VRIJE UNIVERSITEIT

Master Business Analytics

RESEARCH PAPER

Predicting Cancer with Deep Learning

Predictive Modeling of Colorectal Cancer using
Deep Learning on Routine Electronic Medical Records

Author:
Jeroen E.L. van Kasteren

June 27, 2018

Predicting Cancer with Deep Learning

Predictive modeling of colorectal cancer using
deep learning on routine electronic medical records

SUBMITTED BY:

Jeroen E.L. van Kasteren (2540489)

SUBMITTED TO:

VU University Amsterdam
FACULTY OF SCIENCE
De Boelelaan 1081
1081HV Amsterdam

SUPERVISOR:

Dr. Mark Hoogendoorn

June 27, 2018

Abstract— **Although deep learning models have achieved remarkable results in computer vision and speech recognition applications, they are rarely used when working with Electronic Medical Records. However, with the tremendous continues growth of Electronic Medical Records available for research, deep learning models may have the potential to outperform commonly used statistical models and other traditional machine learning models. Recently, multiple studies demonstrated successful implementations of deep learning models for predicting diseases.**

The research question is: Can deep learning models predict colorectal cancer better than traditional machine learning models using electronic medical records of the general practitioner? This research focusses on multiple deep learning models, namely, Neural Networks, Recurrent Neural Networks and Convolutional Neural Networks. Logistic Regression is also implemented to compare the deep learning models with a traditional machine learning model. The influence of normalising the data, adding engineered features and using different model implementations are studied.

The results of this research show that the Neural Network performance does not significantly differ from the Logistic Regression performance. Using only age and gender as features obtained similar results as earlier research. In contradiction with earlier research, adding features did not improve predicting performance. Moreover, the Convolutional Neural Network and Recurrent Neural Network could not extract useful temporal features from the data.

An important notice is that Deep Learning models need more time to learn than most statistical and machine learning models. Deep learning models have more hyperparameters to tune and appeared to be more sensitive to slight changes in hyperparameters. In conclusion, answering the research question, deep learning models have the potential to perform as well as traditional machine learning models when predicting CRC. Nevertheless, they often get stuck in local optima due to the sparse and unbalanced nature of Electronic Medical Records. Furthermore, they could not extract useful temporal features from the records.

Keywords: Deep learning, EMR, LR, CNN, RNN, NN, CRC, text mining, temporal patterns.

I. Introduction

Nowadays it is common practice for health care organisations to store routine care data in Electronic Medical Records (EMRs). This replaced the paper charts in the clinicians office and improved safety and efficiency of health systems. The data is used for real-time decision support systems [10], [23], providing an easy way to access (critical) information and reducing unnecessary testing. Since 2011, the Office of the National Coordinator for Health Information Technology of the United States [17], for instance, is working to create Electronic Health Records (EHRs), which are EMRs that can be created, managed, and consulted by authorized staff across more than one healthcare organization. Software businesses as Epic, Chipsoft and InterSystems are making EHRs possible in countries all over the world [33].

The storage of this great amount of health data paves the way for analysing health care date. With data analysing tools insights can be extracting that can hold great value. It was pointed out by Goldstein et al. [19] that using EMRs for clinical risk predictions is increasingly common. They show that risk predictions are studied in different areas of health care, predicting for example mortality probabilities or service utilizations. The question is whether these models can help with, for instance, early detection of diseases. As an example, take colorectal cancer (CRC), the third deathliest cancer for both men and women in the United States [43]. The earlier CRC is detected, the higher the chance it is still localised. This results in significant higher survival probabilities [29].

Therefore, it is a great contribution to the domain of health if models could identify high risk groups in an early stage, help to facilitate proactive care, and support in selecting the right treatment [26]. Goldstein et al. [19] remarked in their review that statistical and machine learning models are commonly used for predicting high risk groups and that the models show promising results. Goldstein et al. summarise that most frequently applied models are generalized linear models, Bayesian methods, random forest and regularized regression. Shickel et al. [42] observe that until the last few years, no deep learning models were used. They mention how deep learning techniques have the ability to construct deep hierarchical features and to capture long-range dependencies, which led to the great success of deep learning in many other domains. They show that in recent years both supervised and unsupervised deep learning models are used in health care problems. Recurrent Neural Networks (RNN) are mostly used for supervised problems, while Deep belief Networks (DBN) and autoencoders are commonly used for unsupervised problems.

Although it is important to choose the right model, it is at least as important to do proper feature engineering. Recent research has shown significant performance improvement when features were added that contained temporal information [24], [26]. They compared multiple traditional machine learning models and obtained the best results with Logistic Regression. To date, however, little attention has been devoted to predicting diseases with deep learning models taking the time dimension into account. Cheng et al. [12] proposed using a Convolutional Neural Network (CNN) to predict the risk of someone having a disease. Nonetheless, no attention was devoted to predicting CRC.

Therefore, this research aims to examine the performance of deep learning models in predicting CRC. The goal is to answer the following research question:
Can deep learning models predict colorectal cancer better than traditional machine learning models using electronic medical records of the general practitioner?
It is tested how a Neural Network (NN), RNN and CNN can perform in comparison with Logistic Regression (LR). This paper is divided into five parts. First, an overview of the related work is presented in section II. Second, a description of the data is given and the extensive preprocessing steps are described in section III. Third, the used models are explained and described in section IV. Fourth, the experimental setup is outlined in section V. Fifth, the results are shown in section VI. Sixth, the results are analysed and discussed in section VII. Finally, conclusions are drawn and further research suggestions are stated in section VIII.

II. Related Work

In this section an overview of previous related researches is given.

Several studies describe the specific characteristics of EMRs. These characteristics should be taken into account while pre-processing the data and choosing the models. Namely, EMRs are highly dimensional, temporal, sparse, highly variable and incomplete [12], [26]. First, the high dimensionality comes from large amount of distinct medical events. Second, the temporality is because patients evolve over time and the sequentiality of medical events can contain critical information. Third, the sparsity is due to irregular and sporadic visits. Fourth, the high variability is as a result of the complexity of the symptoms and diseases. Last, the incompleteness is due to patients who can decide to not complaint while symptoms are present or a physician's lack of observation competence.

Kop et al. [26] discussed that it can be of great contribution if universal preprocessing steps can be developed or models proposed that can cope with these challenging characteristics. Their proposed pre-processing pipeline is taking into account while cleaning the data and engineering features. Moreover, they compared multiple traditional machine learning models and got the best performance with an off-the-shelf LR after extensive tuning. They showed that using solely age and gender as predictors, resulted in a surprisingly good benchmark. By taking temporal patterns into account, the benchmark and a pure hypothesis-driven approach (the Bristol-Birmingham equation [30]) were significantly outperformed.

Cheng et al. [12] proposed another approach. They represented the EMRs per patient as a temporal matrix with time on one dimension and event on the other dimension. An event can be a laboratory measurement, a comorbidity, consultation journal code, prescribed medication or a referral to a specialist. Afterwards, a CNN model was build for extracting (temporal) features and perform predictions. They used four different architectures which influence the way the time dimension was taken into account: a basic CNN model, temporal late fusion CNN, temporal early fusion CNN, and temporal slow fusion CNN. In short, the basic model fuses time and events simultaneously in multiple sequential layers, to extract patterns. The temporal early fusion model fuses time in the first layer and fuses events separately in the last layer. The late fusion model first extracts local temporal patterns in the first layers. In the sequential layer it fuses the local temporal pattern and the events. The slow fusion model compromises the early and late fusion models by extracting local temporal patterns and fusing them in multiple sequential layers. Similar, it fuses events in the last layer. The models will be further elaborated on in section V, Experimental Setup.

This research will validate the proposed CNN architectures and compare them with other deep learning models and LR. To do this, the data first needs to be pre-processed. The pre-processing is discussed in the next section.

III. Data Description and Preparation

In this section the provided data are described and afterwards the pre-processing steps are discussed. The pre-processing part is divided in cleaning and feature engineering.

A. Data Description

In this research an anonymised primary care dataset is analysed. The dataset is from a network of general practitioners (GPs) centred around the Utrecht University Medical Center in the Netherlands. The recordings are from the period between July 1, 2006 and December 31, 2011. The dataset is divided in six categories, shown in table I and described afterwards. The information contained per category is presented in the second column of the table. All categories contained a patient identifier (ID). The number of observations are shown in the raw size column. The observations are cleaned from missing values, removing duplicates and removing patients under 30[1], the number of clean observations are shown in the last column.

TABLE I
THE DATASET DIVIDED IN SIX CATEGORIES

Category	Dimensions	Raw size	Cleaned
1) Patient Data	ID, Sex, Registration, Unsubscribe, Birth Date	156,176	90,992
2) Measurement	ID, Date, Code, Value, Reference Value	5,661,666	4,798,578
3) Comorbidity	ID, Start Date, End Date, Description	66,898	24,172
4) Journal	ID, Date, SOAP, ICPC, episode_ICPC	14,620,846	5,274,513
5) Medication	ID, Date, ATC code	4,204,615	3,491,983
6) Specialism	ID, Date, Specialism	30,411	25,466

1) General patient data containing date of birth, gender, registration date and unsubscribe date of roughly 155,000 patients with 722 cases of patients with CRC.
2) Laboratory measurements data from measurements performed by the GP or received from an external lab. It contains codes to indicate which value was measured, the measurement outcome, the date and the maximum and minimum reference value. The coding scheme is specific for the GP information system from which the dataset was exported. Approximately 5.6 million measurements were recorded.
3) Comorbidity data describing the presence of additional diseases or disorders co-occurring with the primary disease. It contains about 65,000 records with a start and end date of the comorbidity.
4) Consults journal data describing diseases and symptoms of a patient before and after a consultation, consisting of almost 15 million records. The diseases and symptoms are reported according to the International Classification of Primary Care (ICPC) coding standard

[1]This is explained in the Data Pre-processing section.

[5], using the Dutch version. The journal also included SOAP[2] notes [8],[47].

5) Prescribed medication data, reporting the active substance of the medicine with the Anatomical Therapeutic Chemical (ATC) Classification System codes [49]. In the dataset 4.2 million prescriptions were present.
6) Referrals to specialists, containing the name of the specialism and the date of the referral of 30 thousand referrals.

To give an impression, a small sample of the data is presented in appendix IX-A. Moreover, note that the data is highly unbalanced with only 0.45% of the patients having CRC. In view of the importance of age and gender as predictors for age and gender [26], a visualisation is given in figure 1. In the top graph the number of patients with a certain age is shown. The graph is similar to the Dutch population pyramid illustrated by the CBS[3]. Difference is the low number of patients with an age around 20 years. This reflects that not every citizen visits the GP often. Furthermore, note that due to the unbalanced nature of the data, the CRC patients are almost impossible to see in the top graph. A small line can be detected at the bottom of the top graph at and around 70 years of age. The bottom graph focusses only on the percentage of patients having CRC given sex and age. Note how the graph shows that CRC is extremely rare in early stages of life. Additional, men have a higher probability of CRC in mid-life, while women have a higher probability of CRC in the older ages. Remarkable is the percentage of patients older than 105 years with CRC. Maybe this is due to registration inconsistency. In further research, this can be tested by consulting a specialist.

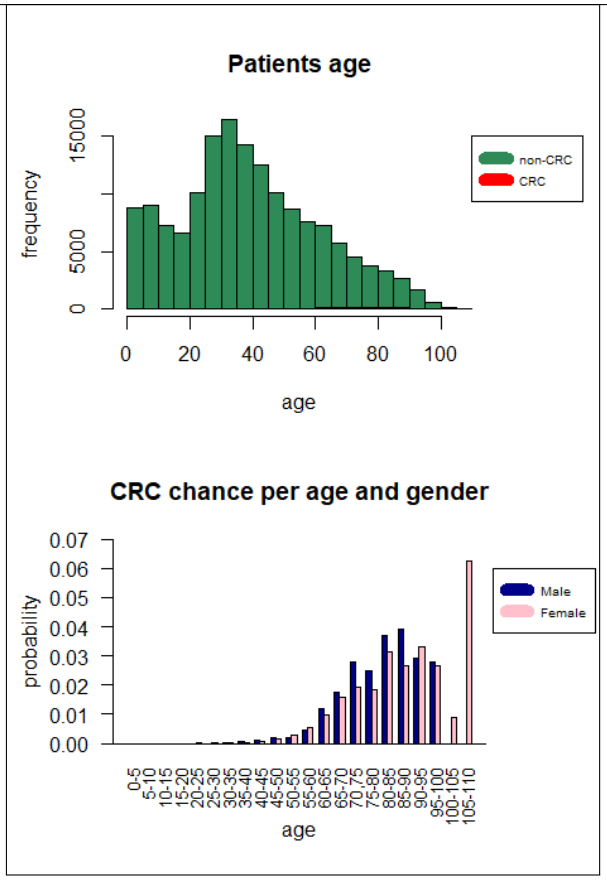

Fig. 1. The top graph represents the number of patients of a certain age present in the dataset. The bottom graph shows the chance to have CRC relative to age and gender.

B. Data Pre-processing, Cleaning

The first step in pre-processing the data is thoroughly cleaning per category. Cleaning is particularly useful for EMR data, because EMR data often contains missing or wrong values [22]. Starting with the cleaning of general patient data, the dataset contained four patients with an incorrect age. These patients were older than the oldest living person alive and were omitted [36].

To stay in line with existing literature, patients under the age of 30 are left out [21], [24], [26], [30]. For developing CRC in that stage of life is rare. Likewise, patients with less than six months of recorded data are dropped. Considering that the used models need at least six months of data per patient. Accordingly for CRC patients, the model needs at least six months of data before CRC was diagnosed. This will further be elaborated on in chapter V, the experimental setup. After patients were dropped following above-mentioned criteria, around 90,000 patients were left in the dataset with 618 cases of CRC. If the registration date was before someone was born, it was removed. The registration and unsubscribe date were only kept when they were inside the window of the the dataset (July 1, 2006 to December 31, 2011) and could be used for the analysis.

As a second step, the laboratory measurements were pre-processed. First, all codes that indicate what was measured were capitalised, to ensure that the same code in different capitalization were not interpreted as different codes. It was observed that about 66,000 times the codes were completely missing. The majority of these missing codes, 99.7%, had exactly the same maximum and minimum reference value.[4] Taken this relation between these records into account, the majority is grouped and given the code 'Unknown', maybe containing valuable insights. The remaining 200 measurements with missing codes were dropped.

For feature engineering it was important that reference values were present. And even though the reference values were missing almost 3 million times, 2.5 million reference values could be recovered. This was achieved by looking at other recorded measurements of the same measured value. If these other recorded measurements were documented with reference values, the missing reference value could be recovered. It is important to note that there was inconsistency in

[2]SOAP is an acronym and stands for subjective, objective, assessment, and plan. Every time a symptom or disease is diagnosed, it is accompanied with one of the letters of SOAP. In case the patient observes symptoms, it is subjective; when the GP detects it, it is objective; the final diagnose is the assessment; and the proposed treatment is the plan.

[3]Centraal Bureau voor de Statistiek [9], translated: Central agency of statistics.

[4]reference minimum 135 and reference maximum 145

the documentation of reference values. Sometimes the same measured value had different reference values documented at other records. Therefore, for the same measured value the most used reference value was used to replace missing reference values.

Unfortunately, the numeric test outcomes were not always well documented, 330,000 times they were non-numeric. Examples of non-numeric test outcomes are[5] "Eet veelzijdig en veel cholesterol arm eten."; "geen afwijkende epitheelcell."; "Zie vorige controle."; "novorapid"; "19-02-2010", "POSITIEF". Although these short texts were discarded in preceding literature, they could contain extra prediction power. In the section III-C, the pre-processing of these non-numeric values is outlined.

The third step is cleaning the comorbidity data. First, records without start and end date were dropped. This considered only three records. The start date was known for all other records. Second, it turned up that sometimes no end date was recorded or that the end date was in conflict with the start data, because the end date was before the start date. In those cases the end date was set equal to the start date. Third, if a comorbidity was outside the time window of the dataset, it was ignored. This is the case when the end date is before 01-07-2006 or the start date is after 31-12-2011.

The fourth step in preprocessing is cleaning the consultation journal. To clean properly, it was needed to know the meaning behind the structure of ICPC codes. ICPC codes are single letters referring to a chapter of ICPC followed by a number. Every chapter defines an area or a body part on which symptoms or diseases have effect. The number specifies the symptom or disease. Small inconsistencies were found in documenting the codes. The same code was sometimes written with or without punctuation or spaces or with different capitalization. To make the system robust against these inconsistencies, white spaces and punctuations were removed and all characters were converted to capitals. A few dozens of records had incorrect documented codes consisting of solely letters or digits, which did not follow the ICPC coding. Moreover, roughly 180,000 records did not report any code. These incorrect and missing codes still represent a visit to the GP. Therefore, the incorrect and missing codes were grouped and reported as visits without (proper) coding. The SOAP notes were never missing.

After cleaning the consultation journal, it was per patient derived if and when patients were diagnosed with CRC. This was possible because CRC was documented in the journal with the ICPC code 'D75'.

Finally, the prescribed medication data and the referrals were examined. The data was clean with the exception of 400,000 missing medication codes and 600 missing specialism names. The missing medication codes were grouped and reported as unknown. The same was done for the missing specialism names.

With the cleaning process described above, the data was cleared from missing values. Additional, in all categories all duplicate recordings were deleted. This created a dataset which was suitable for the used models. In the next section the features extracted from the clean data are presented.

[5]The examples are in Dutch, because the dataset originates from Dutch GPs

C. Data Pre-processing, Feature Engineering

In this section is described how certain features were explicitly extracted from the data and added as additional input for the model. Kop et al. [26] showed that adding contextualization to measurements and adding frequent temporal patterns improved prediction performance. The proposed modified CNN architecture by Cheng et al. [12] is believed to extract temporal patterns by itself. Nonetheless, the architecture prevents the model from the possibility to extract the context of measurements. Therefore, the contextualization is explicitly engineered as recommended by Kop et al.

The first step in contextualization is absolute grounding. Consider patient p at time t measuring value a with minimum reference value $r_{min}(a)$ and maximum reference value $r_{max}(a)$ and test outcome $test(p,t,a)$. Then the absolute grounding $ground_{abs}(p,t,a)$ comes from comparing the test outcome with its maximum reference and minimum reference value. If the test outcome was below the minimum reference value, the absolute grounding was too low, l. In case the outcome was above the maximum reference value, the absolute grounding was too high, h. If both reference values were missing, the absolute grounding was not available, na. The absolute grounding was normal, n, otherwise. When the reference values are present, absolute grounding follows the following formula:

$$ground_{abs}(p,t,a) = \begin{cases} l, & \text{if } test(p,t,a) < r_{min}(a) \\ h, & \text{if } test(p,t,a) > r_{max}(a) \\ n, & \text{otherwise} \end{cases} \tag{1}$$

The second step in contextualization is relative grounding, $ground_{rel}(p,t,a)$. This extracts per patient information about consecutive test outcomes of the same measured value, comparing $test(p,t_{i-1},a)$ with $test(p,t_i,a)$, where $t_{i-1} < t_i$. In this illustration t_{i-1} and t_i are two consecutive times that value a was measured for patient p. In case the test outcome at t_i is lower than the the test outcome at t_{i-1}, relative grounding is said to have decreased, d. If the test outcome is higher than the last time, relative grounding is said to have increased, i. Relative grounding is not available when there is no preceding measurement or the preceding measurement does not contain a numeric value. Relative grounding is stable, s, when the absolute difference between $test(p,t_{i-1},a)$ and $test(p,t_i,a)$ is zero or not more than 5% of the absolute difference between $r_{min}(a)$ and $r_{max}(a)$. For clarity, define the absolute difference needed to be stable as $r_s(a) = |r_{min}(a) - r_{max}(a)| \times 0.05$. The boundary at 5% was chosen for it divided the number of times the absolute grounding was decreasing, increasing or stable in three roughly equally big groups. When relative grounding is available, the formula is as follows,

$$ground_{rel}(p,t,a) =$$

$$= \begin{cases} i, & \text{if } test(p,t_{i+1},a) - test(p,t_i,a) > r_s(a) \\ d, & \text{if } test(p,t_i,a) - test(p,t_{i+1},a) > r_s(a) \\ s, & \text{otherwise} \end{cases} \quad (2)$$

After adding the contextualisation of the measurements, a way to extract features from the non-numeric measurements outcomes was investigated. Some of the non-numeric outcomes contained text. If the text contained for instance the word 'novorapid',[6] it indicates the person has diabetes and patients with diabetes have a significant higher chance of developing CRC [27]. Furthermore, words as 'negatief' indicate an unfavourable test result. Such information may contain predictive insights and should not be neglected. However, natural language text is by nature non-ordered. To gain insights in text, text mining tools were used to extract meaningful features. The approach was to use a bag-of-words model as presented by Salton and McGill [40]. The text mining pipeline is,

- First, the texts were tokenised to obtain analysable words. This omitted punctuation. The abbreviation 'Neg.' became the token 'Neg'. It was ensured that the punctuation in floating points (5.6), scales (13:2) and dates (19-02-2007) were kept unharmed.
- Then all letters were converted to capitals, to make the short texts homogeneous. This ensures for instance that negatief and NEGATIEF can be recognised as the same word.
- Words with frequent occurrences (key words) were clustered as shown in table II, with the aim of creating meaningful groups with significant prediction value. A cluster should contain at least 500 items to be considered as feature. The requirement for a cluster to contain at least 500 items, insures there will not be dozens of small insignificant clusters.
- Afterwards, every short text was assigned to one of the twelve clusters, when the short text contained a corresponding key word. If a short text contained multiple key words and therefore belonged to multiple clusters, it was assigned to the cluster with highest rank, the ranks are shown in table II. If a key word was part of a compounded word, the parts of the compounded word were treated as separate words. In this way the key word was still recognized.
- In some cases a record did contain a numeric measurement but no contextualization was possible. This is when the record had no reference values and no preceding measurements. Then the record was assigned to the cluster No Grounding.

[6]NovoRapid® is used to lower high blood sugar levels for adults, youngsters or childeren from the age of one with diabetes [13].

[7]This considers all remaining values with text that do not belong to another group.

[8]If no numeric or text value was present, the value was missing and set to 'Missing'.

TABLE II

CLUSTERS OF SHORT TEXT

Rank	Cluster Name	Key words	Size
1	Negative	Negatief, Neg, Slecht, Nee, Dubieus, Overleden, Afwijking	102,400
2	Positive	Positief, Pos, Goed, Ja, Redelijk, Normaal, Stabiel, Gezond, Prima	64,700
3	Increased	Verhoogd	600
4	Allergie	Allergie, Gras, Pinda, Eik	2,000
5	Feet	Voet	1,700
6	TBD	n.t.b., Volgt	7,100
7	Unknown	Onbekend	1,800
8	None	Geen	10,800
9	Diabetes	Novo, Insulatard, Diabet	1,400
10	Text	remainder non-numeric values[7]	96,794
11	Missing	missing value[8]	44,330
12	No Grounding	numeric without contextualization	96,112

TABLE III

MERGING CODES TO EVENTS FOR JOURNAL DATA

ID	Date	SOAP	ICPC	Event
2	29-11-2006	S	P01	P01-S
2	18-06-2008	S	T92	T92-S
2	29-12-2010	S	T92	T92-S
⋮	⋮	⋮	⋮	⋮
4	24-09-2008	E	K86	K86-E
4	02-06-2011	E	K86	K86-E
4	20-07-2009	O	A99	A99-O
⋮	⋮	⋮	⋮	⋮

Eventually, the categories of the dataset[12] were transformed to a homogeneous format, with daily events, to make sure they could be given to the model. Although the prescribed medication data and the referrals to specialist data were already in a good format, the other categories needed some manipulation. First, the comorbidity data described diseases with a time span of multiple days or even years. To be in line with the rest of the data, a record was created for every day the comorbidity was present. Second, the journal contained a SOAP code and an ICPC code per

[9]$ground_{rel}$

[10]$ground_{abs}$

[11]Left out for readability

[12]Laboratory measurements, Comorbidity data, Consults Journal, Prescribed medication and Referrals to specialist

TABLE IV

MERGING CODES TO EVENTS FOR MEASUREMENT DATA

ID	Date	Code	Value	Rel[9]	Abs[10]	Event
4		GV	20-02-2008	NA	NA	GV-TEXT
14	. . .[11]	RRDI	100	d	h	RRDI-I
15		BORG	Negatief	NA	NA	RRDI-H BORG-NEGATIEF
⋮	⋮	⋮	⋮	⋮	⋮	⋮

record. These two codes were concatenated to create a new code, referred to as an event, as shown in the table III. Third, laboratory measurements had two kinds of records, one with and one without contextualization. The dimensions were concatenated to fit the events with the format of the model[13]. For records with contextualization, the code of the measured value was concatenated with the absolute and relative grounding coding, as shown in the second and third row of table IV. When the record had no contextualization, the record was assigned to one of the clusters in table II. After the manipulations, the mentioned categories of the dataset were in the same format, every record had an event on a specific day with a patient identifier.

In contrast to the other categories, general patient data was static and had no event per day. This is due to gender, birthday and patient identifier being independent of time. Therefore, the patient category had its own format. In summary, the number of features for all categories are presented in table V.

TABLE V
NUMBER OF FEATURES

Category	Features	Engineered features	Total
1) Patient Data	2[14]	0	2
2) Measurement	1160	3300	4,460
3) Comorbidity	30	0	30
4) Journal	1842	4189	6,031
5) Medication	1,238	0	1,238
6) Specialism	87	0	87
Total	4361	7489	11848

With the data pre-processed, the data could be given to a model. How this is realised, is presented in section V, Experimental Setup. First, the models are described in section IV, Methodology.

IV. METHODOLOGY

In this section the used models will be explained in depth, to create a full understanding of the models. First, the concept of deep learning and neural networks is introduced and then the models are described.

A. Neural Networks

Deep learning refers to the group of models based on neuron cells, also known as perceptrons, able to learn non-linear relations. The idea of using a neuron as a basis for a model was first introduced by McCulloch & Pitts [32]. Given is an input vector $X \in \mathbb{R}^n$ of length n with corresponding weight vector $w \in \mathbb{R}^n$ of equal length and bias $b \in \mathbb{R}$. Given some function g mapping $\mathbb{R} \to \mathbb{R}$, the output $Y \in \mathbb{R}$ of the neuron will be,

$$Y = g(wX + b). \tag{3}$$

[13]Take into account that every event must eventually be represented in a binary way, indicating if it is present or not. Because multiple events can take place per time unit, not concatenating them would trouble with the temporal dimension.

[14]Age and gender.

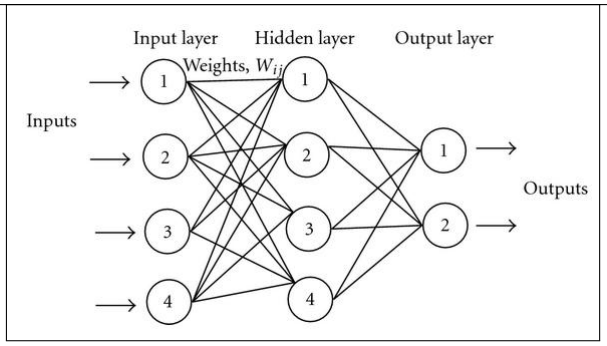

Fig. 2. Example of an architecture of a Neural Network.[16]

This is similar to generalised linear models (GLM) [31]. Nonetheless, generalised linear models can only learn a specific linear or non-linear relation. Different from GLMs is the architecture of deep learning models. The architecture can have multiple layers, with every layer consisting of multiple stacked neurons. Due to this architecture, deep learning models can learn more complex non-linear relations. An illustration of such a architecture is given in figure 2. As example, given an input X, the input is fed to a layer with m neurons. Every neuron will get the whole input. Nonetheless, the weights of the neurons will differ and every neuron will amplify other input signals. The output of every neuron is $y_m \in \mathbb{R}$, thus the output of the layer is $Y \in \mathbb{R}^m$. This output Y can again be fed to the neurons in the next layer. In this way 'deep' non-linear relations can be learned by the multilayer model. It is essential to use a non-linear (activation) function for g, otherwise, only linear relations can be learned. For a sum of linear functions is again a linear function. The following non-linear functions are mostly used,[15]

$$\begin{aligned} Sigmoid, &\quad g(x) = \tfrac{1}{1+e^{-x}} \\ Tanh, &\quad g(x) = \tfrac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \\ ReLU[16], &\quad g(x) = \max(0, x) \end{aligned} \tag{4}$$

The weights of deep learning models are initialised randomly. As discussed by Glorot and Bengio [18], it works best if the initial weights are small. Usually the weights are drawn from a normal distribution with mean zero and small variance. To ensure all random drawn weights are small, the truncated normal distribution is often used.

Nevertheless, such multilayer architectures were impossible to train. Till the idea of backpropogation learning with gradient descent was coined in 1986 by Rumelhart, Hinton and McClelland [39]. Their backpropogation algorithm has been optimized since.[17] Currently the adaptive moment estimation method (Adam) of Kingma and Ba is one of the best optimizations of the backpropogation

[15]For an explanation why Rectified Linear Unit is currently the most used activation function, I refer to the article of Walia [46]. Here is discussed why ReLU solves the vanishing gradient problem.

[16]Rectified Linear Unit (ReLU).

[17]For a clear summary about the optimized algorithms, I refer to Ruder [38].

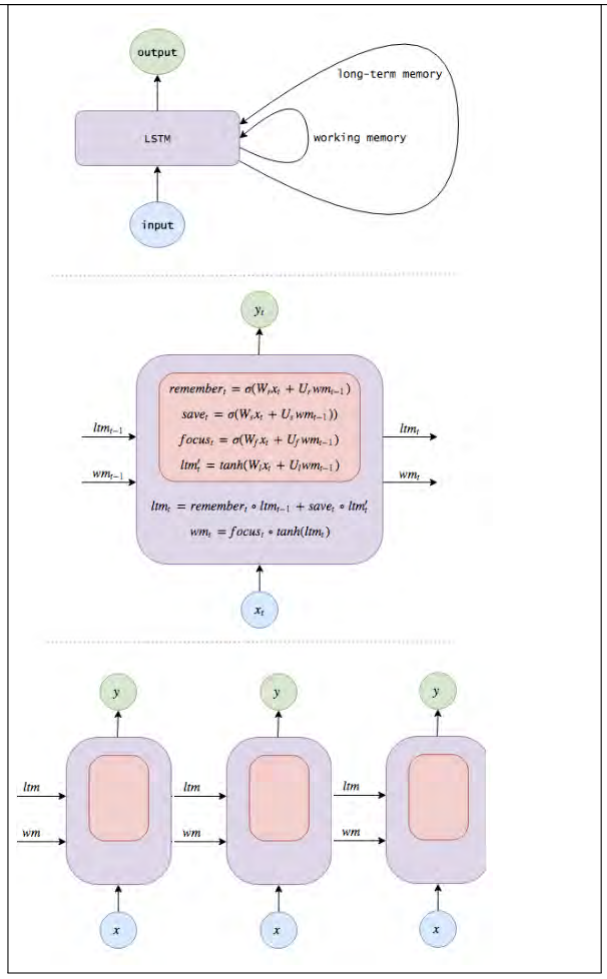

Fig. 3. Illustration of a Long Term Short Memory cell. At the top a conceptual view, in the middle the mathematical formulas and at the bottom the unfolded perspective [11].

algorithm known [25]. With these algorithms the training time of multilayer architectures became reasonable. Since then, multilayer architectures are widely used and mostly referred to as (Artificial) Neural Networks.

A downside of a NN is that it can only have a one dimensional input vector. It is therefore not suitable for directly extracting temporal patterns, which are two dimensional. Nevertheless, the models CNNs and RNNs are created to directly extract temporal patterns.

B. Recurrent Neural Networks

First RNN will be explained. A RNN contains recurrent neurons. This means the output of a neuron at time $t-1$, is given as additional input at time t. This is also shown at the top of figure 3. In formula form, given input X_t, weight w_t and bias b_t as before, but depending on time, define $Y_t \in \mathbb{R}$ the output at time t and $w_t \in \mathbb{R}$ its corresponding weight. The output Y_t of the recurrent neuron will be

$$Y = g(w_t X_t + w_{t-1} Y_{t-1} + b_t). \tag{5}$$

The difficulty is training a network with recurrent connections. To cope with the recurrent connections while training,

Fig. 4. How a feature map is created in a convolution or max pooling layer [41].

the backpropogation algorithm of NNs was adjusted to backpropogation through time (BPTT) [48]. BPTT views a neuron as it is unfolded over time, illustrated at the bottom of figure 3. Note that training the weights of a neuron in BPTT is changed to training the weights of that neuron and the weights of that neuron in all preceding times. This made training time increase exponentially with time when adding layers. It can be partially solved by using truncated learning, were only a fixed number of steps back in time are considered while learning.

Another problem is that BPTT has difficulty in learning long-term dependencies [4]. As a reaction Long Term Short Memory cells (LTSM) were designed. This cell adds a connection between neurons through which long term dependencies can be learned and do not vanish.[18] Every LSTM cell contains four neurons which regulate the input signal.

C. Convolutional Neural Networks

The other extension of NNs are CNNs. CNNs have the ability to learn two dimensional patterns [3]. A CNN does not just give the input X to a neuron in the convolution layers, but gives it in larger batches which contain local information. Illustrated in figure 4, the input is two-dimensional and a certain local area, or so called window, is considered per output pixel. This window is two dimensional and has a width and height and a weight vector of size width times height. In figure 4 can be seen how every time a different

[18] Olah gives a clear explanation of how LSTM work in his article [35].

local part is selected to eventually cover the whole picture. The outputs are stored together in a new two dimensional picture, this is called a feature map. Let the window in figure 4 have width $x \in \mathbb{N}$, height $y \in \mathbb{N}$, weight vector $w \in \mathbb{R}^{xy}$ and bias $b \in \mathbb{R}$. Take the first input and output, illustrated with an a in figure 4. The input of the window is flattened to a one dimensional vector, $X_a \in \mathbb{R}^{xy}$. This vector is given to a neuron as,

$$Y_{kl} = g(wX_a + b). \tag{6}$$

Here k is the row and l the column, representing the coordinate of the output. As example, for input a the output coordinate is $k = 1$ and $l = 1$, while for input n, $k = 2$ and $l = 4$ in figure 4 .

In the top example of figure 4, the window shifts every time one pixel to the right till it covered the whole width. Afterwards, it goes one pixel down and covers again the whole width. It is possible to skip pixels and to shift the window n pixels per movement. How the window moves over the picture is called the stride. At the top illustration in figure 4 is shown a stride of one in width and length, while at the bottom illustration in figure 4 a stride of two in width and length is shown.

Furthermore, zero padding is shown at the bottom illustration in figure 4. Padding is how to handle the edges of the picture where the window falls partly outside the input. Zero padding is adding zero's to the edge, to make sure all input information can be taken into account. In the width of the bottom illustration in figure 4 zero padding is used and makes it possible to create output z and a. When no padding is used and the window falls partly outside the input, the edges are neglected that are not part of a complete window. At the bottom illustration of figure 4 no padding is used in the hight and the window will be partly over the edge for the bottom row. Therefore, the bottom row is neglected.

Recall that every time an output pixel is generated, equation 3 is used with some function g. When for g one of the common activation functions is used, as described in equation 4, it is called convolution. Usually, multiple feature maps are created per convolution layer. Different weights are used for the sliding windows constructing the feature maps and therefore the feature maps differ from each other.

Another commonly used function for g is,

$$Y_{kl} = \max(X_a). \tag{7}$$

This kind of neuron is used to filter out noise and only select the strong signals. If a layer is constructed with this function, it is known as a max pooling layer. Per max pooling layer only one feature map is generated. Creating multiple feature maps is redundant, for the feature maps will be identical when the same input is given. A common way to build a CNN is to alternate convolution and max pooling layers.

After the convolution and max pooling layers, a CNN has as output multiple two dimensional feature maps and needs to convert this to a prediction. However, a prediction can be one dimensional. To overcome this, all feature maps are flattened and stacked together. This one dimensional stack of feature maps is given as input to a neuron network with one or more layers. The neural network layers of the CNN are usually fully connected. Therefore, these layers are called the fully connected layers. The outcome is produced by the last fully connected layer of a CNN.

The proposed architectures of NNs, RNNs and CNNs can learn deep hierarchical features and capturing long-range dependencies. In practice these models have achieved great success in many domains [42]. However, the training time to adjust weights is still quite long and one of the known bottlenecks of the models. Moreover, the models need a lot of data and there is no guaranty to converge to the optimal solution.

D. Logistic Regression

To measure the performance of the deep learning models on EMR data, they are compared with LR. LR is chosen, because Kop et al. [26] showed that with LR good results can be achieved on predicting CRC using EMR data. LR is a Generalised Linear Model commonly used in classification. LR can be viewed as single neuron, as in equation 3, with for g the sigmoid function. LR can be trained with gradient descent. In the next section is explained how the parameters are going to be tuned and how the models are going to be tested.

V. Experimental Setup

In this section it is discussed how the data is transformed, how the models are compared and the parameters are tuned.

Start with considering the format of the data. For every patient to be comparable, a time window of equal size is selected. Previous literature [21], [24], [26] considers a time window of six months. This regards the expectation that symptoms of diseases could be detected a few months before the actual diagnose. For CRC patients this means that a period of six months prior to the first diagnosis of CRC was considered. For all other patients, a random window of 6 months was selected between the first and last recorded event of a patient. The windows were selected randomly, to ensure they would represent the real underlying population without bias. Nevertheless, it could happen that a window was selected without an event[19]. After selecting the windows, the corresponding age in that window of time was derived from the patients birth year and normalised by dividing by the maximum age of 115 years [36].

First, to mimic the pipeline of Kop et al. [26], a static format was created. In this case every feature is the number of occurrences of the corresponding event in the chosen time window. In this static format, normalised age and gender are artificially added as extra features. For the normalisation of the other features, two ways were tested. The first way is a binary representation and a the second a normalised frequency representation. In the latter, the number of occurrences is divided by the maximum number of occurrences to transform all numbers to a number between zero and one.

[19]This means that a patient did not receive medication and did not visit the GP in those six months.

In the binary representation, on the other hand, a feature is a one if it has occurred at least once and zero otherwise.

Second, to mimic the pipeline of Cheng et al. [12], a matrix representation was created. In this representation the events are on the row dimension of the matrix and the time is on the column dimension. Additionally, age and gender are events that are present everyday.

As discussed in section III, Data Description and Preparation, EMRs are sparse. The considered dataset would be 99.96% zero in daily matrix form. If too much noise is presented in the data, a model may not be able to extract the significant features. Therefore, also a weekly representation is tested which is 99.7% zero. The static format is used as benchmark and is is 92% zero. The age and gender format is also a benchmark and is never zero[20], for age and gender were always present by convention.

Regarding the sparsity of the data, four options are tested. First, all features are kept to test if the model can handle the sparsity. The other options consider different techniques for removing unimportant features and only keeping the essential ones. With a Random Forest (RF) [7] is measured which features influence the dependent variable, having CRC or not. Afterwards, only the m most important features are kept. Different values for m are tried: 500, 200, 20, 10 and 5. The hyperparameters for RF were already tuned by earlier research on predicting CRC with EMR data [26]. Therefore, those hyperparameters values[21] are used. An off-the-shelf implementation of RF is used from the Python module Scikit-learn [37]. Another way to decrease sparsity is by removing the engineered features described in section III-C. The last option uses the pipeline of Van Kampen [24] by leaving out measurement data, comorbidity data, SOAP notes of the journal and all engineered features.

The considered models are CNN, RNN, NN and LR. The models are implemented with the Python module Tensorflow [1], that allows GPU computing[22]. Nonetheless, earlier research [21], [24], [26] used LR and implemented it with the Python module Scikit-learn. To make results comparable, LR was also implemented with the Python module Scikit-learn. The Scikit-learn implementation came with off-the-shelf L1[23] and L2[24] regularization. The Tensorflow implementation only had off-the-shelf L2 regularization. Regularization is used to select the most important feature and prevent overfitting.

Scikit-learn and Tensorflow have different solvers implemented. Scikit-learn recommended solvers are LIBLINEAR[25] and SAGA[26], while Tensorflow default solver is the earlier described Adam optimizer [25]. In addition, Scikit-learn requests the training data to fit in working memory, while Tensorflow allows to give the data in batches[27].

The weights of the models are randomly initialised from a normal distribution with mean zero and variance one. For all hidden neurons in the CNN, RNN and NN models the Rectified Linear Unit activation function is used as proposed by Walia [46]. For the output neuron, on the other hand, the Sigmoid activation function was used. This ensured the prediction was between zero and one.

The error for patient p, $e(p)$, is the weighted sqaured difference between the prediction $y_{pred}(p)$ and the actual value $y_{act}(p)$ plus regularization costs on the weights w_{cost}. The actual value y_{act} is binary, indicating the presence (1) or absence (0) of CRC in a patient. As mentioned in section III, Data Description and Preparation, only 0.45% of the patients have CRC. This results in a highly imbalanced dataset. Therefore, the squared difference is weighted to give more importance to the less frequent occurring class, having CRC. The weight for patient p, $w_{factor}(p)$, is inversely proportional to the occurrence in the data of the class patient p belongs to. This is in line with previous research [21], [24], [26] and is illustrated in equation 8. The number of patients is indicated with n_{crc} and the total number of patients is N.

$$\begin{aligned} e(p) = \quad & (y_{pred}(p) - y_{act}(p))^2 * w_{factor}(p) \\ & + w_{cost}, \end{aligned} \tag{8}$$

$$w_{factor}(p) = \begin{cases} 1 - \frac{n_{crc}}{N}, & \text{if } p \text{ has CRC,} \\ \frac{n_{crc}}{N}, & \text{else.} \end{cases}$$

It was also tried to balance the dataset by oversampling patients with CRC. For that method, the weights for both classes $w_{factor}(p)$ was set equal to one.

The prepared data is given in age and gender format and static format to LR and the NN to act as a benchmark. The temporal weekly data is given to all models, with exception[28] of LR implemented with Scikit-learn. To take the temporal weekly data as input, for LR and the NN, the weeks are stacked to a one dimensional vector. Lastly, CNN and RNN are tested with both the temporal daily data as the temporal weekly data.

Due to long training times, extensively tuning hyperparameters was outside the scope of this research. To select reasonable hyperparameters, a local grid search was used. The considered hyperparameter values are presented in appendix IX-B. The hyperparameters include the solver[29], the number

[20]Zero is used as synonym for not present. Therefore, being male having gender 0, is not viewed as zero. For the gender is present.

[21]The best results in earlier research were obtained with a maximum tree depth of 5 and with a minimum number of 50 samples per leaf node.

[22]NVIDIA Corporation explains [28]: "GPU-accelerated computing is the use of a graphics processing unit (GPU) together with a CPU to accelerate calculations."

[23]Also known as LASSO regression [45], LASSO stands for Least Absolute Shrinkage and Selection Operator. It puts cost on the sum of absolute weights, forcing the weights of unimportant features to zero.

[24]Also known as Ridge regression [20]. Puts cost on the sum of squared weights, forces weights to not grow too big.

[25]A Library for Large Linear Classification [15].

[26]A Fast Incremental Gradient Method [14].

[27]The batch method of Tensorflow means that training samples are grouped and given at once to the model. The error per training sample in the batch is calculated and all errors are summed. Afterwards, the summed error is used to update the weights.

[28]The exception was made, because Scikit-learn requests all training data to fit in the working memory. The daily format takes 70 GB of memory and the weekly format needs 25 GB of memory. Clearly, that does not fit in the working memory of a normal computer.

[29]Multiple solvers are available and it is tested which one fits the best.

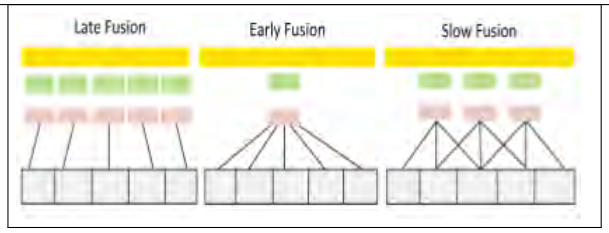

Fig. 5. Architectures to extract information over temporal dimension. The colour grey indicates the input layer, red the convolution layer, green the max pooling layer and yellow the fully connected layer. [12]

of epochs[30], which architecture to use for deep learning models and the regulazation strength. The tuning was done separately for every model and data format[31]. When tuning hyperparameters, the number of epochs are increased until the model converged. The hyperparameters are tuned using a train, validation and test set. The sets are obtained by randomly assigning patients in three disjoint groups,

- 60% of the patients are assigned to the training set. The training data will be given to the models to tune their inner parameters;
- 20% of the patients are assigned to the validation data. The validation data is used to validate the performance of the models.
- 20% of the patients are assigned to the test data. The test data is used to measure actual performance after validation.

Furthermore, the proportion of CRC and control patients are kept equal in all three sets. This is known as stratification and ensures that all three sets are representable for the whole dataset.

For the NN and RNN optimal architectures needed to be found. Stanley and Miikulainen [44] claim that incrementally growing the architecture from minimal structure is one of the reasons why their Neural Evolution through Augmenting Topologies algorithm (NEAT) is efficient. Therefore, for NN and RNN simple architectures using only one hidden note are tested first and the architectures are incrementally grown till satisfactory results are obtained.

For the CNN the four proposed architectures of Cheng et al. [12] are tested. These are the basic, temporal early fusion, temporal late fusion and temporal slow fusion architecture. The illustration by Cheng et al. of the three temporal fusion architectures are showed in figure 5. All architectures first have a convolution layer, followed by a max pooling layer and fully connected neural network layer. The basic architecture is a normal CNN with symmetric windows, different sizes of windows and different strides are tested. It will be used as a benchmark to test if the fusion architectures improve performance. The fusion architectures have windows with a height of one, for convolution over multiple events is not meaningful. Additionally, the stride is strictly one in height, making sure all events are taken into account.

The difference in the fusion models is the width of the windows and strides. In early fusion the width of the window covers the whole time dimension at once. This fuses the temporal patterns to one number per event.[32] In late fusion the strides in width are as big as the window size, to make the windows non-overlapping. The temporal patterns will eventually be fused by the fully connected layer.[33] Slow fusion is a compromise between early and late fusion. In width the strides will be smaller than the window sizes to create fusion in the convolution and max pooling layer. Temporal information not fused in the preceding layers, will be fused in the fully connected layer.

The results are described in the next section. To measure performance the confusion matrix with corresponding precision, recall, accuracy and f1-score are calculated. To stay in line with existing literature, the Area Under the Receiver Operating Characteristic Curve (ROC-curve) is taken as final performance measure [6], [26]. It is important to indicate the robustness of the prediction to be comparable. Therefore, the test set was sampled 500 times with replacement. For every sample the ROC-curve was calculated and afterwards a 95% confidence bootstrap interval could be derived for the ROC-curve, by bootstrapping 100 times.

TABLE VI
SIX MONTHS WINDOWS

	With events	Without events	Total
Non-CRC patients	65,366	24,938	90,304
CRC patients	579	39	618
Total	65,945	24,977	90,922

VI. RESULTS

In this section the results are shown. First, the results are described. Thereafter, the most important results are presented in tables.

First, a six month time window is selected per patient. For 27% of all patients the selected window did not contain any event. For CRC patients this was 6% (39 of 618). The prediction for those patients needed to be based solely on their age and gender. The numbers are presented in table VI.

Consider the way the data was given to the models. Normalisation increased performance and made the models less sensitive for small changes in hyperparameters. Slightly better performance was obtained with the frequency normalisation method than with the binary normalisation method. Additionally, using batches did reduce training time, but decreased performance tremendously. Oversampling CRC patients also did not improve performance.

[30] An epoch is one full training cycle, where every sample of the training data is given to the model once.

[31] As example, the hyperparameters are tuned separately for a CNN model with temporal daily data and for a CNN model with temporal weekly data.

[32] Note that this results in the first layer (the convolution layer) having as output one number per event. This makes the max pooling layer ineffective.

[33] To make sure the temporal patterns are not already fused by the max pooling layer, the max pooling layer is given window size of one in width. This makes the max pooling layer ineffective, in line with previous research [12].

No improvement was found when engineered features were added. Reducing the number of features by selecting only the most important with RF did improve performance. Better results were obtained by just removing measurement data, comorbidity data, SOAP notes and engineered features[34].

Moreover, different hyperparameter values were tested. A regularization strength[35] of order 1 worked best for both LR implementations, while a strength of order 0.0001 worked best for all deep learning models. Considering the solvers, LIBLINEAR was extremely fast, easy to tune, robust and gave reasonable results. SAGA could obtain better results, but had a ten times longer training time and was sensitive to small changes in hyperparameters. The Adam optimizer had the longest training time and often got stuck in local optima predicting all cases only negative or only positive. However, with a right random initialisation, Adams prediction performance was the best of all optimizers.

Considering the architecture of the NN, best results were obtained with two hidden layers with each ten or twenty hidden nodes. Additionally, performance worsened when the sigmoid function was not used at the output node.

For the LR, the feature importance[36] is presented in table VII. The corresponding confusion matrix[37] and measures for the train, validate and test set are shown in table VIII and IX. The presented features are related to known CRC symptoms such as iron anemia, abdominal pain, and constipation [49], [34], [50]. Furthermore, known risk groups are elderly and those with pulmonary, cardiovascular and metabolic chronic diseases [26]. This is reflected with the drug for hypertension, the drug for high cholesterol level and the influenza vaccination[38]. Additionally, when using solely gender and age as features, the feature importance is 0.995 for age and 0.005 for gender.

Table X shows the best obtained results when solely the features age and gender are taken into account. The results show equal prediction performance with LR from Tensorflow and the NN[39]. The LR from Scikit-learn performs worse, but the difference is not significant, for the 95% bootstrap intervals do overlap. Table XI shows the performance when adding medication data, specialism referrals and journal data without SOAP notes. No significant difference in prediction performance is observed between the models.

Table XIII and XII show the performance when the

[34]Thus, only using medication data, specialism referrals and engineered journal data without SOAP notes

[35]Note that when only age and gender were given as features, adding regularization made performance worsen.

[36]The feature importance is measured by multiplying the coefficient times the standard deviation of that variable.

[37]Note that a positive (P) prediction stands for predicting the patient positive for CRC, while a negative (N) prediction stands for predicting the patient not having CRC.

[38]Not reflecting that the vaccination causes CRC, rather that patients that qualify for the vaccination are those already at risk. Similar, hearing damage and bruising do not cause CRC, but are probably correlated with age and therefore significant.

[39]Note that all deep learning models are implemented with the Tensorflow module.

TABLE VII

FEATURE IMPORTANCE

Feature	Importance	Description
Age	1,349	-
A06AD65	0,145	Macrogol, constipation drug
Gender	0,177	-
D01	0,079	Abdominal pain symptoms
B80	0,064	Iron deficiency anemia
R441	0,060	Influenza vaccination
N01	0,060	Headache
D16	0,058	Rectal loss of blood
A99	0,052	Non specified diseases
B82	0,050	Iron deficiency anemia
C09DA01	0,043	Iosartan and diuretics, hypertension drug
D18	0,038	Alternation of stool
C10AA03	0,036	Pracastatin, high level cholesterol drug
S16	0,034	Bruising
R44	0,029	Influenza vaccination
A06AC01	0,027	Ispaghula, constipation drug
M01AB05	0,025	Diclofenac, anti-inflammatory painkiller drug
H02	0,023	Hearing damage
D11	0,020	Diarrhoea
D93	0,019	Irritable bowel syndrome

TABLE VIII

CONFUSION MATRICES

Train		Predicted	
		N	P
Actual	N	41113	13069
	P	68	302

Validate		Predicted	
		N	P
Actual	N	13680	4381
	P	23	101

Test		Predicted	
		N	P
Actual	N	13630	4431
	P	21	103

TABLE IX

MEASURES

Set	Precision	Recall	F1	Accuracy	ROC-curve
Train	0,023	0,816	0,044	0,759	0,788 (0,768 - 0,810)
Validate	0,023	0,815	0,044	0,758	0,786 (0,751 - 0,827)
Test	0,023	0,831	0,044	0,755	0,793 (0,759 - 0,826)

TABLE X

RESULTS FOR AGE AND GENDER

Model	Module	ROC-curve
LR	Scikit-learn	0.790 (0.767 - 0.818)
LR	Tensorflow	0.836 (0.808 - 0.865)
NN	Tensorflow	0.836 (0.818 - 0.857)

TABLE XI

RESULTS FOR STATIC FEATURES

Model	Module	Regularization	ROC-curve
LR	Scikit-learn	L1	0.801 (0.764-0.836)
LR	Scikit-learn	L2	0.797 (0.758-0.834)
LR	Tensorflow	L2	0.833 (0.800 - 0.865)
NN	Tensorflow	L2	0.834 (0.808 - 0.858)

TABLE XII

RESULTS FOR WEEKLY MATRIX FORMAT

Model	Module	ROC-curve
LR	Tensorflow	0.5 (0.5 - 0.5)
NN	Tensorflow	0.5 (0.5 - 0.5)
RNN	Tensorflow	0.5 (0.5 - 0.5)
CNN	Tensorflow	0.5 (0.5 - 0.5)

TABLE XIII

RESULTS FOR THE DAILY MATRIX FORMAT

Model	Module	ROC-curve
RNN	Tensorflow	0.5 (0.5 - 0.5)
CNN	Tensorflow	0.5 (0.5 - 0.5)

features are presented in weekly matrix format or in daily matrix format. After extensive hyperparameters tuning, there was still no model that could extract useful features from the matrix format. All models predicted only negative or positive for CRC, making predicting performance as bad as random.

VII. DISCUSSION

In this section several obstacles are discussed that were encountered while performing this research.

First, it was encountered that EMRs are not created to be analysed. Measurement values, journal codes and subscribe dates were often missing. Moreover, patients do not frequently visit the GP and can decide not to complain while various symptoms are already present. This makes the data sparse and noisy. The models needed to cope with this characteristics.

Furthermore, creating additional features also increased sparsity[40] and sparsity was already one of the downsides of EMRs. Consequently, adding such features will not necessarily result in better performance. When the data gets sparser, relatively more data is needed to get proper results.

Another obstacle was that the temporal matrix representation takes up a lot of memory. Not implementing the pipeline very memory efficient would crash a computer. Nevertheless, because of the trade-off between memory and speed, implementing the pipeline most memory efficient, would result in a running time of several days[41]. It was a major challenge to create a fast memory efficient pipeline, mainly because no available Python module supports manipulating sparse matrices without temporarily converting it to a dense matrix[42].

It was also encountered that the solvers were very sensitive for local optima. For many sets of hyperparameters the models often got stuck in predicting all patients only negative or only positive for CRC. To ensure this was not due to an unlucky random initialisation, multiple random seeds needed to be tested per hyperparameter set. This severely extended tuning time[43]. While tuning hyperparameters, the prediction performance was significantly below the performance obtained in the literature. To improve performance many sets of hyperparameters were tested, the data was normalised and training was performed in batches. Analyses about differences between the solvers, about the influence of normalising and about training in batches, is described in appendix IX-C.

Remarkable was that performance decreased when adding features[44]. Prediction performance finally became in line with the benchmark when all measurement data, comorbidity data, SOAP notes and engineered features were omitted. This can indicate that, among others, the engineered features increased sparsity too much, making the data unsuitable for the models.

Finally, it is noteworthy that in recent research 5-fold cross-validation was used to train and test models. This means that their model trained on 80% of data versus 60% in this research. Additionally, the pipeline is implemented independently. It could be that different implementation choices in this research increased sparsity. It is expected that those differences led to the differences in predicting performance.

VIII. CONCLUSION

First, the main findings will be summarised in this section. Second, recommendations for further research are presented.

A. Conclusions

In this research was studied if deep learning models can predict colorectal cancer better than traditional machine learning models using electronic medical records of the general practitioner. Adding temporal patterns improved performance in recent research. Therefore, it was most interesting if a Convolutional Neural Network (CNN) could extract temporal patterns from a matrix representation of electronic medical records (EMRs) and outperform the traditional machine learning model Logistic Regression (LR). Additional research has been done in pre-processing of the data, implementation of LR and using the deep learning models Neural Network (NN) and Recurrent Neural Network RNN.

The results show that non of the models, including the CNN, could extract useful features from the temporal matrix representation of EMRs. As benchmark, LR and NN are trained on different formats of static features. The simplest format is using only age and gender as features. Both LR and NN could extract useful features from this and got

[40]For example, contextualisation added per measurement six extra elements describing if the value had increased, decreased or remained stable and if the value was higher, lower or between the reference values.

[41]This was not practical, because the data needed to be loaded every time a different hyperparameter set was tested for a model.

[42]A sparse matrix is a matrix with most elements zero. Specific methods exists to store such matrices memory efficient by only saving the non-zero elements. Nevertheless, when such matrices are manipulated or used for training models, all modules temporarily converted these matrices to the dense form.

[43]Because it could already take an hour or longer for a deep learning model to train, it was outside the scope of this research to do an exhaustive grid search to tune the hyperparameters.

[44]Usually, when significant features are added, it improves performance. When insignificant features are added, it should not affect performance.

an Area Under the Receiver Operating Characteristic Curve (ROC-curve) of 0.836[45]. Other formats of static features contained all features, only the most important features or only basic features. Similar results were obtained on the train, validate and test set, indicating that the models were not overfitting. Nevertheless, adding static features did not improve performance. This was opposed to recent research [21], [24], [26]. It is plausible that different choices in the pipeline implementation are causing this.

In conclusion, the proposed matrix format for EMRs is not suitable for the considered models. A plausible cause is the sparsity of EMRs. This research has shown that adding contextualisation as well as extracting features from non-numeric text outcomes[46] did not improve performance. Furthermore, it was confirmed that age and gender are good predictors for CRC. Normalising the data improved prediction performance and using batches decreased performance. Lastly, LR and a NN performed equally in predicting CRC, showing that deep learning can be used in predicting diseases, but do not outperform traditional machine learning models. Nevertheless, deep learning models have more hyperparameters to tune, are more sensitive to small changes and have longer training times. Thus, although the traditional machine learning model LR did not outperform the deep learning models, LR can still be preferred for it is easier to use and faster to train.

B. Further Research

It is likely that differences in pipeline implementation caused that the deep learning models could not extract useful temporal patterns form EMRs. In other studies, patients with a too short relevant history are excluded and the models are trained on 80% of the data. Naturally, excluding irrelevant patients and using more data will improve prediction performance. It is interesting to investigate the exact influence of using different criterion on excluding patients and using lesser training data.

Taking the unsatisfactory results of the temporal matrix format into account, it is interesting if the proposed forward filling approach by Amirkhan et al. [2] can be mimicked and make the format useful. Results are expected to improve with this method. However, this makes it less comparable with other studies using solely six months of data.

Considering the results of static features, it is remarkable how the same model shows significant difference when implemented with other Python modules. It is interesting to investigate when which implementation with which solver is preferred and why.

Further research is also needed in generalizable features that can be extracted from EMRs to significantly increase performance when predicting diseases.

[45]This result is in line with recent research.

[46]Contextualisation features and features from non-numeric text outcomes are the engineered features.

C. Acknowledgements

This research paper is part of the master programme Business Analytics. It is the task to produce a thesis to demonstrate the ability to describe a problem and solution in a clear manner. I want to thank Mark Hoogendoorn for his guidance, patience and support during the this project. Additionally, I want to thank the GPs from the Julius General Practitioners Network (JHN), the Academic Network of General Practice VU Medical Center Amsterdam (ANH-VUmc) and the Leiden General Practice Registration Network (RNUH-LEO LUMC) in the Netherlands for sharing anonymized routine general practice care EMR datasets to be studied.

REFERENCES

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from http://tensorflow.org/.

[2] Amirkhan, R., Hoogendoorn, M., Numans, M. E., & Moons, L. (2017). Using recurrent neural networks to predict colorectal cancer among patients. In Computational Intelligence (SSCI), 2017 IEEE Symposium Series on (pp. 1-8). IEEE.

[3] Bengio, Y., LeCun, Y., & Henderson, D. (1994). Globally trained handwritten word recognizer using spatial representation, convolutional neural networks, and hidden Markov models. In Advances in neural information processing systems (pp. 937-944).

[4] Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks, 5(2), 157-166.

[5] Bentsen, B. G. (1986). International classification of primary care. Scand J Prim Health Care; 4(1):43-50.

[6] Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern recognition, 30(7), 1145-1159.

[7] Brieman, L. (2001). Random Forest. Machine Learning, 45, 5-32.

[8] Cameron, S., & TurtleSong, I. (2002). Learning to write case notes using the SOAP format. Journal of Counseling & Development, 80(3), 286-292.

[9] Centraal Bureau van Statistiek. (2018). Bevolkingspiramide. Retrieved April, 1, 2018, from www.cbs.nl

[10] Chaudhry, B., Wang, J., Wu, S., et al. (2006). Systematic review: impact of health information technology on quality, efficiency, and costs of medical care, Ann. Intern. Med. 144 (10) 742752

[11] Chen, E. (2017). Exploring LSTMs. Retrieved March 28, 2018, from http://blog.echen.me/

[12] Cheng, Y., Wang, F., Zhang, P., & Hu, J. (2016). Risk prediction with electronic health records: A deep learning approach. In Proceedings of the 2016 SIAM International Conference on Data Mining (pp. 432-440). Society for Industrial and Applied Mathematics.

[13] Chlup, R., Zapletalová, J., Seckar, P., Chlupová, L., Táncosová, S., & Rezncková, M. (2004). Benefits of insulin aspart vs phosphate-buffered human regular insulin in persons with type 1 diabetes treated by means of an insulin pump. Biomed Pap Med Fac Univ Palacky Olomouc Czech Repub, 148(1), 27-32.

[14] Defazio, A., Bach, F., & Lacoste-Julien, S. (2014). SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In Advances in neural information processing systems (pp. 1646-1654).

[15] Fan, R., E., Chang, K., W., Hsieh, C., J., Wang, X., R., & Lin., C., J. (2008). LIBLINEAR: A library for large linear classification Journal of Machine Learning Research, 9, 1871-1874.

[16] Gajdos, M. (2016). Fun With Neural Networks in Go. Retrieved March 28, 2018, from http://mlexplore.org/

[17] Garrett, P., Saidman, J. (2011). EMR vs EHR - What is the difference. Office of the National Coordinator for Health Information Technology. Retrieved March 23, 2018, from https://www.healthit.gov/

[18] Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics (pp. 249-256).

[19] Goldstein, B. A., Navar, A. M., Pencina, M. J., & Ioannidis, J. (2017). Opportunities and challenges in developing risk prediction models with electronic health records data: a systematic review. Journal of the American Medical Informatics Association, 24(1), 198-208.
[20] Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12(1), 55-67.
[21] Hoogendoorn, M., Szolovits, P., Moons, L. M., & Numans, M. E. (2016). *Utilizing uncoded consultation notes from electronic medical records for predictive modeling of colorectal cancer.* Artificial intelligence in medicine, 69, 53-61.
[22] Jensen, P. B., Jensen, L. J., & Brunak, S. (2012). Mining electronic health records: towards better research applications and clinical care. Nature Reviews Genetics, 13(6), 395-405.
[23] Jha, A. K., Doolan, D., Grandt, D., Scott, T., & Bates, D. W. (2008). The use of health information technology in seven nations. International journal of medical informatics, 77(12), 848-854.
[24] van Kampen, J. (2016). Predicting colorectal cancer with the aid of temporal patterns. Business Analytics Research Paper. VU University Amsterdam, Faculty of Science.
[25] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
[26] Kop, R., Hoogendoorn, M., ten Teije, A., Bchner, F. L., Slottje, P., Moons, L. M., & Numans, M. E. (2016). *Predictive modeling of colorectal cancer using a dedicated pre-processing pipeline on routine electronic medical records.* Computers in biology and medicine, 76, 30-38.
[27] Krämer, H. U., Schöttker, B., Raum, E., & Brenner, H. (2012). Type 2 diabetes mellitus and colorectal cancer: meta-analysis on sex-specific differences. European journal of cancer, 48(9), 1269-1282.
[28] Krewell, K. (2009). Whats the Difference Between a CPU and a GPU? NVIDIA Corporation. Retrieved March 28, 2018, from http://nvidia.com/
[29] Levin, B., Lieberman, D. A., McFarland, B., Andrews, K. S., et al. (2008). Screening and surveillance for the early detection of colorectal cancer and adenomatous polyps. A joint guideline from the American Cancer Society, the US Multi-Society Task Force on Colorectal Cancer, and the American College of Radiology. Gastroenterology, 134(5), 1570-1595.
[30] Marshall, T., Lancashire, R., Sharp, D., Peters, T. J., Cheng, K. K., & Hamilton, W. (2011). The diagnostic performance of scoring systems to identify symptomatic colorectal cancer compared to current referral guidance. Gut, 60(9), 1242-1248.
[31] McCullagh, P. (1984). Generalized linear models. European Journal of Operational Research, 16(3), 285-292.
[32] McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4), 115-133.
[33] Monegain, B. (2017). InterSystems, Epic land more global business than any other EHR vendors. Retrieved, June 20, 2018, from www.healthcareitnews.com
[34] Nederlands Huisartsen Genootschap (2018). Find ICPC codes using the NHG ICPC thesaurus. ICPC-online. Retrieved, June 22, 2018, from www.nhg.org.
[35] Olah, C. (2015). Understanding LSTM Networks. Retrieved March 26, 2018, from http://colah.github.io/
[36] Oliver, M. (2004). Joan Riudavets-Moll, 114; World's Oldest Man. Los Angeles Times. Retrieved March 24, 2018, from http://articles.latimes.com/2004/mar/08/local/me-joan8.
[37] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830, 2011
[38] Ruder, S. (2016). An overview of gradient descent optimization algorithms. Retrieved March 29, 2018, from http://ruder.io
[39] Rumelhart, D. E., Hinton, G. E., & McClelland, J. L. (1986). A general framework for parallel distributed processing. Parallel distributed processing: Explorations in the microstructure of cognition, 1, 45-76.
[40] Salton, G., & McGill, M. J. (1983). Introduction to Modern Information Retrieval. McGraw Hill Book Co., New York.
[41] Shankar, S., Lasenby, J., & Cipolla, R. (2013). Semantic Transform: Weakly Supervised Semantic Inference for Relating Visual Attributes. Proceedings of the IEEE International Conference on Computer Vision, 361-368.
[42] Shickel, B., Tighe, P. J., Bihorac, A., & Rashidi, P. (2017). Deep EHR: A Survey of Recent Advances in Deep Learning Techniques for Electronic Health Record (EHR) Analysis. IEEE Journal of Biomedical and Health Informatics. arXiv:1706.03446v2
[43] Siegel, R., DeSantis, C., & Jemal, A. (2014). Colorectal cancer statistics, 2014. CA: a cancer journal for clinicians, 64(2), 104-117.
[44] Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. Evolutionary computation, 10(2), 99-127.
[45] Tibshirani, R., Wainwright, M., & Hastie, T. (2015). Statistical learning with sparsity: the lasso and generalizations. Chapman and Hall/CRC.
[46] Walia, A. S. (2017). Activation functions and its types-Which is better? Retrieved March 25, 2018, from https://towardsdatascience.com/.
[47] Weed, Lawrence L. (1964). Medical records, patient care, and medical education. Irish Journal of Medical Science. 39 (6): 271282
[48] Williams, R. J., & Peng, J. (1990). An efficient gradient-based algorithm for on-line training of recurrent network trajectories. Neural computation, 2(4), 490-501.
[49] World Health Organization. (1996). Guidelines for ATC classification and DDD assignment. In Guidelines for ATC classification and DDD assignment. World Health Organization.
[50] Zorginstituut Nederland (2018). The pharmacotherapeutic compass, describing the processes, indications and symptoms of active drug substances. Retrieved June 22, 2018, from www.farmacotherapeutischkompas.nl.

IX. Appendix

A. Sample of the data

In this section a small sample of the anonymised primary care dataset per category is presented. In table XIV the general patient data is shown; in table XV the laboratory measurements data is presented; in table XVI the comorborbidity data is displayed; in table XVII the consults journal data is demonstrated; table XVIII shows a sample of the prescribed medication data and table XIX contains the referrals to specialists.

TABLE XIV
A SAMPLE OF GENERAL PATIENT DATA

Patient ID	Birth date	Sex	Registration	Unsubscribe
14	1958	V	01-01-1988	NA
23	1979	V	04-01-2008	NA
32	1964	M	02-12-2989	NA
35	1969	V	06-26-2006	NA
44	1952	V	01-01-1988	NA
94	2010	M	07-20-2010	11-11-2010
⋮	⋮	⋮	⋮	⋮

TABLE XV
A SAMPLE OF MEASUREMENT DATA

Patient ID	Date	Code	Value	Reference [Min, Max]
2	03-10-2010	GLUC	6.3	[4.00, 6.40]
3	11-24-2010	ROOK	1	[NA, NA]
3	11-24-2010	SGGM	12	[0.00, 0.00]
4	10-12-2012	RDDI	80	[NA, 89.00]
4	10-12-2012	RRSY	140	[NA, 139.00]
4	10-10-2008	MCHC	20.4	[20.30, 22.20]
⋮	⋮	⋮	⋮	⋮

TABLE XVI

A SAMPLE OF COMORBIDITY DATA

Patient ID	Start Date	End Date	Description
2	06-15-2008	NA	Gout
4	01-12-2006	NA	Hypertension
4	06-15-2006	NA	Gout
10	01-13-2007	NA	Asthma
14	11-17-2006	NA	Hypertension
23	01-28-2011	25-03-2011	Pregnancy
⋮	⋮	⋮	⋮

TABLE XVII

A SAMPLE OF JOURNAL DATA

Patient ID	Date	SOAP	ICPC	episode_ICPC
1	11-28-2011	S	NA	R74
1	11-30-2011	P	NA	H71
1	01-29-2011	E	W90	W90
⋮	⋮	⋮	⋮	⋮
2	11-29-2006	S	NA	P01
2	06-18-2008	S	NA	T92
2	12-29-2010	S	NA	T92
⋮	⋮	⋮	⋮	⋮

TABLE XVIII

A SAMPLE OF MEDICATION DATA

Patient ID	Date	atc_code
1	30-11-2011	J01FA10
2	02-02-2006	NA
2	02-03-2006	N06AB04
2	02-03-2006	N06AB04
⋮	⋮	⋮
4	10-09-2008	C01AA05
4	10-09-2008	C10AA01
⋮	⋮	⋮

TABLE XIX

A SAMPLE OF SPECIALISM REFERRALS

Patient ID	Date	Specialism
9	09-09-2008	Scopy-department
10	01-30-2008	Laboratory
35	09-09-2008	ophthalmology
37	08-12-2008	ophthalmology
44	12-05-2007	RNTgenealogie
44	12-04-2007	RNTgenealogie
⋮	⋮	⋮

B. Results on validation set

In this section the results on the validation set while tuning the parameters are presented. In table XX the results are shown using LR on age and gender; in table XXI the results are presented for LR using basic features; in table XXII the results are presented for LR using normalised basic features; in table XXIII the results are displayed for LR using all normalised features; table XXIV shows results for NN using advanced features with different seeds; table XXV shows results for NN using age and gender format and table XXVI contains the Results using CNN. The RNN is tried on weekly data and with 100 nodes and 10 nodes, both gave a result of 0.5.

TABLE XX

RESULTS FOR LR ON AGE & GENDER FORMAT

Module	ROC-curve[47]	Solver	Regul.[48]
Scikit-learn	0.77	Liblinear	l2 C=1
Scikit-learn	0.77	Liblinear	l2 C=0.001
Scikit-learn	0.77	Liblinear	l2 C=100
Scikit-learn	0.77	Liblinear	l1 C=1
Scikit-learn	0.77	Liblinear	l1 C=0.001
Scikit-learn	0.77	Liblinear	l1 C=100
Tensorflow	0.82	adam	L2 C=0
Tensorflow	0.84	adam	L2 C=0.0001
Tensorflow	0.5	adam	L2 C=0.001
Tensorflow	0.5	adam	L2 C=1

TABLE XXI

RESULTS FOR LR USING SOLELY BASIC FEATURES

Module	ROC-curve	Solver	Regul.
Scikit-learn	0.61	Liblinear	l2, C=10
Scikit-learn	0.74	Liblinear	l2, C=1
Scikit-learn	0.68	Liblinear	l2, C=0.1
Scikit-learn	0.70	Liblinear	l2, C=0.001
Scikit-learn	0.64	Liblinear	l1, C=10
Scikit-learn	0.75	Liblinear	l1, C=1
Scikit-learn	0.76	Liblinear	l1, C=0.65
Scikit-learn	0.75	Liblinear	l1, C=0.1
Scikit-learn	0.5	Liblinear	l1, C=0.001
Scikit-learn	0.67	Saga	l1, C=1
Scikit-learn	0.7	Saga	l1, C=1
Scikit-learn	0.68	Sag	l2, C=1
Scikit-learn	0.69	Sag	l2, C=1e-5
Scikit-learn	0.5	Sag	l2, C=1e-10
Scikit-learn	0.7	Sag	l2, C=1e-6
Scikit-learn	0.66	lbfgs	l2, C=1
Scikit-learn	0.67	lbfgs	l2, C=0.1
Scikit-learn	0.66	lbfgs	l2, C=0.01
Scikit-learn	0.66	Newton-cg	l2, C=0.1
Tensorflow	0.65	adam	l2 0.0001
Tensorflow	0.5	adam	l2 0.000001

TABLE XXII

RESULTS FOR LR USING SOLELY NORMALISED BASIC FEATURES

Module	ROC-curve	Solver	Regul.
Scikit-learn	0.7	Liblinear	l2, C=0.5
Scikit-learn	0.79	Liblinear	l1, C=0.5
Scikit-learn	0.78	Saga	l1, C1=0.5
Scikit-learn	0.78	Saga	l1, C1=0.8
Scikit-learn	0.77	Saga	l1, C1=1
Scikit-learn	0.77	Saga	l1, C1=0.1
Scikit-learn	0.7	Saga	l2, C1=1
Scikit-learn	0.72	Saga	l2, C1=0.1
Scikit-learn	0.68	Saga	l2, C1=0.01
Scikit-learn	0.72	Newton-cg	l2, C=0.1
Scikit-learn	0.68	Newton-cg	l2, C=0.01
Tensorflow	0.76	adam	0.0001
Tensorflow	0.5	adam	0.000001
Tensorflow	0.5	adam	0.001
Tensorflow	0.78	adam	0.0001
Tensorflow	0.66	adam	0.00001
Tensorflow	0.3	adam	0
Tensorflow	0.7	adam	0.0001
Tensorflow	0.79	adam	0.0005
Tensorflow	0.75	adam	0.0001

TABLE XXIII

RESULTS FOR LR USING ALL FEATURES

Module	ROC-curve	Solver	Regul.
Scikit-learn	0.78	liblinear	l1, C=0.5
Scikit-learn	0.77	liblinear	l1, C=1
Scikit-learn	0.77	liblinear	l1, C=0.1
Scikit-learn	0.79	liblinear	l1, C=0.2
Scikit-learn	0.77	liblinear	l1, C=0.3
Scikit-learn	0.79	liblinear	l1, C=0.2
Scikit-learn	0.79	liblinear	l1, C=0.2
Scikit-learn	0.70[49]	saga	l1, C=0.2
Scikit-learn	0.67	liblinear	l2, C=0.2
Scikit-learn	0.79	saga	l2, C=0.2
Tensorflow	0.76	adam	l2, C=0.0001
Tensorflow	0.63[50]	adam	l2, C=0.0001

TABLE XXIV

RESULTS FOR NN USING ADVANCED FEATURES, DIFFERENT SEEDS

Size	ROC-curve	Remark
500x500[51]	0.43	batches
500x500	0.41	batches
100x100	0.54	batches
100x100	0.52	batches
100x100	0.48	batches
100x100	0.59	normal
100x100	0.50	normal
100x100	0.68	normal
100x100	0.49	Balanced
100x100	0.49	Balanced
100x100	0.5	400 features
100x100	0.68	200 features
10x10	0.68	200 features
10x10	0.78	10 features
10x10	0.81	10 features
10x10	0.81	10 features
20x20	0.78	10 features
20x20	0.69	10 features
5x5	0.8	10 features
20x20x20	0.67	20 features

TABLE XXV

RESULTS FOR NN USING AGE AND GENDER

Size	ROC-curve	Remark
100x100	0.78	normal
20x20	0.83	balanced
20x20	0.82	normal
20x20	0.42	no sigmoid
20x20	0.28	batches
20x20	0.23	batches
10x10	0.84	normal
1	0.5	L2 C=0.001
3x3	0.5	L2 C=0.001
2x2	0.5	L2 C=0.001
2x2	0.5	L2 C=0.0001
2x2	0.5	L2 C=0.00001
2x1	0.82	L2 C=0.0001

TABLE XXVI

RESULTS FOR CNN

Type	data	C[52]size	C stride	MP[53]size	MP strides	FC[54]	ROC[55]
Basic	week	10x10	1x1	5x5	2x2	100	0.5
Basic	week	5x5	1x1	2x2	2x2	100	0.5
Basic	week	10x10	1x1	5x5	2x2	100	0.5
Basic	week	10x10	1x1	5x5	2x2	100	0.5
Early	week	1x27	1x1	1x1	1x1	100	0.5
Late	week	1x2	1x2	1x1	1x1	10	0.5
Late	week	1x4	1x4	1x1	1x1	100	0.5
Slow	week	1x2	1x1	1x2	1x1	100	0.5
Slow	week	1x2	1x2	1x2	1x2	100	0.5
Slow	day	1x4	1x1	1x4	1x1	100	0.5

C. Discussing normalisation and batches

In the results was discussed that normalising the data improved performance. This is not surprising, considering the used models. All the weights of the models are random initialised around zero and kept small by regularization, as discussed in section V, the Experimental Setup. When for example the weekly matrix format is considered, most patients have events that only occur once or twice per week. The models will adopt to this. However, consider someone in the test data has an event that occurs a dozens of times per week. The strength of that signal will likely overwhelm all other input signals, taking into account that the weights are not tuned to deal with strong input signals. A strong input signal can have unwanted influence on the prediction. This can cause the prediction to be dependent on this event that occurs many times, while maybe this event has in practice little to do with CRC. Normalisation will prevent a signal to overwhelm all other signals and

[47]Abbriviation for Area Under the Reciever Operating Characteristic Curve.

[48]Abbriviation for Regularization. C stands for the strength.

[49]Without normalisation.

[50]Without normalisation.

[51]First layer nodes and second layer nodes.

[52]Abbriviation for convolution layer.

[53]Abbriviation for max pooling layer.

[54]Abbriviation for the number of nodes in the fully connected layer.

[55]Abbriviation for Area Under the Reciever Operating Characteristic Curve.

will result in more robust models. In this case it turned out that the model performance improved the most with the normalised frequency representation, more than with the binary representation.

Although the training time was reduced by working in batches, it did decrease performance. This is probably because of the sparse and unbalanced nature of the data. As illustration, take batches of size 25 and consider that only 0.45% of the patients have CRC. This means that on average one in the ten batches contains a CRC patient. When the model predicts this patient negative, it will get a relatively big error due to the fact that errors are weighted. This error is, however, summed with all other errors in the batch. This will result in big changes in activated weights. The weights are activated by the input signals of the positive sample, but also of all negative samples. Nonetheless, the weights activated by the input signals of the negative samples are not responsible for the big error. Therefore, those weights will be over-adjusted. This combined with the sparsity of input signals, resulted in models that could not properly learn and performed worse than random. Noteworthy is that the models were not able to learn to predict zero or 1 by itself. The best results were obtained by using the sigmoid function as output node to scale the prediction.

The results showed that the Adam optimizer performed better than the Scikit-learn solvers. However, it does not show that the Adam optimizer is always preferred. In this case training time took hours for the Adam optimizer, while similar slightly worse results are obtained with the Scikit-learn solvers in seconds. Why the Scikit-learn solvers performed worse than the Adam optimizer and why the Adam optimizer did not improve performance when features were added, is unknown. It could be that with different hyperparameters and different pre-processing steps, more logical results can be obtained.