has only been done for A and B since C only includes arrays that passed the quality control. The figure shows that there are no replicates that have been classified as *discard* in dataset A. Hence, dataset A is not really suitable to use as training data: networks trained on A will not learn any information about the array deviation for arrays labelled *discard*. Next, there seems to be no agreement between the array deviation and the visual QC. It is only for the six extreme arrays in dataset B where *discard* and array deviation agree. These are,

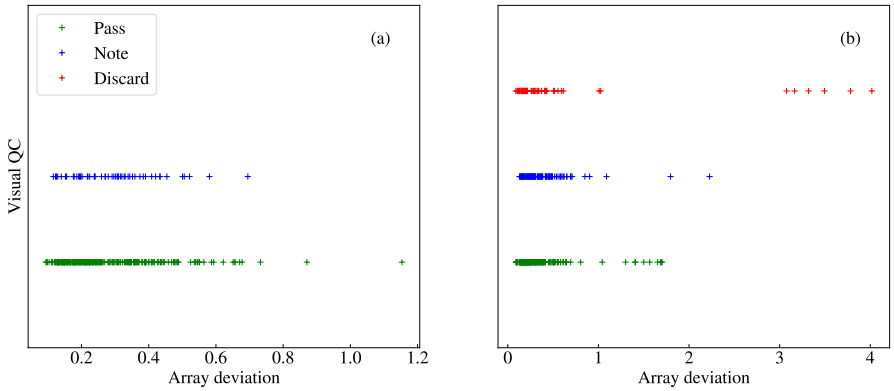


Figure 16: The separation of the visual QC due to the array deviation. Subfigure (a) contains the arrays from dataset A and subfigure (b) the arrays from dataset B.

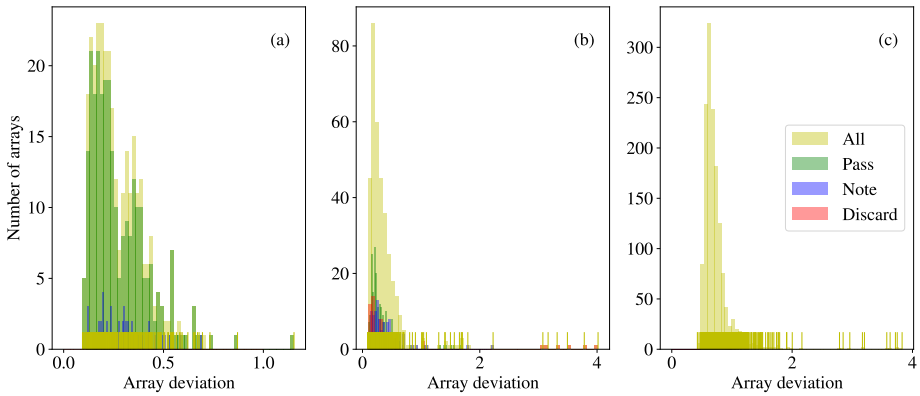


Figure 17: Distribution of the array deviation where the ranges have been divided into 60 bins. Subfigure (a) shows the distribution of dataset A, (b) shows dataset B and (c) shows dataset C.

in fact, the same six arrays that were pointed out earlier in Section 3.3, suggesting that they have not only a very low background intensity, but also a low signal, resulting in a large array deviation.

From Figure 16, it is clear that the different visual classifications overlap at the same values of the array deviation; however, it is not really possible to see the distribution of the microarrays' array deviations. This is, instead, plotted in Figure 17. From this figure, it appears that most of the arrays have an array deviation between 0.1 and 0.6 for datasets A and B. For dataset C, the bulk of the microarrays have an array deviation between 0.5 and 1.0. It is not strange that the range is different for C since the distribution of the array deviation may depend on the design and purpose of the original study.

Finally, before attacking this problem with networks, we will examine if there is any correlation between any of the QC-variables and the array deviation by considering the scatter plots in Figure 18 and Figures 25 and 26 in Appendix A. There is little to no clear trend in any of them, except Figure 26 (a). In Figure 25, the six arrays from dataset B which we have seen deviating from the rest of the set in earlier figures, continue to do so here; although, since it is only those six, it cannot really be called a clear trend throughout the dataset.

5.3 Learning the array deviation with ANN

If ANNs can be trained to recognise which features of an array yield a specific array deviation, it would be possible to predict the array deviation of an array which is not part of a replicate, and, hence, give it a quality score anyway which is independent of the subjectivity of manual classification.

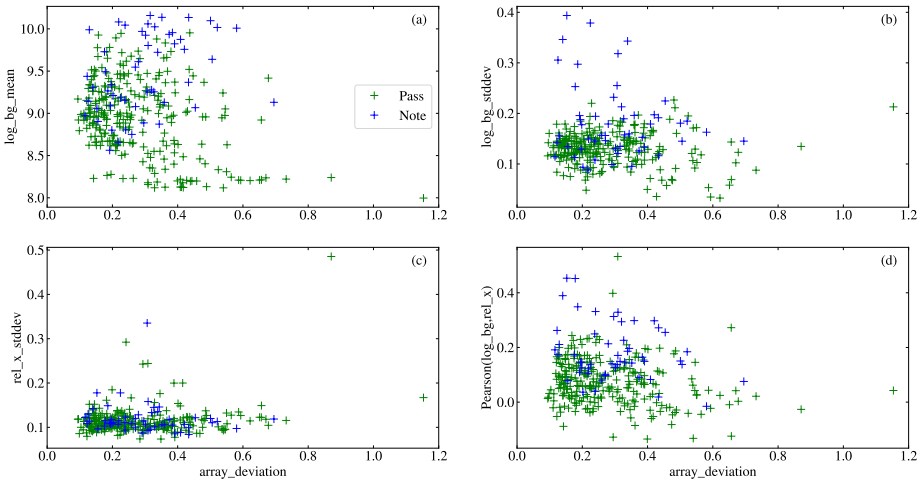


Figure 18: Relations between each of the four QC-variables and array deviation for dataset A.

Prediction of array deviation is a regression problem, and we used networks with one output node and linear output activation function. We used mean square error (MSE) as the loss function, and used mean absolute error (MAE) as a score. As an additional score, we used the MSE divided by the variance of the array deviation of the dataset. If this measure is equal to 1, it means that there is no useful information, while if it is smaller than 1, it suggests that useful predictions can be made, as motivated in the supplements to [24]. Otherwise, we used the same network setups as for the classification problems. If nothing else is stated, the networks were trained for 1 000 epochs. The results for the three datasets for networks trained on the QC-variables can be seen in Figure 19 and Figures 27 to 31 in Appendix A.

The striking difference between the networks that have been trained on the QC-variables compared to those that have been trained on some version of the Fourier data is that for the QC-variables, the MSE/var is mostly well below 1, while for the Fourier data, it is more or less always above 1. Once again, we see that the network architecture with two hidden layers and ten hidden nodes that trained on the mean and the standard deviation of the background is among the best networks, although it is not *the* best for datasets A and B. The best performing network setup for dataset A reaches MAE = 0.095 which can be compared to the median array deviation of the set which is 0.24, i.e. the MAE is 0.40 of the median. For dataset B, the best network setup yields MAE = 0.17 and the median array deviation is 0.28 which gives the ratio 0.61. The corresponding numbers for dataset C are: best MAE = 0.12, median of array deviation = 0.67 with the ratio

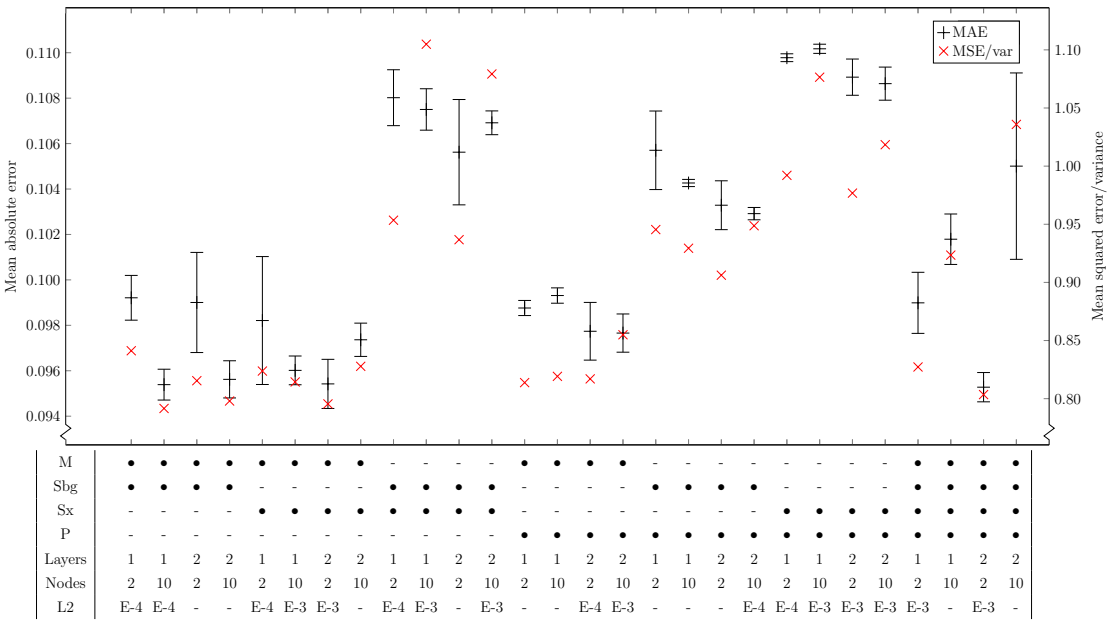


Figure 19: The performance of a selection of different network setups for the regression problem for dataset A with QC-variables as input. The left vertical axis is MAE while the right vertical axis is the MSE/var. The table denotes the network architecture, which input were trained on, and the strength of the L2-regularisation.

0.18. The best MSE/var-values are 0.79, 0.32 and 0.31 for datasets A, B and C respectively. This means that the networks clearly could find features in the input which said something about the array deviation, even though the ratios between the MAE and median array deviation would ideally be lower, especially for datasets A and B. The fact that the best networks actually could extract useful data can be confirmed by Figure 20. This figure shows scatter plots between the arrays' array deviation and the predicted values as predicted by networks with two layers and ten hidden nodes trained on the background mean and standard deviation. Ideally, all the microarrays would lie on a straight line with a 45° slope (the dashed grey line in the plots), which they, of course, do not. However, it was possible to make a linear fit where the null hypothesis of a horizontal line could be discarded with a p-value $< 10^{-10}$.

Finally, we made predictions with networks on the different datasets. We trained a network on a full dataset and then used it to make predictions on the other two. The MAE between the predicted values and the actual array deviation was used as a score. The result of this can be seen in Table 3. The prediction MAEs are notably worse than the validation MAEs. The network trained on A performs roughly as well on B as a network trained on B performs on A. Networks trained on C performs roughly equally well on A as on B, while networks trained

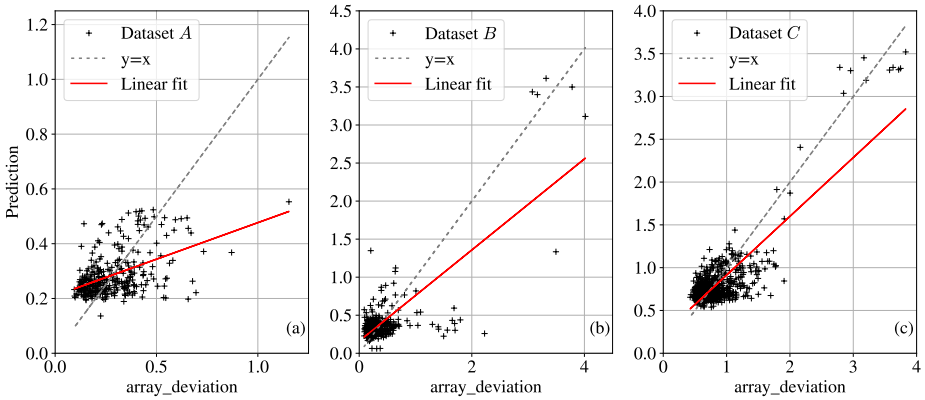


Figure 20: The predicted array deviations and the actual array deviations for each array as predicted by networks with two hidden layers with ten nodes per layer which trained on background mean and standard deviation. The linear fits all have p-value $< 10^{-10}$.

Table 3: The MAE between predictions and actual array deviations as predicted when a network with two hidden layers and ten nodes per layer trains on dataset X and predicts on dataset Y.

		Trained on		
		A	B	C
Predicted on	A	-	0.269	0.462
	B	0.201	-	0.473
	C	0.467	0.390	-

on A and B respectively, performs similarly on C. The problematics with worse prediction scores than validation scores probably derive from the difference in the distributions of array deviation in Figure 17.

6 Concluding remarks

The quality control of microarray data is an important intermediate step in the endeavour of diagnosing complex diseases from a standard blood sample using antibody microarrays. In this master project, we have investigated the possibilities of automating the quality control by using ANN so that it can become reproducible. We constructed variables to feed into the networks and found that the variable of most importance to the quality is the background mean of an array. All the best performing network setups took this variable as input, and in many cases, it was helped by the standard deviation of the background. Furthermore, it sufficed with

very simple networks to reach high accuracy: it was enough with one hidden layer with two nodes in order to reach an accuracy above 0.97.

When we added additional information in the form of the scaled coefficients of the two-dimensional Fourier transform of the background, the performance did not improve; in fact, it decreased. However, the fact that networks that trained only on some version of the Fourier data could score up to 0.94 in accuracy, without the help of any other input variables, shows that there exists information about the quality to be extracted from the Fourier data. It might be possible to improve the performance with more advanced networks. A future investigation could include to do a more detailed study of the scaled Fourier intensities, and to find out how the information contained in the different components transfers between datasets with differently shaped microarrays and Fourier spaces.

In order to be free from the subjectivity embedded in the labels from the visual quality control, we introduced a new measure of array quality: the array deviation. This is a measure which describes how much an array deviates from other arrays containing the same sample, where low array deviation corresponds to high quality and vice versa. This measure did, however, not correlate with the visual classification, except that it seemed to be able to capture low-quality arrays with very low background intensity. We found that it was possible for networks to extract useful information about array deviation from the input data. Once again, the background mean was the most important variable.

A crucial aspect for application on future datasets is if a network that trained on one dataset could perform well on another. We found for the classification problems, both the binary and ternary, that the networks predicted and validated on a dataset equally well, no matter which of datasets A and B it had trained on. This suggests that a trained network might be transferable between different datasets, although this would, of course, be needed to test on more datasets before anything can be concluded with certainty. When it comes to how the trained networks performed on the other datasets for the problem of predicting array deviation, the networks did not transfer as well. They could still extract some features, but could not make predictions with as high accuracy as during validation. This is not particularly strange since the range and distribution of the array deviation differed between the datasets. Furthermore, there could be differences in the experimental protocols that were used to generate the datasets. It is not reasonable to expect a neural network to be able to transfer between datasets generated with different protocols, not even for a future, finalised automatic quality control.

In this project, we have seen that it is possible to reproduce the binary visual classification using simple networks. We can also say with high certainty that the visual classification is mostly based on the mean of the background with some help of its standard deviation. Furthermore, the newly introduced array deviation

seems to be a reasonable measure of quality; however, for now, we have not been able to make predictions with high enough precision to be satisfying, but this might be possible with more complex networks. Additionally, it would need to be investigated if the array deviation actually is a good measure of quality and how one, in that case, finds a suitable threshold.

A Supplementary data

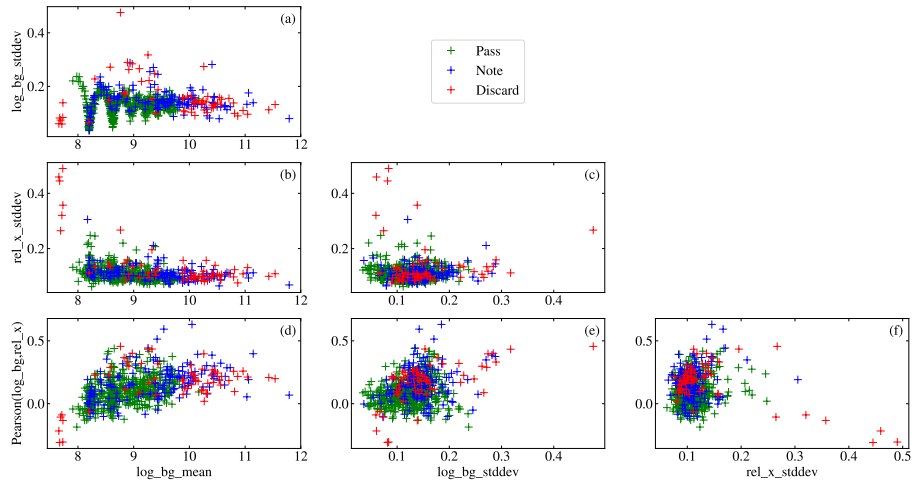


Figure 21: Two out of the four QC-variables are plotted against each other in each subfigure for dataset *B*. Each marker denotes the position for each array and the colours indicate the array's visual classification.

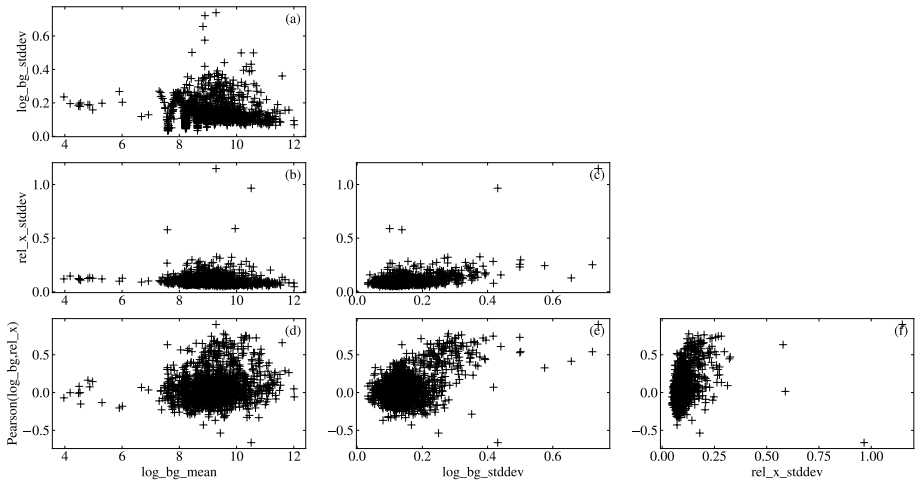


Figure 22: Two out of the four QC-variables are plotted against each other in each subfigure for dataset *C*.

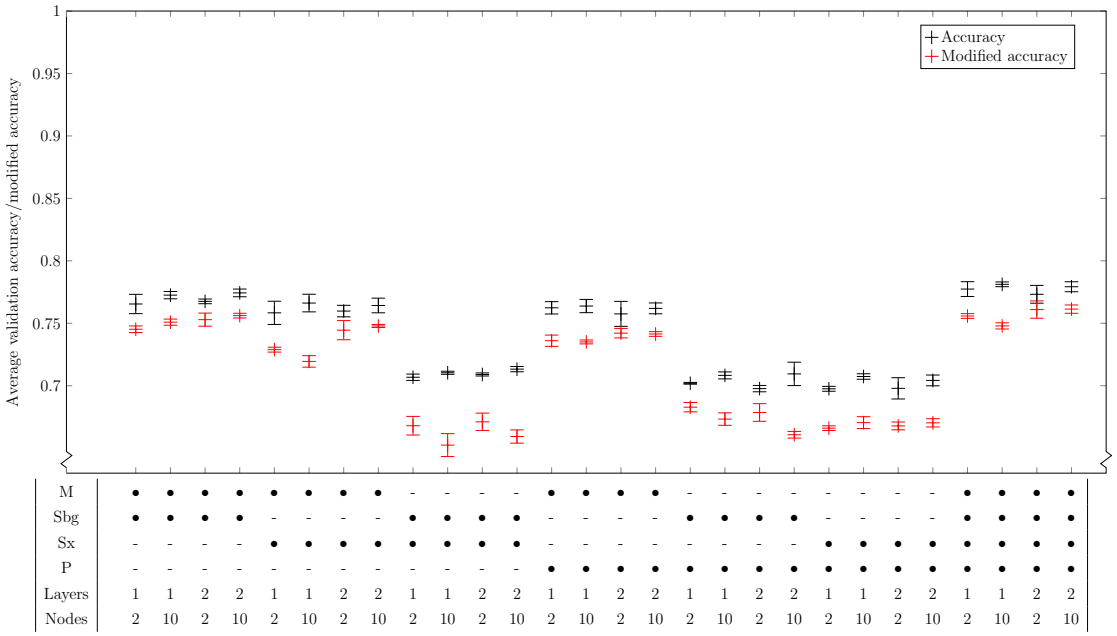


Figure 23: The performance of a selection of different network setups for the ternary classification problem for dataset B with QC-variables as input. The vertical axis is the accuracy or modified accuracy. The table denotes the network architecture, which inputs were trained on, and the strength of the L2-regularisation.

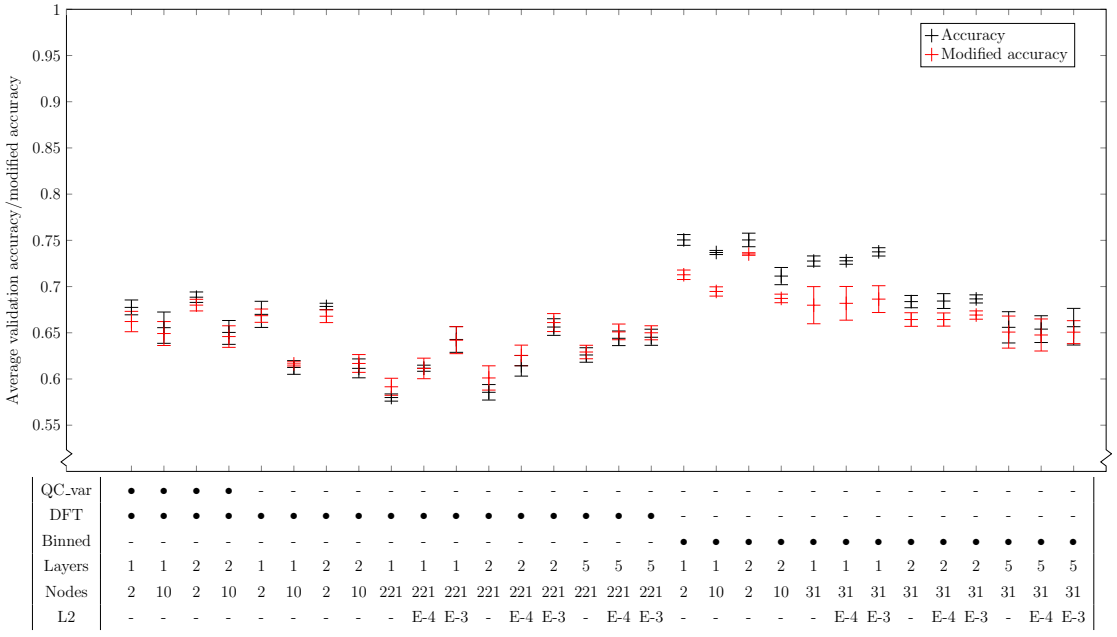


Figure 24: The performance of a selection of different network setups for the ternary classification problem for dataset B with Fourier data (complete or binned) and QC-variables as input. The vertical axis is the accuracy or modified accuracy. The table denotes the network architecture, which inputs were trained on, and the strength of the L2-regularisation.

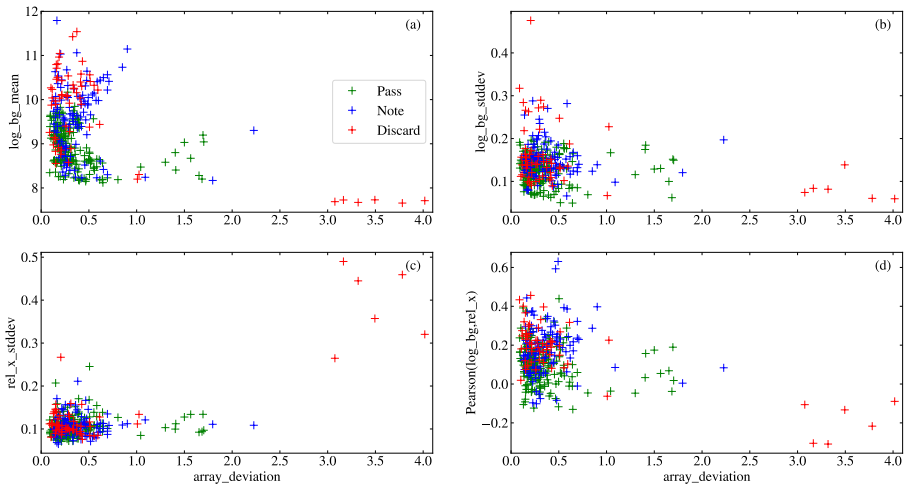


Figure 25: Relations between each of the four QC-variables and array deviation for dataset B.

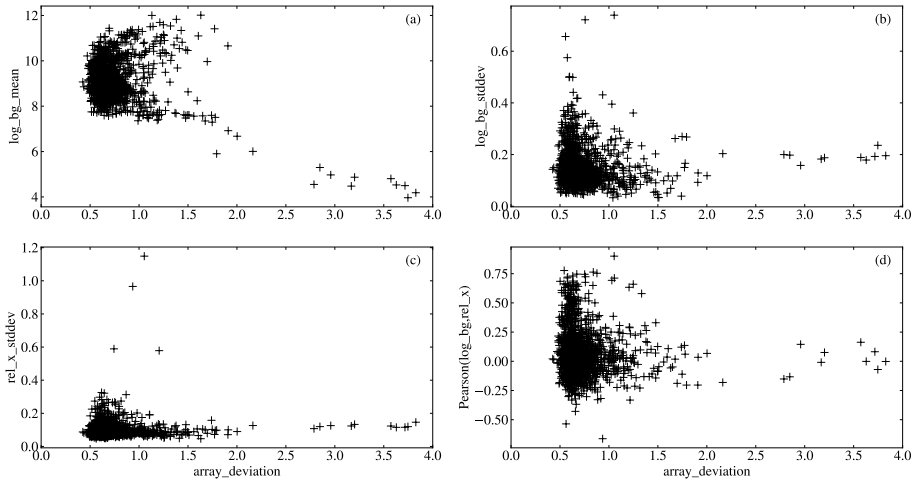


Figure 26: Relations between each of the four QC-variables and array deviation for dataset C.

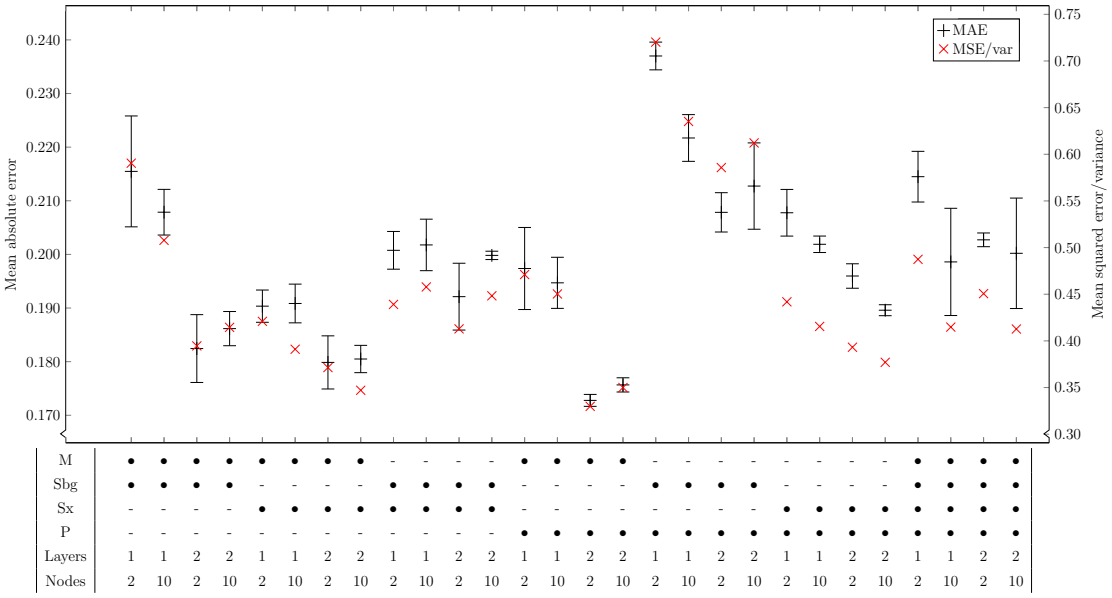


Figure 27: The performance of a selection of different network setups for the regression problem for dataset *B* with QC-variables as input. The left vertical axis is MAE while the right vertical axis is the MSE/var. The table denotes the network architecture and which input were trained on.

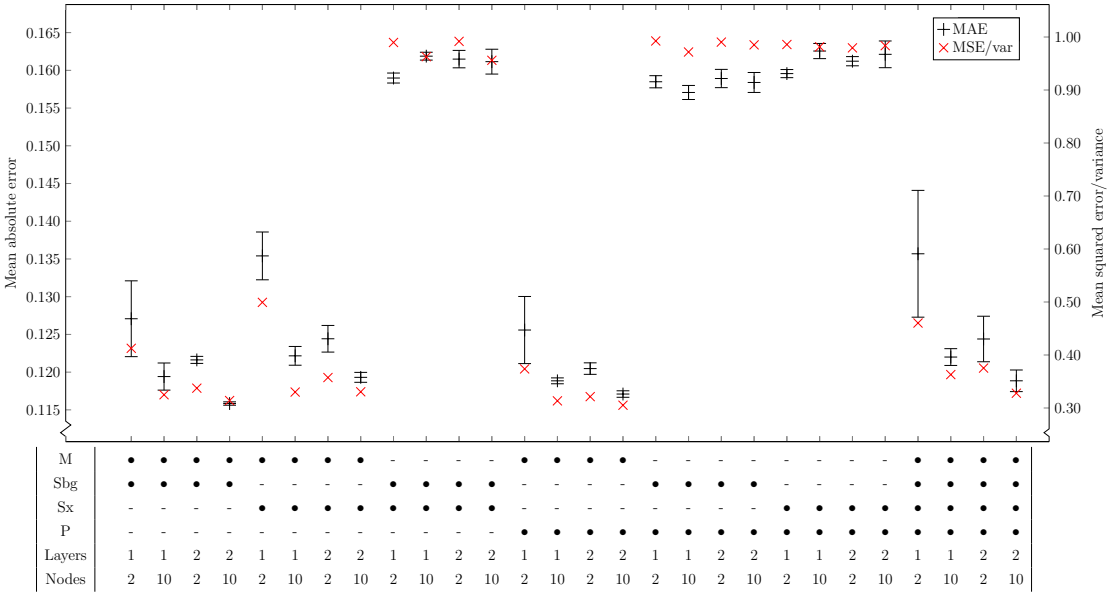


Figure 28: The performance of a selection of different network setups for the regression problem for dataset *C* with QC-variables as input. The left vertical axis is MAE while the right vertical axis is the MSE/var. The table denotes the network architecture and which input were trained on.

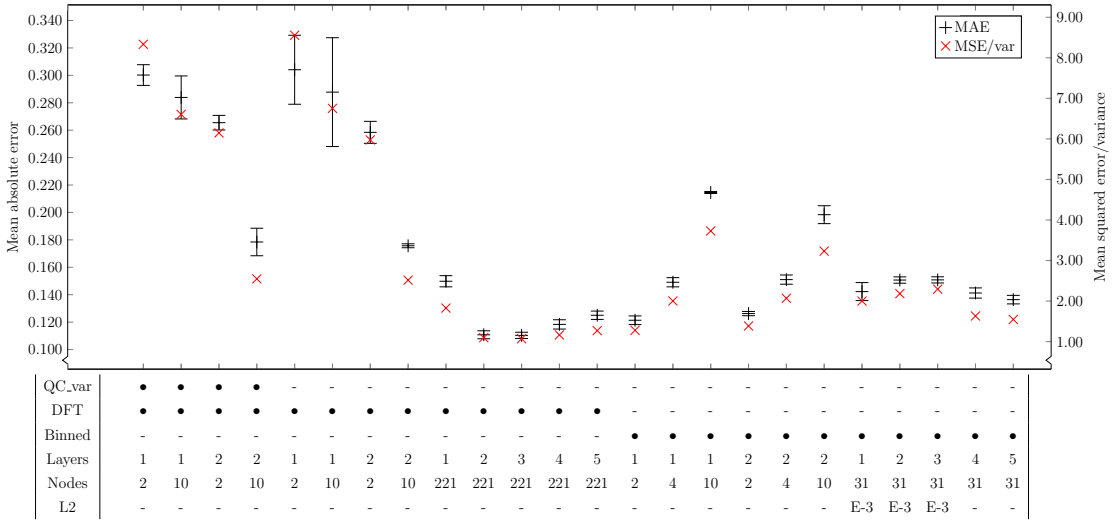


Figure 29: The performance of a selection of different network setups for the regression problem for dataset A with Fourier data (complete or binned) and QC-variables as input. The left vertical axis is MAE while the right vertical axis is the MSE/var. The table denotes the network architecture, which inputs were trained on, and the strength of the L2-regularisation.

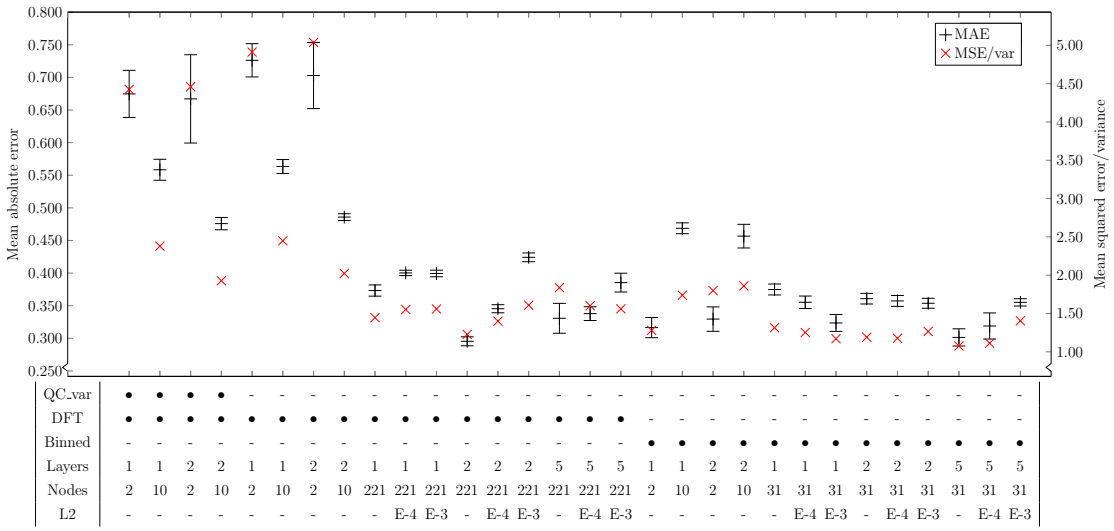
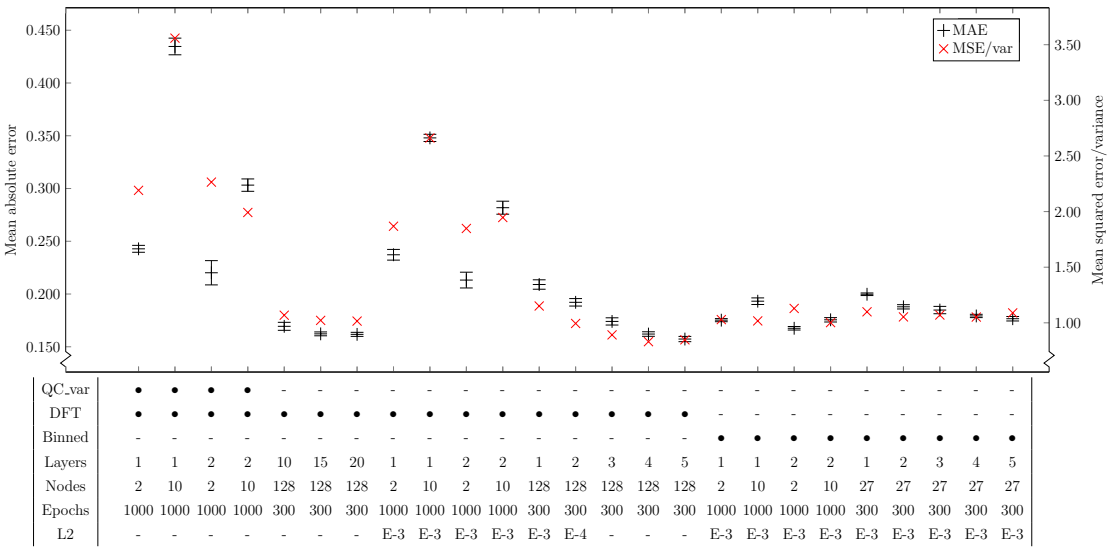


Figure 30: The performance of a selection of different network setups for the regression problem for dataset B with Fourier data (complete or binned) and QC-variables as input. The left vertical axis is MAE while the right vertical axis is the MSE/var. The table denotes the network architecture, which inputs were trained on, and the strength of the L2-regularisation.



QC_var	•	•	•	•	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DFT	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	-	-	-	-	-	-	-	-
Binned	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	•	•	•	•	•	•	•	•
Layers	1	1	2	2	10	15	20	1	1	2	2	1	2	3	4	5	1	1	2	2	1	2	3	4	5
Nodes	2	10	2	10	128	128	128	2	10	2	10	128	128	128	128	128	2	10	2	10	27	27	27	27	27
Epochs	1000	1000	1000	1000	300	300	300	1000	1000	1000	1000	300	300	300	300	300	1000	1000	1000	1000	300	300	300	300	300
L2	-	-	-	-	-	-	-	E-3	E-3	E-3	E-3	E-3	E-4	-	-	-	E-3	E-3	E-3	E-3	E-3	E-3	E-3	E-3	E-3

Figure 31: The performance of a selection of different network setups for the regression problem for dataset C with Fourier data (complete or binned) and QC-variables as input. The left vertical axis is MAE while the right vertical axis is the MSE/var. The table denotes the network architecture, which inputs were trained on, the number of epochs of training, and the strength of the L2-regularisation.

References

[1] S. C. Sealfon and T. T. Chu, *RNA and DNA Microarrays*, pp. 3–34. Totowa, NJ: Humana Press, 2011.

[2] J. B. Rampal, *Microarrays: Volume I: Synthesis Methods*, vol. 1. Springer Science & Business Media, 2007.

[3] J. B. Rampal, *Microarrays: Volume II: Applications and Data Analysis*, vol. 2. Springer Science & Business Media, 2007.

[4] C. Wingren, *Antibody-Based Proteomics*, pp. 163–179. Cham: Springer International Publishing, 2016.

[5] C. Wingren, A. Sandström, R. Segersvärd, A. Carlsson, R. Andersson, M. Löhr, and C. A. K. Borrebaeck, "Identification of serum biomarker signatures associated with pancreatic cancer," *Cancer Research*, vol. 72, no. 10, pp. 2481–2490, 2012.

[6] C. A. Borrebaeck and C. Wingren, "Design of high-density antibody microarrays for disease proteomics: Key technological issues," *Journal of Proteomics*, vol. 72, no. 6, pp. 928 – 935, 2009. Special Section: Clinical Proteomics in Oncology.

[7] C. Wingren, J. Ingvarsson, L. Dexlin, D. Szul, and C. A. K. Borrebaeck, "Design of recombinant antibody microarrays for complex proteome analysis: Choice of sample labeling-tag and solid support," *PROTEOMICS*, vol. 7, no. 17, pp. 3055–3065, 2007.

[8] C. Wingren and C. A. Borrebaeck, "Antibody microarray analysis of directly labelled complex proteomes," *Current Opinion in Biotechnology*, vol. 19, no. 1, pp. 55 – 61, 2008. Analytical biotechnology.

[9] D. Amaratunga and J. Cabrera, *Exploration and Analysis of DNA Microarray and Protein Array Data*. John Wiley & Sons, Inc., ch. 4, 2004.

[10] J. Benesty, J. Chen, Y. Huang, and I. Cohen, *Pearson Correlation Coefficient*, pp. 1–4. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[11] W. Burger and M. J. Burge, *The Discrete Fourier Transform in 2D*, pp. 1–26. London: Springer London, 2009.

[12] K. F. Riley, M. P. Hobson, and S. J. Bence, *Mathematical methods for physics and engineering: a comprehensive guide*. Cambridge university press, third ed., 2006.

[13] Documentation for seaborn.boxenplot. `https://seaborn.pydata.org/generated/seaborn.boxenplot.html`, 2019-05-16.

[14] H. Hofmann, K. Kafadar, and H. Wickham, "Letter-value plots: Boxplots for large data," tech. rep., had.co.nz, 2011.

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

[16] Keras: The Python Deep Learning library, version 2.2.4. `https://keras.io/`, 2019-05-16.

[17] TensorFlow, version 1.13.1. `https://www.tensorflow.org/`, 2019-05-16.

[18] M. A. Nielsemn, *Neural Networks and Deep Learning*. Determination Press, 2015. `http://neuralnetworksanddeeplearning.com/`.

[19] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-prints*, p. arXiv:1412.6980, Dec 2014.

[20] T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861 – 874, 2006. ROC Analysis in Pattern Recognition.

[21] F. Chollet, *Deep Learning with Python*. Manning Publications Co., 2018.

[22] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Computation*, vol. 29, no. 9, pp. 2352–2449, 2017. PMID: 28599112.

[23] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *Artificial Neural Networks – ICANN 2010* (K. Diamantaras, W. Duch, and L. S. Iliadis, eds.), (Berlin, Heidelberg), pp. 92–101, Springer Berlin Heidelberg, 2010.

[24] S. K. Gruvberger-Saal, P. Edén, M. Ringnér, B. Baldetorp, G. Chebil, Å. Borg, M. Fernö, C. Peterson, and P. S. Meltzer, "Predicting continuous values of prognostic markers in breast cancer from microarray gene expression profiles," *Molecular Cancer Therapeutics*, vol. 3, no. 2, pp. 161–168, 2004.