Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Informatik

Deep Learning zur Repräsentation von Elektronischen Gesundheitsdaten und Krankheitsvorhersage mit der Anwendung: Identifikation von sensitiven ROIs für Alzheimer

Seminar Deep Learning

Leipzig, März 2018

Karsten Schreiblehner
Studiengang M.Sc. Informatik

Betreuer: Victor Christen, Abteilung Datenbanken Leipzig

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Der Extraktion von Wissen aus strukturierten, teilstrukturierten (auch semistrukturierten genannt) und unstrukturierten Daten kommt eine immer bedeutendere Rolle in der Informatik zu. Dies wird durch den immensen Anwuchs datenproduzierender Prozesse unterstützt. Folglich reicht es nicht mehr, die Daten in ihrer ursprünglichen Form zu speichern, sondern die neue Aufgabe ist viel mehr die Analyse bestehender Datensammlungen. Als eine solche Menge bilden hier elektronische Gesundheitsdaten die Grundlage zur Analyse.

Elektronische Gesundheitsdaten (abgekürzt EHR für electronic health records) werden vorrangig in Krankenhäusern erzeugt. Diese speichern Patientenakten großteils in elektronischer Form. Dabei werden unter Anderem Behandlungen, Medikation, Laborergebnisse, Krankheitsverläufe und Arztnotizen pro Patient gesammelt und gespeichert. Ein solche Framework ist beispielsweise das Mount Sinai Health System. Dieses stellt die Patientendaten in anonymisierter Form in einem Data Warehouse zur Verfügung.

Auf Grund der vielen verwendeten Frameworks stellt sich die Herausforderung eine geeignete Datenrepräsentation zu schaffen, da die EHRs in der Regel heterogen über die verschiedenen Plattformen gespeichert werden. Eine solche Datenrepräsentation sollte einen effizienten Zugriff auf die Informationen bieten und durch die hohe Menge an Daten auch gering im Speicherverbrauch sein.

Neuronale Netze (kurz NN) sind vor Allem in den letzten Jahrzehnten wieder aufgekommen. Dies liegt begründet in der nun verfügbaren Rechenleistung und vor allem der Problematik schwer beherrschbarer Datenmengen. Dabei werden die neuronalen Verbindungen von Säugetieren nachgeahmt. Hauptbestandteile neuronaler Netze sind sogenannte (künstliche) Neuronen die miteinander verbunden sind. Jedes Neuron besitzt dabei eine Aktivierungsfunktion, die abhängig von den zumeist gewichteten Eingängen des Neurons ist und einen Schwellwert zur Aktivierung. Ferner wird zwischen Eingabeschicht, verdeckter Schicht und Ausgabeschicht in neuronalen Netzen unterschieden. Die Eingabeschicht dient der Entgegennahme von Eingabedaten, sowie deren grundlegende Transformation (z.B. zur Normierung). In der verdeckten Schicht werden die eigentlichen Berechnungen durchgeführt, wobei die Eingaben der ersten verdeckten Schicht aus der Eingabeschicht kommen und die Eingaben der weiteren Schichten jeweils von den Verbindungen zwischen den Neuronen in der verdeckten Schicht abhängen. Grundlegend unterscheidet man zwischen rekurrenten (rückgekoppelten) neuronalen Netzen (oder

auch rnn von recurrent neuronal networks), wo Zyklen innerhalb der neuronalen Verbindungen erlaubt sind und feedforward Netzen, wo die Eingabe sukzessiv durch die Schichten geleitet werden. Die Ergebnisse werden in der Ausgabeschicht ausgegeben. Weiterhin müssen diese Konstrukte vorab trainiert werden, um ein zur Berechnung geeignetes Modell zu erstellen. Ferner bilden neuronale Netze die Grundlage für das Deep Learning (kurz DL) was ein Teilzweig des maschinellen Lernens (kurz ML) darstellt.

Deep Learning und allgemein maschinelles Lernen wird zur Erkennung von bestimmten Mustern in Daten eingesetzt. Dabei steht insbesondere die Erstellung von Modellen im Vordergrund, wodurch diese zur automatischen Generierung von Wissen genutzt werden können. Dabei ist jedoch zu beachten, das solche Modelle im Allgemeinen nur zur maschinellen Verarbeitung geeignet sind, da sie für den Menschen nicht lesbar sind. Dies ist begründet durch die maschineninterne Darstellung der neuronalen Netze, wodurch den einzelnen Schichten die genau Funktion nicht zugeordnet werden kann. Ein großer Vorteil liegt jedoch in der Erkennung von Transitionen in Daten über sehr viele Hierarchieebenen, also Mustern in den Daten, die auf den ersten Blick nicht in Verbindung gebracht werden können.

Eine grundsätzliche Abtrennung zwischen Deep Learning und maschinellen Lernen besteht darin, dass beim ML vorgegebene Algorithmen genutzt werden, die Entscheidungen zumeist auf der Grundlage von Entscheidungsbäumen treffen. Beim DL werden diese Entscheidungen durch die jeweilige Aktivierung der Neuronen bestimmt. Dieser Vergleich wird in [Lec17] näher betrachtet.

Abschließend werden die oben genannten Verfahren zur Vorhersage von Krankheiten genutzt und auf die Alzheimervorhersage als weiteres Anwendungsbeispiel genauer eingegangen.

2 Grundlagen

In diesem Kapitel werden die notwendigen Grundlagen für die EHR-Prozessierung mithilfe von DL besprochen. Dabei wird näher auf feedforward Netze und somit allgemein auf den Aufbau von neuronalen Netzen eingegangen. Erläutert werden dabei die Grundbausteine von NNs, die Neuronen, sowie bestimmte Konstrukte wie sogenannte Autoencoder (kurz AE) und Denoising Autoencoder (kurz DAE) daraus abgeleitet. Weiterhin werden kurz Algorithmen des maschinellen Lernens wie die Principle Component Analysis (PCA), das k-means Clustering als Beispiel kurz vorgestellt, womit die Deep Learning Ansätze verglichen werden.

2.1 Neuronen

Neuronen bilden die Hauptbestandteile neuronaler Netze. Die Vorgänger von Neuronen sind Schwellenwertelemente (oder auch McCulloch-Pitts-Neuronen), die 1943 von W.S. McCulloch und W.H. Pitts in [MP43] vorgestellt wurden. Diese bestehen aus einem Schwellenwert, sowie gewichteten Eingängen. Ist die Summe der gewichteten Eingänge größer als der Schwellenwert, so wird eine eins ausgegeben, andernfalls eine null. Formal kann dies so definiert werden (siehe [KBB+15]).

Definition 2.1.1 *Ein Schwellenwertelement ist eine Verarbeitungseinheit für reelle Zahlen mit n Eingängen $x_1, \ldots, x_n$ und einem Ausgang y. Der Einheit als Ganzer ist ein Schwellenwert θ und jedem Eingang x_i ein Gewicht w_i zugeordnet. Ein Schwellenwertelement berechnet die Funktion*

$$y = \begin{cases} 1, \textit{ falls } \sum_{i=1}^{n} w_i x_i \geq \theta, \\ 0, \textit{ sonst.} \end{cases} \tag{2.1}$$

Darauf aufbauend wurden von F.Rosenblatt 1962 in [Ros58] das Perzeptron vorgestellt, worauf hier nicht näher eingegangen werden soll.

Leider können mit einem Schwellenwertelement (kurz SWE) nur linear separable Funktionen berechnet werden. Diese wären zum Beispiel das logische AND und OR, jedoch die Biimplikation und das XOR nicht. Da die meisten realen Probleme jedoch nicht linear separabel sind, entstanden Netze von Schwellenwertelementen. Während einzelne SWE leicht zu trainieren waren (siehe [KBB+15], Seite 21ff.), war dies mit Netzen von SWEs nicht so einfach. Erst durch die Entdeckung der Fehler-Rückübertragung [Wer74] wurde dies möglich.

Neuronen bauen auf SWEs auf. Ein Neuron u bildet dabei eine Einheit aus einer Eingabefunktion über den Eingaben in_u und den Gewichten der Eingaben $W \in \mathbb{R}^{|in_u|}$ in das Neuron, der Aktivierungsfunktion und der Ausgabefunktion:

- Eingabefunktion: $f_{net}^{(u)} : \mathbb{R}^{2|in_u|} \to \mathbb{R}$,
- Aktivierungsfunktion: $f_{act}^{(u)} : \mathbb{R} \to \mathbb{R}$,
- Ausgabefunktion: $f_{out}^{(u)} : \mathbb{R} \to \mathbb{R}$.

Die Gewichte können dabei als Adjazenzmatrix dargestellt werden, wobei ein Eintrag w_{uv_i} das Gewicht der Verbindung von Neuron $v_i, i \in \{1, \dots n\}$ zu Neuron u darstellt. Ferner können in die Eingabefunktion, die Aktivierungsfunktion und die Ausgabefunktion noch weitere externe Eingaben erfolgen. Zusammenfassend kann ein Neuron wie nachfolgend dargestellt werden.

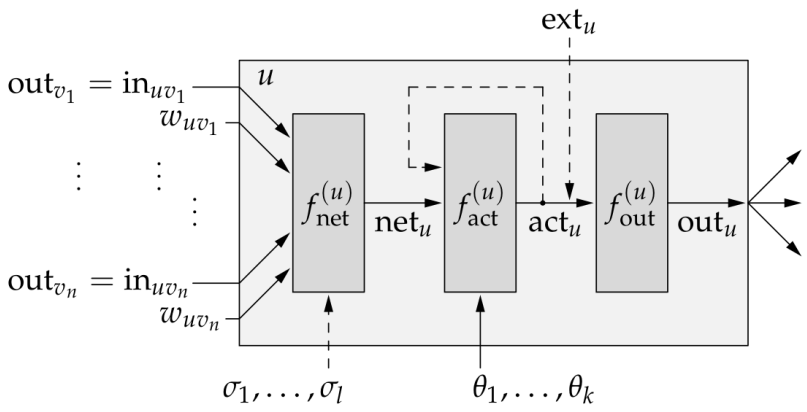

Abbildung 2.1: Aufbau eines allgemeinen Neurons. Dabei sind $out_{v_1}, i \in \{1, \dots, n\}$ die Ausgabe des Neurons v_i, in_{uv_i} die Eingabe des Neurons v_i in u, w_{uv_i} die Einträge der Gewichts-Adjazenzmatrix, $\sigma_1, \dots, \sigma_l$, $\theta_1, \dots, \theta_k$, ext_u externe Eingabe in die jeweile Neuronenfunktion. Die Abbildung stammt aus [KBB+15], Seite 37.

2.2 Neuronale Netze

Nachdem im vorhergehenden Abschnitt Neuronen vorgestellt wurden, geht es nun um kompliziertere Strukturen aus Neuronen, den neuronalen Netzen. Ein NN kann man sich als gerichteten Graph $G(V, E)$ vorstellen, wobei $V = \{v_1, \dots, v_n\}$ die

Menge der Knoten und $E \subseteq V \times V$ die Menge der Kanten darstellt. Insbesondere deutet man die Knoten als Neuronen. Dabei existiert genau dann eine Kante $(u, v) \in E$ zwischen zwei Neuronen $u, v \in V$, wenn Neuron v eine Eingabe von u erhält. Weiterhin unterscheidet man in Eingabeschicht (auch input layer), also die Neuronen, die Eingaben von Außen bekommen und Transformationen der Eingabedaten durchführen, der verdeckten Schicht (auch hidden layer), die Neuronen, die die eigentlichen Berechnungen ausführen und der Ausgabeschicht (auch output layer), also den Neuronen, die Daten an die Umwelt zurückgeben.

2.2.1 Spezielle Neuronale Netze

In diesem Abschnitt wird kurz auf spezielle neuronale Netze eingegangen, insbesondere den Autoencodern und darauf aufbauend den entrauschenden (denoising) Autoencodern.

Autoencoder Ein Autoencoder ist ein NN, dass aus einem Encoder und einem Decoder besteht. Diese werden hauptsächlich zur Komprimierung und Dekomprimierung von Daten genutzt. Ein eingehendes Signal $x \in \mathbb{R}^n$ wird dabei über eine Funktion $f_\theta(x) = s(Wx + b) = y \in \mathbb{R}^m$, $\theta = \{W, b\}$, $m \leq n$ komprimiert. Der Decoder wandelt dann dieses verdichtete Signal wieder mittel der Funktion $g_{\theta'}(y) = s(W^T y + b') = z$, $\theta = \{W^T, b'\}$ um. Dabei ist $s(\cdot)$ zumeist eine hyperbolische Tangensfunktion oder eine sigmoide Funktion. Diese werden so gewählt, da man von nichtlinearen Zusammenhängen in den Daten ausgeht, für die es ein Modell zu entwerfen gilt. Bei der Transformation entsteht dementsprechend ein Rekonstruktionsfehler $L(x, z)$. Abbildung 2.2 veranschaulicht den Autoencoder.

Entrauschende Autoencoder Entrauschende AEs sind Erweiterungen von Autoencodern, bei denen als Zwischenschritt vor der Kodierung der Daten eine stochastische Abbildung $q_D(x)$ auf die Eingabedaten angewandt wird. Dabei werden gewisse Daten im Eingabevektor gelöscht beziehungsweise null setzt. Somit können fehlerbehaftete Daten (z.B. fehlende Medikation in EHRs) simuliert und 'overfitting' vermieden werden. Ebenfalls entsteht hier der Rekonstruktionsfehler $L_H(x, z)$. Die nachstehende Abbildung 2.3 zeigt den allgmeinen Aufbau.

2.2.2 Training neuronaler Netze

In diesem Abschnitt wird kurz auf das Training von NNs mittels Fehler-Rückübertragung (engl. error backpropagation) mithilfe des Gradienten-Abstiegsverfahrens (kurz GAV) eingegangen. Dabei werden sowohl das Online-, als auch das Batch- beziehungsweise das Mini-Batch-Training betrachtet.

Prinzipiell wird beim Training von NNs nach [KBB+15] zwischen freien und festen Lernaufgaben unterschieden. Eine feste Lernaufgabe ist demnach gegeben

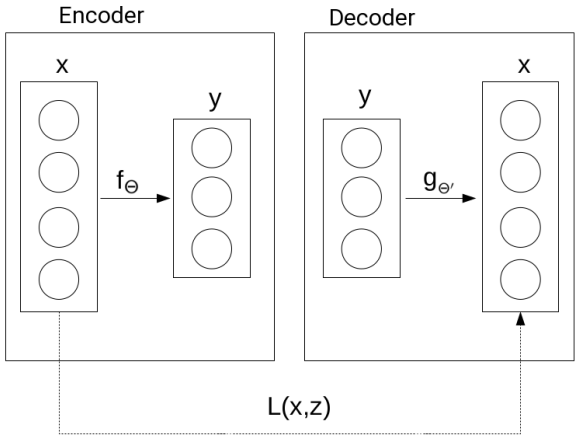

Abbildung 2.2: Aufbau eines Autoencoders mit Eingabe x, komprimierten Daten y (über $f_\theta(x)$), Dekomprimierung z (über $g_{\theta'}(y)$) und $L(x, z)$ als Rekonstruktionsfehler. Die Abbildung ist abgewandelt aus [MLKD16].

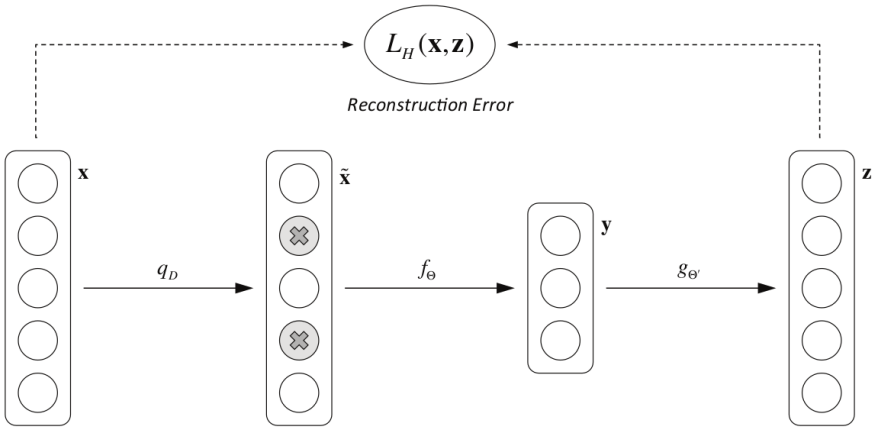

Abbildung 2.3: Aufbau eines entrauschenden Autoencoders mit Eingabe x, komprimierten Daten y (über $f_\theta(\tilde{x})$), Dekomprimierung z (über $g_{\theta'}(y)$) und $L_H(x, z)$ als Rekonstruktionsfehler. Dabei wird vor der Komprimierung ein stochastisches Mapping $q_D(x) = \tilde{x}$ angewandt. Die Abbildung ist aus [MLKD16].

durch eine Menge von Tupeln, wobei jedes Tupel aus einer Eingabe und der zugehörigen erwarteten Ausgabe besteht. Freie Lernaufgaben bestehen dabei nur aus einer Menge von n Eingaben $\{ext^l \mid l = 1, \ldots, n\}$, wobei zu beachten ist, dass die Ähnlichkeit dieser Eingaben messbar sein soll. Oft werden diese daher z-skaliert. Demnach berechnet sich die skalierte Eingabe $ext^l_{\text{neu}} = \frac{ext^l - \mu}{\sigma}$, wobei μ der Mittelwert und σ die Standardabweichung der Eingaben ist.

Gradientenabstieg Der Gradientenabstieg ist eine Methode zum Training neuronaler Netze. Dabei kann allgemein gesagt werden, dass der entstehende Fehler beim Training des NNs an den Ausgabeneuronen, also der Unterschied zwischen tatsächlich erhaltener und erwarteter Ausgabe, nur von der Eingabe des Netzes abhängig ist. Ferner ergibt sich für eine Lernaufgabe $l = (i^l, o^l) \in L$ der Fehler über die Summe der Fehler für diese Ausgabe pro Ausgabeneuron U_{out}, also $e^{(l)} = \sum_{u \in U_{\text{out}}} e_u^{(l)}$. Es wird also der Gradient der Fehlerfunktion betrachtet, welcher sich als

$$\forall u \in U_{\text{out}} : \nabla_{w_u} e_u^{(l)} = \frac{\partial e_u^{(l)}}{\partial w_u} = -2(o_u^{(l)} - out_u^{(l)}) \frac{\partial out_u^{(l)}}{\partial net_u^{(l)}} in_u^{(l)} \tag{2.2}$$

ergibt. Dabei ist w_u der zugehörige Gewichtsvektor und $o_u^{(l)}$ die von der Lernaufgabe l erwartete Ausgabe am Neuron u und $net_u^{(l)} = w_u in_u^{(l)}$ die Netzeingabe mit erweitertem Netzeingabevektor $in_u^{(l)} = \{1, out_{p_1}^{(l)}, out_{p_2}^{(l)}, \ldots, out_{p_m}^{(l)}\}$ mit den Vorgängern des Neurons u als $\text{pred}(u) = \{p_1, \ldots, p_m\}$. Somit ist die allgemeine Gewichtsänderung gegeben durch

$$\forall u \in U_{\text{out}} : \Delta w_u = -\frac{\eta}{2} \nabla_{w_u} e_u^{(l)} = \eta \underbrace{(o_u^{(l)} - out_u^{(l)}) \frac{\partial out_u^{(l)}}{\partial net_u^{(l)}} in_u^{(l)}}_{\delta_u^{(l)}} \tag{2.3}$$

mit der Lernrate η. Die genaue Herleitung ist unter Anderem in [KBB+15] nachzulesen. Da die Ausgabe von der Netzeingabe über die Ausgabefunktion und der Aktivierungsfunktion abhängt, kann dies nicht allgemein bestimmt werden. In Abbildung 2.4 ist das grundlegende Vorgehen anhand der logistischen Aktivierungsfunktion dargestellt.

Online-Training vs (mini-) Batch-Training Das Training des NNs spielt eine große Rolle in Bezug auf die Anpassung der Parameter. Dies ist vom jeweiligen Anwendungstyp abhängig und muss somit dementsprechend gewählt werden. Beim Online-Training werden nach jedem Lernmuster die Gewichte der Neuronen angepasst. Somit ist ein großer Aufwand in der häufigen Berechnung des Gradienten die Folge, was mit der Anzahl der Schichten im NN und der Zahl der Neuronen

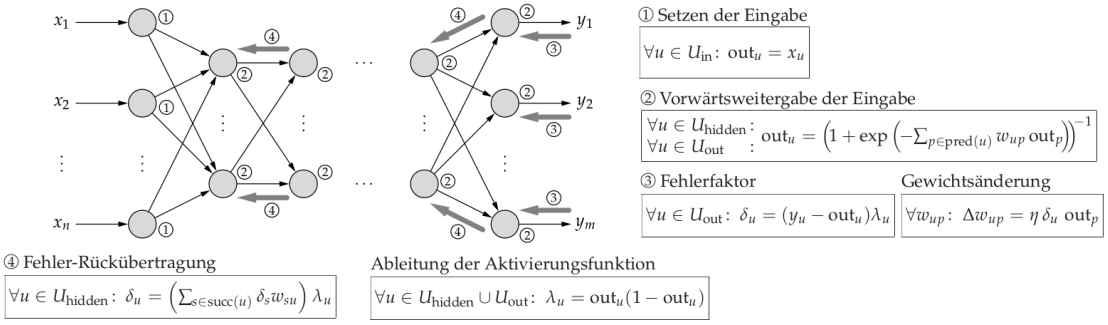

Abbildung 2.4: Ablauf der Fehlerrückübertragung mit Gradientenabstieg am Beispiel der logistischen Funktion. Die Abbildung ist aus [KBB+15].

pro Schicht korreliert. Beim Batch-Training hingegen werden die Fehler nach jedem Durchlauf eines Trainingsmusters gespeichert und aufsummiert, die Änderung jedoch nicht vorgenommen. Die Gewichte der Neuronen bleiben also während der gesamten Trainingsperiode gleich. Erst am Ende folgt dann die Anpassung der Gewichte. Als Zwischenstufe existiert das mini-Batch-Training. Dort wird über Teilmengen der Trainingsbeispiele ein Batch-Training vorgenommen, sodass nach dem Training über eine Teilmenge die Gewichte angepasst werden.

2.3 Verfahren des maschinellen Lernens

Maschinelles Lernen beruht grundsätzlich auf der Abarbeitung vorher definierter Algorithmen und der Erstellung von Entscheidungsbäumen [Lec17]. Dabei werden Entscheidungen gefällt, die durch erhobene Daten gestützt werden. Insbesondere werden zur Lösung von Klassifikationsproblemen oft maschinelle Lernverfahren genutzt.

2.3.1 Entscheidungsbäume und Random Forest

Ein Hauptkonstrukt des maschinellen Lernens sind Entscheidungsbäume und daraus generierte Random Forests.

Entscheidungsbäume sind binäre Bäume, wobei in jedem Blatt eine binäre Entscheidung zu einem Attribut gefällt wird. Ferner kann mit einem Entscheidungsbaum im Allgemeinen auch nur genau ein Problem beziehungsweise eine Fragestellung gelöst werden. Dabei arbeitet man sich von der Wurzel zu einem Blattknoten, sodass man dort die Entscheidung ablesen kann. Vorteile der Nutzung von Entscheidungsbäumen ist, dass die Regeln gut abgelesen werden können und zumeist nachvollziehbar sind. Ein Nachteil liegt in der Güte von Klassifikation, wenn reale

Probleme betrachtet werden. Hier kommt es oft zur Überanpassung der Daten und weiterhin zu extrem großen Bäumen.

Random Forests sind Mengen von Entscheidungsbäumen, die vorwiegend zur Klassifikation genutzt werden. Insbesondere darf hier jeder Entscheidungsbaum einen Entscheidung für die Klassenzugehörigkeit eines Problems treffen. Dabei gewinnt die Klasse mit den meisten Stimmen. Zur Generierung von solchen Random Forests existieren viele Algorithmen. Im Wesentlichen werden aber die einzelnen Bäume aus einer Teilmenge von Entscheidungen gebildet. Dies verhindert in den meisten Fällen eine Überanpassung an ein Problem.

2.3.2 Principle Component Analysis

Grundlegend geht es bei der Principle Component Analysis (abgekürzt PCA) darum, eine Korrelation zwischen bestimmten Eigenschaften zu erkennen und diese durch eine möglichst geringe Anzahl von neuen Eigenschaften widerzuspiegeln. Mathematisch gesehen wird dabei eine Hauptachsentransformation ausgeführt. Angenommen es werden p Eigenschaften gemessen. Ziel ist es, diese in einem $r < p$-dimensionalen Raum darzustellen.

Algorithmus: Sei eine Punktwolke im p-dimensionalen Raum gegeben. Man wählt nun eine Gerade durch den Mittelpunkt. Jeder Abstand zwischen einem Punkt und dem Mittelpunkt der Daten kann nun in einen Anteil entlang der vorher gewählt Geraden und einem senkrecht darauf stehenden Anteil dargestellt werden. Minimiert man den senkrecht auf der Gerade stehenden Anteil, so maximiert man den Anteil entlang der Geraden. Quadriert man diesen, spiegelt er die Varianz der Daten entlang der Geraden wider. Die nächste Achse wird senkrecht auf der Vorhergehenden Geraden gewählt, wobei ebenfalls wieder die Varianz maximiert wird. Dabei wird die Summe der Varianzen einer Achse als Achsenvarianz bezeichnet. Die Summe der p entstehenden Achsenvarianzen bezeichnet man als totale Varianz. Ergibt die Varianz der ersten $r < p$-Achsen nun einen hinreichenden Anteil der totalen Varianz, so können die Daten dadurch gut reflektiert werden. Somit erhält man eine Dimensionsreduzierung der Daten.

2.3.3 k-means clustering

Beim k-means clustering werden die Daten geclustert, um die Zugehörigkeit von Datenpunkten zu einer bestimmten der k Klassen festzulegen. Dabei betrachtet man die Zentren der Klassen und ordnet die Punkte im Allgemeinen nach der euklidischen Distanz zu.

Algorithmus: Der am häufigsten verwendet Algorithmus ist der Lloyd-Algorithmus. Dabei wählt man zufällig k-Mittelwerte aus den Daten. Anschließend ordnet man jeden Punkt genau dem Cluster zu, dessen Clustervarianz am geringsten erhöht wird. Darauffolgend werden die Clustermittelpunkte neu berechnet. Die Zuordnung der Punkte zu den Klassen und die Neuberechnung der Mittelpunkte wird solange wiederholt, bis sich die Zuordnung nicht mehr ändert. Es ergibt sich eine Klassifikation der Punkte, indem jeder Punkt genau einem Mittelpunkt zugeordnet ist.

3 Krankheitsvorhersage

Patientendaten werden zumeist in Krankenhäusern, aber auch in Arztpraxen und Ähnlichem, elektronisch erfasst. Dabei speichert man Diagnosen, Behandlungen, sowie deren Erfolg oder Misserfolg, Behandlungsverläufe, Medikation, Laborberichte und Weitere. Dementsprechend ergeben sich große Datenmengen, die es auszuwerten gilt, sodass die medizinische Versorgung verbessert werden kann. Große Probleme treten dabei in der Vielfältigkeit der Daten auf. Beispielsweise kann eine Krankheit auf verschiedene Art dokumentiert werden, als ICD-9, Mesh oder SnoMed Code, als Beschreibung im Freitext und viele Weitere. Weiterhin existieren verschiedene Frameworks zur Speicherung von Patientendaten, sodass die Datenbestände durchaus heterogen sind.

3.1 EHR-Prozessierung

Die zu verarbeitenden Daten wurden dem Mount Sinai Health System entnommen, in dem sie in einem Data Warehouse gespeichert sind. Dort existieren eine Menge an un-, semi- und strukturierten Daten, also sowohl Freitexte wie Notizen, als auch direkt per ICD-9 Code getaggte Krankheiten. Insbesondere werden die Patientendaten dort seit 2003 gespeichert, wobei ältere Daten seit 1980 manuell eingepflegt wurden. Dies entspricht ungefähr 1,3 Millionen Patienten mit mindestens einem ICD-9 Eintrag. Dabei existieren pro Patient durchschnittlich 88,9 Einträge.

3.1.1 Vorverarbeitung

Als erstes werden die Daten aus dem Data Warehouse extrahiert und liegen danach als Patientvektor mit Medikation, Diagnosen und so weiter vor. Anschließend werden die Daten weiterverarbeitet, indem klinisch relevante Phenotypen extrahiert werden und eine Normalisierung stattfindet. Dies geschieht mittels des Open Biomedical Annotator, um einheitliche Deskriptoren (zum Beispiel Medikationen markenunabhängig) zu erhalten. Ferner sind Notizen verarbeitet wurden, indem negierte Aussagen gelöscht und anschließend ein Topicmodelling durchgeführt wurde. Für jeden Patienten behielt man demnach höchstens eine Notiz mit einem von 300 möglichen Topics. Anschließend wurde ein ICD-9 Mapping auf 231 Krankheiten durchgeführt, wonach unvorhersagbare Krankheiten (zum Beispiel durch

soziales Verhalten ausgelöste Krankheiten) manuell aussortiert wurden. Es ergabe sich 78 relevante Krankheiten. Zum Entfernen von Rauschen wurden klinische Deskriptoren, die mehr als 80% und weniger als fünf Patienten besaßen ebenfalls gelöscht.

3.1.2 Deep Patient Repräsentation

Die vorverarbeiteten Patientendaten werden nun mittels eines neuronales Netzes weiter verarbeitet. Dieses NN besteht aus einem Stack aus entrauschenden Autoencodern (siehe Abschnitt 2.2.1). Dadurch können fehlende Medikationen, Behandlungen in anderen Krankenhäusern, etc., also Fehler in den Daten, simuliert werden. Dabei wird für die entsprechenden Neuronen jeweils eine sigmoide Aktivierungsfunktionen, sowie die Identitätsfunktion als Ein- beziehungsweise Ausgabefunktion genutzt. Weiterhin erfolgt das Training über die Minimierung des Rekonstruktionsfehlers, der als Kreuzentropiefunktion angenommen wird. Pro verdeckter Schicht werden 500 entrauschende Autoencoder, bei drei verdeckten Schichten angenommen, wobei alle Neuronen die gleichen Parameter besitzen. Ferner wird das Mini-Batch Verfahren zum Lernen der Eigenschaften genutzt. Für die entrauschenden Autoencoder ist ein Rausch-Korruptionsfaktor von $\nu = 5\%$ vorgesehen. Man erhält somit eine neue Darstellung der Patienten, folgend Deep Patient genannt. Abbildung 3.1 veranschaulicht nochmals den bisherigen Ablauf. Der Aufbau des Neuronalen Netzes wird in Abbildung 3.2 gezeigt.

Lernen der Deep Patient Repräsentation Alle Patienten mit mindestens einem ICD-9 Eintrag werden in drei Mengen unterteilt, sodass jeder Patient in höchstens einer Menge vorkommt. Die erste Menge besteht aus Patienten, die einen oder mehr Einträge in 2014 haben und mehr als neun Einträge in den Jahren davor. Dies sind 81.214 Patienten. Davon werden 5000 Patienten für die Evaluation genutzt und 76.214 für den Test des überwachten Lernens. Eine Menge von 704.587 Patienten mit mindestens fünf Einträgen vor 2014 wird zufällig gewählt und als Trainingsmenge genutzt. Diese enthalten 60.238 Deskriptoren. Nach der oben genannten Restriktion verbleiben 41.072 Deskriptoren. Insgesamt enthält die Patienten-Deskriptor Matrix 200 Millionen nicht-null Einträge.

Evaluation der Deep Patient Repräsentation Die Menge von 704.587 Patienten durchlaufen das neuronale Netz zum Lernen der Parameter. Anschließend werden die Ergebnisse mit anderen Feature-Lern-Verfahren (insbesondere PCA(100 Hauptkomponenten), k-means clustering, Gaussian mixture model, ICA(100 Hauptkomponenten) und RawFeat) verglichen. RawFeat ist dabei die originale Repräsentation ohne die Vorverarbeitung der Daten.

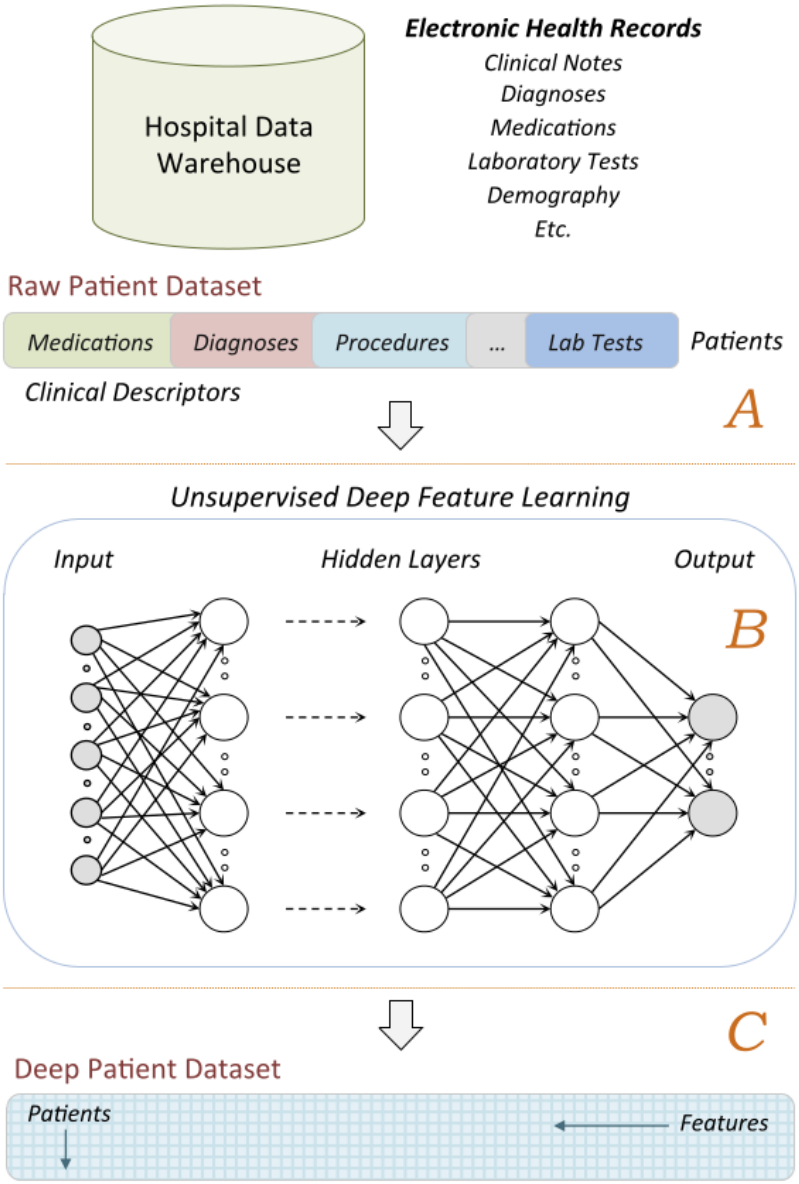

Abbildung 3.1: Ablauf der Erstellung der Deep Patient Repräsentation [MLKD16].

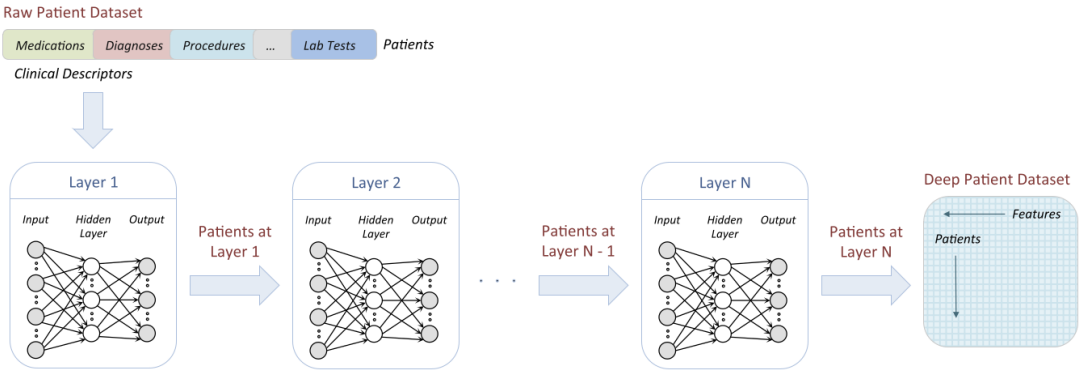

Abbildung 3.2: Neuronales Netz für die Deep Patient Repräsentation [MLKD16].

Zur Evaluation wird die Vorhersage von Krankheiten mit den oben genannten Feature Repräsentationen genutzt. Dafür erstellt man einen Random Forest, der mit 200.000 Patienten trainiert wird, wobei pro Klassifikator 100 Bäume zum Einsatz kommen. Ferner wird das one-vs.-all Lernen genutzt.

Für jeden Patienten wird nun ein Wahrscheinlichkeitsvektor erstellt, der die Wahrscheinlichkeit eine Krankheit zu entwickeln beinhaltet. Dabei wird ab einem Grenzwert von 0.6 angenommen, dass ein positives Ergebnis für die Krankheit vorliegt.

Die Evaluation erfolgt mittels der Zuordnung von Patienten zu Krankheiten mit Hilfe verschiedener Repräsentationen, sowie der Zuordnung von Krankheiten zu Patienten in verschiedenen Zeitfenstern.

Genutzte Gütemaße Es werden folgende Gütemaße genutzt, wobei 'true' bedeutet, dass das Ergebnis richtig erkannt wurde und 'positive', dass die Krankheit auch tatsächlich entwickelt wird. 'false' beziehungsweise 'negative' entsprechen somit genau dem Gegenteil.

- AUC-ROC: Fläche unter der false-positive, true-positive Kurve
- precision:
$$\frac{\text{true-positive}}{\text{true-positive} + \text{false-positive}}$$
- recall:
$$\frac{\text{true-positive}}{\text{true-positive} + \text{false-negative}}$$
- Accuracy:
$$\frac{\text{true-positive} + \text{true-negative}}{\text{alle untersuchten Fälle}}$$

- F-Score:
$$F_\alpha = \frac{(1+\alpha^2) \cdot \text{precision} \cdot \text{recall}}{\alpha^2 \cdot \text{precision} + \text{recall}}$$
- precision@k: Durchschnittlich korrekt zugewiesene Krankheiten innerhalb der besten k Krankheiten

Ergebnisse Um zu messen, wie gut die Deep Patient Repräsentation bei der Vorhersage von Krankheiten ist, testet man, ob bei einem Patienten innerhalb eines Jahres eine vorhergesagte Krankheit eintritt. Dafür misst man für jede Krankheit die Wahrscheinlichkeit, die man für die Patienten in der Testmenge erhält und vergleicht diese mit der Repräsentation durch die anderen Klassifikationsalgorithmen. Die folgende Abbildung 3.3 stellt die Ergebnisse zusammen. Dabei sieht man, dass

Time Interval = 1 year (76,214 patients)			
		Classification Threshold = 0.6	
Patient Representation	**AUC-ROC**	**Accuracy**	**F-Score**
RawFeat	0.659	0.805	0.084
PCA	0.696	0.879	0.104
GMM	0.632	0.891	0.072
K-Means	0.672	0.887	0.093
ICA	0.695	0.882	0.101
DeepPatient	**0.773***	**0.929***	**0.181***

Abbildung 3.3: Ergebnisse für die Krankheitsklassifikation [MLKD16]

die durch Deep Patient erhaltenen Ergebnisse sowohl im AUC-ROC und Accuracy Score, als auch im im F-Score (F_1-Score) besser gegenüber den anderen Repräsentationen sind. Insbesondere zum RawFeat kann eine Verbesserung im Accuracy um 15% und im F-Score um 54% erzielt werden. (*)Ferner ist zu erwähnen, dass die Ergebnisse statistisch signifikant sind, was durch einen T-Test mit $p < 0.05$ getestet wurde.

Weiterhin wurde getestet, wie gut die Ergebnisse pro Patienten sind. Hier wurden die Krankheitsvorhersagen mit einer Wahrscheinlichkeit größer als 0.6 behalten und über verschiedene Zeitfenster (30, 60, 90 und 180 Tage) gemessen, ob ein Patient eine Krankheit in diesem entwickelt. Die Ergebnisse sind nachfolgender Abbildung 3.4 zu entnehmen. Hier wurde die precision@k Methode verwendet,

Time Interval	Metrics	UppBnd	Patient Representation			
			RawFeat	PCA	ICA	DeepPatient
30 days *(16,374 patients)*	Prec@1	1.000	0.319	0.343	0.345	**0.392***
	Prec@3	0.492	0.217	0.251	0.255	**0.277***
	Prec@5	0.319	0.191	0.214	0.215	**0.226***
60 days *(21,924 patients)*	Prec@1	1.000	0.329	0.349	0.353	**0.402***
	Prec@3	0.511	0.221	0.254	0.259	**0.282***
	Prec@5	0.335	0.199	0.216	0.219	**0.230***
90 days *(25,220 patients)*	Prec@1	1.000	0.332	0.353	0.360	**0.404***
	Prec@3	0.521	0.243	0.257	0.262	**0.285***
	Prec@5	0.345	0.201	0.219	0.220	**0.232***
180 days *(33,607 patients)*	Prec@1	1.000	0.331	0.361	0.363	**0.418***
	Prec@3	0.549	0.246	0.261	0.265	**0.290***
	Prec@5	0.370	0.207	0.221	0.224	**0.236***

Abbildung 3.4: Ergebnisse für die Krankheitsklassifikation im Vergleich mit originalen Deskriptoren und vorverarbeitet durch PCA und Deep Patient [MLKD16]

wobei k hier für die Anzahl der Top-k-Ergebnisse steht. Es ist zu sehen, dass die Vorhersage mittels Deep Patient gegenüber allen anderen Repräsentationen besser ist. Ferner wurde mit dem Modell der theoretischen Obergrenze (UppBnd) gearbeitet, was die besten möglichen Ergebnisse darstellt. Insbesondere erzielt die Vorhersage mittels Deep Patient einen Wert von circa 55% richtiger Vorhersagen, wenn drei oder mehr Krankheiten unabhängig vom Zeitintervall pro Patient vorhergesagt werden sollen. (*) Die Ergebnisse wurden wieder positiv mittels eines T-Tests bei $p < 0.05$ auf statistische Signifikanz getestet.

3.2 Identifikation sensitiver ROIs für die Alzheimer Vorhersage

Die Vorhersage von Alzheimer spielt eine große Rolle in der medizinischen Vorsorge. Dies ist begründet durch die weite Verbreitung dieser Krankheit. Während 2006 noch ungefähr 26,6 Millionen Erkrankte registriert waren, vermutet man eine Verdoppelung dieses Wertes aller 20 Jahre. Dementsprechend sind 2046 circa 1,2% der Weltbevölkerung davon betroffen.

Als Vorstufe gilt das sogenannte Mild Cognitive Impairment (MCI). Dies kann sich schleichend zu Alzheimer entwickeln. Dementsprechend ist es notwendig, eine Markierung von MCI zu cMCI (converter MCI) beziehungsweise ncMCI (non

converter MCI) vorzunehmen. Man erhält also eine Klassifikationsaufgabe.

Ein großes Problem bei der Identifikation sind die Daten, die aus bildgebenden Biomarkern wie zum Beispiel MRI oder PET gewonnen werden können und aus hochdimensionalen Eigenschaftsräumen bestehen.

Somit ergibt sich das Ziel der genauen Identifikation von Regions of Interest (ROI), wobei die Nutzung von a-priori Wissen minimiert werden soll. Dies liegt begründet in dem hohen Aufwand des Einbindens dieses Wissens, was bei sehr vielen Algorithmen zum Flaschenhals wird. Bisher wurden Verfahren aus dem Bereich des maschinellen Lernens verwendet, man erreichte aber im Allgemeinen eine schlechte Performanz, durch die Limitierung dieser Lernverfahren bei sehr großen Featuremengen. Es gilt also, sich den Herausforderungen des Entwirrens der komplexen und strukturellen Abhängigkeiten zu stellen.

3.2.1 Datengrundlage

Die genutzten Daten sind neuro-abbildende Daten von der Alzheimer disease Neuroimaging Initiative (ADNI) Datenbank. Dabei wurden 311 Bilderdaten genutzt, die sich in 65 Alzheimer Bilder, 67 cMCI Bilder, 102 ncMCI Bilder und 77 Bilder ohne Erkrankung aufteilen. Diese Bilder sind in 83 funktionelle Regionen aufgeteilt. Ferner sind alle Eigenschaften auf das [0,1] Intervall normalisiert, um sigmoide Aktivierungsfunktionen in den Neuronen zuzulassen.

3.2.2 Verwendetes Neuronales Netz

Als neuronales Netz kommen geschichtete entrauschende Autoencoder zum Einsatz, um die Regionen zu identifizieren. Ferner wurde eine Softmax-Regressionsschicht hinzugefügt, um die Ausgaben nach den Mustern keine Erkrankung (NC, von normal control), cMCI, ncMCI und AD zu ordnen.

Zur Aktiverung a eines Neurons wurde folgende Gleichung 3.1 genutzt:

$$a = \begin{cases} a_{(l)}^{(i)} = x^{(i)}, \text{ falls } l = 1, \\ a_{(l)}^{(i)} = \sigma(W^l a + b), \text{ falls } l > 1 \end{cases} \tag{3.1}$$

und $h(W, b, x) = a^{(N)}$ als Repräsentation der Eingabedaten beziehungsweise die Aktivierung an der Ausgabeschicht, wobei $\{x^{(i)}\}_{i=1}^{m}$ die ungelabelten Daten sind, W ist die Gewichtsmatrix, b der Bias Term und $\sigma(\cdot)$ die Aktivierungsfunktion (sigmoid oder hyberbolische Tangensfunktion).

Die Fehlerfunktion zum Training war

$$L(W, b, x, z) = \min_{W,b} E(W, b, x, z) + \gamma \|W\|_2^2 + \beta K(W, b, x) \tag{3.2}$$

mit $E(W, b, x, z) = \|h(W, b, x) - z\|_2^2$ als quadratischer Fehlerterm, $\gamma\|W\|_2^2$ führt zu kleinen Gewichten und $K(W, b, x) = \sum_j^n ID_{KL}(\rho\|\frac{1}{m}\sum_{i=1}^m h_j(x^{(i)}; W, b))$ als Kullback-Leibler Divergenz. Die Faktoren γ und β steuern den Einfluss dieser Terme. Diese wurde mittels Gradienten-Abstiegs-Verfahren optimiert. Ferner wurde ein Online-Training angewandt.

Als Softmax-Aktiverungsfunktion wurde

$$h_i^l = \frac{e^{W_i^l h^{l-1} + b_i^l}}{\sum_j e^{W_j^l h^{l-1} + b_j^l}} \tag{3.3}$$

genutzt. Dabei ist W_i^l die i-te Zeile von W^l, b_i^l der i-te Bias Term der letzten Schicht und h_i^l ein Schätzer für $P(Y = i \mid x)$, wobei Y das zugehörige Label des Eingabevektor x ist.

3.2.3 Identifikation der ROIs

Es wurde erkannt, dass die Neuronen der ersten, verdeckten Schicht verschiedene Muster der Eingabedaten entwickeln. Diese können genutzt werden, um die für Alzheimer sensitiven ROIs zu identifizieren. Sei nun x_j^* das Eingabemuster, dass das Neuron a_i maximal aktiviert. In diesem Fall gilt $x_{ij}^* = \frac{W_{ij}^{(1)}}{\|W\|_2}$. Anschließend kann x in m Sichten auf die Features aufgeteilt werden. Man berechnet nun die Varianz $D_j^{(m)}$ aller $x_j^{(m)}$ der gleichen ROI über die Aktivierung verschiedener versteckter Neuronen. Ist die Varianz niedrig, ist die ROI stabiler für die Alzheimer Diagnose, als Regionen mit hoher Varianz. Die Gesamtstabilität einer Eigenschaft S_j der j-ten ROI ist demnach

$$S_j = \sum_m \frac{\sum_j D_j^{(m)}}{D_j^{(m)}}. \tag{3.4}$$

3.2.4 Ergebnisse

Man kann nun die Eigenschaftsstabilität auf 3D MRI Bilder mit 83 ROIs abbilden (Abbildung 3.5). Dabei wird ein Gaußscher Filter eingesetzt, um die Unterschiede der Regionen zu verdeutlichen. Hellere Regionen sind dabei betroffener von Alzheimer Progression als dunklere. In Tabelle 3.6 sind die Durchschnittswerte des binären Klassifikationsproblems in % dargestellt. Der Vergleich findet gegen bereits bekannte SVM-basierte Verfahren (SK-SVM und MK-SVM) [ZWZ+11] statt. Man kann gut erkennen, dass die Deep Learning Methode bei der AD-NC Klassifikation insgesamt mit der besten accuracy (ACC) abschneidet und genauso gut bei MCI vs. NC Klassifikation ist. Ebenfalls werden sehr gute Werte bei der Spezifität (SPE) und der Sensitivität (SEN) erreicht.

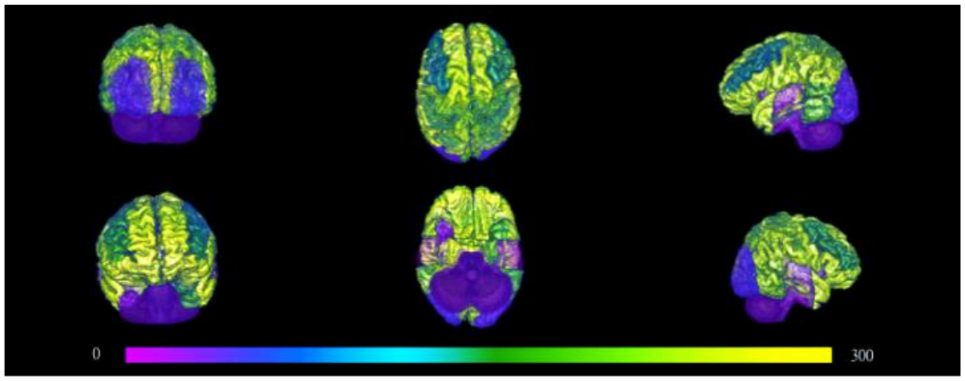

Abbildung 3.5: Mapping der Gesamtstabilität, dunkle Regionen sensitiver für Progression von AD und MCI [LLC+14]

Methods	AD vs. NC			MCI vs. NC		
	ACC	SEN	SPE	ACC	SEN	SPE
SK-SVM	84.40	84.64	84.31	76.81	56.14	86.24
MK-SVM	86.42	84.98	87.83	77.25	55.48	87.10
Proposed	87.76	88.57	87.22	76.92	74.29	78.13

Abbildung 3.6: Durchschnittswerte der binären Klassifikationsperformanz in % mit vorberechneten MRI und PET Bildern [LLC+14]

4 Zusammenfassung

Im ersten Kapitel wird kurz auf die Notwendigkeit der Prozessierung von EHR-Daten und Vorhersage von für Alzheimer relevanten Regionen eingegangen. Ferner wird ein Überblick durch die darauf folgenden Kapitel gegeben.

Das zweite Kapitel behandelt eine Einführung in die Grundlagen zum Verständnis der darauf aufbauenden nächsten Sektionen. Es wird kurz die Geschichte von Neuronen, den Schwellenwertelementen, behandelt und danach mit dem Aufbau von allgemeinen, künstlichen Neuronen fortgeführt. Weiterhin werden neuronale Netze eingeführt, wobei auch auf spezielle neuronale Netze, wie dem Autoencoder und dem entrauschenden Autoencoder eingegangen wird. Abschließend werden Verfahren des maschinellen Lernens, insbesondere Entscheidungsbäume, random forests, das PCA und das k-means clustering behandelt. Dies soll als Vorstellung möglicher Verfahren angesehen werden.

Im anschließenden Kapitel Drei wird auf die Krankheitsvorhersage mittels EHR-Prozessierung durch Deep Learning eingegangen. Dabei entstand eine neuartige Repräsentationsform von Patientendaten, Deep Patient genannt, durch die qualitativ sehr gute Ergebnisse im Vergleich mit maschinellen Lernverfahren erzielt werden konnten. Insbesondere ist die Verwendung von denoising Autoencodern hervorzuheben, durch die fehlende Punkte in den Patientendaten simuliert wurden. Abgeschlossen wurde diese Sektion mit der Identifikation von für Alzheimer sensitiven Regionen im Gehirn. Die Schwierigkeit lag dort vor allem in der hochdimensionalen Darstellung von Features. Es wurde ein Mapping von den erhaltenen Daten auf ein MRI Bild vorgenommen. Dies geschah über die Varianz von Eingabemustern. Dabei konnten ebenfalls mit der Nutzung von entrauschenden Autoencoder sehr gute Ergebnisse erzielt werden.

Gerade in Bezug auf die medizinische Auswertung von Daten und allgemein bei hochkomplexen Fragestellen ist zukünftig eine vermehrte Nutzung von Deep Learning absehbar. Die vorgestellten Paper zeigen, dass sowohl die Performanz von solchen Systemen zur Analyse in Bezug auf die Qualität und die Verarbeitungsgeschwindigkeit von großen Datenmengen durch die Nutzung von Deep Learning gesteigert werden kann.

Literaturverzeichnis

[KBB+15] R. Kruse, C. Borgelt, C. Braune, F. Klawonn, C. Moewes, and M. Steinbrecher. *Computational Intelligence*. Springer Vieweg, 2015.

[Lec17] Jérôme Lecat. Ki, maschinelles lernen und deep learning – das sind die unterschiede. https://www.bigdata-insider.de/ki-maschinelles-lernen-und-deep-learning-das-sind-die-unterschiede-a-588067/, 2017. Stand: 12.03.2018.

[LLC+14] S. Liu, S. Liu, W. Cai, S. Pujol, R. Kikinis, and D. Feng. Early diagnosis of alzheimer's disease with deep learning. *2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI)*, 2014.

[MLKD16] R. Miotto, L. Li, B.A. Kidd, and J.T. Dudley. Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. *Scientific Reports*, 2016.

[MP43] W.S. McCulloch and W.H. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, vol. 5:pp. 115–133, 1943.

[Ros58] F. Rosenblatt. The perceptron: A probabilistic modell for information storage and organization in the brain. *Psychological Review*, vol. 65:pp. 386–408, 1958.

[Wer74] P.J. Werbos. Beyond regression: New tools for prediction and analysis in the behavioral sciences. *Ph.D. Thesis, Harvard University*, 1974.

[ZWZ+11] D. Zhang, Y. Wang, L. Zhou, H. Yuan, and D. Shen. Multimodal classification of alzheimer's disease and mild cognitive impairment. *Neuroimage,*, vol. 55:pp. 856–867, 2011.