Julian Krappinger, BSc

Classification of cancers of unknown primary with deep learning using gene expression data.

Master's Thesis

to achieve the university degree of
Diplom-Ingenieur
Master's degree programme: Biomedical Engineering

submitted to

Graz University of Technology

Supervisor

Dr. Gerhard Thallinger

Institute of Neural Engineering

Graz, May 2020

AFFIDAVIT[1]

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Graz, _____ _____

Date Signature

[1] Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Abstract

About 5% of all metastatic cancer cases fall into the category of cancer of unknown primary site (CUP). As the origin of cancer plays a critical role in choosing the optimal therapy for patients with metastatic cancer, there is an urgent need for a firm histologic diagnosis. Several studies propose gene expression profiling (GEP) for facilitating the identification of the site of origin of the primary tumour. Tothill *et al.* (Cancer Res. 2005. 65(10):4031–40) developed support vector machine (SVM) classifiers by using cDNA microarray data. In a previous project, the classifier was reconstructed in R using the publicly available microarray data. Comparable accuracy was achieved with the microarray data, but using the classifier on 21 Medical University Graz RNA-seq samples did not yield any meaningful results.
The overall aim of this project was to investigate the classification possibilities of RNA-seq CUP samples, firstly test how using raw microarray intensity data influences classification accuracy of SVM and neural network models and whether it improves the classification accuracy of RNA-seq data. The highest validation accuracy was achieved with SVMs trained on log_2-transformed intensity data (84.6%) and the lowest with raw intesity values (69.2%)
Further three different types of neural networks were realized and trained on TCGA RNA-seq data (feed forward neural network [FFNN], deep learning FFNN and a convolutional neural network [CNN]). While adjusting network shapes and parameter two approaches for classification have shown promising results. A single hidden layer FFNN and a CNN yielded 97.4% and 98.5% classification accuracy respectively, while deep learning DLNN yielded 79.7% classification accuracy.
The realized neural network and SVM models were then used to classify 21 CUP samples provided by the MU Graz. This led to varying results depending on the methods used (highest accuracy: 52.38%).

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

Cancer that develops in a single origin may spread in the body (for example gastro-abdominal cancer may spread to the lung) representing a secondary cancer. If the source, or primary cancer cannot be determined, this type of cancer is called cancer of unknown primary (CUP).
The classification of CUP plays a critical role in choosing the optimal therapy for patients with metastatic cancer [5, 7–9]. Tothill *et al.* developed a support vector machine (SVM) classifier using gene expression data measured on cDNA microarrays, comprising primary and metastatic tumour samples. They demonstrated 89% classification accuracy in classifying 13 carcinoma [5]. In a previous project, the classifier was reconstructed in R [10] using their publicly available microarray data. Comparable accuracy was achieved with microarray data, but using the classifier on newly generated RNA-seq samples did not yield any meaningful results.
The phenotypic manifestation of a gene through transcription and translation is called gene expression [11]. During gene expression analysis procedures mRNA and other gene products are measured in order to determine the structure of a gene and how strong it is expressed. A wide range of different experimental techniques and tools of varying complexity are used for this purpose (e.g. microarrays and RNA-seq).
Tothill *et al.* uses microarray data for their classification. These microarrays have microscopic spots attached on their surface in order to measure gene expressions or genotypes through hybridization of cRNA and cDNA [12]. A sample of a particular condition is measured against a reference sample under normal conditions. mRNA is extracted from cells and reverse transcribed into cDNA; sample of interest and reference sample are labeled with different flourescent dyes. The labeled cDNA is

hybridized to the micoarray. The amount of cDNA hybridized is proportional to the initial number of RNA molecules present for the gene in both samples. The spots are then excited by a laser and an amount of fluorescence is emitted that corresponds to the amount of bound nucleic acid. The intensity and the ratio of the intensity in comparison with the background and the reference is contained in the raw micrarray data [13].
RNA-seq data determines the nucleotide sequence of RNA based on next generation sequencing methods (NGS) [14]. It is in comparison less error prone than microarray processes (noise, skewedness, partial hybridizations) and offers the ability to detect novel transcripts with a higher specificity and sensitivity [15–17]. RNA-seq is mostly used for transcriptome profiling, SNP identification, RNA editing and differential gene expression analysis [18]. Total RNA is extracted and either mRNA enriched or rRNA depleted [19]. Complemetary DNA (cDNA) is synthesized from single stranded RNA through the use of the enzyme reverse transcriptase (DNA is more stable for further sequencing). cDNA is fragmented and a sequencing library is generated which is then further analyzed through different sequencing methods (Illumina, PacBio, ...) in order to determine gene expression profiles [20–22]. With tools like Bowtie [23] these short sequences are mapped to a reference genome or transcriptome and read counts for annotated genes are derived from these mappings [24].

Support Vector Machines (SVM) and Neural Networks (NN) can be used to classify a given set of samples, like the results of a sequencing pipeline or micoarray analysis. SVMs are usually supervised learning algorithms analyzing data using classification and regression analysis [25]. A set of training samples is given with marked categories (classes). This training set is used to build a model that assigns new samples to the different classes. The examples represent points in space, mapped so that the examples of separate classes are divided by a hyperplane with a maximum distance to any of the samples. New samples are then mapped into the same space and predicted to belong to a class based on which side of the hyperplane they fall (Figure 1.1). There are one vs. one and one vs. all approaches.

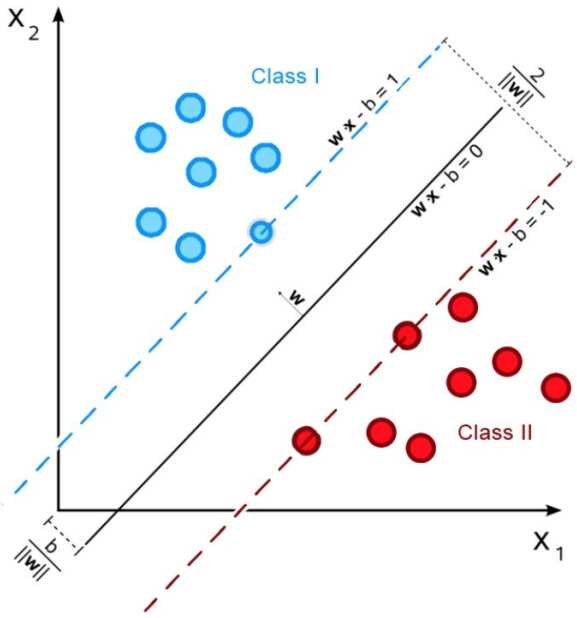


Figure 1.1: A diagram representing a SVM classification process between two classes. Hyperplanes separating two example classes, created and taken from [1]

NN are able to recognize patterns in input data (numerical values contained in vectors) and assign labels or cluster data into classes. In order to classify data, a neural network needs labeled input data. This process is called supervised learning. Further the generated model can be used to predict future events [26].

A NN represents a stacked structure of neural network layers. Each layer consists of neurons representing a series of numerical inputs adjusted in significance by weights and evaluated by an activation function. The activation function determines if the signal reaches the next node or not (true or false). In principle, each output signal is input for every other node in the next layer of the neural network. The first layer of a neural network is a so called input layer, followed by a number of hidden layers and by the output layer at the end. The number of nodes in the output layer is equal to the number of classes. Each layer trains a more sensitive set of features based on the previous layer's output. Commonly, if there are more than three hidden layers, a NN is classified as deep learning neural network (DLNN) [26].
The created model is trained on a set of training data. Each input is weighted and evaluated per action function. The whole training data are separated into equal sized batches, using a single batch each epoch. Through this process the model

learns the properties and features of each training data batch and uses them to determine the class of the sample. The idea is to increase the accuracy of the model with each batch of training data until no further significant increase is recorded or no more training data is available. Accuracy and loss functions are calculated to represent numerical values of how well a model fits the training data. Through optimization the error in predicting new samples can be reduced.
There are different types of neural networks, the ones used in this thesis are most commonly known as:

- Feed Forward Neural Network (FFNN).
- Convolutional Neural Networks (CNN).

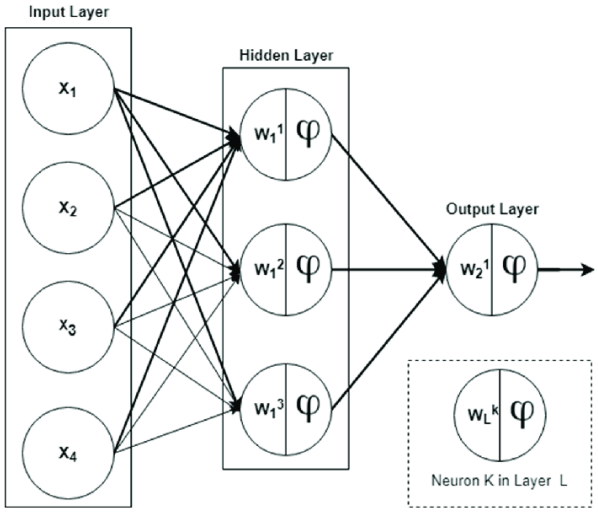


Figure 1.2: Principal structure of a feed forward neural network. Each node is connected to each node of the next layer adjusted by weights and activation function. Created and taken from [2]

FFNNs (Figure 1.2) want to minimize the error as fast as possible. After the sample input, the network simply guesses the significance of each input and adjusts weights and used features for further batches via the calculated loss function (compared to the actual classes). This process is then repeated until the increase of accuracy comes to a halt or no more input data is available.
The CNN consists of convolutional layers, a max pooling layers, a flatten layer, a set of hidden layers and ends with a output layer (Figure 1.3):

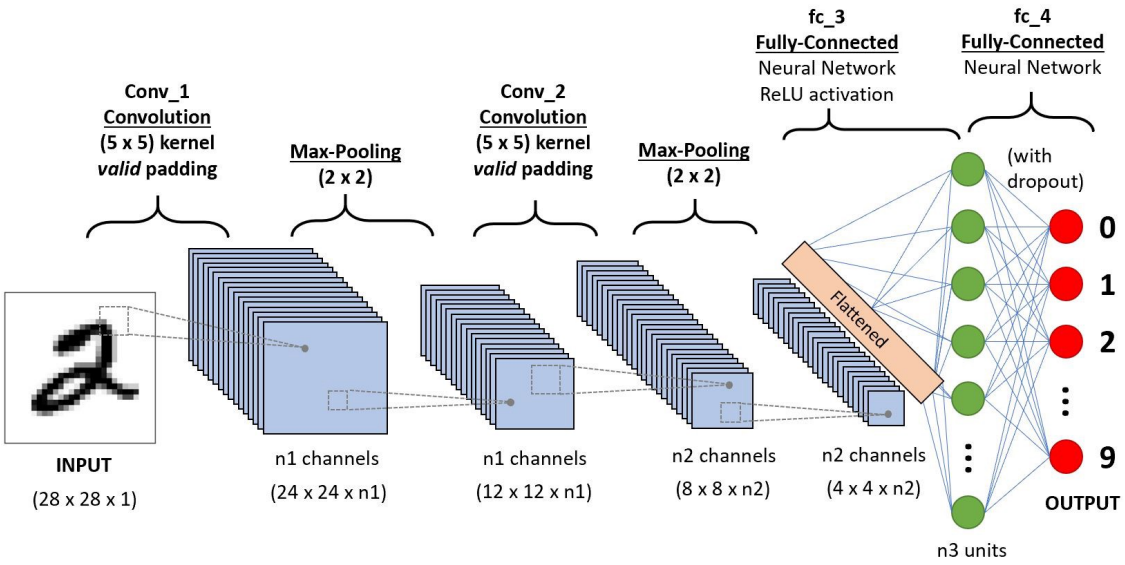


Figure 1.3: Principal structure of a convolutional neural networks. A convolutional layer extracts features through a filter, which is followed by max pooling, a flatten layer and connected to a conventional NN structure. Created and taken from [3]

CNNs are often used for image classification, clustering by similarity and object recognition. These types of networks process images as 4D tensors. A filter is convoluted with the input signal. Depending on the filter type different image features are recognized. For example, in terms of a picture's horizontal or vertical lines can be recognized for border detection. The CNN creates a map of the place each feature occurs. Each filter is only able to recognize a single feature, therefore a number of different filters are applied to the input data. The input is seen as a volume - the height, width and depth of the picture. The pixel values behind each color channel are the raw network input [27].
All this is done in square patches of pixels in order to accommodate filter kernels. These filter kernels represent masks that extract different features from the image (blurring, sharpening, edge detection, ...) through convolution [28].
The dot product of the image and filter is created. Depending of the matching pixel pattern of the mask and picture, the dot product can either be high or low representing the chosen features. The filter starts in the upper left corner, moves to the right and from the top to the bottom of the input data. The dot products are written in an activation map, a separate one for each channel and each used filter kernel. Through max pooling/downsampling the most important features of the activation map are selected and at the end of the convolutional part of the network

flattend to accommodate a conventional feed forward network structure [27].

For further studies, data is always very important. The Cancer Genome Atlas TCGA contains data of collected and characterized high quality tumor and matched normal samples from over 11,000 patients, molecularly characterized over 20,000 primary cancers and matched normal samples spanning 33 cancer types (comprising genomic, epigenomic, transcriptomic, and proteomic data) [29]. The project with their respective samples are available at GDC Genomic Data Commons.

The MU Graz provided 21 different RNA-seq CUP samples, which are classified by SVMs and NNs trained using Tothill microarray and TCGA RNA-seq gene expression data, to determine a model with high classification accuracy through comparison.

1.2 Aim of the Thesis

The overall aim of this thesis is to apply support vector machines and neural networks trained on microarray and RNA-seq data to classify CUP samples of the Medical University Graz.
Specific aims of this thesis are:

- Create SVM and NN models from raw microarray intensity data and assess their accuracy.
- Create SVM and NN models from TCGA RNA-seq data and assess their accuracy.
- Compare accuracy and misclassifications based on the machine learning approach, data used for training and NN structure.
- Apply the above models for the classification of the samples of Medical University Graz.
- Investigate the influence of unbalanced group size on classification performance.

2 Methods

2.1 Datasets

The chosen datasets include:

- Tothill *et al.* microarray data.
- 12 classes TCGA RNA-seq data.
- 19 classes TCGA RNA-seq data.
- 33 classes TCGA RNA-seq data.
- MU RNA-seq data.

Microarray data provided by Tothill *et al.* serves as a reference for further classification with NN trained on RNA-seq data. Further, each RNA-seq dataset was also classified by an SVM and compared in terms of accuracy with NN results. In the course of optimizing and testing different neural network models with the aim of creating a classifier using all 33 available classes on TCGA, subsets of 12 (common with Tothill *et al.*) classes and 19 classes (of single primary) were created. These smaller datasets were used to test (i) and optimize neural networks, (ii) save time while doing so, (iii) provide a clean differentiation between classes (iv) and a clear reference to Tothill *et al.* data (v). The gained knowledge of the neural network models and the datasets was then applied to create a classifier with the complete number of classes. This model was then used to classify three different versions of CUP RNA-seq data provided by the MU Graz. Each dataset name consists out on data type, data source, number of classes and an indicator which operations were performed on this data set (Table 2.1).

Table 2.1: Overview of data sets used to train and validate SVM and NN models and each data set of MU Graz CUP data for subsequent classification.

Dataset name	*Comment*
MA_Tothill_13_processed	Processed MA data set provided by Tothill *et al.* without CUP samples.
MA_Tothill_13_mapped	Processed MA data set provided by Tothill *et al.* with features common to MU samples.
MA_Tothill_13_raw	Raw MA intensity data without CUP samples provided by Tothill *et al.* .
MA_Tothill_13_raw_log2	log_2 transformed background corrected MA intensitiy data without CUP samples.
MA_Tothill_13_raw_range	Range transformed background corrected MA intensitiy data without CUP samples.
MA_Tothill_13_raw_bg	Background corrected MA intensity data without CUP samples.
RNAseq_TCGA_33	RNA-seq data downloaded from TCGA, 33 classes.
RNAseq_TCGA_33_mapped	RNA-seq data downloaded from TCGA, 33 classes with features common to MU samples.
RNAseq_TCGA_33_balanced	RNA-seq data downloaded from TCGA, 33 balanced classes with features common to MU samples.
RNAseq_TCGA_19	RNA-seq data downloaded from TCGA, 19 classes of unique primaries.
RNAseq_TCGA_19_mapped	RNA-seq data downloaded from TCGA, 19 classes of unique primaries with common MU features.
RNAseq_TCGA_12	RNA-seq data downloaded from TCGA, 12 classes common with Tothill *et al.*.
RNAseq_MU_RPM	MU Graz RNA-seq data, 21 samples of CUP represented as RPM.
RNAseq_MU_voom	MU Graz RNA-seq data, 21 samples of CUP represented as voom transformed RPM.
RNAseq_MU_ratio	MU Graz RNA-seq data, 21 samples of CUP represented as ratio transformed RPM. *et al.*.

2.1.1 Microarray data (MA_Tothill_13)

The data used in Tothill *et al.* [5] includes microarray data of 229 primary and metastatic tumors representing 14 tumor types and further 13 samples of unknown primary. These were classified using a 13 class model (Table 2.2). This data was reduced to a dataset without CUP samples for training purposes. The respective CUP samples were later classified by the generated model and compared with their true labels.

Table 2.2: Breakdown of Tothill *et al.* in tumor type, number of samples and features.

tumor type	#samples	#features
breast	34	10239
colon	23	10239
gastro-intestinal	15	10239
melanoma	11	10239
mesothelioma	8	10239
ovary	50	10239
pancreas	9	10239
prostate	8	10239
renal	13	10239
testicular	3	10239
SCCo	14	10239
uterus	9	10239
lung	32	10239

This gene expression data is available at Array Express (Accession Number: e-mexp-113) [30]. It was preprocessed to represent the intensity of the according genes for each sample. The preprocessed data from Array Express represent the log_2 fold-change of the test channel divided by fluorescent intensity from control channel - *MA_Tothill_13_processed*.
According to Tothill *et al.*(Appendix GenePix File Formats), the raw data files were generated using the GenePix Version 4.1 file format [31]. These files contain among others raw intensity data of the sample, both for spot foreground and background. The sample is labeled using a Cyanine-5 Fluorophor (red 650-670 nm peak) while the reference uses a Cyanine-3 Flurophor (orange 550-570 nm peak):

- Foreground Cyanine-5 Flurophor sample (F635).
- Background Cyanine-5 Flurophor sample (B635).
- Foreground Cyanine-3 Flurophor reference (F532).
- Background Cyanine-3 Flurophor reference (B532).

The processed data files from Tothill *et al.* [5] were used as reference for the adapted script and the base reference for following neural networks. It contains 242 samples of CUP and non CUP patients with each 10239 features after preprocessing. Further, notable Patient IDs and scan names are present as well as gene reporter names, reporter types, GenBank, RefSeq- and UniGene names are provided. For the adapted SVM script the raw intensity data of the sample (F635 and B635) were used. For each sample a separate microarray intensity data file was present. These raw data files

were combined into a set of 242 samples with 10,944 features for further processing (Table 2.3) - *MA_Tothill_13_raw*.
Both data sets were later divided into 13 CUP and 229 non-CUP samples.

Table 2.3: Structure of imported microarray data from Tothill *et al.*[5] providing raw intensity values for each sample.

sample_id	**reporter_name**	**reporter_name**	**reporter_name**	**reporter_name**	**...**
sample_1	raw_data	raw_data	raw_data	raw_data	...
sample_2	raw_data	raw_data	raw_data	raw_data	...
sample_3	raw_data	raw_data	raw_data	raw_data	...
sample_4	raw_data	raw_data	raw_data	raw_data	...
...	...	...	...	...	...

Tothill *et al.* raw intensity data files further contain a broad variety of calculated intensity values. The composition of such files is available in the GenePix Version 4.1 documentation [31].
As described in the Results section, the column headers of the downloaded raw data do not represent the respective data values. In order to determine the columns needed to import raw intensities, ratio of means (RoM) value was used. The GenePix Version 4.1 file format defines the RoM as:

$$RoM = \frac{mean\_F635 - median\_B635}{mean\_F532 - median\_B532} \tag{2.1}$$

In order to determine different possible classification results the following raw and transformed intensity values were used:

- raw intensities

$$I = mean\_F635 \tag{2.2}$$

- background corrected intensities

$$I = mean\_F635 - median\_B635 \tag{2.3}$$

- range transformed background corrected intensities with the array index i

$$a_i = mean\_F635 - median\_B635 \tag{2.4}$$

$$I = \frac{a_i - min(a_i)}{max(a_i) - min(a_i)} \tag{2.5}$$

- log_2-transformed intensities

$$I_i = log_2(mean\_F635 - median\_B635) \tag{2.6}$$

For log_2-transformed data the resulting data frame has also been corrected as various measurements are marked as 0 (zero). The resulting $log_2(0)$ transformation is negative infinity and cannot be handled by the classifier. Therefore the data frame was corrected in this case and cells with negative infinity replaced with zero. Another reasonable approach would be adding a very small value (e.g. 1) before each log_2 calculation to counter this behaviour.

2.1.2 TCGA RNA-seq data

RNA-seq data available in TCGA:
TCGA The Cancer Genome Atlas contains data of collected and characterized high quality tumor and matched normal samples from over 11,000 patients [29]. A set of the National Cancer Institute (NCI) supported projects with their respective samples are available at GDC Genomic Data Commons. The NCI provides guidelines for their own and also non-NCI supported programs:

- Clinical information about patients.
- Metadata about samples.
- Histopathology slide images from sample portions.
- Molecular information derived from samples.

RNA-seq TCGA data files available can be divided by project name and cancer type (Table 2.4). TCGA project names are representative of different cancer types and used as class names during classification. A project represents mutated cancer genes and distribution of cases, divided by primary site, program - collecting the data, disease type, data category and experimental strategy.

Table 2.4: Available RNA-seq projects on TCGA divided into project, primary, the number of cases, the number of sequence reads and number of files. Table adapted from [6].

project	primary	cases	files
TARGET-NBL	Nervous System	1127	2809
TCGA-BRCA	Breast	1098	31524
TARGET-AML	Blood	988	2459
TARGET-WT	Kidney	652	1408
TCGA-GBM	Brain	617	11996
TCGA-OV	Ovary	608	15166
TCGA-LUAD	Bronchus and lung	585	17052
TCGA-UCEC	2 Primary Sites	560	16174
TCGA-KIRC	Kidney	537	15091
TCGA-HNSC	13 Primary Sites	528	15289
TCGA-LGG	Brain	516	14728
TCGA-THCA	Thyroid gland	507	14421
TCGA-LUSC	Bronchus and lung	504	15324
TCGA-PRAD	Prostate gland	500	14288
TCGA-SKCM	Skin	470	12725
TCGA-COAD	2 Primary Sites	461	14279
TCGA-STAD	Stomach	443	12846
TCGA-BLCA	Bladder	412	11711
TCGA-LIHC	Liver and intrahepatic bile ducts	377	10815
TCGA-CESC	Cervix uteri	307	8595
TCGA-KIRP	Kidney	291	8507
TCGA-SARC	13 Primary Sites	261	7494
TCGA-LAML	Hematopoietic/reticuloendothelial sys.	200	4434
TCGA-ESCA	2 Primary Sites	185	5271
TCGA-PAAD	Pancreas	185	5307
TCGA-PCPG	7 Primary Sites	179	5035
TCGA-READ	5 Primary Sites	172	4925
TCGA-TGCT	Testis	150	4284
TCGA-THYM	2 Primary Sites	124	3445
TCGA-KICH	Kidney	113	2325
TCGA-ACC	Adrenal gland	92	2547
TCGA-MESO	2 Primary Sites	87	2339
TCGA-UVM	Eye and adnexa	80	2180
TARGET-RT	Kidney	75	381

Table 2.4: (Continued) Available RNA-seq projects on TCGA divided into project, primary, the number of cases, the number of sequence reads and number of files. Table adapted from [6].

project	primary	cases	files
TCGA-DLBC	14 Primary Sites	58	1229
TCGA-UCS	Uterus, NOS	57	1659
TCGA-CHOL	3 Primary Sites	51	1349
		14157	321411

TCGA RNA-seq data were generated as follows: Frozen paraffin slides and scrolls/ribbons are used for RNA sequencing. Frozen tissue samples are submitted to the Biospecimen Core Resource (BCR) for consideration in genomic research projects. The tissue is subdivided into portions as determined by specific project protocols by the Logistics department and each is reviewed independently. DNA/RNA/Protein kits (Qiagen AllPrep) are used to isolate total RNA from small quantities of starting material (depending on the clinical disease case, mostly tumor tissue and blood samples). The flow through from AllPrep DNA column is taken and total RNA is isolated with the *mir*Vana kit from Applied Biosystems (Life Technologies). The AllPrep kit utilizes the RNeasy prep, which excludes small RNAs. An Agilent Bioanalyzer is used to determine RNA quantity and integrity via the RNA nano assay. Melanin associated with RNA isolated from melanoma tumor samples is reduced using the *mir*Vana™ miRNA Isolation Kit [32–43]. No information on library preparation methods was found in the SOPs. The TCGA Molecular Characterization Platforms for RNA-seq sequencing using an Illumina HiSeq® 2000 System or Illumina Genome Analyzer II*x* depending on the research establishment (BC Cancer Agency, Harvard Medical School and the University of North Carolina). Details on the standard operating procedure (SOP) for TCGA data can be found in the Biospecimen Research Database (BRD) on their website [44].
The GDC reference genome and alignment workflow for RNA-seq data states on their website: *Using a two-pass method with STAR [45]. first each group is aligned separately and then merged into one alignment. The first alignment to the reference genome is done to detect splice junctions, the second uses the gained information to increase the alignment quality. This results in a genomic BAM file, an aligned transcriptomic BAM file and an aligned chimeric BAM file. Quality assessment is performed pre-alignment with FASTQC [46] and post-alignment with Picard Tools [47]. Read counts are calculated using HTSeq*

and normalized using the Fragments Per Kilobase of transcript per Million mapped reads (FPKM) and FPKM Upper Quartile (FPKM-UQ) methods [48, 49]. The RNA-seq analysis pipeline is part of the GDC documentation and can be accessed on their website. For the download and further work with samples on TCGA, TCGA-Assembler 2 was used [50]. The respective script can be found in the Appendix (Listing 3).

2.1.3 TCGA RNA-seq data set covering 33 classes (RNAseq_TCGA_33)

This represents 33 projects with a wide range of samples. After downloading with TCGA-Assembler 2 [50] this 33 separate files are compacted to one dataset with 10464 samples and 20502 features representing normalized RNA-seq expression data (Table 2.5).

Table 2.5: Structure of imported TCGA RNA-seq data after downloading and preprocessing with TCGA Assembler 2.

sample_id	**gene_id**	**gene_id**	**gene_id**	**gene_id**	**...**
sample_1	normalized_data	normalized_data	normalized_data	normalized_data	...
sample_2	normalized_data	normalized_data	normalized_data	normalized_data	...
sample_3	normalized_data	normalized_data	normalized_data	normalized_data	...
sample_4	normalized_data	normalized_data	normalized_data	normalized_data	...
...	...	...	...	...	...

During the process of modeling different neural network types, this dataset was separated into smaller subsets in order to test structures and simplify problem solving. The overall aim was to create neural network models covering all 33 TCGA classes using RNA-seq data. However, smaller models, with a lower number of classes using only unique primaries, were used in order to test network structures, options and served as an optimization tool. The following datasets were used in this context:

***RNAseq_TCGA_33* - TCGA RNA-seq subset containing 33 projects:**
As a final result, after the modeling processes and testing of network structures were completed, the best performing network structure was trained and evaluated

with the complete TCGA data set.

***RNAseq_TCGA_19* - TCGA RNA-seq subset containing 19 classes:**
In order to determine the best possible training set for the proposed neural network types only 19 of the original 33 classes from TCGA were used for modeling purposes. From this data set, all cancer classes assigned to multiple primary cancers were eliminated. This ensures that each sample should have only a single correct class. Used project are:

Remaining classes: ACC, BLCA, BRCA, CESC, GBM, KICH, KIRC, KIRP, LAML, LGG, LIHC, LUAD, LUSC, OV, PAAD, PRAD, SKCM, STAD, TGCT, THCA, UCS and UVM.

The projects KIRC, KIRP and KICH as well as LUAD and LUSC were combined as they all stand for renal cell carcinoma and lung cancer respectively.

***RNAseq_TCGA_12* - TCGA RNA-seq subset containing 12 classes:**
To compare the classification accuracy of neural networks using TCGA data with SVMs using Tolhill *et al.* [5] microarray data, a data set containing the same cancer classes was created from the complete TCGA data. Only 12 cancer classes from the original microarray data are present in the TCGA RNA-seq data set:

Tothill *et al.* [5] classes: brea, colo, gast, mela, meso, ovar, panc, pros, rena, test, SCCo, uter, lung

Corresponding TCGA classes: BRCA, CHOL, STAD, SKCM, MESO, OV, PAAD, PRAD, RENA, TGCT, UCS, LUNG

While the projects KIRC, KIRP, KICH as well as LUAD, LUSC and STAD, LIHC were combined to represent to renal cell carcinoma, lung cancer, and gastro-intestinal cancer respectively.

***RNAseq_TCGA_33_balanced* - TCGA RNA-seq subset containing 33 projects and equal class sizes:**
To investigate how balanced data sets and unequal class sizes influence the final

model, using 33 classes, a further data set containing an equal number of samples of each primary was created. Each class was reduced to the smallest occuring class size (45 samples). In order to prevent a bias, participating samples were chosen randomly without replacement.

2.1.4 MU Graz CUP RNA-seq data (MU_data)

An RNA-seq dataset, containing genes of 21 CUP samples were contributed by the Medical Univerity Graz (Karl Kashofer).
The following information about wetlab procedures and transcriptome analysis was obtained from the MU Graz: RNA was extracted from FFPE slides using the Maxwell 16 FFPE RNA extraction system (Promega) according to manufacturers instructions, where the necessary tumor area was identified by a board certified pathologist before microdissection to enrich tumor content. RNA was quantified using Ribogreen fluorescence and a Qbit instrument (both Thermo Fisher Scientific). 100 ng total RNA were used to prepare transcriptome libraries using the Ion Amliseq Human Gene Expression Kit (CatNr.: A26325, Thermo Fisher Scientific) following the manufacturers protocol. Libraries were quantified using the Ion Library Taqman Quantification Kit (CatNr.: 446802, Thermo Fisher Scientific) and sequenced on an Ion Proton System using the 200 Bp workflow of the Ion PI Hi-Q Sequencing Kit and Ion Pi Chip Kit (Thermo Fisher Scientific). Each library was sequenced to approximately 10 million reads. Bioinformatic analysis was done using Ion Torrent Suite 5.10. Briefly, reads were mapped to the human transcriptome reference (hg19_AmpliSeq_Transcriptome_21K.v1, Thermo Fisher Scientific), counted and summarized to RPM (reads per million) values to normalize for library sequencing depth. RPM values for each of the 20812 transcripts annotated in the reference file were then exported to TSV files [51]. Each of 21 CUP samples comprises 20,812 features. Further gene annotations are given by way of gene_id, contig_id, contig_srt, contig_end, region_id and attributes.

Table 2.6: MU Graz samples with corresponding truel labels and respective TCGA RNA-seq labels and Tothill *et al.* labels.

sample	true label	TCGA label	Tothill *et al.* label
K1999-6	colon	COAD/READ	colo
K2170-6	melanom	SKCM	melan
K2953-5	lung	LUSC/LUAD	lung
K4824-4	pancreas	PAAD	panc
K5108-24	ovar	OV	ovar
K5216-6	breast	BRCA	breast
K6266-2	melanom	SKCM	mela
K6267-2	lung	LUSC/LUAD	lung
K6268-2	pancreas	PAAD	panc
K6269-2	stomach	STAD/ESCA	gast
K6270-2	stomach	STAD/ESCA	gast
K6271-2	kidney	KIRC/KIRP	rena
K6272-2	kidney	KIRC/KIRP	rena
K6365-2	colon	COAD/READ	colo
K6366-2	colon	COAD/READ	colo
K6367-2	colon	COAD/READ	colo
K6368-2	lung	LUSC/LUAD	lung
K6369-2	breast	BRCA	brea
K6370-2	breast	BRCA	brea
K6371-2	prostata	PRAD	pros
K6372-2	prostata	PRAD	pros

Reads per million (RPM) were used for classification and further ratio and voom transformed to evaluate eventual impacts on classification performance.

- ratio: Transform RNA-seq RPM values to log_2 ratios using the sum of expression in all samples as the reference.
- voom: Transform RNA-seq total counts to log_{CPM} values using the *voom()* function in the limma package.

After importing, the used data was combined to represent a specific format (Table 2.7):.

Table 2.7: The structure of the 21 samples of imported MU RNA-seq data provided by the MU Graz.

sample_id	**contig_id**	**contig_id**	**contig_id**	**contig_id**	**...**
sample_1	RPM_data	RPM_data	RPM_data	RPM_data	...
sample_2	RPM_data	RPM_data	RPM_data	RPM_data	...
sample_3	RPM_data	RPM_data	RPM_data	RPM_data	...
sample_4	RPM_data	RPM_data	RPM_data	RPM_data	...
...	...	...	...	...	...

Through feature selection methods (SNR) and mapping to features common to MU Graz RNA-seq data the final data sets are reduced in their feature count.

2.1.5 Neural network optimization procedures

In order to compare neural network and SVM classification results, during optimization TCGA RNA-seq data sets were imported in already feature reduced form. This way time is saved as feature extraction steps do not have to be executed all over again. Data is split by random sample subsetting and together with a vector contaning the primaries as factors fed into an SVM for training and validation purposes.
The used feature reduced data sets are:

- *RNAseq_TCGA_33.txt* and *primary_frame.txt* containing all 33 classes of TCGA data and their primaries.
- *RNAseq_TCGA_19.txt* and *primary_frame_selected.txt* containing selected 19 classes of TCGA data and their primaries.
- *RNAseq_TCGA_12.txt* and *primary_frame_Tothill.txt* containing 12 classes of TCGA data and their primaries common with Tothill *et al.* data [5].

2.2 Hardware and software environment

2.2.1 Hardware environment and operating system

All analyses were performed on a ThinkPad E480 notebook.

- Memory 7,6 GiB
- Processor Intel® Core™ i5-8250U CPU @ 1.60GHz × 8
- Graphics Intel® UHD Graphics 620 (Kabylake GT2)
- Ubuntu 18.04.3 LTS
- GNOME 3.28.2
- OS type 64-bit

2.2.2 R-Project

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms [10]. For scripts in this thesis, R version 3.6.1 (2019-07-05) was used. Used packages for SVMs and neural networks are the following (Table 2.8):

Table 2.8: Required R packages for creation of the proposed SVMs and neural networks.

R-package	**Package Vers.**	**Ref.**	**Comment**
e1071	1.0.2	[52]	Provides functions for SVMs.
limma	3.40.6	[53]	Data analysis, linear models and differential expression for microarray data.
devtools	2.2.1	[54]	Tools to make developing R packages easier.
keras	2.2.5	[55]	Provides functions for NNs.
kerasR	0.6.1	[56]	Provides functions for NNs.
HGNChelper	0.8.1	[57]	Identifying and correcting HGNC human gene symbols.
httr	1.4.1	[58]	Tools for Working with URLs and HTTP.
RCurl	1.95-4.12	[59]	General Network (HTTP/FTP/...) Client Interface for R.
rjson	0.2.20	[60]	Converts R object into JSON objects and vice-versa.
stringr	1.4.0	[61]	Simple, Consistent Wrappers for Common String Operations.

2.2.3 RStudio

RStudio is an integrated development environment (IDE) for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management [62]. In this thesis RStudio Version 1.1.453 was used.

2.2.4 Keras

In this thesis different network types are realized through the use of *Keras* and *Keras in R* [63, 64]. The aforementioned APIs allow for modeling, training, testing and visualization of neural networks as well as needed tools for data processing.
Keras is a high-level neural network API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano [63]. It can be used directly with Python in a console environment, IDEs or also with R in RStudio. In this thesis Keras version 2.3.1 / 7 was used.

2.2.5 Keras in R

Keras packages are also available for CRAN and provide an interface to Keras. It allows code to run on the CPU or on GPU, has build in support for convolutional neural networks and recurrent neural networks. It also supports arbitrary network architectures [64].

2.2.6 TCGA-Assembler 2

In order to download data from TCGA, the scripts of TCGA-Assembler 2 (Version: 2.0.6) were used [50]. TCGA-Assembler consists of different modules: Module A and Module B were used for this thesis. For downloading TCGA data using Module A functions:

```
DownloadRNASeqData(cancerType = sCancer,
                   assayPlatform = "gene.normalized_RNAseq",
                   inputPatientIDs = NULL, saveFolderName = sPath1)
```

Further in order to perform basic processing of downloaded data using Module B functions:

```
ProcessRNASeqData(inputFilePath = path_geneExp[1],
                  outputFileName = paste(sCancer, "geneExp", sep = "__"),
                  dataType = "geneExp", outputFileFolder = sPath2)
```

2.3 Data preprocessing and feature selection

The imported data sets have to be usable for later classification processes via SVMs and neural networks. They are prepared via a number of preprocessing steps and a feature extraction algorithm adapted from Tothill *et al.* [5].

2.3.1 Preprocessing of raw intensity microarray data

After importing and extracting the needed columns from the raw data files, a resulting matrix is structured as follows: Columns represent samples per sample ID and rows represent different genes. The input matrix is transposed as SVMs and neural networks need a sample as rows input structure. A sample annotation provides information about gene names and IDs for future mapping to MU RNA-seq data and TCGA data. Features in microarray data are labeled with their reporter name.
Known primaries for samples are provided, thus electing missing primaries or vice versa allows to separate the data into samples without CUP and CUP samples.
Each respective primary for CUP and non CUP samples is provided by Tothill *et al.* [5] and converted into factors. Factors represent categorical data in R and are stored as integers with associated labels [65]. This is important for later classification as SVMs and neural networks need to know which class a sample belongs to in order to train the model. This also remains true for later validation with test or even new data. For classification with neural networks, each primary is assigned an integer value as NN only work with numeric values starting at 0 (zero) in Python based Keras.

2.3.2 Preprocessing of TCGA RNA-seq data

Downloaded data from TCGA has the following structure: Columns mark different samples via sample ID and rows represent separate genes. This means the data set has to be transposed for use in SVMs and neural networks. Features are labeled with gene IDs and EntrezIDs. Further, a separate data frame is created containing the primaries of each sample ordered by ID. Primaries are both labeled with a character

string containing the project name as well as a numeric ID. Numeric IDs for example span from 0 to 32 representing 33 cancer classes as neural network only work with numerics. These primaries are also converted into factors for later classification with SVMs. All 33/19/12 data sets representing different TCGA projects were combined in a single dataset for further preprocessing using R functions.

2.3.3 Preprocessing of MU RNA-seq data

The AmpliSeq annotation data and the CUP count data (RPM) were imported into R. With preexisting functions MU RNA-seq data is further voom- and ratio transformed.The AmpliSeq annotation data comprises information about gene names and IDs map features to microarray data and TCGA data. In this script features of MU RNA-seq data are labeled by their contig_id. For feature extraction purposes, these features have to be mapped to either microarray or TCGA features. Used MU RNA-seq data:

- RPM data: Reads per million.
- ratio transform: Transforms all RNA-seq values to log_2 ratios using the sum of the expression in all samples as reference.
- voom transform: Transforms all RNA-seq total counts to logCPM values using the *voom()* function of the limma package.

2.3.4 Feature selection

Microarray and RNA-seq data contain a huge amount of features, where many of these features do not contribute to the classification. In order to reduce the feature count and only select select potentially important ones for classification a SNR ratio as described in Tothill *et al.* [5] is calculated. A data frame of SNR values for all features for each tumor type is created. The top n-features based on their SNR value are selected. A loop over all tumor types selects these top n-features and a final gene list is created by merging the individual feature lists. The SNR values for a specific

feature is calculated as follows:

$$SNR = \left| \frac{\mu(x_{in\_class}) - \mu(x_{out\_class})}{\sigma(x_{in\_class}) + \sigma(x_{out\_class}))} \right| \tag{2.7}$$

Where μ is the mean of the values, σ represents the standard deviation, x_{in_class} represents a selected tumor type and x_{out_class} all others.
With the top 50 features as calculated with SNR, a pool of features for further classification is created. A subset of the original data frame containing only the most important features is selected with these feature pools.

2.3.5 Feature mapping

For the classification of MU RNA-seq samples the feature set was reduced to the common ones, both occuring in TCGA/MA data and MU data. To classify samples it is important to train the model on features that exist within the data set. As TCGA and MU Graz RNA-seq are produced by different means the feature set may be different. A common set of features is created for training and validation of SVMs and NNs. This is done through merging both datasets by their Gene Symbol, maintaining only common entries.

2.3.6 Determining training- and test data

For training and validation of NN, the data set has to be split into two distinct training and validation sets, commonly named X_train and y_test. In contrast SVMs are trained on the whole dataset and validated on a randomly selected subset, in the scope of the R-code also named X_train and y_test. For LOOCV, the complete dataset and the function *loocv* is used for the given dataset and the specified classes. It returns a data frame with the probabilities of the class assignments for each sample and a confusion matrix.
Thus the data set as well as primary set is split into the train and test data by random sample subsetting. This results in the following data sets for neural network classification:

- X_train: Training set for neural networks.

- y_train: True labels of trainings samples.
- X_test: Validation set for neural networks.
- y_test: True labels of validation samples.

X_train and X_test data sets are matrices containing the respective features of a sample as columns (with the samples as rows).
The y_data is a character vector with corresponding cancer primary sites (classes). To prepare this data for training, the vectors are one-hot encoded into binary class matrices using the Keras *to_categorical()* function. One-hot encoding is a process converting categorical variables in a form used in machine learning. This results in a vector of each possible cancer type for each sample. Each primary is assigned an integer value from 0 to the number of primaries -1. One-hot encoding results in a matrix assigning each sample each possible cancer type with 0 meaning false and 1 for true.
CNNs need four dimensional tensors as input. The input shape follows the form: input_shape = (batch_size, height, width, depth), in terms of an image the height and width in pixels and depth as the number of color channels (e.g. RGB equals a depth of 3, greyscale equals a depth of 1). This is important as CNNs are mostly used for image and time series classification. While providing the required 4D tensors is a simpler task with pictures, gene expression data is provided as a 2D matrix with columns as features and rows representing samples.
This means we have to provide an input shape in the following form (Figure 2.1):

```
img_rows = 1
img_cols = number_of_features
X_train_CNN <- array_reshape(X_train_TCGA,c(nrow(X_train_TCGA),img_rows, img_cols,-1))
X_test_CNN <- array_reshape(X_test_TCGA,c(nrow(X_test_TCGA),img_rows, img_cols,-1))
input_shape <- c(img_rows, img_cols, 1)
```

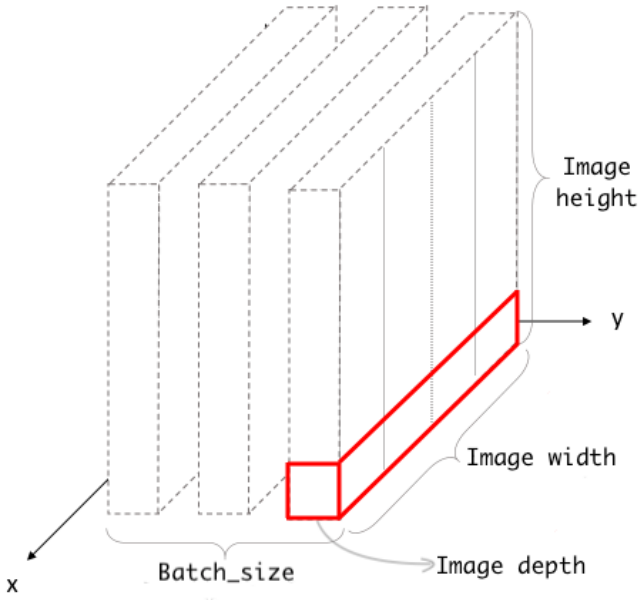


Figure 2.1: The input shape of convolutional neural networks. The red line represents the gene expression input after reshaping. Figure generated with and adapted from [4].

In the case of RNA-seq data the number of image rows equals 1, the number of image columns equals the number of features and the depth equals also 1. The batch_size in this thesis is dynamic. The *array_reshape()* function allows to reshape the output to a certain shape without altering the information contained in the 2D RNA-seq matrices.

2.4 SVM training and validation

Different SVMs were created during optimization (Table 2.9). These models were compared to the original Tothill *et al.* model and NN classification performance to evaluate different network structures and choose the best possible configurations going forward.

Table 2.9: An Overview of SVM models trained using the different datasets during optimization processes. The model name given is used throughout the document to refer to a specific model.

Model name	Description
model_SVM_MA_13_processed	SVM using processed MA data set provided by Tothill *et al.*.
model_SVM_MA_13_mapped	SVM using processed MA data set provided by Tothill *et al.* with features common to MU samples.
model_SVM_MA_13_raw_mapped	SVM using raw intesnsity MA (range normalized background corrected) data sets provided by Tothill *et al.* with features common to MU samples.
model_SVM_RNAseq_33	SVM using TCGA RNA-seq data, 33 classes.
model_SVM_RNAseq_19	SVM using TCGA RNA-seq data, 19 classes of single primary.
model_SVM_RNAseq_12	SVM using TCGA RNA-seq data, 12 classes common with Tothill *et al.*.

A string of character primaries was created and then converted into factors as required for SVM input. Leave one out cross-validation (LOOCV) of the MA datasets (MA_Tothill_13_processed, MA_Tothill_13_mapped and MA_Tothill_13_raw) provides a first estimation of the SVM model classification accuracy. LOOCV is a particular case of cross-validation calculating a statistic on the left out sample in order to test all possible ways of dividing the data set in training and validation parts [66]. It is used to determine how accurate the predictive model will perform in practice and can serve as a first waypoint in optimizing a model.
Validation method used for SVM classification:

- Train the classifiers with the expression of the reporters on the microarray matching the RNA-seq amplicons.
- Classify a random subset of the microarray samples (non-CUP), the 13 CUP samples and the full set of MU Graz RNA-seq samples.

The datasets are used to train SVM models using the *trainClassifier()* function. This function requires a preprocessed training data set, respective primaries as factors and a character vector of all possible primaries and returns a model. *trainClassifier()* uses the *e1071* function *svm()* with the parameter: kernel = linear, scale = F, cost= 10, probability = T to train a model [67]. *classifySamples()* requires the trained model, validation data set, the number of possible classes for a single sample and respective primaries as factors and returns a data frame with each sample in the rows, the actual label (if known) and predicted tumor type, the decision margin level, and the

classification probabilities for the individual tumor types in columns.

For final classification results of optimized models, confusion tables were created and are provided in the Appendix (Tables A27-A35). The classification performance is visualized in matrix form, rows representing the predicted label and columns the true label of a class. Correct classifications are in the principal diagonal while misclassifications appear in off-diagonal elements. A perfect model would show all classifications in the principal diagonal.

2.5 NN validation and training

A series of different NN were created in order to test different models and their effect on classification accuracy. A comparison with literature and testing procedures yielded three main network types which were compared. First, it has to be evaluated if deep learning is the correct approach to classifying gene expression data. The comparison of the classification performance of DNN with simpler structures, follows the choice of the ideal type of neural network (Table 2.10):

Table 2.10: Used hyper parameter for neural network classifications using MLPs and CNNs.

parameter	**settings**
Activation function	(relu, elu, selu, hard_sigmoid, linear,sigmoid, softmax, softplus, softsign, tanh, exponential)
Number of hidden layers	1,2,4
Number of units per layer	[12:700]
overfitting strategy	SNR feature selection & dropout strategy
input dropout ratio	[0.25:0.6]
hidden dropout ratio	[0.2:0.5]

First each neural network model was tested on smaller subsets of the complete TCGA RNA-seq data to save time modeling and optimizing each model. The gained knowledge was then applied to create a version of the best performing neural network classifying all 33 available classes (Table 2.11):

Table 2.11: An overview of NN models trained using the different datasets during optimization processes and final training with complete 33 class TCGA data set to classify 21 MU Graz CUP samples. The model name given is used throughout the document to refer to a specific model.

Model name	Description
model_FFNN_Tothill_MA_13_processed	NN using processed MA data set provided by Tothill *et al.*.
model_FFNN_Tothill_MA_13_mapped	NN using processed MA data set provided by Tothill *et al.* with features common to MU samples.
model_CNN_RNAseq_33	CNN using TCGA RNA-seq data, 33 classes.
model_CNN_RNAseq_33_mapped	CNN using TCGA RNA-seq data, 33 classes with features common to MU samples.
model_CNN_RNAseq_33_balanced	CNN using TCGA RNA-seq data, 33 balanced classes.
model_FFNN_RNAseq_19	FFNN using TCGA RNA-seq data, 19 classes of unique primaries.
model_FFNN_RNAseq_19_mapped	FFNN using TCGA RNA-seq data, 19 classes of unique primaries with features common to MU samples.
model_DEEP_RNAseq_19	Deep learning FFNN using TCGA RNA-seq data, 19 classes of unique primaries.
model_DEEP_RNAseq_19_mapped	Deep learning FFNN using TCGA RNA-seq data, 19 classes of unique primaries with features common to MU samples.
model_CNN_RNAseq_19	CNN using TCGA RNA-seq data, 19 classes of unique primaries.
model_CNN_RNAseq_19_mapped	CNN using TCGA RNA-seq data, 19 classes of unique primaries with features common to MU samples.
model_FFNN_RNAseq_12	NN using TCGA RNA-seq data, 12 classes common with Tothill *et al.*.

2.5.1 Multilayer neural networks

Depending on the data set and on relevance a varying number of different network types were introduced. In this neural network models, weights are calculated automatically based on input and output shape. The input shape depends on the data set. Using gene expression data with MLPs, an integer providing the number of features is sufficient, for CNNs an input shape in the form of a 4D tensor is required. If no batch size is defined, a dynamic size is assumed based on the received input data. All data sets were trained on a simple MLP with one hidden layer. For classification of microarray data (MA_Tothill_13) and 12 class RNA-seq

data (RNAseq_TCGA_12) the following model was used:

```
model %>%
  layer_dense(units = 256, activation = "relu", input_shape = c(input_nodes)) %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 128, activation = "relu") %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 13, activation = "softmax")
```

A sequential model with dense layers used for input, output and hidden layer was created for FFNNs and DLNN. For training and validation of TCGA data sets different activation functions were applied in order to observe their effect on classification accuracy. A dropout strategy was chosen in order to avoid overfitting. Here, randomly selected neurons are ignored during training. The neural network, thus becomes less sensitive to specific weights and neurons, is better generalized and less likely to overfit. The input shape is defined by the number of features.
A similar structure was used for selected TCGA data of 19 cancer classes (RNAseq_TCGA_19) to study the effect of deep learning (with more than 3 hidden layers) models on classification accuracy.
A sequential model with dense layers used for input, output and hidden layer was created and for classification of TCGA data different activation functions were used in order to observe their effect on classification accuracy. A dropout strategy was chosen in order to avoid overfitting. The input shape is defined by the number of features.

```
model_deep %>%
  layer_dense(units=700,activation='softsign',input_shape=c(input_nodes)) %>%
  layer_dropout(rate=0.6) %>%
  layer_dense(units=350,activation='softsign') %>%
  layer_dropout(rate=0.5) %>%
  layer_dense(units=150,activation='softsign') %>%
  layer_dropout(rate=0.3) %>%
  layer_dense(units=100,activation='softsign') %>%
  layer_dropout(rate=0.3) %>%
  layer_dense(units=50,activation='softsign') %>%
  layer_dropout(rate=0.2) %>%
  layer_dense(units = 19, activation = 'softmax')
```

2.5.2 Convolutional neural networks

As for the CNN model used on selected data with 19 classes (RNAseq_TCGA_19) and the complete data set with 33 classes (RNAseq_TCGA_33):

```
19  layer_conv_2d(filters = 32, kernel_size = c(1,71), activation = 'softsign', input_
        shape = input_shape) %>%
20  layer_conv_2d(filters = 64, kernel_size = c(1,71), activation = 'softsign') %>%
21  layer_max_pooling_2d(pool_size = c(1, 2)) %>%
22  layer_dropout(rate = 0.25) %>%
23  layer_flatten() %>%
24  layer_dense(units = 128, activation = 'softsign') %>%
25  layer_dropout(rate = 0.5) %>%
26  layer_dense(units = num_classes, activation = 'softmax')
```

A sequential model with dense layers used for output and hidden layer was created. The input layer is a convolutional layer, whose input is RNA-seq data as a 4D tensor. 32 filters are used on the input data with a kernel size of $(1,71)$ as we are not confident that there is a correlation between neighboring RNA-seq values. Therefore a large kernel size is chosen in order to capture only the global features associated with this kernel [68]. A second convolutional layer follows the first with 64 filters and the same kernel size. A max pooling layer reduces the important information from the previous layers. A flatten layer flattens the 4 dimensional tensors to 2 dimensional input values for a further dense hidden layer. This neural network structure was adapted from Mostavi *et al.* [68]. For classification of TCGA data different activation functions were used in order to observe their effect on classification accuracy. A dropout strategy was chosen in order to avoid overfitting.

2.6 Validation strategy

Keras and *KerasR* provide visualization methods [69] which were used in order to assess model accuracy and training progress. The Keras package [64] calculates accuracy and loss during training and the *evaluate()* function calculates accuracy and loss while validating with test data sets. High accuracy lets assume a good model, fit for classifying new data, e.g. MU RNA-seq data. MU data were then classified using only common features and compared between different transformations and neural network models. Loss is calculated per categorical crossentropy:

$$L(y,\hat{y}) = -\sum_{j=0}^{M}\sum_{i=0}^{N}(y_{ij} * log(\hat{y}_{ij})) \tag{2.8}$$

Where $\hat{y}$ is the predicted value. The distribution of the prediction is compared to the true distribution. The loss is minimized when the NN output is close to the one-hot encoded vector of true labels [70, 71].

The categorical accuracy is a parameter based on the keras metrics.py file on GitHub:

```
def categorical_accuracy(y_true, y_pred):
    return K.cast(K.equal(K.argmax(y_true, axis=-1), K.argmax(y_pred, axis=-1)),
                          K.floatx())
```

This function computes how often the predicted label matches the true label. This frequency is ultimately returned as categorical accuracy [72].

2.7 Analysis of balanced training sets on classification performance

An SVM treats all classes equally and is therefore not well suited to unbalanced class sizes [73].
Tuning datasets with unbalanced class size is a difficult task and needs to be done with great care. What method makes sense in a given circumstance is dependent on a variety of factors and considerations. In many cases having an unbalanced data set can even profit the model as it learns the natural number of occurences depending on each class. If the classification of new samples is not accurate enough, model tuning can be an approach to correct these failings. The data type and specifics as false negative rate, accuracy, area under the curve (AUC) and precision can help in selecting a tuning method. In the case of this thesis, a simple balancing approach using equal class sizes of random samples was chosen to compare results with the unbalanced models.
There are different methods trying to adjust to this problem depending on circumstances, here is a summary of a few relevant [74–81]:

- Model Tuning
- Alternate Cutoffs
- unequal case weights
- Sampling methods

- Cost sensitive training

2.7.1 Model Tuning

Model tuning is a way to produce an analysis that responds exactly to a particular set of data [80, 81]. This can be done beside other options by over-fitting or under-fitting the data.

Random under-sampling: Balancing by randomly eliminating majority class samples until balanced out [74–77].
Random over-sampling: Increases the number of samples in minority class by randomly replicating them [74–77].
Cluster based over-sampling: A k-means clustering algorithm is independently applied to majority and minority class instances [74–77].
k-fold cross validation: A data set is split into a k-number of groups and the skill of a machine learning procedure to predict unseen data is estimated [78, 79, 81].

- shuffle data set randomly
- split in k-groups
- for each group:
    1. Take group as hold out or test data set.
    2. Take remaining groups as training data set.
    3. Fit model on training set and evaluate on test set.
    4. Retain evaluated score and discard model.
- summarize the skill d the model using the sample of model evaluation scores.

Choose k to be representative. k is often chosen as 10 as it generally results in a model model skill estimate with a low bias and modest variance. k = n in leave one out cross validation (LOOCV).
k-fold cross validation identifies clusters in the data set and each cluster is over-sampled such that all clusters of the same class have a equal number of instances and all classes have the same size.

Informed over-sampling: Using e.g. a synthetic minority over-sampling technique (SMOTE) overfitting can be avoided. A subset of data is taken from a minority class

as an example and then new synthetic similar instances are created and added to the original data set. A standard model tuning approach could be:

- divide data set: 80% training and 20% validation
- use k-foold cross validation on training set to tune model [78, 79, 81].
- repeat until optimized.
- use model to predict on validation set.

2.7.2 Alternate cut-offs

Can be used for models that have two possible outcomes. Here the definition for a predicted event is changed. The threshold for choosing between two events can be adjusted if a smaller confidence level for choosing one event is required. For this technique the receiver operating characteristic (ROC) curve can be used as it calculates sensitivity (SN) and specificity (SP) across a continuum of cutoffs in order to determine an appropriate balance between SN and SP.

2.7.3 Unequal case weights

Many predictive models have the ability to use case weights where each individual data point can be given more emphasis in the model training phase.

2.7.4 Other sampling methods

Instead of balancing the model, the class frequencies are balanced. This eliminates the fundamental imbalance issue. There is a huge amount of different possible methods, completely dependent on personal preference and application.

2.7.5 Cost sensitive training

The cost of misclassification is taken into consideration and then the cost is minimized.

3 Results

3.1 Generation and Processing of MA and TCGA RNA-seq data

A series of processing steps creates MA and TCGA RNA-seq data sets for training and validation purposes. Each Tothill *et al.* sample has its raw microarray information contained in a single CSV file. Because of corrupted data files, an algorithm was used on raw intensity MA files to determine the correct intensity columns, which are then used for further calculations. Other than extraction and correction methods, preprocessing steps for MA and RNA-seq data are often similar.
Relevant features were extracted from both data sets and then divided into training and test sets for later classification. The reader is referred to Chapter 2 for the detailed processes. The following sections summarize these steps in the scope of the results of this thesis and provide detailed information about specific steps in the classification processes.

3.1.1 Determination of raw data columns

Investigation of the raw data files revealed that the column headers did not seem to represent the data in the columns. The raw data files were created with the help of GenePix Version 4.1 [5]. In order to determine the relevant data from the raw files [5] a script calculates the correlation between all variables of the raw data while calculating the RoM (ratio of means) for every column the difference between the column and the RoM as described in the Gene Pix Version 4.1 file format [31] using a combination of four other columns.
The ratio of means equals the ratio of the arithmetic median intensities for each

feature and each wavelength, with the median background subtracted [31]. Each possible 4 column combination of the raw data is once used to calculate a respective RoM. This is repeated for each row and for each file. Through comparison of the result with every possible column the ones most likely to contain the desired data are determined (with a difference smaller than 0.001). The RoM itself is not actually needed, but the single variables that calculate the ratio. Under the assumption that the columns themselves are still in order and only the column headers are at the wrong positions, a combination of four columns used for the calculation must equal the unknown columns for the RoM. In order to salvage the data the following steps were applied:

1. Select the first four columns as A, B, C, and D, respectively.
2. Calculate the RoM for columns A, B, C, and D.
3. If the result equals (maximal difference of 0.001) another column in the raw data file it is most likely the RoM.
4. If the result does not equal any other column in the raw data file:
    - If there are still columns to choose left set A to the next column and start again at 2).
    - If all possibilities for A are exausted and there are still new values left for B chose a new Column for B and start again at 2) and reset the counter for A.
    - If all possibilities for A,B are exausted and there are still new values left for C chose a new Column for C and start again at 2) and reset the counter for A,B.
    - If all possibilities for A,B,C are exausted and there are still new values left for D chose a new Column for D and start again at 2) and reset the counter for A,B,C.
5. Write out the most likely combination for the RoM.
6. Repeat for each row of the raw data set.
7. Repeat for each raw data file.
8. Choose the combination that leads to the RoM the highest number of times.

The respective script can be reviewed in the Appendix (Listing 1). The GenePix raw data file contains 42 relevant columns with MA data. This results in $42^4 = 3,111,696$

RoM calculations and comparisons. From the analysis above, the following column assignment was determined:

- mean_F635 = median_B635
- median_B635 = mean.of.ratios
- mean_F532 = sum.of.means
- median_B532 = F.Pixels

3.1.2 Data import and export

For the adaption and extension of the existing script the raw data files had to be imported. The raw data is distributed in 242 files. Each file corresponds to a specific patient. Different calculation executed with Gene Pix Version 4.1 [5] are ordered in columns while each row represents a reporter identifier (Table 3.1).

Table 3.1: An example of one of the raw intensity data files as provided by Tothill *et al*, generated by GenePix Version 4.1.

Reporter identifier	**Composite Sequence identifier**	**RoM**	**mean_F635**	**median_B635**	**...**
ID	seq	data	data	data	...
ID	seq	data	data	data	...
ID	seq	data	data	data	...
ID	seq	data	data	data	...
...	...	...	...	...	...

The relevant data column is extracted and added to a data frame in R. Each extracted column can be transformed as seen in the next chapter. The function *readAllRawData()* returns a data frame with samples and their respective data. Each file is connected to the respective sample name via the table "E-MEXP-113.sdrf.txt" to obtain the same structure as used in RNA-seq data (Table 3.2). While importing and determining useful columns in the raw intensity data files each dataframe is transposed to accommodate SVM and NN classification requirements.

Table 3.2: An example of imported raw intensity data, after combining each data file into a single data frame for further use in the R workflow.

sample name	**sample name**	**sample name**	**sample name**	**sample name**	**...**
transformed_data	transformed_data	transformed_data	transformed_data	transformed_data	...
transformed_data	transformed_data	transformed_data	transformed_data	transformed_data	...
transformed_data	transformed_data	transformed_data	transformed_data	transformed_data	...
transformed_data	transformed_data	transformed_data	transformed_data	transformed_data	...
...	...	...	...	...	...

Raw microarray data are imported through the following functions: *readRawData()* reads a single raw microarray data file.
The function *readAllRawData()* repeats this process for all raw microarray data files. The return values are formatted as needed for further training and classification processes. The structure of this function can be seen in the Appendix (Listing 2). Each raw data file is imported, the relevant columns extracted and assigned to a data frame for further calculations.

In the script *tcga.R* RNA-seq data were downloaded from TCGA with TCGA-Assembler 2 (using functions from Modules A and B) combined into a single data set consisting of 33, 19 and 12 cancer classes, respectively. A separate data set containing primaries as numeric IDs and a character string for each sample was created and imported for further preprocessing.

- *RNAseq_TCGA_33.txt* and *primary_frame.txt* containing all 33 classes of TCGA data and their primaries.
- *RNAseq_TCGA_33_balanced.txt* and *primary_frame_balanced.txt* containing all 33 balanced classes of TCGA data and their primaries (45 samples each).
- *RNAseq_TCGA_19.txt* and *primary_frame_selected.txt* containing all 19 classes of TCGA data and their primaries.
- *RNAseq_TCGA_12.txt* and *primary_frame_Tothill.txt* containing 12 classes of TCGA data and their primaries common with Tothill *et al.* data [5].
- Processed data from Tothill *et al.* containing all 13 classes of microarray data and their primaries [5].

3.1.3 TCGA-Assembler 2 Problems

TCGA-Assembler 2 provides modules to download data from TCGA [50]. Download was performed using the Module A function *DownloadRNASeqData()* and Module B - function *ProcessRNASeqData()* to execute some processing steps.
The downloading of all 33 data sets went without problems. However, the datasets for *TGCT* and *BRCA* were not processed correctly. The processing step did finish correctly when selecting a random subset of samples from the two data files. It is possible that corrupted data is contained within these files, which lead to errors when using the TCGA functions. Further investigation showed that per default a boxplot of the said data was created and the function was not able to do so. In order to correctly process and use the data for this thesis the responsible functions *CombineRedundantFeature()*, *CheckGeneSymbol()* and *ProcessRNASeqData()* from the TCGA-Assembler Module B were altered and corrected in the script *tcga.R*. *ProcessRNASeqData()* is responsible for processing downloaded data files and uses *CombineRedundantFeature()* and *CheckGeneSymbol()* to do so. Some parts of these functions, used to create and access boxplots of the data, were disabled.

3.1.4 Feature extraction

The resulting data sets were imported and transposed for compability. Feature extraction was done via the SNR algorithm proposed by Tothill *et al.* [5] resulting in data sets with the following characteristics for microarray data:

- *exp_total_woCUP_genes_selected* with 229 samples of 607 features without CUP samples.
- *exp_total_CUP_genes_selected* with 13 samples of 607 features of only CUP samples.
- *exp_total_woCUP_genes_rnaseq* with 229 samples of 541 (common with MU Graz RNA-seq data) features without CUP samples.
- *exp_total_CUP_genes_rnaseq* with 13 samples of 541 (common with MU Graz RNA-seq data) features of only CUP samples.

Names of datasets were kept as proposed in the original script to provide a common naming convention. Feature extraction results in the following data sets for TCGA

RNA-seq data:

- *exp_total_TCGA_12* with 5673 samples of 521 features of TCGA RNA-seq data with common Tothill *et al.* features in 12 classes.
- *exp_total_TCGA_19* with 8314 samples of 902 features of TCGA RNA-seq data in 19 classes.
- *exp_total_TCGA_33* with 10464 samples of 1444 features of TCGA RNA-seq data in 33 classes.
- *exp_total_TCGA_19_mapped* with 8314 samples of 835 common features between MU and TCGA RNA-seq data in 19 classes.
- *exp_total_TCGA_33_mapped* with 10464 samples of 1342 common features between MU and TCGA RNA-seq data in 33 classes.
- *exp_total_TCGA_33_balanced* with 1485 samples of 1369 common features between MU and TCGA RNA-seq data in 33 balanced classes.

Feature extraction results in the following data sets for MU Graz RNA-seq data:

- *rpm_RNAseq_Tothill* with 21 samples of 541 features of MU Graz CUP samples for the use with Tothill *et al.* microarray data.
- *rpm_RNAseq_19* with 21 samples of 834 features of MU Graz CUP samples for the use with TCGA RNA-seq data of 19 classes.
- *rpm_RNAseq_33* with 21 samples of 1341 features of MU Graz CUP samples for the use with TCGA RNA-seq data of 33 classes.

MU Graz RNA-seq data is further voom and ratio transformed. For details of these data transformations, see Section 2.1.4.

3.1.5 Training & test data

Training and test sets were created via random sample subsetting (*X_train, y_train, X_text and y_test*) and primary vectors, one-hot encoded into binary class matrices for FFNN. For CNN, 4D tensors were created and fed into the network.

3.2 Neural network realization

Training and test data sets were used to train neural networks. This was a stepwise process, as there is no definitive neural network structure, but rather each problem is in need of individual optimization. The first step was the creation of a data set using the same cancer classes as the original Tothill *et al.* data MA_Tothill_13 (created dataset: RNAseq_TCGA_12). This makes the results directly comparable to the microarray classification, results in a small number of classes and in a quick training time of the network.
After optimizing this first prototype, the next step was to create a larger pool of classes, excluding TCGA classes with more than one primary in order to ensure there is no overlap between classes (RNAseq_TCGA_19).
After satisfactory results were achieved, a neural network model using all 33 TCGA RNA-seq classes was created in order to provide a high accuracy, high classification and precision (RNAseq_TCGA_33). Further different network types were compared to maximize classification accuracy (FFNN, deep learning FFNN and CNN). These 33 class models are presented as the finalized results of this thesis. The following chapter explains the modeling of these networks.

3.2.1 Modeling of neural networks

A number of sequential neural network models, representing a linear stack of layers, were created. Input shape is defined in Chapter 2. Batch size was chosen as dynamic. The input layer of CNNs represents a convolutional layer, followed by a max pooling and flatten layers. A range of one to four hidden layers was chosen to represent simple and deep learning structures. Different types of activation functions were chosen and their influence on accuracy can be seen in Section 3.3. For this thesis, MLPs and CNNs were compared as recurrent neural networks (RNNs) are not really suited for tabular data and lead to a bad performance [82]. The following loss functions, metrics and optimizer were used for the NN:

- loss function: *categorical crossentropy*
- metric: For multy-class classification *categorical accuracy* has to be chosen.
- optimizer: The *rmsprop* optimizer was chosen for FFNNs as it divides the gradient by a running average of its recent magnitude.

- optimizer: The *adam* optimizer was chosen for deep FFNNs.
- optimizer: The robust *Adadelta* optimizer was chosen for CNNs as it adapts learning rates based on a moving window of gradient updates.

The output layer has nodes equal to the number of classes and always uses a *softmax* activation function to guarantee an output between 0 and 1 representing the class membership. This is the (from a NN model) calculated probability that a sample belongs to a class in the scope of this classification independent from any other classes.
Classifications were realized in the following scripts:

- *NN_CUP_TCGA_33.R*: Classification of 33 TCGA cancer classes.
- *NN_CUP_TCGA_19.R*: Classification of selected 19 TCGA cancer classes of single primaries.
- *NN_CUP_TCGA_12.R*: Classification of 12 TCGA cancer classes common with Tothill *et al.* classes.
- *NN_CUP_Tothill_13.R*: Classification of 13 cancer classes using Tothill *et al.* microarray data [5].
- *SVM _Tothill_CUP_III_v.q.5.R*: Classification of TCGA and Tothill *et al.* cancer classes with SVMs.

3.2.2 Neural network overview

Different models of neural networks were used to classify MU Graz RNA-seq data (Table 3.3). For all proposed neural network models *categorical crossentropy* was chosen as loss function, *accuracy* as training metric, a 0.2 validation split and a *softmax* output activation function. All neural networks in this thesis are sequential models.

Table 3.3: Overview of each neural network, used to classify MU Graz RNA-seq data during the different prototype phases and with the finalized neural network model. Interim stages of models used for optimization purposes are not included.

NN name	NN type	layer type	# hidden layer	act_fct	optimizer	# epochs	batch size	units/layer	dropout L_i
model_FFNN_MA_13_processed	FFNN	dense	1	relu	rmsprop	10	20	256/128/13	0.4/0.3
model_FFNN_RNAseq_12	FFNN	dense	1	relu	rmsprop	46	100	256/128/13	0.4/0.3
model_FFNN_RNAseq_19	FFNN	dense	1	softsign	rmsprop	67	100	256/128/13	0.4/0.3
model_DEEP_RNAseq_19	FFNN	dense	4	softsign	adam	67	100	700/350/150/100/50	0.6/0.5/0.3/0.3/0.2
model_CNN_RNAseq_19	CNN	conv./dense	2	softsign	adadelta	67	67	128	0.25/0.5
model_FFNN_RNAseq_33	CNN	conv./dense	1	softsign	adadelta	70	100	128	0.25/0.5

Table 3.4: Summary of neural network layer structure. The name and type of a layer are represented, output shape (dynamic batch size marked as *None* with a static number of output nodes) and trainable params.

network name	NN type	# hidden layer	output shape/layer*	#params/layer	total params
model_RNAseq_MA_13_processed	FFNN	1 dense	None/256/128/13	155648/32896/1677	190221
model_FFNN_RNAseq_12	FFNN	1 dense	None/265/128/13	133632/32896/1677	168205
model_FFNN_RNAseq_19	FFNN	1 dense	None/265/128/13	213760/32896/2451	249107
model_DEEP_RNAseq_19	FFNN	4 dense	None/700/350/150/100/50/19	584500/245350/52650/15100/5050/969	903619
model_CNN_RNAseq_19	CNN	1 conv./ 1 dense	(None, 1, 764, 32)/ (None, 1, 694, 64)/ (None, 1, 347, 64)/ (None, 22208)/ (None, 128)/ (None, 19)	2304/145472/2842752/2451	2992979
model_FFNN_RNAseq_33	CNN	1 conv./ 1 dense	(None, 1, 1271, 32)/ (None, 1, 1201, 64)/ (None, 1, 600, 64)/ (None, 38400)/ (None, 128)/ (None, 33)	2304/145472/4915328/4257	5067361

* Output shapes of CNNs represent a series of 4D tensors of the form output_shape = (batch_size, height, width, depth), batch_size = None represents a dynamic batch size.

The *summary(model)* function provided by Keras compiles a structural overview of the neural network layer (Table 3.4). The total parameter amount of each neural network is trainable with no untrainable parameter (cannot be optimized with training data) as stated by the summary function. Dropout layer do not add to the total number of trainable parameters. A detailed representation of each NN summary output can be viewed in the Appendix (Listings 4-9).

3.3 Training and validation classification results

After modeling, the different network types and prototypes were evaluated in order to determine which model structures result in the highest classification accuracy during validation. An overview of the classification results of SVMs and NN is provided in the following subsections.

3.3.1 SVM Classification results

Table 3.5: Overview of the highest reached SVM classification results of Tothill *et al.* [5, 7, 8] data and TCGA RNA-seq data as a comparison of neural network results. Divided into data type, number of samples and classes, features per total feature count, accuracy of the model (acc_M), validation accuracy (acc_{val}), Tothill *et al.* CUP sample classification accuracy (acc_{CUP}) and training time.

data	#samples	#classes	#features	acc_M	acc_{val}	acc_{CUP}	t [s]
D1	229	13	607	96.6%	100%	84.6%	1
D2	229	13	541	95.00%	100%	84.60%	1
D3	229	13	504	94.8%	100%	84.6%	1
D4	229	13	536	85.6%	100%	76.9%	1
D5	229	13	540	85.2%	100%	69.2%	1
D6	229	13	541	85.2%	100%	69.2%	1
D7	5674	12	521	-	98.10%	-	66
D8	10464	33	1444	-	93.30%	-	245
D9	8314	19	902	-	97.80%	-	1231

[D1] *MA_Tothill_13_processed* without CUP samples.
[D2] *MA_Tothill_13_mapped*.
[D3] *MA_Tothill_13_raw_log2* without CUP samples.
[D4] *MA_Tothill_13_raw_range* without CUP samples.
[D5] *MA_Tothill_13_raw* without CUP samples.
[D6] *MA_Tothill_13_raw_bg* without CUP samples.
[D7] *RNAseq_TCGA_12* with common TCGA and Tothill classes.
[D8] *RNAseq_TCGA_19* with selected 19 classes of single primaries
[D9] *RNAseq_TCGA_33* with 33 classes of TCGA data set.

Classification results of different raw intensity data transformations (transformations: raw intensity data, background corrected intensities, range transformed background corrected intensities and log_2-transformed intensities) using SVMs are compared to SVM classification of TCGA RNA-seq data (Table 3.5). Other transformations of raw intensity data do not yield an improvement of classification accuracy. A more detailed representation of the classification results is part of the Appendix (Tables A1-A15).

Newly generated models trained on TCGA RNA-seq data are compared with the original Tothill *et al.* processed microarray data. The number of samples and possible cancer classes is much higher when using variations of TCGA RNA-seq data. Training and validation with SVMs shows that classification accuracy is still comparable to the original, the process however takes significantly longer. These results also serve as a point of comparison for the following classification with neural networks.

3.3.2 NN Classification results

Classification results with different neural network models depend heavily on the chosen hyper parameters, ranging between various settings (Table 3.6). A strong dependency of classification accuracy on the used activation function can be observed. The activation functions *softsign* and *softplus* show the highest increase in accuracy. The use of CNN results in the highest accuracy, followed by FFNN and with the lowest classification accuracy DLNN. After a training time of 7700 seconds, an accuracy of 93.88% with a calculated loss of 0.43 was reached for the finalized NN version - model_CNN_RNAseq_33_mapped using CNN and 33 TCGA classes.

The classification of Tothill *et al.* microarray data yielded comparable results when using neural networks for classification (Tables 3.5 and 3.6). The SVM model yielded a 89% classification accuracy while the neural network provides approximately 87% accuracy. When classifying Tothill *et al.* CUP samples, SVM yielded 84.6% classification accuracy (69.2% when using raw microarray data). Neural networks performed better with 92.3% (85% when using common MU Graz features) accuracy when classifying CUP samples.
The classification of TCGA RNA-seq data of 19 classes yielded an accuracy of 97.4% classification accuracy for simple neural networks with one hidden layer. Using common MU Graz features 96.9% accuracy was achieved. Further 79.7% (81.8% when using common MU Graz features) accuracy when training with deep learning models and 98.5% accuracy when training CNN models with common MU Graz features.
Further the training and validation process of the complete 33 classes of TCGA RNA-seq data yielded an accuracy of 98.3% and 93.9% when using common MU Graz features. A more detailed depiction of single classification results can be found in the Appendix (Figures A1-A3 and Tables A27-A35).
Neural network models achieved a higher accuracy than SVMs, especially CNNs. The rather small increase in accuracy comes with a high increase in time needed for training. Classification accuracy of Tothill *et al.* CUP samples was higher with neural networks.

Table 3.6: Overview of the highest reached NN classification results of Tothill *et al.* [5, 7, 8] data and TCGA RNA-seq data. Divided into data type, NN type, number of samples and classes, number of features, accuracy of the model (acc_M), loss of the model ($loss_M$), number of layers, batch size per epoch, activation function (f_a), validation accuracy and loss, Tothill *et al.* CUP sample classification accuracy (acc_{CUP}) and training time.

data	type	#samples	#classes	#features	acc_M	$loss_M$	#layer	batch/ep.	f_a	acc	loss	acc_{CUP}	t [s]
D1	FFNN	242	13	607	95.98%	0.12	1	10/20	relu	86.97%	0.29	92.31%	10
D2	FFNN	242	13	541	98.63%	0.08	1	10/20	relu	93.49%	0.33	85.00%	10
D7	FFNN	5673	12	521	27.36%	11.71	1	100/46	relu	26.30%	11.89	-	46
D8	FFNN	10464	33	1444	04.81%	15.26	1	100/84	relu	04.30%	15.43	-	90
D8	FFNN	8314	19	902	20.38%	12.80	1	100/67	relu	20.39%	12.83	-	33
D8	FFNN	8314	19	457	21.23%	12.70	1	100/67	relu	21.23%	12.70	-	33
D8	FFNN	8314	19	902	11.75%	14.23	4	500/14	relu	11.75%	14.23	-	14
D8	FFNN	8314	19	902	61.64%	1.19	4	500/14	sigmoid	61.64%	1.19	-	67
D8	FFNN	8314	19	902	67.53%	1.09	4	500/14	tanh	67.53%	1.09	-	67
D8	FFNN	8314	19	902	79.74%	0.62	4	500/14	softsign	79.74%	0.62	-	67
D8	FFNN	8314	19	902	86.50%	0.43	1	100/67	tanh	86.50%	0.43	-	67
D8	FFNN	8315	20	902	20.20%	12.86	1	100/67	elu	20.20%	12.86	-	33
D8	FFNN	8316	21	902	87.13%	0.39	1	100/67	sigmoid	87.13%	0.39	-	33
D8	FFNN	8316	21	902	90.40%	0.31	1	100/67	hard_sigm.	90.44%	0.31	-	33
D8	FFNN	8316	21	902	10.70%	14.39	1	100/67	linear	10.70%	14.39	-	33
D8	FFNN	8316	21	902	18.28%	13.17	1	100/67	softplus	18.28%	13.17	-	33
D8	FFNN	8316	21	902	96.75%	0.10	1	100/67	softsign	97.42%	0.10	-	33
D8	CNN	8314	19	902	14.47%	13.78	2	100/67	relu	14.47%	13.78	-	4020
D8	CNN	8314	19	902	99.87%	0.09	2	100/67	sigmoid	98.50%	0.06	-	4020
D10	FFNN	8314	19	834	96.93%	0.09	1	100/67	softsign	96.93%	0.09	-	33
D10	FFNN	8314	19	834	81.84%	0.54	4	100/67	softsign	81.84%	0.54	-	67
D10	CNN	8314	19	834	98.32%	0.08	2	100/67	softsign	98.32%	0.08	-	4020
D11	CNN	10464	33	1341	98.67%	0.22	2	100/70	softplus	93.88%	0.24	-	7700
D12	CNN	1485	33	1369	89.90%	0.43	2	100/70	softplus	89.90%	0.43	-	1120

D1 *MA_Tothill_13_processed* without CUP samples.
D2 *MA_Tothill_13_mapped*.
D7 *RNAseq_TCGA_12* with common TCGA and Tothill classes.
D8 *RNAseq_TCGA_19* with selected 19 classes of single primaries
D10 *RNAseq_TCGA_19_mapped* - selected 19 classes of TCGA data with common MU Graz RNA-seq features.
D11 *RNAseq_TCGA_33_mapped* - with all 33 classes of TCGA data with common MU Graz RNA-seq features.
D12 *RNAseq_TCGA_33_balanced* - using 33 balanced classes of each 45 samples with common MU Graz RNA-seq features.

3.3.3 Misclassifications

Misclassifications can give indicators about NN behaviour, issues with the training data set and help to better understand the classification results. Properties of the used dataset, classes with a low sample count, uneven class size, NN type, chosen hyperparameter and many more circumstances can lead to misclassification of samples. Misclassifications of SVM *model_SVM_RNAseq_33*, NN *model_CNN_RNAseq_33_mapped* and NN *model_CNN_RNAseq_33_balanced* using balanced data sets were compared (Figure 3.1).

***model_SVM_RNAseq_33* misclassifications:** There were 160 misclassification across all 33 classes (Figure 3.1. A random subset (2093 samples) of the dataset `RNAseq _TCGA_33` was used to validate the created SVM model *model_SVM_RNAseq_33*. The percentage of misclassifications was higher than with comparable NN models like *model_CNN_RNAseq_33_mapped* reaching up to 17.02% in classes like Cholangiocarcinoma *CHOL*, which had also shown the highest misclassification percentage when working with NN models *model_CNN_RNAseq_33_mapped*. In general, it seems the amount of samples per class is even more important when using SVM classification as most classes with a low sample count exhibit a higher misclassification rate in this case.

***model_CNN_RNAseq_33_mapped* misclassifications:** There were 123 misclassifications rather unevenly spread over the classes. The highest amount of misclassifications, a rate of 10.64% occured in the class (*CHOL*) with a total of 47 samples used for training. However, no definitive decision can be made about the influence of small class sizes on the misclassification rate of all classes, as it ranges between no misclassifiactions (e.g. Adrenocortical Carcinoma [*ACC*] and Lymphoid Neoplasm Diffuse Large B-cell Lymphoma [*DLBC*]) and 10.64% for *CHOL*. Some cancer classes could need a higher amount of features to be classified correctly while others are able to be classified with a low sample count in comparison. Classes with significantly more samples (176 and 330 respectively for training) like Esophageal Carcinoma (*ESCA*) or Colon Adenocarcinoma (*COAD*) have also shown a misclassification rate of 7.58% and 5.76%. The rate of misclassifications seems to generally lower the higher the amount of samples per class is as seen with Bladder Urothelial Carcinoma (*BLCA*), Kidney Chromophobe (*KIRC*), Head and Neck Squamous Cell Carcinoma (*HNSC*),

LGG, Lung Adenocarcinoma (*LUAD*), Lung Squamous Cell Carcinoma (*LUSC*) and Prostate Adenocarcinoma (*PRAD*), which all consist out of more than 500 samples per class. In this case a trend can be observed that using more samples per class to train neural networks results in less misclassifiactions per class. The amount of misclassifications per class of TCGA RNA-seq data varies between classes (Figure 3.1). Classes with more than a single primary do not seem to result in significantly higher rate of misclassifications. Classes with a low amount of samples often show no higher rate of misclassifications than *BRCA* with the highest amount of samples. *CHOL*, *ESCA* and *KICH* show the highest rate of misclassifications. In the Appendix a misclassification overview of a finalized neural network version using TCGA RNA-seq data with 33 cancer classes using the dataset *exp_total_TCGA_33_mapped* is provided (Table A36). Features are all common with MU Graz RNA-seq data. The full list of features and classification probabilities are contained in the files *misclassifications_33_CNN_MUG.txt* and *misclassifications_33_CNN_MUG_prob.txt*.

***model_CNN_RNAseq_33_balanced* misclassifications:** The misclassification per balanced class differ from models without balancing procedure (Figure 3.1). While the number of total misclassifications is higher than without balanced data sets, the number of total samples used for training is much smaller too. Misclassifications are unevenly spread over all possible classes indicating little improvement over the unbalanced CNN model. Using unbalanced data did show little improvement on classes like *CHOL* which had shown the highest rate of misclassifications. There were also instances of classes, recording misclassification which did not show any before and vice versa. This could be due to NN needing more samples to properly discern between some classes or be the product of white noise. The class Lymphoid Neoplasm Diffuse Large B-cell Lymphoma *DLBC* shows an huge increase in misclassifications.

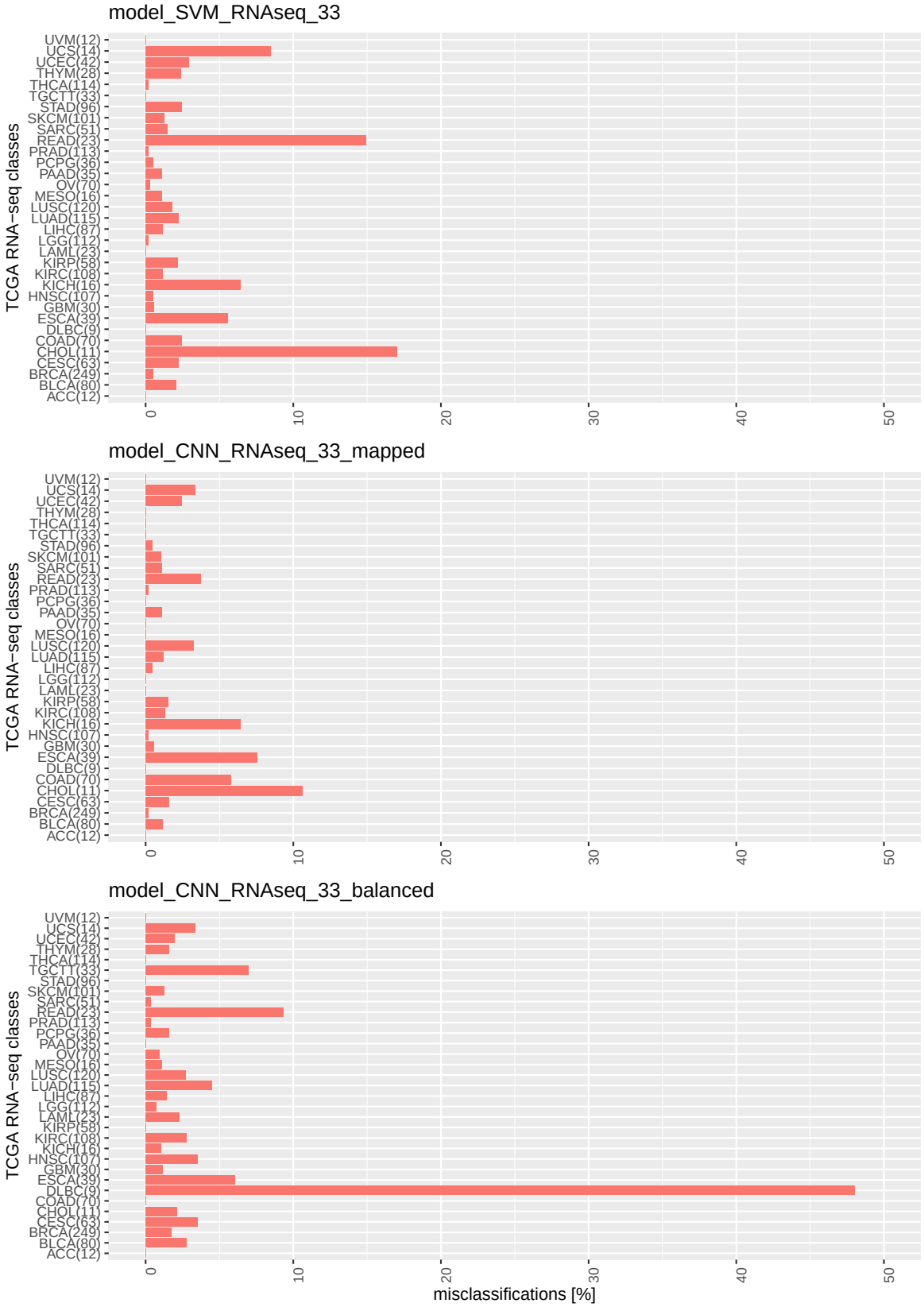


Figure 3.1: Misclassifications as percentage in each TCGA RNA-seq data class of in order of occurence: *model_SVM_RNAseq_33* (subset of 2093 samples), `model_CNN_RNAseq_33_mapped` (subset of 2093 samples) and *model_CNN_RNAseq_33_balanced* (subset of 297 samples). The total number of samples in each class is mentioned beside the class name.

3.3.4 Influence of hyper parameter settings on classification performance of NN

A range of different hyperparameters was used to test models and improve classification performance. The neural network was trained in batches of 100 samples for TCGA RNA-seq data and batches of 10 for microarray data. The number of nodes per layer was chosen corresponding to the respective network structure. The metric function used to judge the performance of the different models was *accuracy*. The optimizer was chosen as already stated in Chapter 3.2.1 . The number of hidden layers ranged between 1 and 4, while in general neural networks with a lower hidden layer count seemed to have a higher accuracy when trained on RNA-seq data. The time spent training a network was much smaller when using simple NNs with a single hidden layer and peaked when training a CNN. The choice of activation function had an impact on classification performance (Table 3.6). While *relu* worked fine with microarray data (accuracy of approximately 87% accuracy), the same set of parameters did not yield any meaningful results with the TCGA data set containing Tothill *et al.* classes - *RNAseq_TCGA_12* (approximately 26% accuracy using 12 classes in common with microarray data). The accuracy was also low for other neural networks trained on RNA-seq data (approximately between 4%-21% accuracy).
For deep learning neural networks, activation functions like *sigmoid, tanh* yielded a far higher accuracy, peaking when using *softsign* with approximately 79% accuracy. For simple FFNN activation functions *sigmoid, hard_sigmaid, tanh, linear, elu*, and *softplus* yieldeld a far higher accuracy peaking when using *sigmoid* with approximately 97% accuracy. For CNNs sigmoid provided an accuracy of 98.5% for selected (19 classes) TCGA RNA-seq data and 93.9% for the complete TCGA RNA-seq data set using common features with MU Graz RNA-seq data.

3.4 Classification of MU Graz cancer samples

In order to increase the classification accuracy for RNA-seq data provided by the Medical University Graz, which did not yield any meaningful results with the normalized MA data from Tothill *et al.*, raw microarray intensity data were used for

training and validation of the classifier. Additional to the raw data, log_2 and range transformed data and the sample intensity was used to train and validate.
The existing R script was adapted and extended to classify CUP samples of RNA-seq data with a classifier of the publicly available raw microarray data contained in the MA_Tothill_13_raw data set.

Table 3.7: Table with classification results of RPM MU Graz data with range normalized background corrected MA data generated model `model_SVM_MA_13_raw_mapped` classified RNAseq_MU_RPM data.

Sample	Actual	Pred	DM	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung	Total
K1999-6	colo	colo,gast,ovar	L	0	1	1	0	0	1	0	0	0	0	0	0	0	3
K2170-6	melan	colo,mela,rena	L	0	1	0	1	0	0	0	0	1	0	1	0	0	4
K2953-5	lung	colo,gast,ovar	L	0	1	1	0	0	1	0	0	0	0	0	0	0	3
K4824-4	panc	brea,colo,meso	L	1	1	0	0	1	1	0	0	0	0	1	0	0	5
K5108-24	ovar	brea,colo,meso	L	1	1	0	0	1	1	0	0	0	0	1	0	0	5
K5216-6	breast	brea,ovar	L	1	0	0	0	0	1	0	0	0	0	0	0	0	2
K6266-2	mela	brea,colo,meso	L	1	1	0	0	1	1	0	0	0	0	0	0	0	4
K6267-2	lung	colo,mela	L	0	1	0	1	0	0	0	0	0	0	0	0	0	2
K6268-2	panc	colo,meso,SCCo	L	0	1	0	0	1	0	0	0	0	0	1	0	0	3
K6269-2	gast	colo,gast,ovar	L	0	1	1	0	0	1	0	0	0	0	1	0	0	4
K6270-2	gast	gast,mela,panc	L	0	0	1	1	0	0	1	0	0	1	1	0	0	5
K6271-2	rena	brea,colo,meso	L	1	1	0	0	1	1	0	0	0	0	1	0	0	5
K6272-2	rena	brea,colo,ovar	L	1	1	0	0	0	1	0	0	1	0	0	0	0	4
K6365-2	colo	colo,pros	L	0	1	0	0	0	0	0	1	0	0	0	0	0	2
K6366-2	colo	colo	H	0	1	0	0	0	0	0	0	0	0	0	0	0	1
K6367-2	colo	colo	H	0	1	0	0	0	0	0	0	0	0	0	0	0	1
K6368-2	lung	brea,colo,meso	L	1	1	0	0	1	1	0	0	0	0	1	0	0	5
K6369-2	brea	brea,colo,gast	L	1	1	1	0	0	1	0	0	0	0	0	0	0	4
K6370-2	brea	brea,colo,ovar	L	1	1	0	0	0	1	0	0	0	0	0	0	0	3
K6371-2	pros	colo,meso,pros	L	0	1	0	0	1	0	0	1	0	0	1	0	0	4
K6372-2	pros	colo,pros,test	L	0	1	0	0	0	0	0	1	0	1	0	0	0	3
acc. [%]		4.76															

In the course of a Master's project the classification accuracy of SVMs using raw microarray intensity data provided by Tothill *et al.* was investigated. The MU Graz RNA-seq samples were separated in a dataset containing RPM, *voomTransformed* [83], and *ratioTransformed* data:

- Results generated using the untransformed RPM data format.
- Results generated using the function *ratioTransform()*; transform RNA-seq RPM values to log_2 ratios using the sum of expression in all samples as the reference.
- Results generated using the class *voomTransform()*; Transform count data to log_2-counts per million (logCPM), estimate the mean-variance relationship and use this to compute appropriate observational-level weights. The data are then ready for linear modelling [83].

Further, as with the Tothill *et al.* [5, 7, 8] datasets, the classification was done, using raw, background corrected intensities, range transformed background corrected and log_2-transformed datasets. This generates 15 separate classification results:
The classification of MU Graz RNA-seq data did not yield any meaningful results. Each sample was assigned multiple cancer types, which generates more than 100% assignments per sample. Respective tables containing the results for MU Graz samples are part of the Appendix (Tables A16-A26). A with range normalized background corrected MA data generated model *model_SVM_MA_13_raw_mapped* classified RNAseq_MU_RPM data with an accuracy of 4.76% (Table 3.7). All other models did not reach a classification accuracy above 0%. No confusion matrix was generated from this data sets because of the poor classification performance of the models.

Different neural network models were used to classify MU Graz CUP data with an emphasis on the performance of model *model_CNN_RNAseq_33_mapped* and other models for comparison. An increase in classification accuracy by the neural network models in comparison to the SVMs performance was achieved (Table 3.8). Model *model_CNN_RNAseq_33_mapped*, which is the result of NN optimization processes while training and validation different prototypes reached the highest accuracy of 52.38%. In general the classification results do not seem to agree with each other. While some samples are classified the same across all models, others differ strongly. The ratio transformed data seems to disagree the most, while RPM and voom transformed data are closer. The model *model_CNN_RNAseq_19* seems to result in similar predictions for ratio and voom transformed data. Other models disagree in most cases, while *LGG* seems to have been classified correctly by most models.

Table 3.8: The highest reached classification results of MUG RNA-seq (in RPM, ratio and voom transformed) data with different neural network models (M_i) as specified in the table footnotes. Each sample classification is compared to the true label $Lab._T$ by using the closest available TCGA label $Lab._{TCGA}$ and resulting accuracies are stated as percentage.

model	$Lab._T$	$Lab._{TCGA}$	**M1**	**M2**	**M3**	**M4**	**M5**	**M1**	**M2**	**M3**	**M4**	**M5**	**M1**	**M2**	**M3**	**M4**	**M5**
sample			RPM	RPM	RPM	RPM	RPM	ratio	ratio	ratio	ratio	ratio	voom	voom	voom	voom	voom
K1999-6	colon	COAD/READ	LGG	LGG	KICH	STAD	CHOL	BLCA	LGG	HNSC	LAML	MESO	LGG	LGG	KICH	STAD	COAD
K2170-6	melanom	SKCM	LAML	LAML	LAML	SKCM	SKCM	LAML	ACC	LAML	SKCM	MESO	LAML	LAML	LAML	SARC	SKCM
K2953-5	lung	LUSC/LUAD	LGG	HNSC	LGG	ESCA	ESCA	BLCA	BRCA	BLCA	BRCA	MESO	LGG	BLCA	HNSC	ESCA	ESCA
K4824-4	pancreas	PAAD	KIRC	LGG	LGG	STAD	STAD	BLCA	ACC	BLCA	BRCA	MESO	LGG	KIRC	KICH	STAD	STAD
K5108-24	ovar	OV	KICH	KICH	KICH	OV	OV	KICH	BRCA	BLCA	BRCA	MESO	KICH	BLCA	KICH	SARC	KICH
K5216-6	breast	BRCA	BRCA	BRCA	BRCA	BRCA	BRCA	BRCA	BRCA	BRCA	BRCA	MESO	BRCA	BRCA	BRCA	SARC	KIRP
K6266-2	melanom	SKCM	KICH	KICH	KICH	OV	OV	KICH	BRCA	BLCA	BRCA	MESO	KICH	KICH	KICH	SARC	KIRP
K6267-2	lung	LUSC/LUAD	LGG	LGG	LGG	STAD	STAD	LGG	LGG	HNSC	KIRC	MESO	LGG	LGG	LGG	ESCA	COAD
K6268-2	pancreas	PAAD	LGG	BLCA	BLCA	STAD	STAD	BLCA	LGG	BLCA	BRCA	MESO	LGG	BLCA	HNSC	ESCA	STAD
K6269-2	stomach	STAD/ESCA	LGG	LGG	LGG	STAD	STAD	LGG	LGG	BLCA	KIRC	MESO	LGG	LGG	KICH	STAD	STAD
K6270-2	stomach	STAD/ESCA	LGG	LGG	LGG	STAD	STAD	LGG	ACC	BLCA	BRCA	MESO	LGG	LGG	KICH	STAD	STAD
K6271-2	kidney	KIRC/KIRP	COAD	COAD	COAD	KIRC	KIRC	COAD	COAD	COAD	KIRC	MESO	COAD	COAD	KICH	SARC	KIRP
K6272-2	kidney	KIRC/KIRP	COAD	COAD	COAD	KIRC	KIRC	BLCA	COAD	HNSC	KIRC	MESO	LGG	BLCA	KICH	SARC	KIRP
K6365-2	colon	COAD/READ	LGG	LGG	LGG	READ	COAD	BLCA	LGG	BLCA	BRCA	MESO	LGG	LGG	KICH	STAD	COAD
K6366-2	colon	COAD/READ	LGG	LGG	LGG	STAD	STAD	BLCA	LGG	BLCA	BRCA	MESO	LGG	LGG	KICH	STAD	COAD
K6367-2	colon	COAD/READ	LGG	LGG	LGG	STAD	STAD	BLCA	BRCA	BLCA	BRCA	MESO	LGG	LGG	KICH	STAD	STAD
K6368-2	lung	LUSC/LUAD	HNSC	LGG	KICH	SARC	SARC	HNSC	HNSC	MESO	SKCM	MESO	HNSC	LGG	DLBC	SARC	STAD
K6369-2	breast	BRCA	BRCA	BRCA	BRCA	BRCA	BRCA	BRCA	BRCA	BRCA	BRCA	MESO	BRCA	BRCA	KICH	SARC	STAD
K6370-2	breast	BRCA	BRCA	BRCA	HNSC	STAD	ESCA	BRCA	BRCA	BRCA	BRCA	MESO	BRCA	BRCA	KICH	SARC	UCS
K6371-2	prostata	PRAD	KIRP	KIRP	KIRP	PRAD	PRAD	BRCA	BRCA	BRCA	BRCA	MESO	LGG	KIRP	KICH	SARC	PRAD
K6372-2	prostata	PRAD	KIRP	KIRP	KIRP	PRAD	PRAD	BRCA	BRCA	BRCA	SKCM	MESO	LGG	KIRP	BRCA	PRAD	PRAD
acc. [%]			14.29	14.29	09.52	52.38	42.86	14.29	14.29	14.29	28.57	0	14.29	14.29	14.29	14.29	47.62

M1 *model_FFNN_RNAseq_19*
M2 *model_CNN_RNAseq_19*
M3 *model_DEEP_RNAseq_19*
M4 *model_CNN_RNAseq_33_mapped*
M5 *model_CNN_RNAseq_33_balanced*

In general, the classification with a CNN trained on 33 classes yielded the highest classification results of up to 52.38%. NN models with a reduced number of classes did not perform well (14.29%). Especially the DLNN model disapointed with only 09.52% classification accuracy of MU Graz CUP samples. *ratio* and *voom* transformations did not increase classification accuracy. A detailed depiction of the classification probabilities for each class per sample can be found in the files *rpm_MUG_classification.txt*, *ratio_MUG_classification.txt* and *voom_MUG_classification.txt* for RPM, ratio- and voom-transformed MU RNA-seq CUP data respectively.

3.4.1 MU RNA-seq data classification results with balanced classes

As seen in chapter 3.3 Classification results, with an accuracy of 89.90% and loss of 0.43 the CNN model using balanced classes did not yield a higher classification accuracy during training and validation (Table 3.6).

A NN model *model_FFNN_RNAseq_33_balanced* based on the *RNAseq_TCGA_33* data set, as it achieved the highest accuracy when using unbalanced data sets, was generated. The model uses balanced TCGA RNA-seq training as shown in data to classify MU Graz CUP samples as RPM, ratio- and voom transformed data. The resulting classifications were compared to the original classifications of `model_FFNN_RNAseq_33_mapped` (Table 3.8). The classifications of MU RNA-seq data transformations do not agree with each other. Further, the balancing of data sets did not increase classification accuracy (47.62% opposed to 52.38% accuracy). A detailed depiction of the classification probabilities for each MU RNA-seq class per sample can be found in the files *rpm_MUG_classification_balanced.txt*, *ratio_MUG_classification_balanced.txt* and *voom_MUG_classification_balanced.txt* for RPM, ratio- and voom-transformed MU RNA-seq CUP data respectively.

4 Discussion

With the aim to investigate the applicability of different SVMs and neural networks on MU Graz RNA-seq CUP data, NNs trained on TCGA RNA-seq data were generated as well as SVM models using raw intensity MA data and TCGA RNA-seq data for comparison. Testing and optimization of various NN models has led to a CNN classifying 21 MU Graz CUP RNA-seq samples into possible 33 classes.
In order to compare SVM and neural network classification accuracy of microarray and RNA-seq data we had to adapt and extend the existing script for the analysis using the raw fluorescence intensity data values from the microarrays. This contained the development of an algorithm to identify the required columns from the corrupted microarray data files.
33 TCGA RNA-seq projects were downloaded and in over the course of this thesis combined to three different datasets (with 12, 19 and 33 classes). Smaller data sets allowed for shorter training durations, which allowed a more streamlined optimization process. In order to prevent misclassifications, only classes with unique primaries were used during testing. These prototypes lead to the final neural network models using the whole TCGA-RNA-seq data set of 33 classes.
Microarray and RNA-seq datasets were used to train and validate SVM and neural network models followed by an assessment of the performance of the SVMs and NN in the classification of RNA-seq CUP samples provided by the MU Graz.
Further the impact of neural network structure and deep learning, as well as the influence of hyper parameter on classification accuracy were studied. A comparison between results provides information about the applicability of different neural networks on RNA-seq data and misclassifications.
A literature search on the influence on unbalanced group size in the training set was performed as well as a respective analysis of the influence of balancing the training set using a specific balancing approach.

4.1 Feature selection and mapping

Feature extraction or reduction methods reduce the dataset to a more manageable size, omitting features which do not contribute to the quality of a model. This for example, reduces training time or reduces the risk of over-fitting the model. As they often improve performance, these methods have become cornerstones of SVM generation and one of the first important steps while developing a model [84]. NN do not require feature reduction methods. Depending on the application, it may be desirable to use the whole data set for training. It does, however reduce training time and if features do not contribute to the final model, reduction methods may present handy. The number of parameters calculated by a NN is directly dependend on the number of features used to train it. A higher amount of features means more calculated parameters and a higher training time of a model.
In order to reduce the number of features in the data sets, an approach based on a signal to noise ratio (SNR) was used [5]. There are other approaches of feature selection for this application, but the SNR feature extraction method was used to provide a comparable classification workflow to Tothill *et al.* [5].
Other methods to avoid overfitting could be cross-validation or stepwise regression for data sets with a low number of features, which is not applicable for TCGA RNA-seq data with 20502 features. Other regularization and overfitting methods are applied to reduce the number for applications with a huge amount of features. Urda *et al.* used regularization methods L1 (Least Absolute Shrinkage and Selection Operator - LASSO) and L2 (RIDGE) to distill the dataset to the most important features [85]. LASSO and RIDGE shrink the less important feature coefficients to zero, removing some features altogether. Penalty terms are the *absolute value of the magnitude of coefficients* and the *squared magnitude of coefficients* respectively. Urda *et al.* used regularization in a LASSO model and compared it to deep learning models in classifying data of three different TCGA RNA-seq classes to study their suitability for analysis of high-throughput sequencing data. They indicate an application of deep learning may not outperform simpler LASSO models [85]. L1 and L2 regularization performs well for feature selection in case of a huge number of features like RNA-seq data [86]. In general neural networks have to use regularization and overfitting methods like L1/L2 regularization, feature reduction or a dropout strategy. When using a dropout strategy randomly selected neurons are ignored during training effectively reducing the total number of features used and

can result in multiple independent internal representations being learned by the network, preventing overfitting and the slow nature of such networks [87].
With 33 classes, over 20000 features and a different scope in this thesis, we chose an approach similar to Tothill *et al.* who showed that a calculation of the absolute value of signal to noise ratio (SNR) selects features which can result in high accuracies in training SVM models, on data sets with a high amount of different unbalanced classes [5].
Another level of feature selection is the mapping of features between datasets is needed to ensure data integrity as different sources are used. During mapping, features are lost because not all features are present in the MU Graz RNA-seq data files. This means every time a model is trained to classify MU Graz CUP samples, a part of potentially representative features are lost which could result in reduced classification accuracy.
Each data set (microarray data, TCGA RNA-seq data and MU Graz CUP samples) uses a different gene annotation standard. As these standards tend to vary and change over time when new discoveries are made, the mapping of genes for classification purposes could have led to some errors and excluded genes. The impact of mapping features on classification accuracy were minimal during training and validation: A model trained on *RNAseq_TCGA_19* data reached 96.75% accuracy and a respective mapped model using *RNAseq_TCGA_19_mapped* data reached 96.93% accuracy. A small increase in accuracy through mapping was reached during this NN training cicle. However, this was not always the case when repeating the training and validation process. The mapping of features seems to have little impact on training and validation of NN. A total of 68 features were lost during the mapping process resulting in a 0.49% loss in validation accuracy. NN trained on TCGA RNA-seq data using 33 classes, were only trained on mapped data sets due to the small impact of feature mapping on validation accuracy.

4.2 Influence of network structure

Research about this topic has shown that for gene expression data a single hidden layer may produce more accurate results than a full fledged deep learning model [68, 85]. Most neural networks are quite flexible and can reach a prediction even when using the wrong network type for the data [82].

Not every neural network model can be used to classify any data set. When to use MLPs, CNNs or RNNs depends on the respective use. While most neural networks are flexible enough to make a prediction even under not ideal circumstances, the optimal performance is reached when choosing the right model for a task.
MLPs are most of all comparable to SVMs where labels are assigned to a set of input data. They are suitable for regression and prediction problems and best used on tabular data. That is why the first prototypes for classification on RNA-seq data was done with a simple FFNN [82]. The size of the used datasets was increased from 12 to using 19 unique primaries and finally all 33 available classes from TCGA. Not every project on TCGA is assigned a single primary. This has a negative, if small, impact on classification accuracy and it was tried to counter this effect while using data set with 19 unambiguous primary classes.After generating a functioning model model_CNN_RNAseq_33, different NN structures were tested to optimize classification accuracy.
A CNN is mostly used on pictures and time series data but also gene expression data [88]. The CNN learns about the input structure and applies this information in the classification process. This is not intuitively applicable for the classification based on gene expression data, but nevertheless has led to good classification performance not only in this thesis, but also in other cases where tabular data was reshaped in a four dimensional tensor format [27, 68].
RNNs were not tested as they are not really used on tabular data nor image data and result in a bad performance for the both of them [82].
In this thesis, the use of CNNs has shown an improvement over the other used types of neural network and SVMs and resulted in a very high classification accuracy of 98.50%, which is higher than the accuracy of FFNN *model_FFNN_RNAseq_19* (97.42%), SVM *model_SVM_RNAseq_33* (97.80%) and Tothill *et al.* SVM version *model_SVM_MA_13_processed* (84.60%).

4.3 Classification of TCGA RNA-seq data using SVMs and NNs

The classification of TCGA RNA-seq data with SVM models resulted in quite a satisfactory accuracy. The training and validation of SVMs with both data types

provided similar accuracies during validation using randomly subsetted validation sets. The training with RNA-seq data took longer because up to 50 times more samples were used. While training a SVMs with microarray data took about a second, using TCGA RNA-seq sample training times span between 66 to 1231 seconds (Table 3.5). Training of neural networks took between 10 and 7700 seconds. Using raw intensity data does not significantly improve or worsen classification accuracy of SVMs. It could be concluded preprocessed Tothill *et al.* data was log_2-transformed, as the accuracy seems to be nearly identical.

4.4 Assessment of NN classification performance

The classification of microarray data yielded comparable results to Tothill *et al.* (between 86.97% to 93.49%) but with a higher accuracy in the classification of their microarray CUP samples. Tothill *et al.* states a classification accuracy of 84.6% [5] which was exactly recreated with models based on *MA_Tothill_13_processed*, *MA_Tothill_13_mapped* and *MA_Tothill_13_raw_log2* datasets for SVM classification (Table 3.5) while using NNs to classify *MA_Tothill_13_processed* data 92.31% accuracy was reached (Table 3.6). *MA_Tothill_13_processed* data provides 229 samples with 607 features, *MA_Tothill_13_mapped* data provides 229 samples with 541 features and the data set *MA_Tothill_13_raw_log2* data provides 229 samples with 504 features produced through SNR feature selection which are classified by an SVM and a FFNN using a single hidden layer, the following output shape (None/256/128/139) and a total of 190221 trainable parameters.
The classification of RNA–seq data yielded also comparable results between the two methods while CNN classification provided the highest validation classification accuracy at the cost of a higher training time (FFNN: 97.42% in 33 seconds and CNN: 98.50% in 4020 seconds).
A number of misclassifications occured. Misclassifications were rather unevenly spread over all classes, not suggesting a specific cause. This suggests maybe some of the representative genetic features are not occuring enough to allow a strict differentiation between classes or are representated in both. Further, not all TCGA-RNA-seq projects are assigned a unique primary which could also be the cause of misclassifications. The NN model *model_CNN_RNAseq_19* and *model_CNN_RNAseq_33* did present a classification accuracy of 98.32% and 93.88% respectively (Table 3.6. Model

model_CNN_RNAseq_19 represents a CNN consisting of 2 hidden layers (one convolutional and one dense layer), classifying datasets *RNAseq_TCGA_19_mapped* with 8314 samples of 834 features and *model_CNN_RNAseq_33* with 10464 samples of 1341 features provided through SNR feature selection. This suggests that unambiguous classes may further decrease the number of misclassifications. However, NN may also learn valuable properties of the training data set through overlapping classes. Further, if a model is used to classify new data, a sample could belong to more than one class. Smaller data sets combining less different class possibilities result in a higher classification accuracy. This behaviour can be seen when comparing the 3 different RNA-seq data sets.
Transformation of the MU RNA-seq data did not improve accuracy (classification accuracy 14.29%). Deep learning did not seem to have any effect on classification accuracy of MU Graz CUP RNA-seq samples (accuracy 9.52%). SVMs trained on Tothill MA data achieved a MU CUP classification accuracy of 4.76%. The highest accuracy was reached when using a large amount of classes with unbalanced sizes (52.38% accuracy) using model *model_CNN_RNAseq_33_mapped*. CNN do seem to yield the best results in comparison to FFNN and DLNN. Classification accuracy could be further improved when using classes with more samples in general. Balancing the data set did not to improve results (Table 3.8).

While Tothill *et al.* describes a validation accuracy of 89%, a CNN outperforms this clearly with an reached accuracy of 98.5%.
Literature on the topic of classifying RNA-seq data with the help of NN is rare at the moment of writing. Mostavi *et al.* which documents 93.9% - 95.0% accuracy using 1D-CNN, 2D-Vanilla-CNN and 2D-Hybrid-CNN models with similar number of neurons and layers as CNN models proposed in this thesis, while taking into account the effect of the tissue of origin to reduce potential bias in the identification of cancer markers. To take into account the impact of tissue of origin a new label consisting of samples without cancer, regardless of their original tissue type designation is created [68]. Their data sets consists of 10,340 samples and 34 classes (33 TCGA classes and tissue of origin) using TCGA RNA-seq data.
Urda *et al.* studied the influence of different network structures on small TCGA RNA-seq data sets with 1 to 3 classes. While the area under the curve (AUC) does not measure accuracy, it measures how well a model performs. The receiver operating characteristic is a probability curve (plotting true positive rate (TPR) on the

y-axis and false positive rate (FPR) on the x-axis) with its AUC being an indicator how well the model can distinguish between classes. As it measures probability the AUC ranges between 0 and 1, while 1 is the maximum [89]. They report an AUC between between 0.57 AUC - 0.77 AUC while testing on different activation functions, number of hidden layers and strategies to avoid overfitting [85].
Zhang *et al.* used RNA-seq data from the UCSC Xena data portal and created a neural network classifier reaching 97.49% validation accuracy with the data set consisting of 11,538 samples and 450,804 molecular features, which were preprocessed and then used to train their OmiVAE model. They used principal component analysis (PCA) and sparse autoencoder to select used features for training (35,877 selected multi-omics molecular features). This model is a deep machine learning model with a training procedure divided in an unsupervised phase, where a hierachical cluster of samples can be formed without the need of labels and a supervised phase whith 10-fold cross validation among 33 TCGA RNA-seq tumor classes [90].

4.5 Influence of hyper parameter settings on classification performance of NN

The selection of different hyper-parameters has shown a huge influence on neural network performance. As stated in literature, a deep learning approach did not yield a satisfactory accuracy for classification of RNA-seq data [68, 85]. Rather FFNNs with a single hidden layer and CNNs provided high accuracies. While CNNs provided the highest accuracy and smallest loss, they did take a lot more time to train. Simple FFNN were rather quick to train and even outperformed SVMs in accuracy and in needed training time.
The activation function *relu* performed well with microarray data, but when training models on RNA-seq data only accuracies of 4.30% to 26.30% could be reached. Other activation functions performed better with *softsign* being the clear favourite for this application.
A dropout strategy after each layer of the neural network was used in order to prevent overfitting. Further measures to handle uneven class sizes were not applied

up until the writing of this thesis, but matter for correct classification and prediction of new data as discussed later in this chapter.

4.6 Classification of MU Graz samples

When applying a 33 class CNN model (*model_CNN_RNAseq_33_mapped*) on CUP samples provided by the MU Graz, no definitive classification results were achieved. This could be due to class sizes as samples belonging to large classes were classified correctly more often. Necessary class sizes could also be dependend on the cancer type as training and validation across models has shown no misclassifications in small classes like *ACC* and *UVM* using CNN.
TCGA RNA-seq data and MU Graz CUP RNA-seq data was generated through completely different wetlab procedures. While no information about TCGA library preparation methods was found, their sequencing pipelines differ. MU Graz RNA-seq data does not contain all exons after mRNA sequencing. These differences during data generation could also have implications and effects on classification performance.
In order to improve the low classification accuracy of MU Graz RNA-seq data when using SVMs the raw mircoarray intensity data provided by Tothill *et al.* [5] was downloaded. Problems with the initial classification process might occur because microarray data represent ratios of two dyes (for a sample and a reference) present on each position of DNA sequence on the MA. MA data MA_Tothill_13 represents log_2 fold-change of the test channel divided by fluorescent intensity from control channel. RNA-seq data is represented as count data like reads per million mapped reads (RPM) or fragments per kilobase million (FPKM) and others:

$$RPM = \frac{\text{number of reads mapped to gene} * 10^6}{\text{total number of mapped reads}} \tag{4.1}$$

$$FPKM = \frac{\text{number of reads mapped to gene} * 10^3 * 10^6}{\text{total number of mapped reads} * \text{gene length in bp}} \tag{4.2}$$

Microarray data and RNA-seq data do not provide the same range of possible values which could lead to complications. The range of genes detected by MAs is dependend on the chip used while whith this procedure even small amounts of genes can be detected. RNA-seq can detect a wider range of features depending on

the sequencing processes involved.
However, the provided raw data files were seemingly corrupted. The column names did not match the content at all. In order to still extract the required information to create an SVM using raw fluorescence intensity data an approach for aligning the column names with their respective data had to be created.
To compare different classification performances, various transformations of the raw intensities were performed. MA data in the form of raw intensities, background corrected intensities, range transformed background corrected intensities and log_2-transformed intensities were used. The classification performance did not improve by using raw fluorescence intensity data instead of preprocessed data [5]. The same accuracy as preprocessed data files was reached when using log_2-transformed data probably because the preprocessed samples were prepared in the same way. Other approaches all yielded worse classification performances with the worst in using raw intensities. Classification accuracy of Tothill *et al.* CUP samples did worsen significantly, while log_2-transformed raw intensities resulted in a model with the same accuracy as Tothill *et al.* (Table 3.5). None of the MA SVM models were able to classify MU Graz CUP samples in a meaningful way as the highest accuracy was 4.76% (Table 3.7).

In general, the use of CNN seems to be the most promising for classifying CUP samples. While the accuracy of NN, when classifying data used during training and validation, is high and slightly outperforms comparable SVMs, the classification results of new data are not convincing. The accuracy of a NN model is increased with the amount of training data available. The proposed CNNs might perform better with more samples to substitude smaller TCGA classes. Different model tuning procedures and other choices for hyper parameters could also increase classification accuracy. In future, with more experiments, the ever increasing range of applications for NN and more comprehensive RNA-seq data libraries for training, generally applicable NN for CUP classification may be possible yet.

Bibliography

[1] Duraisamy V, Ramakrishnan SS and Vidhya R: **Mapping invasive plant prosopis juliflora in arid land using high resolution remote sensing data and biophysical parameters.** *Indian Journal of Geo-Marine Sciences* 2017. 46(6):1135–1144.

[2] Adeyemi T, Grove I, Peets S and Domun Y: **Dynamic Neural Network Modelling of Soil Moisture Content for Predictive Irrigation Scheduling.** *Sensors (Basel)* 2018. 18(10):3408.

[3] Saha S: **A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way**. `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53`, 19.08.2019.

[4] Verma S: **Understanding Input Output shapes in Convolution Neural Network — Keras**. `https://towardsdatascience.com/understanding-input-and-output-shapes-in-convolution-network-keras-f143923d56ca`, 03.03.2020.

[5] Tothill RW, Kowalczyk A, Rischin D, Bousioutas A, Haviv I and van Laar RK: **An expression-based site of origin diagnostic method designed for clinical application to cancer of unknown origin.** *Cancer Research.* 2005. 65(10):4031–40.

[6] Grossman RL, Heath AP, Ferretti V, Varmus HE, Lowy DR, Kibbe WA *et al.*: **Toward a Shared Vision for Cancer Genomic Data.** *New England Journal of Medicine* 2016. 375(12):1109–1112.

[7] Tothill RW, Shi F, Paiman L, Bedo J, Kowalczyk A, Mileshkin L *et al.*: **Development and validation of a gene expression tumour classifier for cancer of unknown primary.** *Pathology* 2015. 47(1):7–12.

[8] Tothill RW, Li AJ, Mileshkin L, Doig K, Siganakis T, Cowin P *et al.*: **Massively-parallel sequencing assists the diagnosis and guided treatment of cancers of unknown primary.** *The Journal of Pathology* 2013. 231(4):413–423.

[9] Ramaswamy S, Tamayo P, Rifkin R, Mukherjee S, Yeang CH, Angelo M *et al.*: **Multiclass cancer diagnosis using tumor gene expression signatures.** *Proceedings of the National Academy of Sciences USA* 2001. 98(26):15149–15154.

[10] R Core Team: **R: A language and environment for statistical computing**. R Foundation for Statistical Computing, Vienna, Austria - `https://www.R-project.org/`, 17.12.2018.

[11] GEO - NCBI: **Gene Expression**. https://www.ncbi.nlm.nih.gov/probe/docs/applexpression/, 03.03.2020.

[12] Pollack JR, Perou CM, Alizadeh AA, Eisen MB, Pergamenschikov A, Williams CF *et al.*: **Genome-wide analysis of DNA copy-number changes using cDNA microarrays.** *Nature Genetics* 1999. 23(1):41–6.

[13] Babu MM: **Computational Genomics**, chapter **Introduction to microarray data analysis**, pages 225–249. Horizon Press, U.K., Cambridge, UK, 2004.

[14] Wang Z, Gerstein M and Snyder M: **RNA-seq: a revolutionary tool for transcriptomics.** *Nature Reviews Genetics* 2009. 10(1):57–63.

[15] Kukurba KR and Montgomery SB: **RNA Sequencing and Analysis.** *Cold Spring Harb Protocols* 2015. 2015(11):951–969.

[16] Ozsolak F and Milos PM: **RNA sequencing: advances, challenges and opportunities.** *Nature Reviews Genetics* 2011. 12(2):87–98.

[17] Rai MF, Tycksen ED, Sandell LJ and Brophy RH: **Advantages of RNA-seq compared to RNA microarrays for transcriptome profiling of anterior cruciate ligament tears.** *Journal of Orthopaedic Research* 2017. 36(1):484–497.

[18] Han Y, Gao S, Muegge K, Zhang W and Zhou B: **Advanced Applications of RNA Sequencing and Challenges.** *Bioinformatics and Biology Insights* 2015. 9(Suppl 1):29–46.

[19] Zhao S, Zhang Y, Gamini R, Zhang B and von Schack D: **Evaluation of two main RNA-seq approaches for gene quantification in clinical RNA sequencing: polyA+ selection versus rRNA depletion**. *Scientific Reports volume 8* 2018. 4781(2018):2045–2322.

[20] Okayama H and Berg P: **High-efficiency cloning of full-length cDNA.** *Molecular and Cellular Biology* 1982. 2(2):161–170.

[21] Gublera U and Hoffman BJ: **A simple and very efficient method for generating cDNA libraries.** *Gene* 1983. 25(2-3):263–269.

[22] Griffith M, Walker JR, Spies NC, Ainscough BJ and Griffith OL: **Informatics for RNA Sequencing: A Web Resource for Analysis on the Cloud.** *PLoS Computational Biology* 2015. 11(8):e1004393.

[23] Langmead B, Trapnell C, Pop M and Salzberg SL: **Ultrafast and memory-efficient alignment of short DNA sequences to the human genome**. *Genome Biology* 2009. 10(3):R25.

[24] Babu MM: **RNA-Seq: a revolutionary tool for transcriptomics.** *Nature Reviews Genetics* 2009. 10(1):57–63.

[25] Cortes C and Vapnik V: **Support-Vector Networks.** *Machine Learning* 1995. 20(3):273–297.

[26] Nicholson C: **A Beginner's Guide to Neural Networks and Deep Learning**. https://pathmind.com/wiki/neural-network, 19.08.2019.

[27] Nicholson C: **A Beginner's Guide to Convolutional Neural Networks (CNNs)**. https://pathmind.com/wiki/convolutional-network, 19.08.2019.

[28] Powell V: **Image Kernels - Explained Visually**. http://setosa.io/ev/image-kernels/, 19.01.2020.

[29] NIH - National Cancer Institute: **The Cancer Genome Atlas TCGA**. https://cancergenome.nih.gov/, 17.12.2018.

[30] Tothill RW, Kowalczyk A, Rischin D, Bousioutas A, Haviv I and van Laar RK: **E-MEXP-113 - Transcription profiling of multiple human tumour specimens of different anatomical origin arrayed against a common reference**. 03.03.2020. URL https://www.ebi.ac.uk/arrayexpress/experiments/E-MEXP-113/.

[31] Molecular Devices Corporation: **Gene Pix Version 4.1 File Format**. Archived from [www.axon.com/GN_GenePix_File_Formats.html]-http://web.archive.org/web/20040604080449/http://www.axon.com/gn_GenePix_File_Formats.html, 11.01.2019.

[32] NIH - National Cancer Institute: **H001 Standard Operating Procedure (SOP) for Hematoxylin and Eosin (H&E) Staining Coverslipping**. https://brd.nci.nih.gov/brd/sop/show/1421, 03.03.2020.

[33] NIH - National Cancer Institute: **H005 Standard Operating Procedure (SOP) for Automated Wright/Giemsa Staining for Leukemia Slides**. https://brd.nci.nih.gov/brd/sop/show/1441, 03.03.2020.

[34] NIH - National Cancer Institute: **H006 Standard Operating Procedure (SOP) for Sectioning Frozen Tissue Samples in Histology**. https://brd.nci.nih.gov/brd/sop/show/1442, 03.03.2020.

[35] NIH - National Cancer Institute: **H013 Standard Operating Procedure (SOP) for Paraffin Slides and Scrolls**. https://brd.nci.nih.gov/brd/sop/show/1443, 03.03.2020.

[36] NIH - National Cancer Institute: **H014 Standard Operating Procedure (SOP) for Scraping Slides for Extraction**. https://brd.nci.nih.gov/brd/sop/show/1444, 03.03.2020.

[37] NIH - National Cancer Institute: **L016 Standard Operating Procedure (SOP) For Preparing Frozen Tissue For Molecular Analysis**. https://brd.nci.nih.gov/brd/sop/show/1445, 03.03.2020.

[38] NIH - National Cancer Institute: **L017 Standard Operating Procedure (SOP) For Preparation Of Tissue Samples For Proteome Characterization Centers**. https://brd.nci.nih.gov/brd/sop/show/1446, 03.03.2020.

[39] NIH - National Cancer Institute: **L020 Standard Operating Procedure (SOP) for Sectioning and Portioning Frozen Tissue Samples in Logistics**. https://brd.nci.nih.gov/brd/sop/show/1447, 03.03.2020.

[40] NIH - National Cancer Institute: **M001 Standard Operating Procedure (SOP) for DNA/RNA Extraction with Allprep (DNA) and MirVana (Total RNA with Small RNA)**. https://brd.nci.nih.gov/brd/sop/show/1450, 03.03.2020.

[41] NIH - National Cancer Institute: **M019 Standard Operating Procedure (SOP) FOR DNA/RNA Extraction with AllPrep (DNA) and mirVana (Total RNA with small RNA) 3 Column-Modified Melanoma Protocol**. https://brd.nci.nih.gov/brd/sop/show/1458, 03.03.2020.

[42] NIH - National Cancer Institute: **M022 Standard Operating Procedure (SOP) For DNA/RNA Extraction With Allprep (DNA) And MirVana (Total RNA With Small RNA) - Modified Leukemia Protocol**. https://brd.nci.nih.gov/brd/sop/show/1460, 03.03.2020.

[43] NIH - National Cancer Institute: **M027 Standard Operating Procedure (SOP) for FFPE RNA/DNA Co-Isolation Using the Allprep FFPE Kit and the Highpure miRNA Kits**. https://brd.nci.nih.gov/brd/sop/show/1461, 03.03.2020.

[44] TCGA: **Biospecimen Research Database**. https://brd.nci.nih.gov/brd/sop-compendium/show/701, 03.03.2020.

[45] Dobin A, Davis CA, Schlesinger F, Drenkow J, Zaleski C, Jha S *et al.*: **STAR: ultrafast universal RNA-seq aligner**. *Bioinformatics.* 2013. 29(1):15–21.

[46] Babraham Bioinformatics: **FastQC**. http://www.bioinformatics.babraham.ac.uk/projects/fastqc/, 03.03.2020.

[47] Broad Institute: **Picard**. http://broadinstitute.github.io/picard/, 03.03.2020.

[48] NIH - National Cancer Institute - Genomic Data Commons: **GDC Pipeline Overviews**. 03.03.2020. URL https://gdc.cancer.gov/about-data/data-harmonization-and-generation/genomic-data-harmonization-0#Pipelines.

[49] GDC Documentation: **RNA-Seq Alignment Workflow**. 03.03.2020. URL https://docs.gdc.cancer.gov/Data/Bioinformatics_Pipelines/Expression_mRNA_Pipeline/#rna-seq-alignment-workflow.

[50] Zhu Y, Qiu P and Ji Y: **TCGA-assembler: open-source software for retrieving and processing TCGA data.** *Nature Methods* 2014. 11(6):599–600.

[51] Thermo Fischer: **Targeted Transcriptome Sequencing for Human and Mouse by Ion Torrent Next-Generation Sequencing**. https://www.thermofisher.com/at/en/home/life-science/sequencing/next-generation-sequencing/ion-torrent-next-generation-sequencing-workflow/ion-torrent-next-generation-sequencing-select-targets/ampliseq-target-selection/ampliseq-transcriptome.html, 03.03.2020.

[52] Meyer D, Dimitriadou E, Hornik K, Weingessel A, Leisch F, Chang CC *et al.*: **e1071: Misc Functions of the Department of Statistics, Probability Theory Group**. https://CRAN.R-project.org/package=e1071, 03.03.2020.

[53] Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W *et al.*: **limma powers differential expression analyses for RNA-sequencing and microarray studies.** *Nucleic Acids Research* 2015. 43(7):e47.

[54] Wickham H, Hester J, Chang W, RStudio and R Core Team: **devtools: Tools to Make Developing R Packages Easier**. https://CRAN.R-project.org/package=devtools, 03.03.2020.

[55] Falbel D, Allaire J, Chollet F, RStudio, Google, Tang Y *et al.*: **keras: R Interface to 'Keras'**. https://CRAN.R-project.org/package=keras, 03.03.2020.

[56] Arnold T: **kerasR: R Interface to the Keras Deep Learning Library**. https://CRAN.R-project.org/package=kerasR, 03.03.2020.

[57] Waldron L and Riester M: **HGNChelper: Identify and Correct Invalid HGNC Human Gene Symbols and MGI Mouse Gene Symbols**. https://CRAN.R-project.org/package=HGNChelper, 03.03.2020.

[58] Wickham H and RStudio: **httr: Tools for Working with URLs and HTTP**. https://CRAN.R-project.org/package=httr, 03.03.2020.

[59] CRAN Team and Lang DT: **RCurl: General Network (HTTP/FTP/...) Client Interface for R**. https://CRAN.R-project.org/package=RCurl, 03.03.2020.

[60] Couture-Beil A: **rjson: JSON for R**. https://CRAN.R-project.org/package=rjson, 03.03.2020.

[61] Wickham H and RStudio: **stringr: Simple, Consistent Wrappers for Common String Operations**. https://CRAN.R-project.org/package=stringr, 03.03.2020.

[62] RStudio Team: **RStudio: Integrated Development Environment for R**. RStudio, Inc., Boston, MA, 2016. URL http://www.rstudio.com/.

[63] Chollet F: **Keras**. https://keras.io, 2015.

[64] Chollet F, Allaire JJ, Falbel D, RStudio, Tang GY *et al.*: **R interface to keras**. https://github.com/rstudio/keras, 2017.

[65] R Core Team: **R-Documentation for Factors**. https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/factor, 26.01.2020.

[66] Kohavi R: **A study of cross-validation and bootstrap for accuracy estimation and model selection**. Morgan Kaufmann, 1995 pages 1137–1143.

[67] Meyer D: **Support Vector Machines**. https://www.rdocumentation.org/packages/e1071/versions/1.7-3/topics/svm, 03.03.2020.

[68] Mostavi M, Chiu YC, Huang Y and Chen Y: **Convolutional neural network models for cancer type prediction based on gene expression**. *ArXiv* 2019. abs/1906.07794:1–34.

[69] Allaire JJ: **fit function in keras**. `https://www.rdocumentation.org/packages/keras/versions/2.0.5/topics/fit`, 02.02.2019.

[70] Peltarion AB: **Categorical crossentropy**. `https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy`, 03.03.2020.

[71] Raúl Gombru: **Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names**. `https://gombru.github.io/2018/05/23/cross_entropy_loss/`, 03.03.2020.

[72] Chollet F, Vijay P, Balakrishnan R, kuza55, Thomé C *et al.*: **/metrics.py**. `https://github.com/keras-team/keras/blob/master/keras/metrics.py`, 03.03.2020.

[73] Jung KM: **Mathematical Problems in Engineering**, chapter **Support Vector Machines for Unbalanced Multicategory Classification**, pages 1–7. Department of Statistics and Computer Science, Kunsan National University, Gunsan 573-701, Republic of Korea, vol. 2015 edition, 2015.

[74] Guzman L: **Data sampling improvement by developing SMOTE technique in SAS**. urlhttps://support.sas.com/resources/papers/proceedings15/3483-2015.pdf, 11.01.2019.

[75] Pavlov DY, Gorodilov A and Brunk CA: **BagBoo: a scalable hybrid bagging-the-boosting model.** 2010 pages 1897–1900. doi:10.1145/1871437.1871758.

[76] Hanifah FS, Wijayanto H and Kurnia A: **SMOTEBagging Algorithm for Imbalanced Dataset in Logistic Regression Analysis**. *Applied Mathematical Sciences* 2015. 9(138):6857–6865.

[77] Galar M, Fernández A, Barrenechea E, Sola HB and Herrera F: **A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches**. *IEEE Transactions on Systems Man and Cybernetics Part C (Applications and Reviews)* 2012. 42(4):463–484.

[78] Sunil R: **Improve Your Model Performance using Cross Validation (in Python and R)**. `https://www.analyticsvidhya.com/blog/2018/05/improve-model-performance-cross-validation-in-python-r/`, 11.01.2019.

[79] Schneider J: **Cross Validation.** `http://www.cs.cmu.edu/~schneide/tut5/node42.html`, 11.01.2019.

[80] Tetko IV, Livingstone DJ and Luik AI: **Neural network studies. 1. Comparison of overfitting and overtraining.** *Journal of Chemical Information and Computer Sciences* 1995. 35(5):826–833.

[81] Cawley GC and Talbot NLC: **On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation.** *Journal of Machine Learning Research* 2010. 11(Jul):2079–2107.

[82] Brownlee J: **When to Use MLP, CNN, and RNN Neural Networks**. https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/, 19.08.2019.

[83] Smyth G, Hu Y, Ritchie M, Silver J, Wettenhall J, McCarthy D *et al.*: **Transform RNA-Seq Data Ready For Linear Modelling**. https://www.rdocumentation.org/packages/limma/versions/3.28.14/topics/voom, 17.12.2018.

[84] Cao LJ and Chong WK: **Feature extraction in support vector machine: a comparison of PCA, XPCA and ICA**. 2002 pages 1001 – 1005 vol.2. doi:10.1109/ICONIP.2002.1198211.

[85] Urda D, Montes-Torres J, Moreno F, Franco L and Jerez JM: **Deep learning to analyze rna-seq gene expression data.** ISBN 978-3-319-59146-9, 2017 pages 50–59.

[86] Nagpal A: **L1 and L2 Regularization Methods**. https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c, 13.10.2017.

[87] Srivastava N, Hinton G, Krizhevsky A, Sutskever I and Salakhutdinov R: **Dropout: A Simple Way to Prevent Neural Networks from Overfitting**. *Journal of Machine Learning Research* 2014. 15(56):1929–1958.

[88] Zhuang Z, Shen X and Pan W: **A simple convolutional neural network for prediction of enhancer-promoter interactions with DNA sequence data.** *Bioinformatics* 2019. 35(17):2899–2906.

[89] Fawcett T: **Introduction to ROC analysis**. *Pattern Recognition Letters* 2006. 27(8):861–874.

[90] Zhang X, Zhang J, Suna K, Xian Yang CD and Guo Y: **Integrated multi-omics analysis using variational autoencoders: Application to pan-cancer classification**. 2019 pages 1–7. doi:10.1109/BIBM47256.2019.8983228.

Appendix

R-script - correlation of variables in order to calculate RoM:

```
#library(ArrayExpress)
#install.packages("e1071")
#source("https://bioconductor.org/biocLite.R")
#biocLite("limma")

library(e1071)
library(limma)

setwd("H:/Master Project/CUP III")
rm(list = ls())

# ------------------------------------------------------------------------------
# Some functions
#

writeLog <- function(textstr)
{
  if (class(textstr) == "character") {
    output_string <- textstr
    cat(paste(output_string, "\n", sep=""))
  } else {
    output_string <- print(textstr)
  }
  flush.console()
  return(output_string)
}

readRawData <- function(fname)
{
  scan.names <- read.table(fname, header=TRUE, sep="\t", stringsAsFactors=FALSE,
                           row.names=NULL)
  return(scan.names)

}
readAllRawData <- function(adir, fpattern, file_nr)
{
  afiles <- dir(adir, pattern=fpattern, full.names=TRUE)
  nfiles <- length(afiles)
  writeLog(sprintf("Reading processed data from %d microarray data files",
           nfiles))

  array_data <- readRawData(afiles[file_nr])

  return(array_data)
}

# ------------------------------------------------------------------------------
# load and prepare data for w data files
#

for (w in 3:10) {

```

```
sample_raw_data <- readAllRawData(adir="data/Tothill/Raw_Data/",
                                  fpattern="E-MEXP-113-raw-data-.*.txt",w)

# get column with ratio of means
ratio = as.numeric(sample_raw_data$Software.Unknown.Normalize);

# prepare raw data, rmove uneeded cols
rownames(sample_raw_data) = as.character(gsub('R:A-MEXP-28:','',
                                        sample_raw_data$Reporter.identifier));
sample_calc_data = sample_raw_data[,-(1:6)];
sample_calc_data[,(1:2)]= NULL;
sample_calc_data[,10]= NULL;
sample_calc_data[,18:20]= NULL;
sample_calc_data[,27]= NULL;
sample_calc_data[,35]= NULL;


# ------------------------------------------------------------------------------
# Start calculation
#
# Ratio of Means = the ratio of the arithmetic mean intensities of each feature
# for each wavelength, with the median background subtracted.

results <- data.frame(matrix(ncol = length(sample_calc_data[1,]), nrow =
                                          length(sample_calc_data[,1])));
rownames(results) = as.character(gsub('R:A-MEXP-28:','',
                                  sample_raw_data$Reporter.identifier));
colnames(results) = colnames(sample_calc_data);

for (i in 1:1000) { # rows
  current_ratio = 2^(as.numeric(ratio[i])); # ratio of current row
  # 2 ^ to transform current ratio back from log 2
  sample_calc_data_vec = as.numeric(sample_calc_data[i,])
  if(!is.na(current_ratio)) {
    for (j in 1:length(sample_calc_data_vec)) { # col
      median_B635 = as.numeric(sample_calc_data_vec[j]); #  current median_F635

      for (k in 1:length(sample_calc_data_vec)) {
        mean_F532 = as.numeric(sample_calc_data_vec[k]); # current mean_F532

        for (l in 1:length(sample_calc_data_vec)) {
          median_B532 = as.numeric(sample_calc_data_vec[l]); # current median

          for (h in 1:length(sample_calc_data_vec)) {
            #calculate mean_F535 with other variables
            mean_F635 = as.numeric(sample_calc_data_vec[h]);
            ROM = ((mean_F635-median_B635)/(mean_F532 - median_B532));

            #
            #print(sprintf("Current ratio:
            #
            if((current_ratio-ROM) <= 0.001 & (current_ratio-ROM) >=
                                    (-0.001) & !is.na(current_ratio-ROM)) { #
```

```
106
107                 results[i,j] = "median_B635";
108                 results[i,k] = "mean_F532";
109                 results[i,l] = "median_B532";
110                 results[i,h] = "mean_F635";
111                 #print("TRUE")
112
113               }
114             }
115           }
116         }
117       }
118     }
119     print(i)
120 }
121
122 #To A Tab Delimited Text File.
123 result_ending =paste0(w,".txt");
124 path_result = paste0("H:/Master Project/Other/RoM_calc_", result_ending);
125 write.table(results, path_result, sep="\t")
126
127 #evaluate how which col is most probable to be part of ROM
128
129 #results<- read.table("Other/RoM_calc_353419619.txt", header=TRUE, sep="\t",
130 #                             stringsAsFactors=FALSE)
131
132 #create table
133 evaluation <- data.frame(matrix(ncol = length(sample_calc_data[1,]), nrow = 4));
134 rownames(evaluation) = c("mean_F635","median_B635","mean_F532","median_B532");
135 colnames(evaluation) = colnames(sample_calc_data);
136
137 #calculate number of variables in columns
138 for (m in 1:length(sample_calc_data[1,])) {
139   current_col = results[,m]
140   possibilities = c("mean_F635","median_B635","mean_F532","median_B532");
141   for (n in 1:length(possibilities)) {
142     index = which(current_col == possibilities[n])
143     nr_found = length(index)
144     evaluation[n,m]=nr_found
145   }
146 }
147 #To A Tab Delimited Text File.
148 eval_ending =paste0(w,".txt");
149 path_eval = paste0("H:/Master Project/Other/RoM_calc_eval", eval_ending);
150 write.table(evaluation, path_eval, sep="\t")
151 #write.table(evaluation, "H:/Master Project/Other/RoM_calc_353419766_eval.txt",
152               sep="\t")
153
154 }
155 #get index of max value in row
156 #this is the value used for further calculation of respective variable in
157 #Tothill_CUP3.R
158 #eg. mean_F635 = Software.Unknown.....
```

```
#this column can then be selected in Tolhill_CUP3.R

#evaluation<- read.table("Other/RoM_calc_3353419766_eval.txt", header=TRUE,
#                         sep="\t", stringsAsFactors=FALSE)

for (u in 1:4) {
  index = which(evaluation[u,]==max(evaluation[u,]))
  x = colnames(evaluation)[index]

  #print variable for respective value from ROM
  print(sprintf(" %s = %s",rownames(evaluation)[u],x))
}
```

Listing 1: Script calculating and determining columns for RoM calculation using the formula provided by the GenePix Version 4.1 documentation.

R-script - Importing raw intensity data files:

```
# Read the raw microarray gene expression data. Return the data
# in a data.frame where the rows represent the individual samples and
# columns indivual genes. Column names represent the reporter name, row
# names the sample names.
# The unusual layout has been choosen to be compatible with the data
# layout for the svm() function of the e1071 package.
#
readRawData <- function(fname)
{
  # Read raw microarray data
  # read reporter_identifier and mean of a sample
  # the position of the identifier is column 6

  scan.names <- read.table(fname, header=TRUE, sep="\t", stringsAsFactors=FALSE)

  mean_FG_sample = as.numeric(scan.names$Software.Unknown.B635.Median)
  median_BG_sample = as.numeric(scan.names$Software.Unknown.Mean.of.Ratios..635.532.)
  mean_FG_ref = as.numeric(scan.names$Software.Unknown.Sum.of.Means)
  median_BG_ref = as.numeric(scan.names$Software.Unknown.F.Pixels)

  results <- data.frame(matrix(ncol = length(scan.names[,1])))
  normalized_values = mean_FG_sample-median_BG_sample;
  # normalized_values = normal(mean_FG_sample-median_BG_sample);
  # normalized_values = log2(mean_FG_sample);
  # normalized_values[is.infinite(normalized_values)] <-0
  results[1,] = normalized_values;

  colnames(results) = gsub(substring(scan.names$Reporter.identifier[1],
                           first = 1, last = 12L),'',
                           scan.names$Reporter.identifier)

  return(results)
}

# Read all microarray raw gene expression data files
#
# Returns a frame with samples and respective reporter names of all raw
# microarray data files.
readAllRawData <- function(adir, fpattern)
{

  afiles <- dir(adir, pattern=fpattern, full.names=TRUE)
  nfiles <- length(afiles)

  writeLog(sprintf("Reading raw microarray data from %d raw microarray
                   data files", nfiles))
  aframe <- readRawData(afiles[1])

  # create raw data frame formed like processed data
  # map scan names with raw data files
  raw_scan_names<- read.table("data/Tothill/E-MEXP-113.sdrf.txt", header=TRUE,
                              sep="\t", stringsAsFactors=FALSE, colClasses =
```

```
                                c(rep("NULL", 37), "character", "character",
                                rep("NULL", 7)))

  # only select unique rows for readability
  raw_scan_names = raw_scan_names[!duplicated(raw_scan_names),]

  for (i in 1:(nfiles-1)) {
    array_data <- readRawData(afiles[i+1])
    TF.names <- which(names(array_data) %in% names(aframe))
    subset = array_data[TF.names]
    aframe <- rbind(aframe, subset)
    print(i)
  }
  for (i in 1:nfiles){
    array_data_file = basename(afiles[i])
    index_scan_name <- which(raw_scan_names$Array.Data.File == array_data_file)
    current_scan_name <- gsub('MBA:MEXP:677:','',raw_scan_names[index_scan_name,])
    print(array_data_file)
    print(current_scan_name[[1]])
    rownames(aframe)[i]<-paste(current_scan_name[1])
    aframe[1:10,1:10]
  }
  writeLog(sprintf("Read %d samples with %d reporters", nrow(aframe), ncol(aframe)))
  return(aframe)
}
```

Listing 2: Functions used for importing raw intensity data which also apply first data processing steps.

TCGA-Assembler sript:

```
  #installed packages

#$ sudo apt-get install libssl-dev

#and then back to R
#install.packages("openssl")
#which was successful
#then
#install.packages("httr")

#packages <- c("HGNChelper", "httr", "RCurl", "rjson", "stringr")
#install.packages(packages, dependencies = T)
#install.packages("httr")
#install.packages("openssl")

#libraries
library(HGNChelper)
library(httr)
library(RCurl)
library(rjson)
library(stringr)

setwd("/home/julian/Documents/Master Project/TCGA/TCGA-Assembler")

#' Load functions
source("Module_A.R")
source("Module_B.R")

#' set data saving path
sPath1 <- "./QuickStartExample/Part1_DownloadedData"
sPath2 <- "./QuickStartExample/Part2_BasicDataProcessingResult"
sPath3 <- "./QuickStartExample/Part3_AdvancedDataProcessingResult"

#' choose a cancer type
sCancer <- "BRCA"

#' choose some patients
vPatientID <- c("TCGA-A7-A13F", "TCGA-AO-A12B", "TCGA-AR-A1AP", "TCGA-AR-A1AQ",
                "TCGA-AR-A1AS", "TCGA-AR-A1AV", "TCGA-AR-A1AW", "TCGA-BH-A0BZ",
                "TCGA-BH-A0DD", "TCGA-BH-A0DG")


#' ============================================================================
#' Part 1: Downloading Data of 7 different platforms using Module A functions
#' ============================================================================

#' Download somatic mutation data
path_somaticMutation <-
  DownloadSomaticMutationData(cancerType = sCancer,
                              assayPlatform = "somaticMutation_DNAseq",
                              inputPatientIDs = vPatientID,
                              saveFolderName = sPath1)
```

```
53
54 #' Download copy number alternation data
55 path_copyNumber <-
56   DownloadCNAData(cancerType = sCancer,
57                   assayPlatform = "cna_cnv.hg19",
58                   inputPatientIDs = vPatientID,
59                   saveFolderName = sPath1)
60
61 #' Download DNA methylation 450 data
62 path_methylation_450 <-
63   DownloadMethylationData(cancerType = sCancer,
64                           assayPlatform = "methylation_450",
65                           inputPatientIDs = vPatientID,
66                           saveFolderName = sPath1)
67
68 #' Download miRNA expression data
69 path_miRNAExp <-
70   DownloadmiRNASeqData(cancerType = sCancer,
71                        assayPlatform = "mir_HiSeq.hg19.mirbase20",
72                        inputPatientIDs = vPatientID,
73                        saveFolderName = sPath1)
74
75 #' Download gene expression data
76 path_geneExp <-
77   DownloadRNASeqData(cancerType = sCancer,
78                      assayPlatform = "gene.normalized_RNAseq",
79                      inputPatientIDs = vPatientID,
80                      saveFolderName = sPath1)
81
82 #' Download RPPA protein expression data
83 path_protein_RPPA <-
84   DownloadRPPAData(cancerType = sCancer,
85                    assayPlatform = "protein_RPPA",
86                    inputPatientIDs = vPatientID,
87                    saveFolderName = sPath1)
88
89 #' Download iTRAQ protein expression data
90 path_protein_iTRAQ <-
91   DownloadCPTACData(cancerType = sCancer,
92                     assayPlatform = "proteome_iTRAQ",
93                     inputPatientIDs = vPatientID,
94                     saveFolderName = sPath1)
```

Listing 3: Functions used downloading and processing TCGA RNA-seq data adapted from TCGA Assembler 2.

Neural network overview:

```
_________________________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================================
dense_25 (Dense)             (None, 256)               155648
_________________________________________________________________________________
dropout_20 (Dropout)         (None, 256)               0
_________________________________________________________________________________
dense_26 (Dense)             (None, 128)               32896
_________________________________________________________________________________
dropout_21 (Dropout)         (None, 128)               0
_________________________________________________________________________________
dense_27 (Dense)             (None, 13)                1677
=================================================================================
Total params: 190,221
Trainable params: 190,221
Non-trainable params: 0
_________________________________________________________________________________
```

Listing 4: Summary represents the neural network *model_RNAseq_MA_13_processed* used to classify Tothill *et al.* microarray data.

```
_________________________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================================
dense_22 (Dense)             (None, 256)               133632
_________________________________________________________________________________
dropout_18 (Dropout)         (None, 256)               0
_________________________________________________________________________________
dense_23 (Dense)             (None, 128)               32896
_________________________________________________________________________________
dropout_19 (Dropout)         (None, 128)               0
_________________________________________________________________________________
dense_24 (Dense)             (None, 13)                1677
=================================================================================
Total params: 168,205
Trainable params: 168,205
Non-trainable params: 0
_________________________________________________________________________________
```

Listing 5: Summary of neural network *model_FFNN_RNAseq_12* used to classify 12 selected TCGA RNA-seq classes common with Tothill *et al.* microarray data.

```
_________________________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================================
dense_11 (Dense)             (None, 256)               213760
_________________________________________________________________________________
dropout_9 (Dropout)          (None, 256)               0
_________________________________________________________________________________
dense_12 (Dense)             (None, 128)               32896
_________________________________________________________________________________
dropout_10 (Dropout)         (None, 128)               0
```

```
_________________________________________________________________
dense_13 (Dense)             (None, 19)                2451
=================================================================
Total params: 249,107
Trainable params: 249,107
Non-trainable params: 0
_________________________________________________________________
```

Listing 6: Summary of neural network *model_FFNN_RNAseq_19* used to classify 19 selected TCGA RNA-seq classes with single primaries.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_16 (Dense)             (None, 700)               584500
_________________________________________________________________
dropout_13 (Dropout)         (None, 700)               0
_________________________________________________________________
dense_17 (Dense)             (None, 350)               245350
_________________________________________________________________
dropout_14 (Dropout)         (None, 350)               0
_________________________________________________________________
dense_18 (Dense)             (None, 150)               52650
_________________________________________________________________
dropout_15 (Dropout)         (None, 150)               0
_________________________________________________________________
dense_19 (Dense)             (None, 100)               15100
_________________________________________________________________
dropout_16 (Dropout)         (None, 100)               0
_________________________________________________________________
dense_20 (Dense)             (None, 50)                5050
_________________________________________________________________
dropout_17 (Dropout)         (None, 50)                0
_________________________________________________________________
dense_21 (Dense)             (None, 19)                969
=================================================================
Total params: 903,619
Trainable params: 903,619
Non-trainable params: 0
_________________________________________________________________
```

Listing 7: Summary of neural network *model_deep_RNAseq_19* used to classify 19 selected TCGA RNA-seq classes with single primaries.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 1, 764, 32)        2304
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 1, 694, 64)        145472
_________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 1, 347, 64)     0
_________________________________________________________________
```

```
dropout_11 (Dropout)              (None, 1, 347, 64)            0
_________________________________________________________________________
flatten_1 (Flatten)               (None, 22208)                 0
_________________________________________________________________________
dense_14 (Dense)                  (None, 128)                   2842752
_________________________________________________________________________
dropout_12 (Dropout)              (None, 128)                   0
_________________________________________________________________________
dense_15 (Dense)                  (None, 19)                    2451
=========================================================================
Total params: 2,992,979
Trainable params: 2,992,979
Non-trainable params: 0
_________________________________________________________________________
```

Listing 8: summary of neural network *model_CNN_RNAseq_19* used to classify 19 selected TCGA RNA-seq classes with single primaries.

```
_________________________________________________________________________
Layer (type)                      Output Shape                  Param #
=========================================================================
conv2d (Conv2D)                   (None, 1, 1271, 32)           2304
_________________________________________________________________________
conv2d_1 (Conv2D)                 (None, 1, 1201, 64)           145472
_________________________________________________________________________
max_pooling2d (MaxPooling2D)      (None, 1, 600, 64)            0
_________________________________________________________________________
dropout_2 (Dropout)               (None, 1, 600, 64)            0
_________________________________________________________________________
flatten (Flatten)                 (None, 38400)                 0
_________________________________________________________________________
dense_3 (Dense)                   (None, 128)                   4915328
_________________________________________________________________________
dropout_3 (Dropout)               (None, 128)                   0
_________________________________________________________________________
dense_4 (Dense)                   (None, 33)                    4257
=========================================================================
Total params: 5,067,361
Trainable params: 5,067,361
Non-trainable params: 0
_________________________________________________________________________
```

Listing 9: Summary of neural network *model_CNN_RNAseq_33* used to classify 33 classes of TCGA RNA-seq data.

Classification results of Tothill *et al.* [5, 7, 8] RNA-seq data:

Classification results of preprocessed data:

In confusion tables the classification performance is visualized in matrix form, rows representing the predicted label and columns the true label of a class. Correct classifications are in the principal diagonal while misclassifications are not contained. A perfect model would show all classifications in the principal diagonal.

Table A1: Confusion table of the loocv results of preprocessed data samples based on 229 samples based on 13 tumor types and 607 genes resulting in a classification accuracy: 222/229 (96.9%)

Actual_Prediction	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung
brea (34)	34[32,0,2]	0	0	0	0	0	0	0	0	0	0	0	0
colo (23)	0	22[21,0,1]	0	0	0	0	1[0,0,1]	0	0	0	0	0	0
gast (15)	0	1[0,1,0]	14[13,0,1]	0	0	0	0	0	0	0	0	0	0
mela (11)	0	0	0	11[10,0,1]	0	0	0	0	0	0	0	0	0
meso (8)	0	0	0	0	8[8,0,0]	0	0	0	0	0	0	0	0
ovar (50)	0	0	0	0	0	50[48,2,0]	0	0	0	0	0	0	0
panc (9)	0	1[1,0,0]	0	0	0	0	8[6,1,1]	0	0	0	0	0	0
pros (8)	0	0	0	0	0	0	0	8[7,1,0]	0	0	0	0	0
rena (13)	0	0	0	0	0	0	0	0	13[11,0,2]	0	0	0	0
test (3)	0	0	0	0	0	0	0	0	0	3[3,0,0]	0	0	0
SCCo (14)	0	0	0	0	0	0	0	0	0	0	12[7,3,2]	0	2[1,0,1]
uter (9)	0	0	0	0	0	1[0,0,1]	0	0	0	0	0	8[8,0,0]	0
lung (32)	0	0	0	0	0	0	0	0	0	0	1[1,0,0]	0	31[29,2,0]

Table A2: Confusion table of the classification results of preprocessed samples without CUP based on 21 samples based on 13 tumor types and 541 genes resulting in a classification accuracy: 21/21 (100.0%).

Actual_Prediction	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung
brea (4)	4[4,0,0]	0	0	0	0	0	0	0	0	0	0	0	0
colo (2)	0	2[2,0,0]	0	0	0	0	0	0	0	0	0	0	0
gast (1)	0	0	1[1,0,0]	0	0	0	0	0	0	0	0	0	0
mela (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
meso (1)	0	0	0	0	1[1,0,0]	0	0	0	0	0	0	0	0
ovar (4)	0	0	0	0	0	4[4,0,0]	0	0	0	0	0	0	0
panc (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
pros (1)	0	0	0	0	0	0	0	1[1,0,0]	0	0	0	0	0
rena (1)	0	0	0	0	0	0	0	0	1[1,0,0]	0	0	0	0
test (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
SCCo (2)	0	0	0	0	0	0	0	0	0	0	2[2,0,0]	0	0
uter (1)	0	0	0	0	0	0	0	0	0	0	0	1[1,0,0]	0
lung (4)	0	0	0	0	0	0	0	0	0	0	0	0	4[4,0,0]

Table A3: Confusion table of the classification results of preprocessed data and CUP samples based on 13 samples based on 13 tumor types and 541 genes resulting in a classification accuracy: 11/ 13 (84.6%).

Actual_Prediction	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung
brea (3)	3[3,0,0]	0	0	0	0	0	0	0	0	0	0	0	0
colo (1)	0	1[1,0,0]	0	0	0	0	0	0	0	0	0	0	0
gast (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
mela (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
meso (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
ovar (1)	0	0	0	0	0	1[1,0,0]	0	0	0	0	0	0	0
panc (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
pros (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
rena (2)	0	0	0	0	0	0	0	0	2[2,0,0]	0	0	0	0
test (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
SCCo (2)	1[0,0,1]	0	0	0	0	0	0	0	0	0	0	0	1[0,0,1]
uter (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
lung (4)	0	0	0	0	0	0	0	0	0	0	0	0	4[3,1,0]

Classification results of log_2 transformed data:

Table A4: Confusion table of the loocv results of log_2 transformed data samples based on 229 samples based on 13 tumor types and 591 genes resulting in a classification accuracy: 217/229 (94.8%).

Actual_Prediction	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung
brea (34)	33[31,2,0]	0	0	1[0,1,0]	0	0	0	0	0	0	0	0	0
colo (23)	0	22[21,1,0]	0	0	0	0	0	0	0	0	0	0	1[0,0,1]
gast (15)	0	0	14[12,2,0]	0	0	1[0,1,0]	0	0	0	0	0	0	0
mela (11)	0	0	0	10[8,2,0]	0	0	0	0	0	0	0	0	1[0,1,0]
meso (8)	0	0	0	0	7[7,0,0]	0	0	0	0	0	1[0,0,1]	0	0
ovar (50)	0	0	0	0	0	50[44,4,2]	0	0	0	0	0	0	0
panc (9)	0	1[0,1,0]	0	0	0	0	8[7,0,1]	0	0	0	0	0	0
pros (8)	0	0	0	0	0	0	0	8[7,1,0]	0	0	0	0	0
rena (13)	0	0	0	0	0	0	0	0	12[11,1,0]	0	0	0	1[1,0,0]
test (3)	0	0	0	0	0	0	0	0	0	3[2,0,1]	0	0	0
SCCo (14)	0	0	0	0	0	0	0	0	0	0	11[6,2,3]	0	3[0,2,1]
uter (9)	0	0	0	0	0	0	0	0	0	0	0	9[8,0,1]	0
lung (32)	0	0	0	0	0	0	0	0	0	0	2[1,0,1]	0	30[23,6,1]

Table A5: Confusion table of the classification results of log_2 transformed samples without CUP based on 21 samples based on 13 tumor types and 504 genes resulting in a classification accuracy: 21/ 21 (100.0%).

Actual_Prediction	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung
brea (4)	4[4,0,0]	0	0	0	0	0	0	0	0	0	0	0	0
colo (1)	0	1[1,0,0]	0	0	0	0	0	0	0	0	0	0	0
gast (1)	0	0	1[1,0,0]	0	0	0	0	0	0	0	0	0	0
mela (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
meso (1)	0	0	0	0	1[1,0,0]	0	0	0	0	0	0	0	0
ovar (4)	0	0	0	0	0	4[4,0,0]	0	0	0	0	0	0	0
panc (1)	0	0	0	0	0	0	1[1,0,0]	0	0	0	0	0	0
pros (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
rena (1)	0	0	0	0	0	0	0	0	1[1,0,0]	0	0	0	0
test (1)	0	0	0	0	0	0	0	0	0	1[1,0,0]	0	0	0
SCCo (1)	0	0	0	0	0	0	0	0	0	0	1[1,0,0]	0	0
uter (2)	0	0	0	0	0	0	0	0	0	0	0	2[2,0,0]	0
lung (4)	0	0	0	0	0	0	0	0	0	0	0	0	4[4,0,0]

Table A6: Confusion table of the classification results of log_2 transformed data and CUP samples based on 13 samples based on 13 tumor types and 504 genes resulting in a classification accuracy: 11/ 13 (84.6%).

Actual_Prediction	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung
brea (3)	3[2,1,0]	0	0	0	0	0	0	0	0	0	0	0	0
colo (1)	0	1[0,1,0]	0	0	0	0	0	0	0	0	0	0	0
gast (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
mela (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
meso (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
ovar (1)	0	0	0	0	0	1[0,1,0]	0	0	0	0	0	0	0
panc (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
pros (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
rena (2)	0	0	0	0	0	0	0	0	2[1,1,0]	0	0	0	0
test (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
SCCo (2)	1[1,0,0]	0	0	0	0	0	0	0	0	0	0	0	1[1,0,0]
uter (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
lung (4)	0	0	0	0	0	0	0	0	0	0	0	0	4[2,1,1]

Classification results of range normalized background corrected data:

Table A7: Confusion table of loocv results of range normalized background corrected data based on 229 samples based on 13 tumor types and 631 genes resulting in a classification accuracy: 196/229 (85.6%).

Actual_Prediction	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung
brea (34)	32[30,1,1]	0	0	1[0,1,0]	0	0	0	0	0	0	0	0	1[0,0,1]
colo (23)	0	21[18,2,1]	0	0	0	1[1,0,0]	0	0	1[0,0,1]	0	0	0	0
gast (15)	0	2[0,0,2]	12[10,2,0]	1[0,0,1]	0	0	0	0	0	0	0	0	0
mela (11)	0	0	1[0,0,1]	8[7,0,1]	0	1[0,0,1]	0	0	0	0	1[0,0,1]	0	0
meso (8)	0	0	0	0	8[5,2,1]	0	0	0	0	0	0	0	0
ovar (50)	0	0	0	0	0	46[36,7,3]	2[0,0,2]	0	0	0	1[0,0,1]	0	1[0,1,0]
panc (9)	0	0	1[0,0,1]	0	0	2[0,1,1]	4[2,0,2]	0	0	0	1[0,0,1]	0	1[0,0,1]
pros (8)	0	0	0	0	0	0	0	8[6,2,0]	0	0	0	0	0
rena (13)	0	0	0	0	0	1[0,0,1]	0	0	11[8,2,1]	0	0	0	1[0,0,1]
test (3)	0	0	0	0	0	0	0	0	0	3[2,1,0]	0	0	0
SCCo (14)	0	0	0	0	0	1[0,0,1]	1[0,0,1]	0	0	0	7[2,2,3]	0	5[1,1,3]
uter (9)	0	0	0	0	0	0	0	0	0	0	0	9[6,3,0]	0
lung (32)	0	1[0,0,1]	0	0	0	0	1[0,0,1]	0	0	0	4[2,1,1]	0	26[17,7,2]

Table A8: Confusion table of the classification results of range normalized background corrected data without CUP samples based on 21 samples based on 13 tumor types and 536 genes resulting in a classification accuracy: 21/ 21 (100.0%).

Actual_Prediction	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung
brea (6)	6[6,0,0]	0	0	0	0	0	0	0	0	0	0	0	0
colo (2)	0	2[2,0,0]	0	0	0	0	0	0	0	0	0	0	0
gast (2)	0	0	2[2,0,0]	0	0	0	0	0	0	0	0	0	0
mela (1)	0	0	0	1[1,0,0]	0	0	0	0	0	0	0	0	0
meso (1)	0	0	0	0	1[1,0,0]	0	0	0	0	0	0	0	0
ovar (4)	0	0	0	0	0	4[4,0,0]	0	0	0	0	0	0	0
panc (2)	0	0	0	0	0	0	2[2,0,0]	0	0	0	0	0	0
pros (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
rena (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
test (1)	0	0	0	0	0	0	0	0	0	1[1,0,0]	0	0	0
SCCo (1)	0	0	0	0	0	0	0	0	0	0	1[0,1,0]	0	0
uter (1)	0	0	0	0	0	0	0	0	0	0	0	1[1,0,0]	0
lung (0)	0	0	0	0	0	0	0	0	0	0	0	0	0

Table A9: Confusion table of the classification results of range normalized background corrected and CUP samples'based on 13 samples based on 13 tumor types and 536 genes resulting in a classification accuracy: 10/ 13 (76.9%).

Actual_Prediction	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung
brea (3)	3[3,0,0]	0	0	0	0	0	0	0	0	0	0	0	0
colo (1)	0	0	0	0	0	0	0	0	0	0	0	0	1[0,0,1]
gast (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
mela (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
meso (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
ovar (1)	0	0	0	0	0	1[0,1,0]	0	0	0	0	0	0	0
panc (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
pros (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
rena (2)	0	0	0	0	0	0	0	0	2[1,0,1]	0	0	0	0
test (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
SCCo (2)	0	0	0	0	0	0	0	0	0	0	1[0,0,1]	1[0,0,1]	0
uter (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
lung (4)	0	0	0	0	0	1[0,0,1]	0	0	0	0	0	0	3[0,1,2]

Classification results of raw intensity data:

Table A10: Confusion table of loocv results of raw intensity data based on 229 samples based on 13 tumor types and 632 genes reulting in a classification accuracy: 195/229 (85.2%).

Actual_Prediction	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung
brea (34)	32[29,2,1]	0	0	1[1,0,0]	0	0	0	0	0	0	0	0	1[0,0,1]
colo (23)	0	21[18,1,2]	0	0	0	1[1,0,0]	0	0	1[0,0,1]	0	0	0	0
gast (15)	0	1[0,0,1]	13[10,2,1]	1[0,0,1]	0	0	0	0	0	0	0	0	0
mela (11)	0	0	1[0,0,1]	8[6,2,0]	0	2[0,0,2]	0	0	0	0	0	0	0
meso (8)	0	0	0	0	8[5,2,1]	0	0	0	0	0	0	0	0
ovar (50)	0	0	0	0	0	46[36,7,3]	2[0,0,2]	0	0	0	1[0,0,1]	0	1[0,1,0]
panc (9)	0	0	1[0,0,1]	0	0	2[0,2,0]	4[2,0,2]	0	0	0	1[0,1,0]	0	1[0,0,1]
pros (8)	0	0	0	0	0	0	0	8[8,0,0]	0	0	0	0	0
rena (13)	0	0	0	0	0	1[0,1,0]	0	0	11[8,2,1]	0	0	0	1[0,0,1]
test (3)	0	0	0	0	0	0	0	0	0	3[1,2,0]	0	0	0
SCCo (14)	0	0	0	0	0	1[0,0,1]	1[0,1,0]	0	0	0	7[3,1,3]	0	5[2,1,2]
uter (9)	0	0	0	0	0	1[0,1,0]	0	0	0	0	0	8[5,2,1]	0
lung (32)	0	1[0,0,1]	0	0	0	1[0,0,1]	0	0	0	0	4[2,1,1]	0	26[18,6,2]

Table A11: Confusion table of the classification results of raw intensity data without CUP samples based on 21 samples based on 13 tumor types and 540 genes resulting in a classification accuracy: 21/ 21 (100.0%).

Actual_Prediction	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung
brea (1)	1[1,0,0]	0	0	0	0	0	0	0	0	0	0	0	0
colo (6)	0	6[6,0,0]	0	0	0	0	0	0	0	0	0	0	0
gast (1)	0	0	1[1,0,0]	0	0	0	0	0	0	0	0	0	0
mela (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
meso (2)	0	0	0	0	2[2,0,0]	0	0	0	0	0	0	0	0
ovar (3)	0	0	0	0	0	3[3,0,0]	0	0	0	0	0	0	0
panc (2)	0	0	0	0	0	0	2[2,0,0]	0	0	0	0	0	0
pros (1)	0	0	0	0	0	0	0	1[1,0,0]	0	0	0	0	0
rena (1)	0	0	0	0	0	0	0	0	1[1,0,0]	0	0	0	0
test (1)	0	0	0	0	0	0	0	0	0	1[1,0,0]	0	0	0
SCCo (1)	0	0	0	0	0	0	0	0	0	0	1[0,1,0]	0	0
uter (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
lung (2)	0	0	0	0	0	0	0	0	0	0	0	0	2[2,0,0]

Table A12: Confusion table of the classification results of raw intensity data with CUP samples based on 13 samples based on 13 tumor types and 540 genes resulting in a classification accuracy: 9/ 13 (69.2%).

Actual_Prediction	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung
brea (3)	3[3,0,0]	0	0	0	0	0	0	0	0	0	0	0	0
colo (1)	0	0	0	0	0	0	1[0,0,1]	0	0	0	0	0	0
gast (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
mela (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
meso (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
ovar (1)	0	0	0	0	0	1[0,1,0]	0	0	0	0	0	0	0
panc (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
pros (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
rena (2)	1[0,0,1]	0	0	0	0	0	0	0	1[1,0,0]	0	0	0	0
test (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
SCCo (2)	1[0,0,1]	0	0	0	0	0	0	0	0	0	1[0,0,1]	0	0
uter (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
lung (4)	0	0	0	0	0	1[0,0,1]	0	0	0	0	0	0	3[0,2,1]

Classification results of background corrected intensitiy data:

Table A13: Confusion table of loocv results of background corrected intensitiy data based on 229 samples based on 13 tumor types and 631 genes resulting in a classification accuracy: 195/229 (85.2%).

Actual_Prediction	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung
brea (34)	32[28,3,1]	0	0	1[0,1,0]	0	0	0	0	0	0	0	0	1[0,0,1]
colo (23)	0	21[17,1,3]	0	0	0	1[1,0,0]	0	0	1[0,0,1]	0	0	0	0
gast (15)	0	1[0,0,1]	13[9,2,2]	1[0,0,1]	0	0	0	0	0	0	0	0	0
mela (11)	0	0	1[0,0,1]	8[6,2,0]	0	2[0,0,2]	0	0	0	0	0	0	0
meso (8)	0	0	0	0	8[6,1,1]	0	0	0	0	0	0	0	0
ovar (50)	0	0	0	0	0	46[36,6,4]	2[0,0,2]	0	0	0	1[0,0,1]	0	1[0,1,0]
panc (9)	0	0	1[0,0,1]	0	0	2[0,1,1]	4[2,0,2]	0	0	0	1[0,1,0]	0	1[0,0,1]
pros (8)	0	0	0	0	0	0	0	8[7,1,0]	0	0	0	0	0
rena (13)	0	0	0	0	0	1[0,1,0]	0	0	11[8,1,2]	0	0	0	1[0,0,1]
test (3)	0	0	0	0	0	0	0	0	0	3[2,1,0]	0	0	0
SCCo (14)	0	0	0	0	0	1[0,0,1]	1[0,1,0]	0	0	0	7[3,1,3]	0	5[2,1,2]
uter (9)	0	0	0	0	0	1[0,1,0]	0	0	0	0	0	8[4,2,2]	0
lung (32)	0	1[0,0,1]	0	0	0	1[0,0,1]	0	0	0	0	4[2,1,1]	0	26[17,5,4]

Table A14: Confusion table of the classification results of background corrected intensitiy data without CUP samples based on 13 samples based on 13 tumor types and 541 genes resulting in a classification accuracy: 9/ 13 (100.0%).

Actual_Prediction	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung
brea (2)	2[2,0,0]	0	0	0	0	0	0	0	0	0	0	0	0
colo (6)	0	6[6,0,0]	0	0	0	0	0	0	0	0	0	0	0
gast (2)	0	0	2[2,0,0]	0	0	0	0	0	0	0	0	0	0
mela (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
meso (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
ovar (4)	0	0	0	0	0	4[4,0,0]	0	0	0	0	0	0	0
panc (2)	0	0	0	0	0	0	2[2,0,0]	0	0	0	0	0	0
pros (1)	0	0	0	0	0	0	0	1[1,0,0]	0	0	0	0	0
rena (1)	0	0	0	0	0	0	0	0	1[1,0,0]	0	0	0	0
test (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
SCCo (1)	0	0	0	0	0	0	0	0	0	0	1[1,0,0]	0	0
uter (1)	0	0	0	0	0	0	0	0	0	0	0	1[1,0,0]	0
lung (1)	0	0	0	0	0	0	0	0	0	0	0	0	1[1,0,0]

Table A15: Confusion table of the classification results of background corrected intensitiy data with CUP samples based on 13 samples based on 13 tumor types and 541 genes resulting in a classification accuracy: 9/ 13 (69.2%).

Actual_Prediction	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung
brea (3)	3[2,1,0]	0	0	0	0	0	0	0	0	0	0	0	0
colo (1)	0	0	0	0	0	0	0	0	0	0	0	0	1[0,0,1]
gast (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
mela (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
meso (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
ovar (1)	0	0	0	0	0	1[0,1,0]	0	0	0	0	0	0	0
panc (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
pros (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
rena (2)	0	0	0	0	0	0	1[0,0,1]	0	1[1,0,0]	0	0	0	0
test (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
SCCo (2)	0	0	0	0	0	0	0	0	0	0	1[0,0,1]	1[0,0,1]	0
uter (0)	0	0	0	0	0	0	0	0	0	0	0	0	0
lung (4)	0	0	0	0	0	1[0,0,1]	0	0	0	0	0	0	3[0,1,2]

Classification results of MU Graz RNA-seq data:

Results generated using the function *ratioTransform()*; transform RNA-seq RPM values to log_2 ratios using the sum of expression in all samples as the reference.

Results generated using the class *voomTransform()*; Transform count data to log_2-counts per million (logCPM), estimate the mean-variance relationship and use this to compute appropriate observational-level weights. The data are then ready for linear modelling [83].

Table A16: Classification results of *RNAseq_MU_ratio* data and a SVM model trained on raw intensity data.

Sample	Actual	Pred	DM	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung	Total
K1999-6	colo	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.727	0.014	0.14	0.011	0.029	0.018	0.616	1.91
K2170-6	melan	panc,lung,mela	L	0.012	0.012	0.064	0.151	0.008	0.108	0.727	0.014	0.141	0.011	0.029	0.018	0.616	1.911
K2953-5	lung	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.727	0.014	0.14	0.011	0.029	0.018	0.617	1.911
K4824-4	panc	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.727	0.014	0.14	0.011	0.029	0.018	0.615	1.909
K5108-24	ovar	panc,lung,mela	L	0.012	0.012	0.064	0.15	0.008	0.108	0.727	0.014	0.14	0.011	0.029	0.018	0.615	1.908
K5216-6	breast	panc,lung,mela	L	0.012	0.012	0.064	0.15	0.008	0.108	0.727	0.014	0.14	0.011	0.029	0.018	0.615	1.908
K6266-2	mela	panc,lung,mela	L	0.012	0.012	0.064	0.15	0.008	0.108	0.727	0.014	0.14	0.011	0.029	0.018	0.616	1.909
K6267-2	lung	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.727	0.014	0.14	0.011	0.029	0.018	0.616	1.91
K6268-2	panc	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.727	0.014	0.14	0.011	0.029	0.018	0.615	1.909
K6269-2	gast	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.727	0.014	0.14	0.011	0.029	0.018	0.615	1.909
K6270-2	gast	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.727	0.014	0.14	0.011	0.029	0.018	0.616	1.91
K6271-2	rena	panc,lung,mela	L	0.012	0.012	0.064	0.15	0.008	0.108	0.727	0.014	0.141	0.011	0.029	0.018	0.616	1.91
K6272-2	rena	panc,lung,mela	L	0.012	0.012	0.064	0.15	0.008	0.108	0.727	0.014	0.141	0.011	0.029	0.018	0.616	1.91
K6365-2	colo	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.727	0.014	0.14	0.011	0.029	0.018	0.616	1.91
K6366-2	colo	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.727	0.014	0.14	0.011	0.029	0.018	0.615	1.909
K6367-2	colo	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.727	0.014	0.14	0.011	0.029	0.018	0.615	1.909
K6368-2	lung	panc,lung,mela	L	0.012	0.012	0.064	0.15	0.008	0.108	0.727	0.014	0.141	0.011	0.029	0.018	0.617	1.911
K6369-2	brea	panc,lung,mela	L	0.012	0.012	0.064	0.15	0.008	0.108	0.727	0.014	0.14	0.011	0.029	0.018	0.616	1.909
K6370-2	brea	panc,lung,mela	L	0.012	0.012	0.064	0.15	0.008	0.108	0.727	0.014	0.14	0.011	0.029	0.018	0.616	1.909
K6371-2	pros	panc,lung,mela	L	0.012	0.012	0.064	0.15	0.008	0.108	0.727	0.014	0.14	0.011	0.029	0.018	0.616	1.909
K6372-2	pros	panc,lung,mela	L	0.012	0.012	0.064	0.15	0.008	0.108	0.727	0.014	0.14	0.011	0.029	0.018	0.616	1.909

Table A17: Classification results of *RNAseq_MU_ratio* data with a SVM model trained on background corrected intensity data.

Sample	Actual	Pred	DM	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung	Total
K1999-6	colo	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.634	0.013	0.098	0.011	0.016	0.023	0.456	1.505
K2170-6	melan	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.634	0.013	0.098	0.011	0.016	0.023	0.455	1.504
K2953-5	lung	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.634	0.013	0.098	0.011	0.016	0.023	0.456	1.505
K4824-4	panc	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.634	0.013	0.098	0.011	0.016	0.023	0.455	1.504
K5108-24	ovar	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.634	0.013	0.098	0.011	0.016	0.023	0.455	1.504
K5216-6	breast	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.634	0.013	0.098	0.011	0.016	0.023	0.455	1.504
K6266-2	mela	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.633	0.013	0.098	0.011	0.016	0.023	0.455	1.503
K6267-2	lung	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.634	0.013	0.098	0.011	0.016	0.023	0.456	1.505
K6268-2	panc	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.634	0.013	0.098	0.011	0.016	0.023	0.455	1.504
K6269-2	gast	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.634	0.013	0.098	0.011	0.016	0.023	0.454	1.503
K6270-2	gast	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.634	0.013	0.098	0.011	0.016	0.023	0.455	1.504
K6271-2	rena	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.634	0.013	0.098	0.011	0.016	0.023	0.455	1.504
K6272-2	rena	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.634	0.013	0.098	0.011	0.016	0.023	0.455	1.504
K6365-2	colo	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.634	0.013	0.098	0.011	0.016	0.023	0.456	1.505
K6366-2	colo	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.634	0.013	0.098	0.011	0.016	0.023	0.455	1.504
K6367-2	colo	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.634	0.013	0.098	0.011	0.016	0.023	0.455	1.504
K6368-2	lung	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.634	0.013	0.098	0.011	0.016	0.023	0.456	1.505
K6369-2	brea	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.634	0.013	0.098	0.011	0.016	0.023	0.455	1.504
K6370-2	brea	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.634	0.013	0.098	0.011	0.016	0.023	0.455	1.504
K6371-2	pros	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.634	0.013	0.098	0.011	0.016	0.023	0.455	1.504
K6372-2	pros	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.634	0.013	0.098	0.011	0.016	0.023	0.455	1.504

Table A18: Classification results of *RNAseq_MU_ratio* data with a SVM model trained on range normalized background corrected intensity data.

Sample	Actual	Pred	DM	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung	Total
K1999-6	colo	brea,colo,ovar	L	1	1	0	0	0	1	1	0	0	1	0	1	1	7
K2170-6	melan	brea,mela,meso	L	1	0	0	1	1	1	1	0	1	1	1	0.595	1	9.595
K2953-5	lung	brea,meso,ovar	L	1	0	0	0	1	1	1	0	1	1	0.001	1	1	8.001
K4824-4	panc	brea,colo,meso	L	1	1	0	0	1	1	1	0	0	0	0	1	1	7
K5108-24	ovar	brea,meso,ovar	L	1	0	0	0	1	1	1	0	1	1	1	1	1	9
K5216-6	breast	brea,ovar,panc	L	1	0	0	0	0	1	1	0	0.653	1	0.022	0	1	5.675
K6266-2	mela	brea,meso,ovar	L	1	0	0	0	1	1	0.43	0	1	1	1	1	1	8.43
K6267-2	lung	brea,colo,ovar	L	1	1	0	0	0.512	1	1	0	0.927	1	0	0.844	1	8.283
K6268-2	panc	brea,ovar,panc	L	1	0	0	0	0.997	1	1	0	1	0.005	1	1	0.604	7.606
K6269-2	gast	gast,ovar,panc	L	0	0.997	1	0	0	1	1	0	1	0.995	0	1	0	6.992
K6270-2	gast	brea,colo,ovar	L	1	1	0	0	0.1	1	1	0	0	1	0	0.959	1	7.059
K6271-2	rena	brea,meso,ovar	L	1	0	0	0	1	1	1	0	1	1	1	1	1	9
K6272-2	rena	brea,meso,ovar	L	1	0	0	0	1	1	1	0	1	1	0	0	1	7
K6365-2	colo	brea,colo,ovar	L	1	1	0	0.996	0.183	1	1	0	0	1	0	1	1	8.179
K6366-2	colo	brea,colo,ovar	L	1	1	0	0	0.21	1	1	0	0	0	0	1	1	6.21
K6367-2	colo	brea,colo,ovar	L	1	1	0	0	0	1	1	0	0	0.999	0	1	0	5.999
K6368-2	lung	brea,meso,ovar	L	1	0	0	0.632	1	1	1	0	1	1	1	0.515	1	9.147
K6369-2	brea	brea,mela,ovar	L	1	0	0	1	0.122	1	1	0	1	1	0	1	1	8.122
K6370-2	brea	brea,meso,ovar	L	1	0	0	0	1	1	1	0	1	1	1	0.948	1	8.948
K6371-2	pros	brea,mela,meso	L	1	0	0	1	1	1	1	1	1	1	0.998	1	1	10.998
K6372-2	pros	brea,meso,ovar	L	1	0	0	0.008	1	1	1	1	1	1	0.999	1	1	10.007

Table A19: Classification results of *RNAseq_MU_ratio* data with a SVM model trained on log_2-transformed intensity data.

Sample	Actual	Pred	DM	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung	Total
K1999-6	colo	panc,lung,gast	L	0	0	0.947	0.003	0.017	0.005	1	0	0.924	0	0	0	1	3.896
K2170-6	melan	mela,panc,lung	L	0	0	0.149	1	0.053	0	1	0	0.899	0	0	0	1	4.101
K2953-5	lung	panc,lung,rena	L	0	0	0.393	0.005	0.052	0	1	0	0.993	0	0	0.002	1	3.445
K4824-4	panc	ovar,panc,lung	L	0	0	0.017	0.003	0.134	1	1	0	0.031	0	0	0	1	3.185
K5108-24	ovar	ovar,panc,lung	L	0	0	0.106	0	0.527	1	1	0	0.488	0	0.001	0	1	4.122
K5216-6	breast	brea,panc,lung	L	1	0	0.098	0.012	0.007	0.264	1	0	0.525	0	0	0	1	3.906
K6266-2	mela	ovar,panc,lung	L	0	0	0.042	0.002	0.399	1	1	0	0.133	0	0	0.003	1	3.579
K6267-2	lung	panc,lung,ovar	L	0	0	0.016	0.029	0.007	0.994	1	0	0.882	0	0	0	1	3.928
K6268-2	panc	ovar,panc,lung	L	0	0	0.896	0.002	0.05	1	1	0	0.526	0	0.003	0	1	4.477
K6269-2	gast	gast,ovar,panc	L	0	0	1	0.004	0.012	1	1	0.001	0.931	0	0	0	1	4.948
K6270-2	gast	ovar,panc,lung	L	0	0	0.899	0.025	0.009	1	1	0	0.325	0	0	0	1	4.258
K6271-2	rena	rena,lung,ovar	L	0	0	0.007	0.014	0.047	0.999	0.999	0	1	0	0	0	1	4.066
K6272-2	rena	ovar,panc,rena	L	0	0	0.003	0.003	0.082	1	1	0	1	0	0	0	1	4.088
K6365-2	colo	panc,lung,gast	L	0	0	0.542	0.033	0.018	0.024	1	0	0.31	0	0	0	1	2.927
K6366-2	colo	panc,lung,ovar	L	0	0	0.008	0.005	0.03	0.999	1	0	0.786	0	0	0.001	1	3.829
K6367-2	colo	ovar,panc,lung	L	0	0	0.006	0.009	0.02	1	1	0	0.29	0	0.007	0	0.998	3.33
K6368-2	lung	ovar,panc,lung	L	0	0	0.005	0.131	0.356	1	1	0.004	0.939	0	0	0	1	4.435
K6369-2	brea	panc,lung,ovar	L	0.07	0	0.059	0.15	0.074	0.666	1	0	0.451	0	0	0	1	3.47
K6370-2	brea	ovar,panc,lung	L	0.663	0	0.011	0.006	0.125	1	1	0	0.93	0	0.003	0	1	4.738
K6371-2	pros	panc,lung,pros	L	0	0	0	0.652	0.149	0.819	1	0.995	0.668	0	0	0	1	5.283
K6372-2	pros	panc,lung,pros	L	0	0	0.02	0.122	0.15	0.293	1	0.997	0.482	0	0	0.001	1	4.065

Table A20: Classification results of *RNAseq_MU_voom* data with a SVM model trained on raw intensity data.

Sample	Actual	Pred	DM	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung	Total
K1999-6	colo	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.726	0.014	0.14	0.011	0.029	0.018	0.614	1.907
K2170-6	melan	panc,lung,mela	L	0.012	0.013	0.064	0.151	0.008	0.108	0.726	0.014	0.14	0.011	0.029	0.018	0.614	1.908
K2953-5	lung	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.726	0.014	0.14	0.011	0.029	0.018	0.614	1.907
K4824-4	panc	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.726	0.014	0.14	0.011	0.029	0.018	0.613	1.906
K5108-24	ovar	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.726	0.014	0.14	0.011	0.029	0.018	0.613	1.906
K5216-6	breast	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.726	0.014	0.14	0.011	0.029	0.018	0.613	1.906
K6266-2	mela	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.726	0.014	0.14	0.011	0.029	0.018	0.613	1.906
K6267-2	lung	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.107	0.726	0.014	0.14	0.011	0.029	0.018	0.613	1.905
K6268-2	panc	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.726	0.014	0.14	0.011	0.029	0.018	0.613	1.906
K6269-2	gast	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.726	0.014	0.14	0.011	0.029	0.018	0.613	1.906
K6270-2	gast	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.726	0.014	0.14	0.011	0.029	0.018	0.613	1.906
K6271-2	rena	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.726	0.014	0.14	0.011	0.029	0.018	0.613	1.906
K6272-2	rena	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.726	0.014	0.14	0.011	0.029	0.018	0.614	1.907
K6365-2	colo	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.726	0.014	0.14	0.011	0.029	0.018	0.613	1.906
K6366-2	colo	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.726	0.014	0.14	0.011	0.029	0.018	0.613	1.906
K6367-2	colo	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.726	0.014	0.14	0.011	0.029	0.018	0.613	1.906
K6368-2	lung	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.726	0.014	0.14	0.011	0.029	0.018	0.614	1.907
K6369-2	brea	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.726	0.014	0.14	0.011	0.029	0.018	0.613	1.906
K6370-2	brea	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.726	0.014	0.14	0.011	0.029	0.018	0.613	1.906
K6371-2	pros	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.108	0.726	0.014	0.14	0.011	0.029	0.018	0.613	1.906
K6372-2	pros	panc,lung,mela	L	0.012	0.013	0.064	0.15	0.008	0.107	0.726	0.014	0.14	0.011	0.029	0.018	0.613	1.905

Table A21: Classification results of *RNAseq_MU_voom* data with a SVM model trained on background corrected intensity data.

Sample	Actual	Pred	DM	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung	Total
K1999-6	colo	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.096	0.632	0.013	0.098	0.011	0.016	0.023	0.453	1.499
K2170-6	melan	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.096	0.632	0.013	0.098	0.011	0.016	0.023	0.453	1.499
K2953-5	lung	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.096	0.632	0.013	0.098	0.011	0.016	0.023	0.453	1.499
K4824-4	panc	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.632	0.013	0.098	0.011	0.016	0.023	0.452	1.499
K5108-24	ovar	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.632	0.013	0.098	0.011	0.016	0.023	0.453	1.5
K5216-6	breast	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.096	0.632	0.013	0.098	0.011	0.016	0.023	0.452	1.498
K6266-2	mela	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.632	0.013	0.098	0.011	0.016	0.023	0.453	1.5
K6267-2	lung	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.096	0.632	0.013	0.098	0.011	0.016	0.023	0.453	1.499
K6268-2	panc	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.632	0.013	0.098	0.011	0.016	0.023	0.453	1.5
K6269-2	gast	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.096	0.632	0.013	0.098	0.011	0.016	0.023	0.452	1.498
K6270-2	gast	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.096	0.632	0.013	0.098	0.011	0.016	0.023	0.453	1.499
K6271-2	rena	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.096	0.632	0.013	0.098	0.011	0.016	0.023	0.453	1.499
K6272-2	rena	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.632	0.013	0.098	0.011	0.016	0.023	0.453	1.5
K6365-2	colo	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.096	0.632	0.013	0.098	0.011	0.016	0.023	0.453	1.499
K6366-2	colo	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.096	0.632	0.013	0.098	0.011	0.016	0.023	0.452	1.498
K6367-2	colo	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.632	0.013	0.098	0.011	0.016	0.023	0.452	1.499
K6368-2	lung	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.096	0.632	0.013	0.098	0.011	0.016	0.023	0.453	1.499
K6369-2	brea	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.097	0.632	0.013	0.098	0.011	0.016	0.023	0.453	1.5
K6370-2	brea	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.096	0.632	0.013	0.098	0.011	0.016	0.023	0.453	1.499
K6371-2	pros	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.096	0.632	0.013	0.098	0.011	0.016	0.023	0.453	1.499
K6372-2	pros	panc,lung	L	0.015	0.017	0.035	0.082	0.008	0.096	0.632	0.013	0.098	0.011	0.016	0.023	0.453	1.499

Table A22: Classification results of *RNAseq_MU_voom* data with a SVM model trained on range normalized background corrected intensity data.

Sample	Actual	Pred	DM	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung	Total
K1999-6	colo	brea,colo,ovar	L	1	1	0	0	0	1	0	0	0	1	0	1	0	5
K2170-6	melan	brea,colo,mela	L	1	1	0	1	0	1	0	0	1	1	1	0.002	0	7.002
K2953-5	lung	brea,colo,ovar	L	1	1	0	0	0	1	0	0	0	1	0	1	0	5
K4824-4	panc	brea,colo,ovar	L	1	1	0	0	0	1	0	0	0	0	0	0	0	3
K5108-24	ovar	brea,ovar	L	1	0	0	0	0	1	0	0	0	0.06	0	0.003	0	2.063
K5216-6	breast	brea,colo,ovar	L	1	1	0	0	0	1	0	0	0	0.011	0	0	0	3.011
K6266-2	mela	brea,colo,ovar	L	1	1	0	0	0	1	0	0	0	0.988	0	0	0	3.988
K6267-2	lung	brea,colo,ovar	L	1	1	0	0	0	0.998	0	0	0	0.098	0	0	0	3.096
K6268-2	panc	brea,colo,ovar	L	1	1	0	0	0	1	0	0	0	0	0.015	0.008	0	3.023
K6269-2	gast	colo,ovar,uter	L	0	1	0.729	0	0	1	0	0	0	0	0	0.782	0	3.511
K6270-2	gast	brea,colo,ovar	L	1	1	0	0	0	1	0	0	0	0.995	0	0	0	3.995
K6271-2	rena	brea,colo,ovar	L	1	1	0	0	0	1	0	0.013	1	0.999	0	0	0	5.012
K6272-2	rena	brea,colo,ovar	L	1	1	0	0	0	1	0	0	1	0.995	0	0	0	4.995
K6365-2	colo	brea,colo,ovar	L	1	1	0	0	0	1	0	0	0	0.963	0	0	0	3.963
K6366-2	colo	brea,colo,ovar	L	1	1	0	0	0	1	0	0	0	0	0	0.006	0	3.006
K6367-2	colo	brea,colo,ovar	L	1	1	0	0	0	1	0	0	0	0	0	0	0	3
K6368-2	lung	brea,colo,ovar	L	1	1	0	0	0	1	0	0.999	0.014	1	0.001	0	0	5.014
K6369-2	brea	brea,colo,ovar	L	1	1	0	0	0	1	0	0	0	0.998	0	0.626	0	4.624
K6370-2	brea	brea,ovar,test	L	1	0.999	0	0	0	1	0	0	0	1	0	0	0	3.999
K6371-2	pros	brea,colo,ovar	L	1	1	0	0	0	1	0	1	0	1	0	0	0	5
K6372-2	pros	brea,colo,pros	L	1	1	0	0	0	0.766	0	1	0	1	0	0	0	4.766

Table A23: Classification results of *RNAseq_MU_voom* data with a SVM model trained on log_2-transformed intensity data.

Sample	Actual	Pred	DM	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung	Total
K1999-6	colo	lung,panc,gast	L	0	0	0.971	0.003	0.005	0.059	0.999	0	0.279	0	0	0.001	1	3.317
K2170-6	melan	mela,lung,panc	L	0	0	0.364	1	0.01	0.011	0.985	0	0.695	0	0	0	1	4.065
K2953-5	lung	lung,panc,rena	L	0	0	0.567	0.026	0.008	0.124	0.996	0	0.82	0	0	0.001	1	3.542
K4824-4	panc	ovar,panc,lung	L	0	0	0.219	0.006	0.025	1	1	0	0.014	0	0	0	0.991	3.255
K5108-24	ovar	ovar,lung,panc	L	0	0	0.288	0.001	0.068	1	0.987	0	0.045	0	0	0	1	3.389
K5216-6	breast	lung,brea,panc	L	0.995	0	0.245	0.01	0.004	0.718	0.975	0	0.035	0	0	0	1	3.982
K6266-2	mela	ovar,lung,panc	L	0	0	0.14	0.002	0.048	1	0.945	0	0.056	0	0	0.001	1	3.192
K6267-2	lung	lung,ovar,panc	L	0	0	0.547	0.006	0.003	0.993	0.981	0.001	0.151	0	0	0	1	3.682
K6268-2	panc	ovar,panc,lung	L	0	0	0.799	0.003	0.008	1	1	0	0.18	0	0.003	0.001	1	3.994
K6269-2	gast	gast,ovar,panc	L	0	0	1	0.004	0.003	0.999	0.999	0.001	0.271	0	0	0	0.998	4.275
K6270-2	gast	lung,panc,ovar	L	0	0	0.914	0.01	0.004	0.997	0.999	0	0.018	0	0	0	1	3.942
K6271-2	rena	rena,lung,ovar	L	0	0	0.061	0.009	0.007	0.982	0.911	0.001	1	0	0	0	1	3.971
K6272-2	rena	ovar,lung,rena	L	0	0	0.101	0.01	0.018	1	0.965	0	0.999	0	0	0	1	4.093
K6365-2	colo	lung,panc,gast	L	0	0	0.791	0.006	0.006	0.282	0.996	0	0.055	0	0	0	0.999	3.135
K6366-2	colo	ovar,panc,lung	L	0	0	0.42	0.004	0.006	0.996	0.995	0	0.11	0	0	0.001	0.975	3.507
K6367-2	colo	ovar,panc,lung	L	0	0	0.347	0.008	0.007	1	0.996	0	0.014	0	0.001	0	0.94	3.313
K6368-2	lung	lung,ovar,panc	L	0	0	0.082	0.047	0.042	0.999	0.906	0.007	0.679	0	0	0	1	3.762
K6369-2	brea	lung,ovar,panc	L	0.092	0	0.4	0.087	0.007	0.938	0.931	0	0.059	0	0	0	1	3.514
K6370-2	brea	lung,ovar,panc	L	0.107	0	0.256	0.021	0.025	0.999	0.979	0	0.212	0	0	0	1	3.599
K6371-2	pros	lung,panc,pros	L	0	0	0.02	0.039	0.016	0.895	0.977	0.97	0.212	0	0	0	1	4.129
K6372-2	pros	lung,pros,panc	L	0	0	0.111	0.026	0.015	0.806	0.904	0.993	0.207	0	0	0.001	1	4.063

Table A24: Classification results of *RNAseq_MU_RPM* data with a SVM model trained on raw intensity data.

Sample	Actual	Pred	DM	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung	Total
K1999-6	colo	panc,lung,mela	L	0.012	0.014	0.065	0.145	0.008	0.117	0.713	0.014	0.127	0.011	0.026	0.017	0.582	1.851
K2170-6	melan	panc,lung,mela	L	0.012	0.013	0.062	0.176	0.008	0.102	0.698	0.013	0.142	0.011	0.028	0.016	0.577	1.858
K2953-5	lung	panc,lung,mela	L	0.013	0.013	0.069	0.138	0.007	0.11	0.67	0.014	0.123	0.011	0.028	0.017	0.566	1.779
K4824-4	panc	panc,lung,mela	L	0.015	0.018	0.055	0.123	0.009	0.099	0.674	0.011	0.081	0.008	0.033	0.013	0.548	1.687
K5108-24	ovar	panc,lung,ovar	L	0.015	0.017	0.049	0.105	0.009	0.116	0.663	0.012	0.093	0.009	0.033	0.015	0.601	1.737
K5216-6	breast	panc,lung,mela	L	0.016	0.011	0.06	0.13	0.007	0.122	0.685	0.014	0.113	0.011	0.027	0.016	0.608	1.82
K6266-2	mela	panc,lung,mela	L	0.013	0.014	0.058	0.123	0.008	0.123	0.681	0.014	0.12	0.011	0.029	0.016	0.591	1.801
K6267-2	lung	panc,lung,mela	L	0.011	0.017	0.062	0.151	0.007	0.09	0.689	0.013	0.116	0.011	0.026	0.015	0.561	1.769
K6268-2	panc	panc,lung,mela	L	0.013	0.013	0.061	0.128	0.008	0.094	0.685	0.013	0.12	0.01	0.036	0.016	0.495	1.692
K6269-2	gast	panc,lung,mela	L	0.012	0.013	0.071	0.142	0.008	0.097	0.708	0.013	0.102	0.01	0.029	0.015	0.54	1.76
K6270-2	gast	panc,mela,lung	M	0.009	0.007	0.13	0.406	0.005	0.075	0.75	0.01	0.044	0.014	0.026	0.007	0.378	1.861
K6271-2	rena	panc,lung,mela	L	0.014	0.014	0.057	0.132	0.008	0.107	0.702	0.013	0.131	0.01	0.03	0.016	0.583	1.817
K6272-2	rena	panc,lung,mela	L	0.013	0.013	0.062	0.142	0.008	0.106	0.699	0.013	0.14	0.011	0.028	0.015	0.581	1.831
K6365-2	colo	panc,lung,mela	L	0.012	0.016	0.063	0.142	0.008	0.101	0.705	0.014	0.13	0.011	0.027	0.017	0.584	1.83
K6366-2	colo	panc,lung,mela	L	0.011	0.021	0.058	0.133	0.008	0.079	0.702	0.013	0.108	0.009	0.026	0.016	0.536	1.72
K6367-2	colo	panc,lung,mela	L	0.012	0.015	0.06	0.133	0.008	0.093	0.711	0.013	0.113	0.01	0.028	0.017	0.567	1.78
K6368-2	lung	panc,lung,mela	L	0.013	0.013	0.059	0.141	0.008	0.109	0.695	0.013	0.132	0.011	0.029	0.016	0.589	1.828
K6369-2	brea	panc,lung,mela	L	0.014	0.012	0.066	0.149	0.008	0.109	0.709	0.014	0.129	0.011	0.028	0.017	0.587	1.853
K6370-2	brea	panc,lung,mela	L	0.014	0.013	0.063	0.141	0.008	0.108	0.69	0.013	0.118	0.011	0.028	0.015	0.609	1.831
K6371-2	pros	panc,lung,mela	L	0.012	0.013	0.058	0.134	0.008	0.1	0.693	0.017	0.123	0.01	0.029	0.016	0.585	1.798
K6372-2	pros	panc,lung,mela	L	0.01	0.012	0.061	0.141	0.008	0.099	0.699	0.021	0.125	0.011	0.027	0.017	0.57	1.801

Table A25: Classification results of *RNAseq_MU_RPM* data with a SVM model trained on background corrected intensities intensity data.

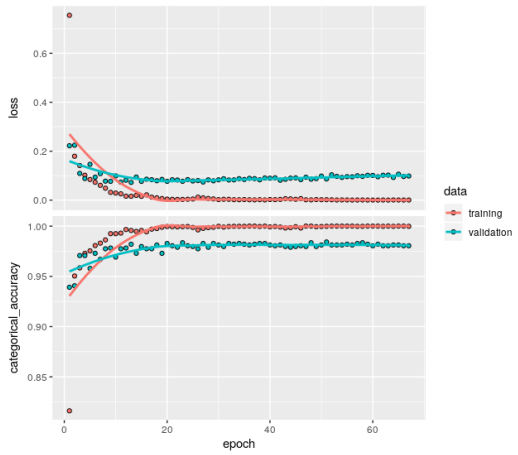
Sample	Actual	Pred	DM	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung	Total
K1999-6	colo	panc,lung,ovar	L	0.015	0.018	0.037	0.079	0.007	0.104	0.614	0.013	0.088	0.011	0.015	0.023	0.42	1.444
K2170-6	melan	panc,lung,rena	L	0.016	0.018	0.033	0.092	0.007	0.096	0.597	0.013	0.1	0.011	0.016	0.022	0.415	1.436
K2953-5	lung	panc,lung	L	0.015	0.018	0.036	0.08	0.007	0.097	0.565	0.013	0.088	0.011	0.016	0.023	0.42	1.389
K4824-4	panc	panc,lung	L	0.018	0.024	0.026	0.084	0.008	0.098	0.597	0.011	0.067	0.008	0.02	0.017	0.378	1.356
K5108-24	ovar	panc,lung,ovar	L	0.019	0.022	0.024	0.072	0.009	0.105	0.578	0.012	0.075	0.009	0.019	0.019	0.437	1.4
K5216-6	breast	panc,lung,ovar	L	0.02	0.016	0.033	0.076	0.007	0.11	0.592	0.013	0.082	0.011	0.015	0.021	0.443	1.439
K6266-2	mela	panc,lung,ovar	L	0.017	0.019	0.03	0.072	0.008	0.11	0.584	0.013	0.087	0.011	0.016	0.021	0.431	1.419
K6267-2	lung	panc,lung	L	0.013	0.022	0.034	0.083	0.007	0.087	0.588	0.013	0.081	0.011	0.014	0.02	0.403	1.376
K6268-2	panc	panc,lung	L	0.016	0.018	0.032	0.074	0.008	0.088	0.591	0.012	0.087	0.01	0.02	0.021	0.351	1.328
K6269-2	gast	panc,lung	M	0.015	0.017	0.038	0.086	0.008	0.094	0.625	0.012	0.077	0.01	0.017	0.02	0.374	1.393
K6270-2	gast	panc,mela,lung	H	0.01	0.01	0.067	0.221	0.005	0.087	0.726	0.011	0.034	0.014	0.015	0.01	0.194	1.404
K6271-2	rena	panc,lung	L	0.017	0.018	0.03	0.077	0.008	0.097	0.611	0.013	0.096	0.01	0.017	0.021	0.419	1.434
K6272-2	rena	panc,lung,rena	L	0.016	0.017	0.033	0.082	0.007	0.096	0.609	0.013	0.1	0.011	0.016	0.02	0.419	1.439
K6365-2	colo	panc,lung	L	0.015	0.021	0.034	0.079	0.007	0.091	0.606	0.013	0.091	0.011	0.015	0.022	0.426	1.431
K6366-2	colo	panc,lung	L	0.013	0.026	0.031	0.078	0.008	0.077	0.607	0.013	0.08	0.01	0.015	0.021	0.385	1.364
K6367-2	colo	panc,lung	L	0.015	0.02	0.032	0.077	0.008	0.09	0.619	0.013	0.083	0.01	0.016	0.022	0.412	1.417
K6368-2	lung	panc,lung	L	0.016	0.018	0.031	0.08	0.008	0.099	0.599	0.013	0.094	0.011	0.017	0.022	0.422	1.43
K6369-2	brea	panc,lung	L	0.017	0.016	0.036	0.082	0.007	0.097	0.617	0.013	0.091	0.012	0.015	0.022	0.424	1.449
K6370-2	brea	panc,lung	L	0.017	0.017	0.033	0.081	0.008	0.097	0.601	0.013	0.085	0.011	0.016	0.02	0.443	1.442
K6371-2	pros	panc,lung	L	0.014	0.018	0.031	0.077	0.008	0.093	0.598	0.016	0.089	0.011	0.016	0.021	0.421	1.413
K6372-2	pros	panc,lung	L	0.012	0.016	0.034	0.077	0.007	0.092	0.598	0.019	0.088	0.011	0.015	0.022	0.41	1.401

Table A26: Classification results of *RNAseq_MU_RPM* data with a SVM model trained on log_2-transformed intensity data.

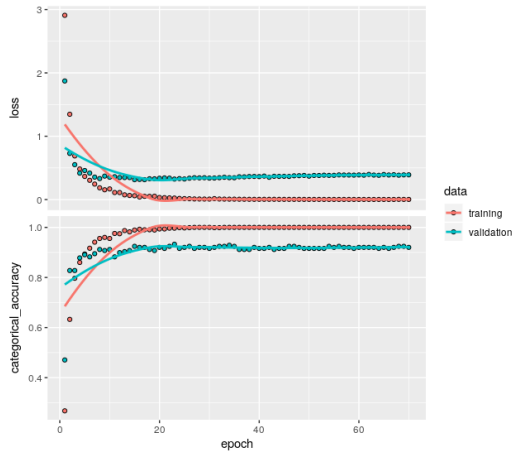
Sample	Actual	Pred	DM	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung	Total
K1999-6	colo	panc,lung,ovar	L	0.015	0.018	0.037	0.079	0.007	0.104	0.614	0.013	0.088	0.011	0.015	0.023	0.42	1.444
K2170-6	melan	panc,lung,rena	L	0.016	0.018	0.033	0.092	0.007	0.096	0.597	0.013	0.1	0.011	0.016	0.022	0.415	1.436
K2953-5	lung	panc,lung	L	0.015	0.018	0.036	0.08	0.007	0.097	0.565	0.013	0.088	0.011	0.016	0.023	0.42	1.389
K4824-4	panc	panc,lung	L	0.018	0.024	0.026	0.084	0.008	0.098	0.597	0.011	0.067	0.008	0.02	0.017	0.378	1.356
K5108-24	ovar	panc,lung,ovar	L	0.019	0.022	0.024	0.072	0.009	0.105	0.578	0.012	0.075	0.009	0.019	0.019	0.437	1.4
K5216-6	breast	panc,lung,ovar	L	0.02	0.016	0.033	0.076	0.007	0.11	0.592	0.013	0.082	0.011	0.015	0.021	0.443	1.439
K6266-2	mela	panc,lung,ovar	L	0.017	0.019	0.03	0.072	0.008	0.11	0.584	0.013	0.087	0.011	0.016	0.021	0.431	1.419
K6267-2	lung	panc,lung	L	0.013	0.022	0.034	0.083	0.007	0.087	0.588	0.013	0.081	0.011	0.014	0.02	0.403	1.376
K6268-2	panc	panc,lung	L	0.016	0.018	0.032	0.074	0.008	0.088	0.591	0.012	0.087	0.01	0.02	0.021	0.351	1.328
K6269-2	gast	panc,lung	M	0.015	0.017	0.038	0.086	0.008	0.094	0.625	0.012	0.077	0.01	0.017	0.02	0.374	1.393
K6270-2	gast	panc,mela,lung	H	0.01	0.01	0.067	0.221	0.005	0.087	0.726	0.011	0.034	0.014	0.015	0.01	0.194	1.404
K6271-2	rena	panc,lung	L	0.017	0.018	0.03	0.077	0.008	0.097	0.611	0.013	0.096	0.01	0.017	0.021	0.419	1.434
K6272-2	rena	panc,lung,rena	L	0.016	0.017	0.033	0.082	0.007	0.096	0.609	0.013	0.1	0.011	0.016	0.02	0.419	1.439
K6365-2	colo	panc,lung	L	0.015	0.021	0.034	0.079	0.007	0.091	0.606	0.013	0.091	0.011	0.015	0.022	0.426	1.431
K6366-2	colo	panc,lung	L	0.013	0.026	0.031	0.078	0.008	0.077	0.607	0.013	0.08	0.01	0.015	0.021	0.385	1.364
K6367-2	colo	panc,lung	L	0.015	0.02	0.032	0.077	0.008	0.09	0.619	0.013	0.083	0.01	0.016	0.022	0.412	1.417
K6368-2	lung	panc,lung	L	0.016	0.018	0.031	0.08	0.008	0.099	0.599	0.013	0.094	0.011	0.017	0.022	0.422	1.43
K6369-2	brea	panc,lung	L	0.017	0.016	0.036	0.082	0.007	0.097	0.617	0.013	0.091	0.012	0.015	0.022	0.424	1.449
K6370-2	brea	panc,lung	L	0.017	0.017	0.033	0.081	0.008	0.097	0.601	0.013	0.085	0.011	0.016	0.02	0.443	1.442
K6371-2	pros	panc,lung	L	0.014	0.018	0.031	0.077	0.008	0.093	0.598	0.016	0.089	0.011	0.016	0.021	0.421	1.413
K6372-2	pros	panc,lung	L	0.012	0.016	0.034	0.077	0.007	0.092	0.598	0.019	0.088	0.011	0.015	0.022	0.41	1.401

Training performance of neural networks:

The following figures show the classification performance of a neural network in terms of accuray and loss per completed epoch. Red represents the training data set and green the respective validation data. Generally loss should be minimized while accuracy should be maximized.

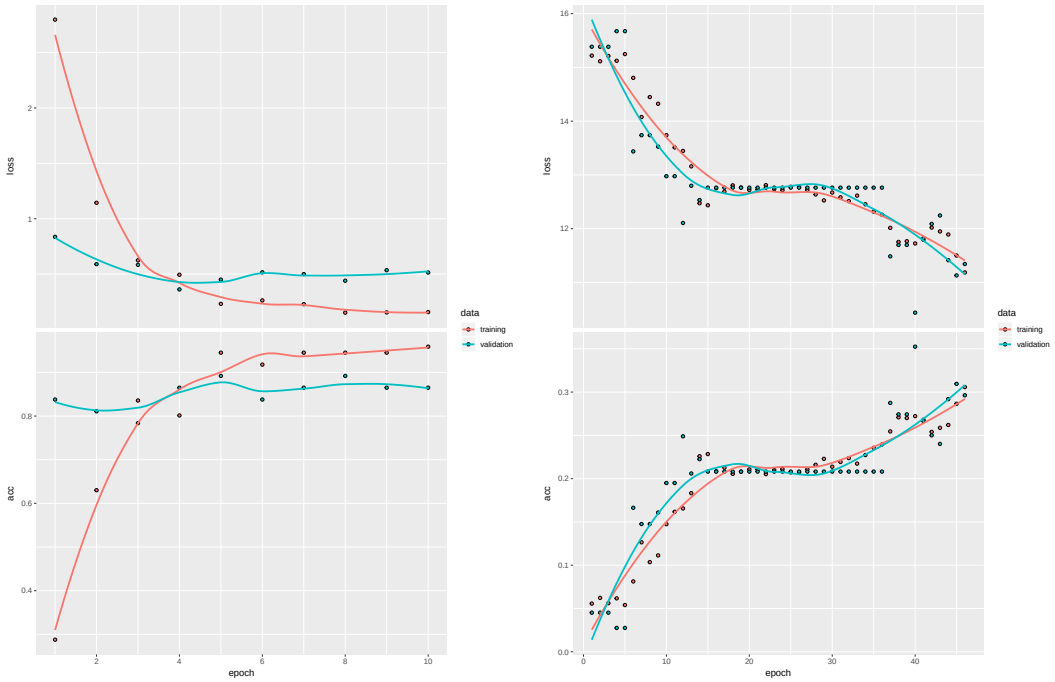


(a) Trainig of `model_CNN_RNAseq_19` with TCGA data `RNAseq_TCGA_19` (19 classes) using a CNN.



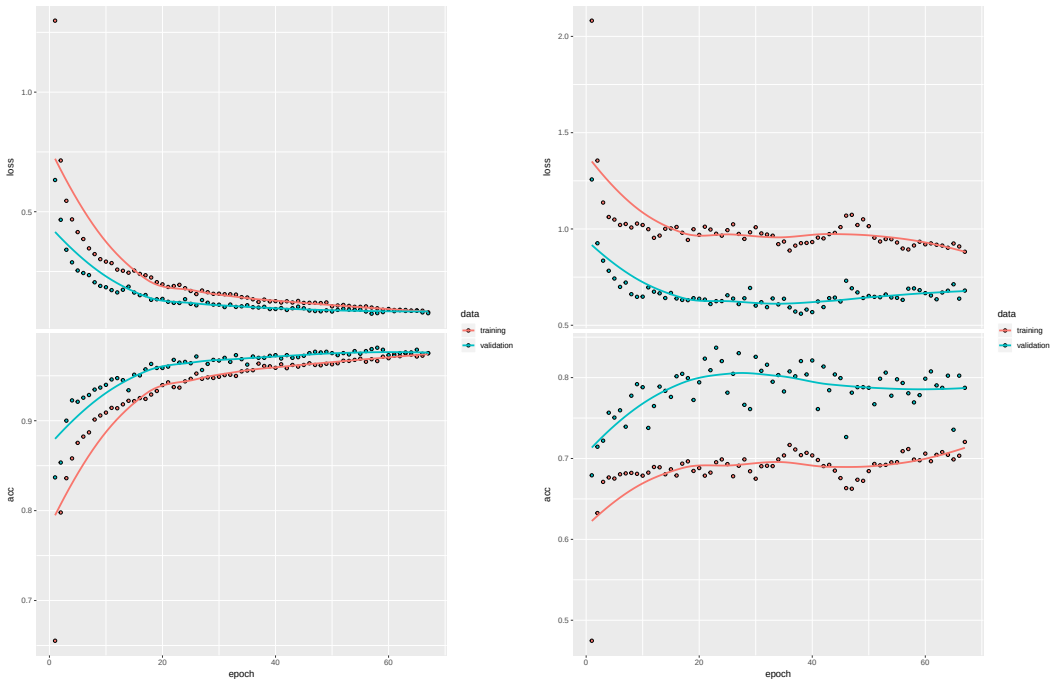
(b) Trainig of `model_CNN_RNAseq_33_balanced` with TCGA data `RNAseq_TCGA_19_balanced` (33 balanced classes) using a CNN.

Figure A1: Training and validation processes of of `model_CNN_RNAseq_19` and `model_CNN_RNAseq_33_balanced`.



(a) Trainig of `model_FNNN_Tothill_MA_13_mapped` with microarray data `MA_Tothill_13_mapped`.

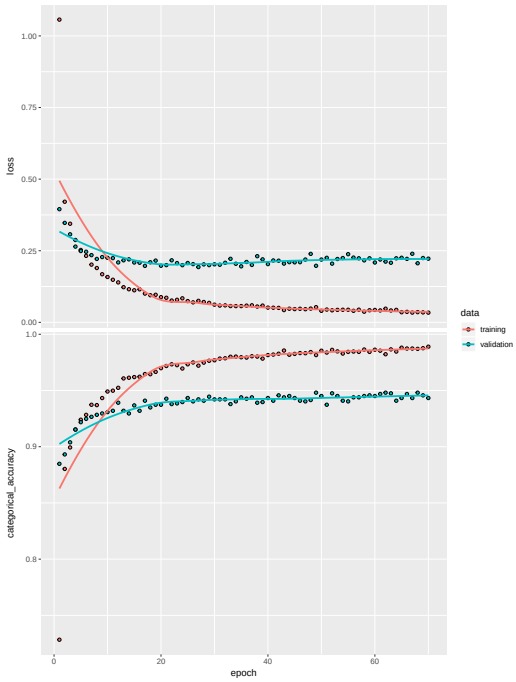
(b) Trainig of `model_FFNN_RNAseq_12` with TCGA data `RNAseq_TCGA_12` (12 classes) for comparison with Tothill *et al.* SVM performace.



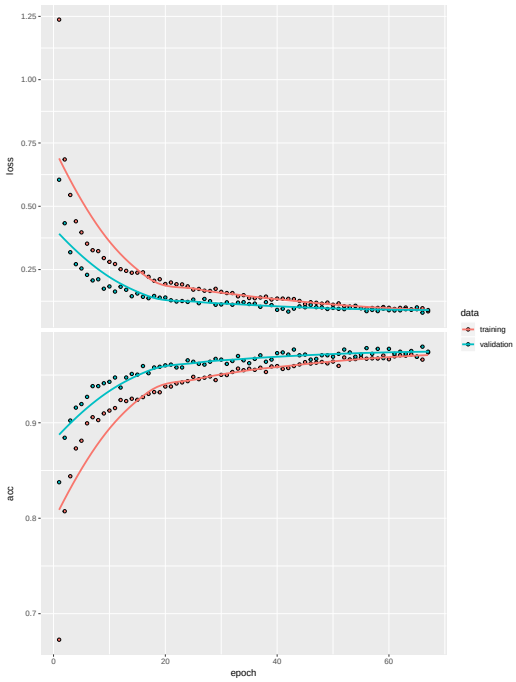
(c) Trainig of `model_FFNN_RNAseq_19` with TCGA data `RNAseq_TCGA_19` (19 classes) using a simple NN.

(d) Trainig of `model_DEEP_RNAseq_19` with TCGA data `RNAseq_TCGA_19` (19 classes) using a deep learning NN.

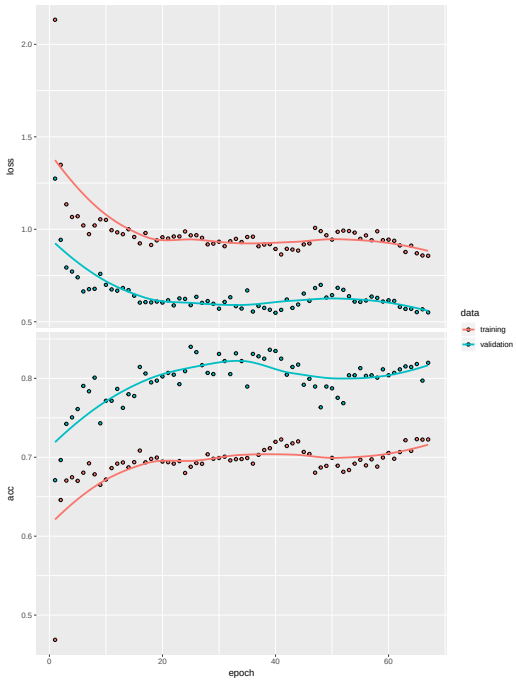
Figure A2: Training and validation processes of of `model_FNNN_Tothill_MA_1_mapped`, `model_FFNN_RNAseq_12`, `model_FFNN_RNAseq_19` and `model_DEEP_RNAseq_19`.



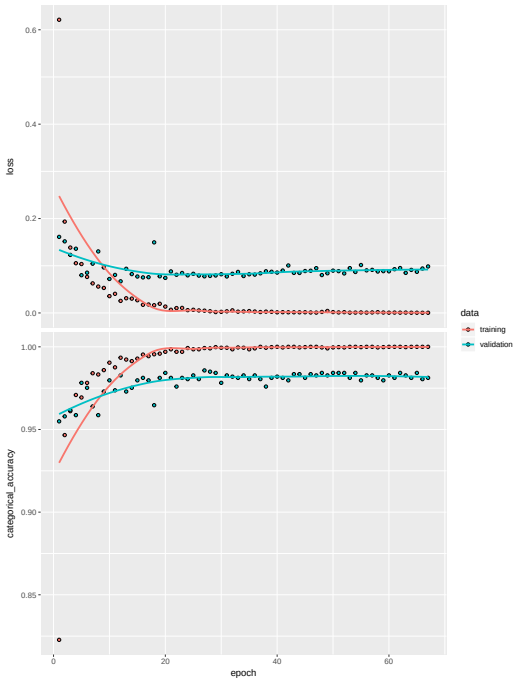
(a) Trainig of `model_CNN_RNAseq_33_mapped` with TCGA data `RNAseq_TCGA_33_mapped` (33 classes) using a CNN.



(b) Trainig of `model_FFNN_RNAseq_19_mapped` with TCGA data `RNAseq_TCGA_19_mapped` (19 classes) using a simple NN.



(c) Trainig of `model_DEEP_RNAseq_19_mapped` with TCGA data `RNAseq_TCGA_19_mapped` (19 classes) using a deep learning NN.



(d) Trainig of `model_CNN_RNAseq_19_mapped` with TCGA data `RNAseq_TCGA_19_mapped` (19 classes) using a CNN.

Figure A3: Training and validation processes of of `model_CNN_RNAseq_33_mapped`, `model_FFNN_RNAseq_19_mapped`, `model_DEEP_RNAseq_19_mapped` and `model_CNN_RNAseq_19_mapped`.

Confusion tables of neural network classifications:

In confusion tables the classification performance is visualized in matrix form, rows representing the predicted label and columns the true label of a class. Correct classifications are in the principal diagonal while misclassifications are not contained. A perfect model would show all classifications in the principal diagonal.

Table A27: Confusion table of the classification results of `model_FFNN_Tothill_MA_13_processed` using MA data *MA_Tothill_13_processed* resulting in a classification accuracy of 86.97%.

classes	brea	colo	gast	mela	meso	ovar	panc	pros	rena	test	SCCo	uter	lung
brea (7)	7[1.00]	0	0	0	0	0	0	0	0	0	0	0	0
colo (6)	0	5[0.83]	0	0	0	0	1	0	0	0	0	0	0
gast (2)	0	0	2[1.00]	0	0	0	0	0	0	0	0	0	0
mela (6)	0	0	0	6[1.00]	0	0	0	0	0	0	0	0	0
meso (1)	0	0	0	0	1[1.00]	0	0	0	0	0	0	0	0
ovar (5)	0	0	0	0	0	5[1.00]	0	0	0	0	0	0	0
panc (4)	0	0	0	0	0	0	4[1.00]	0	0	0	0	0	0
pros (3)	0	0	0	0	0	0	0	3[1.00]	0	0	0	0	0
rena (2)	1	0	0	0	0	0	0	0	1[0.50]	0	0	0	0
test (1)	0	0	0	0	0	0	0	0	0	1[1.00]	0	0	0
SCCo (1)	0	0	0	0	0	0	0	0	0	0	1[1.00]	0	0
uter (1)	0	0	0	0	0	0	0	0	0	0	0	1[1.00]	0
lung (7)	0	0	0	0	0	1	0	0	0	0	0	0	6[0.86]

Table A28: Confusion table of the classification results of *model_FFNN_RNAseq_19* using data *RNAseq_TCGA_19 resulting* in a classification accuracy of 97.42%.

classes	ACC	BLCA	BRCA	CESC	GBM	RENA	LAML	LGG	LIHC	LUNG	OV	PAAD	PRAD	SKCM	STAD	TGCT	THCA	UCS	UVM
ACC (24)	24[1.00]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BLCA (96)	0	86[0.90]	1	5	0	0	0	0	0	0	0	0	0	0	2	1	0	1	0
BLCA (229)	0	2	226[0.99]	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CESC (68)	0	2	0	66[0.97]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GBM (31)	0	0	0	0	27[0.87]	0	0	3	0	0	0	0	0	0	0	0	0	1	0
RENA (194)	1	1	0	0	0	192[0.99]	0	0	0	0	0	0	0	0	0	0	0	0	0
LAML (38)	0	0	0	0	0	0	38[1.00]	0	0	0	0	0	0	0	0	0	0	0	0
LGG (107)	0	0	0	0	3	0	0	104[0.97]	0	0	0	0	0	0	0	0	0	0	0
LIHC (79)	0	0	0	0	0	0	0	0	79[1.00]	0	0	0	0	0	0	0	0	0	0
LUNG (227)	0	4	0	3	0	0	0	0	1	216[0.95]	1	1	0	0	0	1	0	0	0
OV (61)	0	0	0	0	0	0	0	0	0	0	61[1.00]	0	0	0	0	0	0	0	0
PAAD (37)	0	0	0	0	0	0	0	0	0	0	0	36[0.97]	0	0	1	0	0	0	0
PRAD (115)	0	0	0	0	0	0	0	0	0	0	0	0	115[1.00]	0	0	0	0	0	0
SKCM (104)	0	1	0	0	0	0	0	1	0	0	0	0	0	100[0.96]	0	0	0	0	2
STAD (87)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	87[1.00]	0	0	0	0
TGCT (32)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32	0	0	0
THCA (106)	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	105[0.99]	0	0
UCS (11)	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	8[0.73]	0
UVM (17)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17[1.00]

Table A29: Confusion table of the classification results of *model_DEEP_RNAseq_19* using data *RNAseq_TCGA_19* resulting in a classification accuracy of 79.74%.

classes	ACC	BLCA	BRCA	CESC	GBM	RENA	LAML	LGG	LIHC	LUNG	OV	PAAD	PRAD	SKCM	STAD	TGCT	THCA	UCS	UVM
ACC (24)	24[1.00]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BLCA (96)	0	66[0.69]	0	0	0	1	0	0	0	7	0	0	0	0	22	0	0	0	0
BRCA (223)	0	6	187[0.84]	0	0	3	0	0	0	17	2	0	0	0	14	0	0	0	0
CESC (68)	0	29	0	0[0.00]	0	0	0	0	0	4	0	0	0	0	35	0	0	0	0
GBM (31)	0	0	0	0	0[0.00]	1	0	28	0	0	1	0	0	0	1	0	0	0	0
RENA (194)	0	1	0	0	0	191[0.98]	0	0	0	1	0	0	0	0	1	0	0	0	0
LAML (38)	0	0	0	0	0	0	38[1.00]	0	0	0	0	0	0	0	0	0	0	0	0
LGG (107)	0	0	0	0	0	0	0	106[0.99]	0	0	0	0	0	0	1	0	0	0	0
LIHC (79)	0	0	0	0	0	1	0	0	72[0.91]	0	0	0	0	0	6	0	0	0	0
LUNG (227)	0	17	0	0	0	1	0	0	0	188[0.83]	0	0	0	0	21	0	0	0	0
OV (61)	0	2	0	0	0	2	0	0	0	2	30[0.49]	0	0	0	25	0	0	0	0
PAAD (37)	0	13	0	0	0	5	0	0	0	0	0	0[0.00]	0	0	19	0	0	0	0
PRAD (115)	0	0	4	0	0	0	0	0	0	0	0	0	111[0.97]	0	0	0	0	0	0
SKCM (104)	0	0	0	0	0	0	0	0	0	0	2	0	0	96[0.92]	4	2	0	0	0
STAD (87)	0	8	0	0	0	1	0	0	0	0	1	0	0	0	77[0.89]	0	0	0	0
TGCT (32)	0	0	0	0	0	0	0	0	0	16	4	0	0	0	1	11[0.34]	0	0	0
THCA (106)	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	104[0.98]	0	0
UCS (11)	0	0	0	0	0	0	0	0	0	10	0	0	0	0	1	0	0	0[0.00]	0
UVM (17)	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0[0.00]

Table A30: Confusion table of the classification results of *model_CNN_RNAseq_19* using data *RNAseq_TCGA_19* resulting in a classification accuracy of 98.50%.

classes	ACC	BLCA	BRCA	CESC	GBM	RENA	LAML	LGG	LIHC	LUNG	OV	PAAD	PRAD	SKCM	STAD	TGCT	THCA	UCS	UVM
ACC (11)	11[1.00]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BLCA (81)	0	75[0.93]	0	2	0	0	0	0	0	2	0	0	0	2	0	0	0	0	0
BRCA (238)	0	0	238[1.00]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CESC (61)	0	1	0	58[0.95]	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
GBM (37)	0	0	1	0	34[0.92]	0	0	2	0	0	0	0	0	0	0	0	0	0	0
RENA (206)	0	0	0	1	0	205[0.99]	0	0	0	0	0	0	0	0	0	0	0	0	0
LAML (36)	0	0	0	0	0	0	36[1.00]	0	0	0	0	0	0	0	0	0	0	0	0
LGG (114)	0	0	0	0	1	0	0	112[0.98]	0	0	0	0	0	0	0	0	0	1	0
LIHC (90)	0	0	0	0	0	0	0	0	90[1.00]	0	0	0	0	0	0	0	0	0	0
LUNG (208)	0	0	0	1	0	0	0	0	0	206[0.99]	0	1	0	0	0	0	0	0	0
OV (59)	0	0	0	0	0	0	0	0	0	0	59[1.00]	0	0	0	0	0	0	0	0
PAAD (47)	0	1	0	0	0	0	0	0	0	0	0	45[0.96]	0	1	0	0	0	0	0
PRAD (117)	0	0	0	0	0	0	0	0	0	0	0	0	117[1.00]	0	0	0	0	0	0
SKCM (102)	1	0	0	0	0	0	0	0	0	2	0	0	0	98[0.96]	0	0	0	1	0
STAD (92)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	92[1.00]	0	0	0	0
TGCT (23)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23[1.00]	0	0	0
THCA (113)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	113[1.00]	0	0
UCS (8)	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	6[0.75]	0
UVM (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20[1.00]

Table A31: Confusion table of the classification results of *model_FFFN_RNAseq_19_mapped* using data *RNAseq_TCGA_19_mapped* resulting in a classification accuracy of 96.93%.

classes	ACC	BLCA	BRCA	CESC	GBM	RENA	LAML	LGG	LIHC	LUNG	OV	PAAD	PRAD	SKCM	STAD	TGCT	THCA	UCS	UVM
ACC (24)	24[1.00]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BLCA (96)	0	78[0.81]	0	2	0	0	0	0	0	11	0	1	0	0	2	2	0	0	0
BRCA (229)	0	1	227[0.99]	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CESC (68)	0	0	0	62[1.00]	0	0	0	0	0	3	2	0	0	0	0	0	0	1	0
GBM (31)	0	0	0	0	27[0.91]	0	0	4	0	0	0	0	0	0	0	0	0	0	0
RENA (194)	0	1	1	0	0	192[0.99]	0	0	0	0	0	0	0	0	0	0	0	0	0
LAML (38)	0	0	0	0	0	0	38[1.00]	0	0	0	0	0	0	0	0	0	0	0	0
LGG (107)	0	0	0	0	1	0	0	106[0.99]	0	0	0	0	0	0	0	0	0	0	0
LIHC (79)	0	0	0	0	0	0	0	0	79[1.00]	0	0	0	0	0	0	0	0	0	0
LUNG (227)	0	2	0	0	0	0	0	0	1	221[0.97]	1	2	0	0	0	0	0	0	0
OV (61)	0	0	0	0	0	0	0	0	0	0	61[1.00]	0	0	0	0	0	0	0	0
PAAD (37)	0	0	0	0	0	0	0	0	0	0	0	37[1.00]	0	0	0	0	0	0	0
PRAD (115)	0	0	0	0	0	0	0	0	0	0	0	0	115[1.00]	0	0	0	0	0	0
SKCM (104)	0	0	0	0	0	0	0	1	0	1	0	0	0	100[0.96]	0	0	0	0	2
STAD (87)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	87[1.00]	0	0	0	0
TGCT (32)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32[1.00]	0	0	0
THCA (106)	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	105[0.99]	0	0
UCS (11)	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	9[0.82]	0
UVM (17)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17[1.00]

Table A32: Confusion table of the classification results of *model_DEEP_RNAseq_19_mapped* using data *RNAseq_TCGA_19_mapped* resulting in a classification accuracy of 81.84%.

classes	ACC	BLCA	BRCA	CESC	GBM	RENA	LAML	LGG	LIHC	LUNG	OV	PAAD	PRAD	SKCM	STAD	TGCT	THCA	UCS	UVM
ACC (24)	24[1.00]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BLCA (96)	0	71[0.74]	2	2	0	0	0	0	0	6	7	0	0	1	5	2	0	0	0
BRCA (228)	0	2	208[0.92]	0	0	0	0	0	0	7	11	0	0	0	0	1	0	0	0
CESC (68)	0	33	2	7[0.10]	0	0	0	0	0	15	5	0	0	0	6	0	0	0	0
GBM (31)	0	0	0	0	0[0.00]	0	0	29	0	0	0	0	0	2	0	0	0	0	0
RENA (194)	0	1	0	0	0	189[0.97]	0	0	0	0	2	0	0	0	2	0	0	0	0
LAML (38)	0	0	0	0	0	0	38[1.00]	0	0	0	0	0	0	0	0	0	0	0	0
LGG (107)	0	0	0	0	0	0	0	106[0.99]	0	0	0	0	0	1	0	0	0	0	0
LIHC (79)	0	0	0	0	0	4	0	0	73[0.92]	0	0	0	0	0	2	0	0	0	0
LUNG (227)	0	13	0	2	0	0	0	0	1	187[0.82]	12	0	0	0	11	1	0	0	0
OV (61)	0	0	0	0	0	1	0	0	0	0	47[0.77]	0	0	0	13	0	0	0	0
PAAD (37)	0	1	0	0	0	2	0	0	0	0	4	0[0.00]	0	0	30	0	0	0	0
PRAD (115)	0	0	0	0	0	0	0	0	0	0	0	0	111[0.97]	0	4	0	0	0	0
SKCM (104)	0	0	0	0	0	0	0	0	0	2	0	0	0	102[0.98]	0	0	0	0	0
STAD (87)	0	2	0	0	0	0	0	0	0	0	17	0	0	0	68[0.78]	0	0	0	0
TGCT (32)	0	0	1	0	0	0	0	0	0	1	13	0	0	5	0	12[0.38]	0	0	0
THCA (106)	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	104[0.98]	0	0
UCS (11)	0	0	0	0	0	0	0	0	0	0	5	0	0	2	0	4	0	0[0.00]	0
UVM (17)	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0[0.00]

Table A33: Confusion table of the classification results of *model_CNN_RNAseq_19_mapped* using data *RNAseq_TCGA_19_mapped* resulting in a classification accuracy of 98.32%.

classes	ACC	BLCA	BRCA	CESC	GBM	RENA	LAML	LGG	LIHC	LUNG	OV	PAAD	PRAD	SKCM	STAD	TGCT	THCA	UCS	UVM
ACC (11)	11[1.00]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BLCA (81)	0	73[0.90]	1	2	0	0	0	0	0	3	0	0	0	1	0	0	0	1	0
BRCA (238)	0	0	238[1.00]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CESC (61)	0	1	0	57[0.93]	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0
GBM (37)	0	0	1	0	35[0.95]	0	0	1	0	0	0	0	0	0	0	0	0	0	0
RENA (206)	0	0	0	1	0	205[0.99]	0	0	0	0	0	0	0	0	0	0	0	0	0
LAML (36)	0	0	0	0	0	0	36[1.00]	0	0	0	0	0	0	0	0	0	0	0	0
LGG (114)	0	0	0	0	2	0	0	111[0.97]	0	0	0	0	0	0	0	0	0	1	0
LIHC (90)	0	0	0	0	0	0	0	0	90[1.00]	0	0	0	0	0	0	0	0	0	0
LUNG (208)	0	0	0	0	0	0	0	0	0	207[0.99]	0	1	0	0	0	0	0	0	0
OV (59)	0	0	0	0	0	0	0	0	0	0	58[0.98]	0	0	0	1	0	0	0	0
PAAD (47)	0	0	0	0	0	1	0	0	0	0	0	45[0..96]	0	1	0	0	0	0	0
PRAD (117)	0	0	0	0	0	0	0	0	0	0	0	0	117[1.00]	0	0	0	0	0	0
SKCM (102)	1	0	0	0	0	0	0	0	0	2	0	0	0	98[0.96]	0	0	0	1	0
STAD (92)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	92[1.00]	0	0	0	0
TGCT (23)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23[1.00]	0	0	0
THCA (113)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	113[1.00]	0	0
UCS (8)	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	6[0.75]	0
UVM (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20[1.00]

Table A34: Confusion table of the classification results of *model_CNN_RNAseq_33_mapped* using data *RNAseq_TCGA_33_mapped* resulting in a classification accuracy of 93.88%.

classes	ACC	BLCA	*BRCA*	CESC	CHOL	COAD	DLBC	ESCA	GBM	HNSC	KICH	KIRC	KIRP	LAML	LGG	LIHC	LUAD	LUSC	MESO	OV	PAAD	PCPG	PRAD	READ	SARC	SKCM	STAD	TGCTT	THCA	THYM	UCEC	UCS	UVM
ACC (13)	12[0.92]	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BLCA (95)	0	86[0.91]	2	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0
BRCA (259)	0	1	255[0.98]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
CESC (64)	0	2	0	55[0.86]	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	3	0	0
CHOL (9)	0	0	0	0	7[0.78]	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
COAD (72)	0	0	0	0	0	63[0.86]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0
DLBC (10)	0	0	0	0	0	0	10[1.00]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ESCA (37)	0	0	0	0	0	0	0	27[0.73]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0
GBM (35)	0	0	0	0	0	0	0	0	31[0.89]	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
HNSC (108)	0	0	0	0	0	0	0	0	0	106[0.98]	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
KICH (14)	0	0	0	0	0	0	0	0	0	0	10[0.71]	0	3	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
KIRC (122)	0	0	1	0	0	0	0	0	0	0	4	111[0.91]	5	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
KIRP (74)	0	0	0	0	0	0	0	0	0	0	1	3	70[0.95]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LAML (24)	0	0	0	0	0	0	0	0	0	0	0	0	0	24[1.00]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LGG (99)	0	0	0	0	0	0	0	0	2	0	0	0	0	0	96[0.97]	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
LIHC (95)	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	91[0.96]	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
LUAD (119)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	106[0.89]	11	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
LUSC (110)	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	9	97[0.88]	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
MESO (19)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	17[0.89]	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OV (64)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	63[0.98]	0	0	0	0	1	0	0	0	0	0	0	0	0
PAAD (32)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32[1.00]	0	0	0	0	0	0	0	0	0	0	0	0
PCPG (31)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31[1.00]	0	0	0	0	0	0	0	0	0	0	0
PRAD (99)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	99[1.00]	0	0	0	0	0	0	0	0	0	0
READ (18)	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12[0.67]	0	0	0	0	0	0	0	0	0
SARC (46)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	45[0.98]	0	0	0	0	0	0	1	0
SKCM (105)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	101[0.96]	0	0	0	0	0	0	2
STAD (89)	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	85[0.96]	0	0	0	0	0	0
TGCTT (27)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27[1.00]	0	0	0	0	0
THCA (98)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	98[1.00]	0	0	0	0
THYM (28)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28[1.00]	0	0	0
UCEC (44)	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	41[0.93]	1	0
UCS (16)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	11[0.69]	0
UVM (18)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18[1.00]

Table A35: Confusion table of the classification results of *model_CNN_RNAseq_33_balanced* using data *RNAseq_TCGA_33_balanced* resulting in a classification accuracy of 89.90%.

classes	ACC	BLCA	*BRCA*	CESC	CHOL	COAD	DLBC	ESCA	GBM	HNSC	KICH	KIRC	KIRP	LAML	LGG	LIHC	LUAD	LUSC	MESO	OV	PAAD	PCPG	PRAD	READ	SARC	SKCM	STAD	TGCTT	THCA	THYM	UCEC	UCS	UVM
ACC (6)	5[0.83]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
BLCA (14)	0	13[0.93]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BRCA (9)	0	0	9[1.00]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CESC (8)	0	0	0	6[0.75]	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
CHOL (8)	0	0	0	0	7[0.88]	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
COAD (8)	0	0	0	0	0	8[1.00]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DLBC (7)	0	0	0	0	0	0	7[1.00]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ESCA (9)	0	0	0	0	0	0	0	9[1.00]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GBM (8)	0	0	0	0	0	0	0	0	7[0.88]	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HNSC (13)	0	0	0	1	0	0	0	0	0	12[0.92]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
KICH (12)	0	0	0	0	0	0	0	0	0	0	11[0.92]	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
KIRC (6)	0	0	0	0	0	0	0	0	0	0	1	5[0.83]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
KIRP (10)	0	0	0	0	0	0	0	0	0	0	1	0	9[0.90]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LAML (11)	0	0	0	0	0	0	0	0	0	0	0	0	0	11[1.00]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LGG (9)	0	0	0	0	0	0	0	0	2	0	0	0	0	0	7[0.78]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LIHC (8)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	7[0.88]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LUAD (12)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11[0.92]	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
LUSC (9)	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	6[0.67]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MESO (5)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5[1.00]	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OV (14)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13[0.93]	0	0	0	0	0	0	0	0	0	1	0	0	0
PAAD (4)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4[1.00]	0	0	0	0	0	0	0	0	0	0	0	0
PCPG (8)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8[1.00]	0	0	0	0	0	0	0	0	0	0	0
PRAD (8)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8[1.00]	0	0	0	0	0	0	0	0	0	0
READ (16)	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11[0.67]	0	0	0	0	0	0	0	0	0
SARC (12)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10[0.83]	0	0	0	0	0	2	0	0
SKCM (12)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10[0.83]	0	0	0	0	0	1	0
STAD (9)	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8[0.88]	0	0	0	0	0	0
TGCTT (4)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4[1.00]	0	0	0	0	0
THCA (7)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7[1.00]	0	0	0	0
THYM (3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3[1.00]	0	0	0
UCEC (5)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4[0.80]	1	0
UCS (15)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	14[0.93]	0
UVM (8)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8[1.00]

Table A36: Overview of misclassifications of a neural network classifying TCGA RNA-seq data with common MU Graz features.

sample	predicted_labels	true_labels
TCGA.BT.A20V.01A.11R.A14Y.07	CESC	BLCA
TCGA.C4.A0F7.01A.11R.A084.07	LUSC	BLCA
TCGA.GD.A3OS.01A.12R.A220.07	UCS	BLCA
TCGA.K4.A5RH.01A.11R.A30C.07	SARC	BLCA
TCGA.XF.AAMH.01A.11R.A42T.07	CESC	BLCA
TCGA.AC.A2QH.01A.11R.A18M.07	UCS	BRCA
TCGA.AC.A7VC.01A.11R.A352.07	SARC	BRCA
TCGA.EA.A556.01A.11R.A26T.07	UCEC	CESC
TCGA.MA.AA3Y.01A.11R.A38B.07	BLCA	CESC
TCGA.R2.A69V.01A.11R.A32P.07	BLCA	CESC
TCGA.VS.A8QH.01A.11R.A37O.07	READ	CESC
TCGA.VS.A9V0.01A.11R.A42T.07	LUAD	CESC
TCGA.W5.AA2G.01A.11R.A41I.07	LIHC	CHOL
TCGA.W5.AA2U.11A.11R.A41I.07	LIHC	CHOL
TCGA.W5.AA2X.11A.11R.A41I.07	LIHC	CHOL
TCGA.W5.AA30.11A.11R.A41I.07	LIHC	CHOL
TCGA.W5.AA36.01A.11R.A41I.07	LIHC	CHOL
TCGA.A6.5662.01A.01R.1653.07	READ	COAD
TCGA.A6.A565.01A.31R.A28H.07	READ	COAD
TCGA.AA.3660.01A.01R.1723.07	READ	COAD
TCGA.AY.5543.01A.01R.1653.07	READ	COAD
TCGA.AZ.4682.01B.01R.A32Z.07	READ	COAD
TCGA.AZ.6603.01A.11R.1839.07	READ	COAD
TCGA.CK.4952.01A.01R.1723.07	READ	COAD
TCGA.CM.5349.01A.21R.1723.07	READ	COAD
TCGA.CM.5864.01A.01R.1653.07	READ	COAD
TCGA.CM.5868.01A.01R.1653.07	READ	COAD
TCGA.CM.6163.01A.11R.1653.07	READ	COAD
TCGA.CM.6172.01A.11R.1653.07	READ	COAD
TCGA.DM.A1D0.01A.11R.A155.07	READ	COAD
TCGA.DM.A1D7.01A.11R.A155.07	READ	COAD
TCGA.DM.A1D9.01A.11R.A155.07	READ	COAD
TCGA.DM.A285.01A.11R.A16W.07	READ	COAD
TCGA.F4.6704.11A.01R.1839.07	READ	COAD

Table A36: Overview of misclassifications of a neural network classifying TCGA RNA-seq data with common MU Graz features.

sample	predicted_labels	true_labels
TCGA.G4.6311.01A.11R.1723.07	READ	COAD
TCGA.QL.A97D.01A.12R.A41B.07	READ	COAD
TCGA.2H.A9GI.01A.11R.A37I.31	STAD	ESCA
TCGA.JY.A938.01A.11R.A37I.31	STAD	ESCA
TCGA.JY.A939.01A.12R.A37I.31	STAD	ESCA
TCGA.KH.A6WC.01A.11R.A336.31	STAD	ESCA
TCGA.L5.A4OF.01A.11R.A260.31	STAD	ESCA
TCGA.L5.A4OP.01A.11R.A260.31	STAD	ESCA
TCGA.L5.A8NE.01A.11R.A37I.31	STAD	ESCA
TCGA.L5.A8NG.01A.11R.A37I.31	STAD	ESCA
TCGA.L5.A8NR.01A.11R.A37I.31	STAD	ESCA
TCGA.L5.A8NU.01A.11R.A36D.31	STAD	ESCA
TCGA.R6.A6KZ.01A.11R.A31P.31	STAD	ESCA
TCGA.R6.A6Y2.01B.11R.A336.31	STAD	ESCA
TCGA.V5.AASX.01A.11R.A38D.31	STAD	ESCA
TCGA.VR.AA4D.01A.11R.A37I.31	STAD	ESCA
TCGA.ZR.A9CJ.01B.11R.A38D.31	STAD	ESCA
TCGA.06.AABW.11A.31R.A36H.07	LGG	GBM
TCGA.BA.7269.01A.11R.2016.07	LUSC	HNSC
TCGA.KL.8329.11A.01R.2315.07	KIRP	KICH
TCGA.KN.8427.01A.11R.2315.07	SKCM	KICH
TCGA.KN.8430.11A.01R.2315.07	KIRP	KICH
TCGA.KN.8433.11A.01R.2315.07	KIRP	KICH
TCGA.KN.8435.11A.01R.2315.07	KIRP	KICH
TCGA.KO.8403.01A.11R.2315.07	KIRC	KICH
TCGA.6D.AA2E.01A.11R.A37O.07	KIRP	KIRC
TCGA.A3.3363.01A.01R.0864.07	KIRP	KIRC
TCGA.AK.3433.01A.02R.1277.07	KICH	KIRC
TCGA.AK.3440.01A.02R.1277.07	KICH	KIRC
TCGA.AK.3453.01A.02R.1277.07	KIRP	KIRC
TCGA.B0.5107.01A.01R.1420.07	KIRP	KIRC
TCGA.B0.5707.01A.11R.1541.07	KIRP	KIRC
TCGA.B8.A54E.01A.11R.A266.07	KIRP	KIRC
TCGA.5P.A9KH.01A.11R.A42S.07	KICH	KIRP

Table A36: Overview of misclassifications of a neural network classifying TCGA RNA-seq data with common MU Graz features.

sample	predicted_labels	true_labels
TCGA.AL.3466.01A.02R.1351.07	KIRC	KIRP
TCGA.BQ.5887.11A.01R.1965.07	KICH	KIRP
TCGA.HE.7130.01A.11R.1965.07	BLCA	KIRP
TCGA.MH.A857.01A.11R.A355.07	KICH	KIRP
TCGA.CC.A9FV.01A.11R.A37K.07	SARC	LIHC
TCGA.FV.A3I0.01A.11R.A22L.07	CHOL	LIHC
TCGA.05.4415.01A.22R.1858.07	LUSC	LUAD
TCGA.05.5420.01A.01R.1628.07	LUSC	LUAD
TCGA.38.4627.11A.01R.1758.07	LUSC	LUAD
TCGA.38.4631.01A.01R.1755.07	LUSC	LUAD
TCGA.44.7670.01A.11R.2066.07	LUSC	LUAD
TCGA.49.4490.11A.01R.1858.07	LUSC	LUAD
TCGA.55.6968.01A.11R.1949.07	LUSC	LUAD
TCGA.21.1078.01A.01R.0692.07	BLCA	LUSC
TCGA.22.4596.01A.01R.1201.07	LUAD	LUSC
TCGA.22.4609.11A.01R.2125.07	LUAD	LUSC
TCGA.22.5489.11A.01R.1635.07	LUAD	LUSC
TCGA.33.4566.01A.01R.1443.07	LUAD	LUSC
TCGA.33.4587.01A.11R.2125.07	CESC	LUSC
TCGA.33.AASB.01A.11R.A405.07	LUAD	LUSC
TCGA.39.5034.01A.01R.1443.07	LUAD	LUSC
TCGA.43.5670.11A.01R.2125.07	LUAD	LUSC
TCGA.56.7730.11A.01R.2125.07	LUAD	LUSC
TCGA.56.8201.11A.01R.2247.07	LUAD	LUSC
TCGA.56.8309.11A.01R.2296.07	LUAD	LUSC
TCGA.77.7338.11A.01R.2045.07	LUAD	LUSC
TCGA.77.8007.01A.11R.2187.07	LUAD	LUSC
TCGA.77.8007.11A.01R.2187.07	LUAD	LUSC
TCGA.90.7964.01A.21R.2187.07	LUAD	LUSC
TCGA.96.A4JL.01A.11R.A24Z.07	THYM	LUSC
TCGA.O2.A5IB.01A.11R.A27Q.07	BRCA	LUSC
TCGA.H6.A45N.11A.12R.A26U.07	SKCM	PAAD
TCGA.L1.A7W4.01A.12R.A36G.07	BLCA	PAAD
TCGA.HC.7740.11A.01R.2118.07	THCA	PRAD

Table A36: Overview of misclassifications of a neural network classifying TCGA RNA-seq data with common MU Graz features.

sample	predicted_labels	true_labels
TCGA.AF.5654.11A.11R.1660.07	COAD	READ
TCGA.EI.6883.01A.31R.1928.07	COAD	READ
TCGA.F5.6465.01A.11R.1736.07	COAD	READ
TCGA.F5.6861.01A.11R.1928.07	COAD	READ
TCGA.DX.AB2J.01A.11R.A38C.07	TGCTT	SARC
TCGA.FX.A2QS.11A.11R.A21T.07	KIRP	SARC
TCGA.IW.A3M5.01A.22R.A21T.07	UCS	SARC
TCGA.D3.A8GE.06A.11R.A37K.07	SARC	SKCM
TCGA.ER.A2NF.06A.11R.A18T.07	UVM	SKCM
TCGA.ER.A42L.06A.11R.A24X.07	ACC	SKCM
TCGA.W3.A828.06A.11R.A352.07	LUAD	SKCM
TCGA.YD.A9TA.06A.11R.A39D.07	LUSC	SKCM
TCGA.CD.8531.01A.11R.2343.13	DLBC	STAD
TCGA.IN.AB1X.01A.11R.A39E.31	ESCA	STAD
TCGA.A5.A1OH.01A.21R.A22K.07	UCS	UCEC
TCGA.AJ.A3NG.01A.11R.A22K.07	UCS	UCEC
TCGA.AX.A3FS.01A.11R.A22K.07	UCS	UCEC
TCGA.AX.A3FT.01A.11R.A22K.07	CESC	UCEC
TCGA.D1.A3JP.01A.31R.A22K.07	PAAD	UCEC
TCGA.N7.A59B.01A.11R.A28V.07	UCEC	UVM
TCGA.N8.A4PM.01A.11R.A28V.07	UCEC	UVM

Microarray Analysis
- GenePix Professional 4200A
- GenePix Autoloader 4200AL
- GenePix 4000B
- GenePix Personal 4100A
- GenePix Comparison Table

- Acuity 3.1
- GenePix Pro 5.0

- GenePix Scanner FAQ
- Acuity FAQ
- Innovative Uses
- Training Seminars

Product Support Pages
- GenePix
- Acuity
- GenePix File Formats
- GAL (Array List) Examples
- GPR Format History
- Developer Info

GenePix® File Formats

GenePix Pro recognizes and uses several different file types:

ATF - Axon Text File format (*.atf)

ATF is a tab-delimited text file format that can be read by typical spreadsheet programs such as Microsoft Excel. It is used for GenePix Array List (GAL) files, and GenePix Results (GPR) files.

An ATF text file consists of **records**. Each line in the text file is a record. Each record may consist of several **fields**, separated by a **field separator** (column delimiter). The tab and comma characters are field separators. Space characters around a tab or comma are ignored and considered part of the field separator. Text strings are enclosed in **quotation marks** to ensure that any embedded spaces, commas and tabs are not mistaken for field separators.

The group of records at the beginning of the file is called the **file header**. The file header describes the file structure and includes column titles, units, and comments.

ATF File Structure

```
First header record     Format: ATF (all caps), Version number
Second header record    Number of optional header records n,
                        Number of data columns (fields) m
1st optional record     ...
2nd optional record     ...
nth optional record     ...
(n+3)th record          Required record containing m fields.
                        Each field contains a column title.
DATA RECORDS            Arranged in m columns (fields) of data.
```

See below under GenePix Array List format for an example of an ATF file.

GAL - GenePix Array List format (*.gal)

Introduction

Download a sample GAL file.
See also:
Making GenePix Array List Files Application Note. [PDF] 281 KB
GAL File Examples specifically for **array and arrayer manufacturers**.

GenePix Array List files describe the size and position of blocks, the layout of feature-indicators in them, and the names and identifiers of the printed substances associated with each feature-indicator.

GenePix Pro includes an integrated Array List Generator which generates GAL files from plain text files; see the GenePix Pro online Help for details.

GAL files conform to the Axon Text File (ATF) format described above. As such, they can be created in Microsoft Excel by saving an Excel spreadsheet as Text (Tab delimited).

To create a GAL file that describes block and feature-indicator positions and geometry, but without substance IDs or names, save a settings file using the Save Settings As command in GenePix Pro (select *.gal as the output file type).

GAL files consist of two sections: the header, and data records. The header contains all the structural and positional information about the blocks; the data records contain all the name and identifier information for each spot.

GenePix Pro assigns block numbers such that the top leftmost block on the image is block #1, and the block numbers increase from left to right and then from top to bottom:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

The order in which a block was created does not matter; GenePix Pro automatically renumbers all blocks to follow this rule.

Example GenePix Array List (GAL) file

The following very simple array list file describes four blocks ("BlockCount=4"), each with 24 columns and 5 rows. For simplicity, we have included the data record information (name, ID, etc) only for the first two features:

```
ATF		1.0
8		5
"Type=GenePix ArrayList V1.0"
"BlockCount=4"
"BlockType=0"
"URL=http://genome-www.stanford.edu/cgi-bin/dbrun/SacchDB?find+Locus+%22[ID]%22"
"Block1= 400, 400, 100, 24, 175, 5, 175"
"Block2= 4896, 400, 100, 24, 175, 5, 175"
"Block3= 400, 4896, 100, 24, 175, 5, 175"
"Block4= 4896, 4896, 100, 24, 175, 5, 175"
"Block"	"Column"	"Row"	"Name"	"ID"
 1		1		1		VPS8		YAL002W
 1		2		1		NTG1		YAL015C
```

Description of header records

The header section describes basic file information and all block properties apart from names and IDs (which are in the data records section). Each record is explained below:

ATF	**1.0**	**(Required)** First line of an ATF file; the same in all GAL files: File format (ATF) and version (1.0).
8	**5**	**(Required)** Second line of an ATF file: 8 (number of optional header records). 5 (number of data columns).

"Type=GenePix Array List v1.0"	**(Required)** Type of file; the same in all GAL files.
"BlockCount=4"	**(Optional)** Number of blocks described in the file.
"BlockType=0"	**(Optional)** Type of block described: 0 = rectangular. 1 = orange-packing #1. 2 = orange-packing #2.
"URL=..."	**(Optional)** The URL for the Go To Web command.
"Supplier=CompanyXYZ"	**(Optional)** The manufacturer that supplied the array or arrayer.
"ArrayerSoftwareName=Printer Robot User Interface"	**(Optional)** The name of the arrayer software.
"ArrayerSoftwareVersion=1.1"	**(Optional)** The version number of the arrayer software.
"ArrayName=MouseApoptosisProteins 4000"	**(Optional)** The name of an array as supplied by an array manufacturer.
"ArrayRevision=2.7"	**(Optional)** The version of an array as supplied by an array manufacturer.
"SlideBarcode=abc0011, abc0012, abc0013"	**(Optional)** Barcodes supported by the GAL file, used for barcode-driven automation.
"Block*n*="	**(Optional)** The position and dimensions of each block. There is one record for each block, and each record contains 7 fields. Each field is separated by a comma followed by a space. *xOrigin* — X position of center of top leftmost feature of current block (in µm). *yOrigin* — Y position of center of top leftmost feature of current block (in µm). *FeatureDiameter* — Diameter of features within the current block (in µm). *xFeatures* — Number of columns of features in current block. *xSpacing* — Column spacing of current block (in µm). *yFeatures* — Number of rows of features in current block. *ySpacing* — Row spacing of current block (in µm). **Note:** Positions on arrays are measured in microns with respect to the origin, which is the top left corner of the array.
"User Defined"	**(Optional)** You may include any number of correctly formatted extra lines in the header. When the GAL file is read as input by GenePix Pro 4.1, these will be passed to the output Results (GPR) file.
"Block" "Column" "Row" "Name" "ID"	**(Required)** Last line of the header, containing column titles for the data records. The quotation marks are advised, but not necessary.

Description of data records

The Data Record section contains records which describe each feature in detail. It includes the block, column, and row numbers for features, as well as descriptive name and identifier information. The GAL Data Record may also optionally contain user-defined fields (column titles) for extra annotation information that you may wish to include.

In GenePix Pro 4.1, any user-defined GAL file data columns are read and output to the Results (GPR) file; in earlier versions they are ignored. Also new in GenePix Pro 4.1 is

that there is no longer a 40-character limit on the *Name* and *ID* fields; entries longer than 40 characters are truncated when read by earlier versions.

There is one record for each feature, containing a field for each of the descriptive columns:

Block	**(Required)** The block number for the feature (required).
Column	**(Required)** The column location within the block (required).
Row	**(Required)** The row location within the block (required).
Name	**(Optional)** Name to be displayed for the given feature (optional; limited to 40 characters in GenePix Pro 4.0 and earlier, no limit in 4.1).
ID	**(Required)** Identifier for each feature (required; limited to 40 characters in GenePix Pro 4.0 and earlier, no limit in 4.1).
"User Defined"	**(Optional)** Annotation information (optional).

Block, Column, Row and ID are required fields. The column titles can be in any order.

Note: If you have empty features, use 'empty' as the feature ID, and the feature is flagged *absent* when the GAL file is opened by GenePix.

A minimal GAL file header

Because most of the GAL file header records are optional, it is relatively simple to construct a GAL file with a very minimal header. The following example also leaves out the Name column, which is also optional:

```
ATF        1.0
1          4
"Type=GenePix ArrayList V1.0"
"Block"  "Column"  "Row"  "ID"
 1         1          1      YAL002W
 1         2          1      YAL015C
```

When you open this GAL file in GenePix, you will be prompted with the New Blocks dialog box to enter block properties. You may find this method of configuring blocks via the New Blocks dialog box more convenient than working out block arrangements by hand.

GPR - GenePix Results format (*.gpr)

GenePix Results data are saved as GPR files, which are in Axon Text File (ATF) format. A Results file contains general information about image acquisition and analysis, as well as the data extracted from each individual feature. Any user-defined feature data contained in a GAL file read by GenePix Pro 4.1 will be included in the output GPR file. As of GenePix Pro version 4.0.1.4, the GPR version number is 3.0.

Read a history of the changes to the GPR file format since GenePix Pro 3, including example GPR files in all the various formats.

GPR Header

A sample GPR file header and a description of each entry are shown below:

Entry	Description
ATF 1.0	File type and version number.
29 48	Number of optional header records and number of data fields (columns).
"Type=GenePix Results 3"	Type of ATF file.
"DateTime=2002/02/09 17:15:48"	Date and time when the image was acquired.
"Settings=C:\Genepix\Genepix.gps"	The name of the settings file that was used for analysis.
"GalFile=C:\Genepix\Demo.gal"	The GenePix Array List file used to associate

	Names and IDs to each entry.
"PixelSize=10"	Resolution of each pixel in µm.
"Wavelengths=635 532"	Installed laser excitation sources in nm.
"ImageFiles=C:\Genepix\demo.tif 0 C:\Genepix\Genepix.tif 1"	The name and path of the associated TIF file(s).
"NormalizationMethod=None"	The type of normalization method used, if applicable.
"NormalizationFactors=1 1"	The normalization factor applied to each channel.
"JpegImage=C:\Genepix\demo.jpg"	The name and path of the associated Jpeg image files.
"StdDev=Type 1"	The type of standard deviation calculation selected in the Options settings.
"RatioFormulation=W1/W2 (635/532)"	The ratio formulation of the ratio image, showing which image is numerator and which is denominator.
"Barcode=00331"	The barcode symbols read from the image.
"BackgroundSubtraction=LocalFeature"	The background subtraction method selected in the Options settings.
"ImageOrigin=0, 0"	The origin of the image relative to the scan area.
"JpegOrigin=390, 4320"	The origin of the Results JPEG image (the bounding box of the analysis Blocks) relative to the scan area origin.
"Creator=GenePix 4.1.1.4"	The version of the GenePix Pro software used to create the Results file.
"Scanner=GenePix 4000B [*serial number*]"	Type and serial number of scanner used to acquire the image.
"FocusPosition=0"	The focus position setting used to acquire the image, in microns.
"Temperature=19.6127"	The temperature of the scanner, in degrees C.
"LinesAveraged=1"	The line average setting used to acquire the image.
"Comment=hyb 2673"	User-entered file comment.
"PMTGain=500 600"	The PMT settings during acquisition.
"ScanPower=100 100"	The amount of laser transmission during acquisition.
"LaserPower=1 1"	The power of each laser, in volts.
"LaserOnTime=5 5"	The laser on-time for each laser, in minutes.
"Filters=<Empty> <Empty>"	Emission filters used during acquisition (GenePix 4100 and 4200 only.)
"ScanRegion=100,100,2000,2000"	The coordinate values of the scan region used during acquisition, in pixels.
"Supplier="	Header field supplied in GAL file.
Data record column headings	Column titles for each measurement (see below).
Data Records	Extracted data.

GPR Data

The list below describes each column of data in the Results file.

Column Title	Description
Block	the block number of the feature.
Column	the column number of the feature.
Row	the row number of the feature.
Name	the name of the feature derived from the Array List (up to 40 characters

	long, contained in quotation marks).
ID	the unique identifier of the feature derived from the Array List (up to 40 characters long, contained in quotation marks).
X	the X-coordinate in µm of the center of the feature-indicator associated with the feature, where (0,0) is the top left of the image.
Y	the Y-coordinate in µm of the center of the feature-indicator associated with the feature, where (0,0) is the top left of the image.
Dia.	the diameter in µm of the feature-indicator.
F635 Median	median feature pixel intensity at wavelength #1 (635 nm).
F635 Mean	mean feature pixel intensity at wavelength #1 (635 nm).
F635 SD	the standard deviation of the feature pixel intensity at wavelength #1 (635 nm).
B635 Median	the median feature background intensity at wavelength #1 (635 nm).
B635 Mean	the mean feature background intensity at wavelength #1 (635 nm).
B635 SD	the standard deviation of the feature background intensity at wavelength #1 (635 nm).
% > B635 + 1 SD	the percentage of feature pixels with intensities more than one standard deviation above the background pixel intensity, at wavelength #1 (635 nm).
% > B635 + 2 SD	the percentage of feature pixels with intensities more than two standard deviations above the background pixel intensity, at wavelength #1 (635 nm).
F635 % Sat.	the percentage of feature pixels at wavelength #1 that are saturated.
F532 Median	median feature pixel intensity at wavelength #2 (532 nm).
F532 Mean	mean feature pixel intensity at wavelength #2 (532 nm).
F532 SD	the standard deviation of the feature intensity at wavelength #2 (532 nm).
B532 Median	the median feature background intensity at wavelength #2 (532 nm).
B532 Mean	the mean feature background intensity at wavelength #2 (532 nm).
B532 SD	the standard deviation of the feature background intensity at wavelength #2 (532 nm).
% > B532 + 1 SD	the percentage of feature pixels with intensities more than one standard deviation above the background pixel intensity, at wavelength #2 (532 nm).
% > B532 + 2 SD	the percentage of feature pixels with intensities more than two standard deviations above the background pixel intensity, at wavelength #2 (532 nm).
F532 % Sat.	the percentage of feature pixels at wavelength #2 that are saturated.
Ratio of Medians	the ratio of the median intensities of each feature for each wavelength, with the median background subtracted.
Ratio of Means	the ratio of the arithmetic mean intensities of each feature for each wavelength, with the median background subtracted.
Median of Ratios	the median of pixel-by-pixel ratios of pixel intensities, with the median background subtracted.
Mean of Ratios	the geometric mean of the pixel-by-pixel ratios of pixel intensities, with the median background subtracted.
Ratios SD	the geometric standard deviation of the pixel intensity ratios.
Rgn Ratio	the regression ratio of every pixel in a 2-feature-diameter circle around the center of the feature.
Rgn R^2	the coefficient of determination for the current regression value.
F Pixels	the total number of feature pixels.
B Pixels	the total number of background pixels.
Sum of Medians	the sum of the median intensities for each wavelength, with the median background subtracted.
Sum of Means	the sum of the arithmetic mean intensities for each wavelength, with the median background subtracted.
Log Ratio	log (base 2) transform of the ratio of the medians.
Flags	the type of flag associated with a feature.
Normalize	the normalization status of the feature (included/not included).

F1 Median - B1	the median feature pixel intensity at wavelength #1 with the median background subtracted.
F2 Median - B2	the median feature pixel intensity at wavelength #2 with the median background subtracted.
F1 Mean - B1	the mean feature pixel intensity at wavelength #1 with the median background subtracted.
F2 Mean - B2	the mean feature pixel intensity at wavelength #2 with the median background subtracted.
SNR 1	the signal-to-noise ratio at wavelength #1, defined by (Mean Foreground 1- Mean Background 1) / (Standard deviation of Background 1)
F1 Total Intensity	the sum of feature pixel intensities at wavelength #1
Index	the number of the feature as it occurs on the array.
"User Defined"	user-defined feature data read from the GAL file (GenePix Pro 4.1).

GPL - GenePix Lab Book format (*.gpl)

The GenePix Lab Book is a binary file that contains a fixed-size structure for each line in the Lab Book.

GPS - GenePix Settings format (*.gps)

GenePix acquisition, analysis and display settings are saved as binary GenePix Settings Files. Settings are organized into a number of different categories (acquisition, analysis and display) all of which are saved together in the GPS file. However, when opening a settings file you can choose which subset of the settings you wish to open.

Acquisition settings include which laser was enabled during the acquisition, the PMT voltages, the lines averaged, and the scan area. Analysis settings include the location and identification of blocks and feature-indicators that were defined on the image. Display settings include brightness and contrast settings, and the color mapping.

JPEG - Joint Photographic Experts Group (*.jpg)

Images can be saved in the JPEG format, which is a lossy compressed image file format. GenePix implements minimal JPEG compression, which is enough to reduce image file size significantly, but which removes only a small amount of data from the image. However, we recommend that you do not use the JPEG format to archive images that are to be analyzed later. Rather, use the JPEG format to store images that are to be used in presentations.

TIFF - Tagged Image File Format (*.tif)

Images acquired in GenePix are by default saved as 16-bit unsigned TIFF images. This is a standard, uncompressed graphic file format that can be read by many graphics and imaging programs. The primary data acquired by GenePix are the single-wavelength images, and by default these are saved as 16-bit grayscale TIFFs in a single multi-image TIFF file. Not all graphics applications can read multi-image TIFF files. You may wish to try opening a multi-image file with your preferred graphics application to see if they are supported. If not, save the single-wavelength images as separate single-image files.

GenePix exports its preview and pseudocolor ratio images as 24-bit color TIFFs, but it does not read them, as data are not extracted from them.

.