Optimierung der Architektur neuronaler Netze für modulare Probleme

Anna Schua

Optimierung der Architektur neuronaler Netze für modulare Probleme

Anna Schua

13. August 2019

Lehrstuhl für Datenverarbeitung
Technische Universität München

Anna Schua. *Optimierung der Architektur neuronaler Netze für modulare Probleme.* Technische Universität München, München, 2019.

Lehrstuhl für Datenverarbeitung, Technische Universität München, 80290 München, http://www.ldv.ei.tum.de/.

Zusammenfassung

Eine Extreme Learning Machine (ELM) initialisiert den Bias und die Gewichte zwischen der Eingabeschicht und den versteckten Schichten willkürlich. Mithilfe der Methode der kleinsten Quadrate können die Gewichte zwischen der versteckten Schicht und der Ausgabeschicht mit wenig Rechenaufwand berechnet werden. Diese Arbeit untersucht, inwiefern Modelle einer Extreme Learning Machine mit modularem Aufbau des Reservoirs, bei Problemen mit modularer Struktur das Lernverhalten beeinflussen. Hierbei bezeichnet das Reservoir die versteckten Schichten der Extreme Learning Machine. Experimente mithilfe verschiedener Beispielfunktionen und auch eines Datensets zur Approximation von Schülerleistungen zeigen, dass zusätzliches Wissen über die Struktur einer Funktion eine Verbesserung erzielen kann, wenn diese Struktur auch im Reservoir-Design wiederzufinden ist. Das heißt, ist es möglich, das Problem in kleinere Teilaufgaben zu unterteilen und diese Teilaufgaben mithilfe von einzelnen Extreme Learning Machines vorherzusagen, erzielt man bessere Lernergebnisse als mit einer einzigen Extreme Learning Machine.

Inhaltsverzeichnis

Abkürzungsverzeichnis

MNN Modulares neuronales Netz

ESN Echo State Network

ELM Extreme Learning Machine

TELM Extreme Learning Machine mit zwei versteckten Schichten

MSE Mittlerer quadratischer Fehler

RMSE Wurzel des mittleren quadratischen Fehlers

STD Standardabweichung

ReLU Rectified Linear Unit

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

Neuronale Netze wurden inspiriert durch die Natur [3]. Ein scheinbar einfaches Gehirn einer Taube wiegt nur wenige Gramm, dennoch kann eine Taube sehr komplexe Aufgaben erledigen, wie beispielsweise zu fliegen und zu navigieren. Angelehnt an biologische Gehirne werden neuronale Netze ebenfalls mit Neuronen und Synapsen ausgestattet. Sie erfreuen sich immer größerer Beliebtheit in verschiedensten Bereichen wie beispielsweise Mustererkennung oder Spracherkennung, aber auch in vielen anderen Gebieten [4]. Damit ein solches Netz die gewünschte Funktion erfüllt, benötigt man eine große Menge an Trainingsdaten und einige Erfahrungswerte, um Hyperparameter auszuwählen. Eine Extreme Learning Machine stellt eine Möglichkeit dar, welche weniger Rechenaufwand beim Training benötigt als klassische neuronale Netze. Dies ist auf die Tatsache zurückzuführen, dass bei einer Extreme Learning Machine nur die Gewichte zwischen der Verborgenen- und der Ausgangsschicht trainiert werden. Die Neuronen innerhalb des Reservoirs, welches die verborgenen Schichten (Hidden Layer) bezeichnet, werden mit zufälligen Werten initialisiert. Als Folge dessen können neuronale Netze dieser Art mit weniger Trainingsdaten und Rechenaufwand ihre Funktionalität anpassen.
Um die Möglichkeiten dieser Netzwerke voll auszuschöpfen wäre es wünschenswert zu wissen, inwiefern das Design des Reservoirs eine Auswirkung auf die erzielten Ergebnisse des Netzes hat.

Die Bachelorarbeit von Herrn Liebald „Exploration of effects of different network topologies on the ESN signal crosscorrelation matrix spectrum“ [5] befasst sich mit einem sehr ähnlichen Thema. Die Experimente wurden mithilfe eines Echo State Network (ESN) durchgeführt, welches ähnlich aufgebaut ist wie eine Extreme Learning Machine. Der Unterschied der beiden Netzwerke besteht darin, dass mithilfe eines ESN rekurrente Funktionen approximiert werden können. In der Arbeit [5] wurden verschiedene Netzwerkarchitekturen eines Echo State Network (ESN) untersucht und überprüft ob ein gezieltes Reservoir-Design ein besseres Lernergebnis erzielt, als ein willkürlich initialisiertes. Herr Liebald kam zu dem Schluss, dass es tatsächlich keine Verbesserung bringt ein spezielles Reservoir-Design zu wählen. Allerdings wurden in der Bachelorarbeit die Ergebnisse an einem NARMA System und der Mackey-Glass Gleichung getestet, beide Funktionen besitzen keinen modularen Aufbau. Was unter einer modularen Funktion verstanden wird, wird im Abschnitt 1.2 erklärt.

In dieser Bachelorarbeit soll nun der Effekt des Reservoir-Designs auf Probleme mit modularer Struktur untersucht werden.

1.2 Systemansatz und Zielsetzung

Die Auswirkung der Reservoir-Architektur auf das Lernverhalten wird mithilfe einer Extreme Learning Machine (ELM) untersucht. Je mehr Kenntnisse man über ein Phänomen besitzt, welches vorhergesagt werden soll, desto genauer und zuverlässiger kann man dieses vorhersagen. Weiß man beispielsweise, dass sich eine Gewitterwolke auf einen zubewegt, kann man mit hoher Wahrscheinlichkeit davon ausgehen, dass es in nächster Zeit regnen wird. Diese Arbeit befasst sich mit der Struktur des vorherzusagenden Problems. Hat man über diese zusätzliche Kenntnisse würde es Sinn machen dieses Wissen in die Vorhersage einzubauen. Weiß man, dass es sich um eine modulare Struktur handelt, beispielsweise eine Funktion der Form $f(x, y) = g(x) + h(y)$ kann man dieses Wissen nutzen. Eine modulare Funktion beschreibt alle Funktionen, welche sich linear aus möglicherweise nichtlinearen Teilfunktionen zusammensetzen. Dabei ist zu beachten, dass die Parameter nicht überlappen, das heißt, jeweils nur in einer jeweiligen Teilfunktion vorkommen. Nutzt man nun die Kenntnisse über die modulare Struktur des Problems kann man auch die Extreme Learning Machine modular aufbauen. Modular bedeutet hierbei eine Reservoir-Architektur aus einer ELM mit dem Input x und einer weiteren ELM mit dem Input y, die einen gemeinsames Ausgangsneuron $f(x, y)$ ansprechen.
Um unterschiedliche Architekturen vergleichen zu können wird außerdem ein additives Model betrachtet, welches sich ebenfalls aus zwei separaten ELMs zusammensetzt. Die beiden Extreme Learning Machines besitzen, wie auch die modulare ELM, je einen Input x und y. Diese beiden Netze werden dann mit den Ergebnissen der jeweiligen Teilfunktion $g(x)$ und $h(y)$ trainiert und deren Ergebnisse anschließend linear zusammengesetzt um den Output $f(x, y)$ zu generieren. Um die Funktionalität der modularen und additiven ELM zu evaluieren werden sie mit einer gewöhnlichen Extreme Learning Machine mit zwei Eingangsneuronen x, y und einem Ausgangsneuron $f(x, y)$ verglichen. Am Ende werden hinsichtlich des Designs und der verwendeten Anzahl an Neuronen und Synapsen die Ergebnisse analysiert und evaluiert.

Das Ziel dieser Arbeit ist herauszufinden, ob Modelle mit modularem Reservoir Vorteile bei Problemen mit modularer Struktur bieten. Es können vorhandene Kenntnisse über die Struktur genutzt werden, um mit dem optimalen Reservoir-Design ein besseres Lernergebnis zu erzielen. Somit ist eine gezieltere Anwendung neuronaler Netze dieser Art möglich und es können im besten Fall Funktionen mit einem geringeren Fehler approximiert werden.

2 Neuronale Netze

Neuronale Netze wurden der Natur nachempfunden [3]. Der Aufbau eines künstlichen neuronalen Netzes ist jedoch bei weitem nicht so komplex und leistungsfähig, wie ein tatsächliches Gehirn eines Säugetiers oder, um auf die Einleitung zurückzugreifen, wie das einer Taube. Das liegt daran, dass Computer die Daten sequentiell verarbeiten, während Gehirne die Daten parallel auswerten. Obwohl ein Gehirn eine deutlich geringere Taktgeschwindigkeit besitzt als ein Computer, ist die Berechnung aufgrund der Parallelität trotzdem signifikant schneller.
Das Gehirn besteht außerdem aus verschiedenen modularen Systemen [6]. Das Verständnis der Modularität eines Gehirns hier, weicht jedoch von der vorher beschriebenen mathematischen Definition ab. Jede Gehirnregion bildet ein Funktionsmodul und alle Regionen sind mit anderen Teilen des Gehirns verbunden. Beispielsweise besteht die Großhirnrinde aus mehreren Regionen, denen Hauptaufgaben in Bezug auf Wahrnehmung und Kognition zugeschrieben werden. Dort tritt Modularität in zwei verschiedenen Aspekten auf, in struktureller und funktioneller Modularität. Die strukturelle Modularität ist erkennbar, durch sehr wenige Verbindungen zwischen verschiedener neuronaler Gruppen, während die Neuronen innerhalb einer Gruppe jedoch sehr dicht verknüpft sind. Die funktionelle Modularität wird sichtbar bei den unterschiedlichen Antwortmustern, die von den Modulen für verschiedene Aufgaben im Bereich der Wahrnehmung und Kognition erzielt werden. Durch die Modularität ist die Gesamtfunktion des Gehirns, auch wenn die Funktion einer Teilregion beeinträchtigt wird, trotz allem weiterhin gegeben.
Ein biologisches Gehirn besteht, wie in Abbildung 2.1 ersichtlich wird, aus Neuronen, Dendriten, Axonen und Synapsen [3]. Sowohl in einem Gehirn als auch in einem künstlichen neuronalen Netz überträgt das Neuron ein Signal [3]. Liegt ein Eingangssignal an, gibt das Neuron ein abgeändertes Signal aus. Jedoch reagieren die Neuronen nicht sofort, sondern erst wenn das Eingangssignal einen gewissen Schwellwert überschreitet. Tritt dieser Zustand ein, feuert das Neuron. Eine Funktion, welche die Ausgabe eines Neurons aus der Summe der Eingangssignale bestimmt, nennt sich Aktivierungsfunktion. Bei künstlichen neuronalen Netzen können unterschiedliche Funktionen als Aktivierungsfunktion verwendet werden, unter anderem zum Beispiel eine Stufenfunktion oder aber auch die Sigmoidfunktion. Ein neuronales Netz besteht aus sehr vielen Neuronen, die in mehreren Schichten angeordnet sind. Die Neuronen jeder Schicht sind meist mit allen Neuronen der Vorgängerschicht und der Nachfolgeschicht verbunden. Damit ein solches Netz auch lernen kann, wird jeder Verbindung ein Gewicht zugewiesen, dass das zu übertragende Signal entweder abschwächt oder verstärkt. Im Trainingsprozess werden diese Gewichte dann mit unterschiedlichen Methoden verändert und angepasst. Eine häufig verwendete

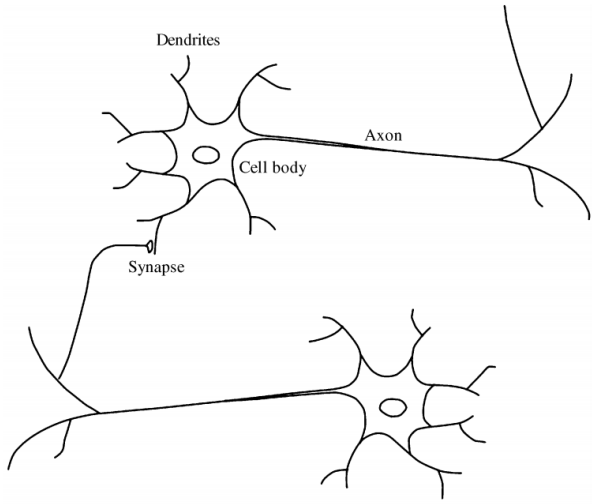


Abbildung 2.1: Schematisches Diagramm zweier biologischer Neuronen (von [1])

Methode um ein Netz zu trainieren ist der Backpropagation-Trainingsalgorithmus. Hierbei werden die Gewichte der Neuronen wieder verwendet um den Fehler anteilig des Gewichts zurück in das Netz zu führen. Der Grund dafür ist, dass kleinere Gewichte theoretisch auch einen kleineren Einfluss auf den Fehler am Ausgang haben. Besteht ein Netz nur aus zwei Layern mit wenigen Neuronen ist das zurückführen des Fehlers kein großes Problem. Hat ein neuronales Netz aber deutlich mehr als zwei Schichten und besitzt viele Neuronen, kann das Training eines Netzes viel Zeit in Anspruch nehmen, da der Rechenaufwand sehr schnell sehr groß wird. Mittels Gradientenverfahren und einer Lernrate α werden während des Trainings die Gewichte aktualisiert. Ist das Netz fertig trainiert wird mit den Testdaten die Funktion überprüft und anschließend die Genauigkeit des neuronalen Netzes ermittelt.

Die Tatsache, dass ein biologisches Gehirn einen modularen Aufbau besitzt und gleichzeitig sehr komplexe Aufgaben lösen kann, motivierte dabei diese Struktur auch bei neuronalen Netzen umzusetzen [6]. Dadurch entstand um die Jahrtausendwende die Entwicklung der modularen neuronalen Netze (MNN). Diese Art von Netzwerk kann ein komplexes Problem mithilfe mehrerer einfacher neuronaler Netze erlernen. Ein MNN erzielt meist eine bessere Generalisierung und die gesamte Komplexität eines MNN ist in der Regel geringer als die eines gewöhnlichen neuronalen Netzes, welches die selbe Aufgabe vorhersagen soll. Die Interaktion der einzelnen Komponenten eines Netzwerks kann mithilfe zweier Modelle beschrieben werden, einem eng und einem locker gekoppelten MNN. Dabei werden bei einem eng gekoppelten modularen neuronalem Netz alle Teilnetzwerke gemeinsam trainiert, so wird ihre Interaktion während einer einzelnen Lern-

phase berücksichtigt. Deshalb müssen auch die Parameter aller Netzwerke gleichzeitig aktualisiert werden, indem sie eine globale Kostenfunktion minimieren. In einem locker gekoppelten MNN hingegen werden beim Training verschiedene Stufen in hierarchischer oder sequentieller Weise durchlaufen. Wobei das Lernen während einer Stufe entweder mit anderen Netzwerken korreliert oder unkorreliert sein kann.
Laut [7] können modulare neuronale Netze auch unterschiedlich designt werden, dies erfolgt in drei Schritten. Als Erstes wird die Aufgabe in kleinere Teilaufgaben unterteilt. Jede Teilaufgabe wird einem Modul des neuronalen Netzes zugewiesen. Dadurch wird die Gesamtaufgabe, die zu erledigen ist vereinfacht. Danach werden die Module je nach Design entweder parallel oder in einer bestimmten sequentiellen Reihenfolge trainiert. Ist das komplette Netzwerk trainiert, wird mithilfe einer Entscheidungsstrategie aus den Teilentscheidungen der Module eine globale Entscheidung für das Netzwerk getroffen. Beim Design des MNN ist darauf zu achten ein Gleichgewicht zwischen der Vereinfachung der Teilaufgaben und der Effizienz der endgültigen Entscheidung des Netzwerkes zu finden. Damit die Flexibilität des Systems nicht eingeschränkt wird, sollte die Modulgröße nicht schon vorher auf die selbe Anzahl an Neuronen festgelegt werden.

3 Extreme Learning Machine

3.1 Grundlagen Extreme Learning Machine

Feedforward Neural Networks werden in vielen Bereichen für verschiedenste Aufgaben eingesetzt, da sie komplexe nichtlineare Abbildungen direkt von den Eingangssignalen approximieren und Modelle für eine große Klasse natürlicher und künstlicher Phänomene liefern können, die mit klassischen parametrischen Techniken schwer zu handhaben sind [8]. Ein Nachteil hingegen stellen die langsamen Trainingsalgorithmen dar, welche oftmals mehrere Stunden oder gar Tage für das Training des Netzes benötigen. Es wird immer noch an Methoden geforscht, welche diese Algorithmen optimieren und beschleunigen, um die Dauer der Berechnungen zu verringern.
Eine neuer Lernalgorithmus, der genau diese Dauer deutlich reduziert, wurde von Huang et al. [4] vor etwa 20 Jahren vorgestellt. Angelehnt an die Tatsache, dass ein biologisches Gehirn auch ohne äußere Einwirkung des Menschen lernt, wird vermutet, dass es Bereiche des Gehirns gibt, in denen die Neuronenkonfigurationen nicht von der äußeren Umgebung beeinflusst werden [2]. Der in [4] vorgeschlagene Lernalgorithmus basiert auf diesem Argument. Die Extreme Learning Machine (ELM) ist ein neuronales Netz mit Eingangsneuronen, versteckten Neuronen und Ausgangsneuronen. Die versteckten Neuronen können in einer oder mehreren Schichten angeordnet sein. Die Gewichte dieser Neuronen und die Eingangsgewichte sind zufällig initialisiert und werden, anders als bei gewöhnlichen neuronalen Netzen, durch das Training nicht verändert. Nur die Gewichte der Ausgangsneuronen werden während des Trainings mit Hilfe von Moore-Penrose Inversen angepasst. Somit erreichen Extreme Learning Machines eine Trainingsgeschwindigkeit die um ein Vielfaches schneller ist, als die traditioneller Feedforward-Netzwerk-Lernalgorithmen, wie beispielsweise Backpropagation, gleichzeitig erreichen sie eine bessere Generalisierung [8]. Mithilfe einer ELM werden außerdem Probleme wie lokale Minima oder eine falsch gewählte Lernrate vermieden. Ein weiterer Vorteil ist, dass der Lernalgorithmus einfacher zu implementieren ist, als der anderer Feedforward-Netzwerke.

Auch in diesem Teilgebiet der künstlichen Intelligenz lassen sich die Lernergebnisse spezieller Aufgaben, mittels modularer Modelle, optimieren. So beispielsweise auch in [9], dort wird mittels einer modular aufgebauten ELM der Flugzeugtyp bestimmt. Modulare Netze evaluieren dabei drei Merkmale, welche Aufschluss über die Form des Flugzeuges geben und bestimmen anschließend den Flugzeugtyp. Diese Merkmale werden zuvor aus einem Bild eines Flugzeuges extrahiert. Jedes dieser modularen Netzwerke besteht aus genauso vielen Klassifizierern wie Flugzeugtypen und wird mittels des Extreme Learning

Machine-Algorithmus trainiert. Bei einer ELM erfolgt die Initialisierung der Parameter der versteckten Neuronen willkürlich und unabhängig der Trainingsdaten, aus diesem Grund teilen sich alle Klassifizierer ein gemeinsames Set an Neuronen. Das Ergebnis der Klassifizierung jedes modularen Netzes wird durch Verwendung einer speziellen Cluster- und Selektionsmethode bestimmt. Die gewichteten Ausgaben der drei modularen Extreme Learning Machines bestimmen den Flugzeugtyp. Es wird eine größere Genauigkeit bei der Klassifizierung festgestellt, als mit einer einzigen ELM. Soll ein neues Merkmal hinzugefügt werden, erfolgt dies einfach durch die parallele Integration eines weiteren modularen Netzes. Dieses neue Netzwerk beeinflusst die anderen, bereits existierenden, Netzwerke nicht.
Es beschäftigt sich noch eine weitere Arbeit von Tang et al. [10] mit einer anderen Art von modularer Extreme Learning Machine. Hierbei handelt es sich um einen evolutionären Multitasking-Algorithmus, welcher unterschiedliche Architekturen gleichzeitig trainieren kann. Der Austausch des Wissens unter den Netzwerken wird durch den ähnlichen Aufbau dieser Netze erleichtert. Dadurch beschleunigt sich der Prozess des modularen Trainings, trotz unterschiedlicher Architekturen, in bestimmten Netzwerken. Die angewandte Methode erzielte qualitativ höherwertige Ergebnisse mithilfe des evolutionärem Multitasking und dem modularen Aufbau, als eine gewöhnliche ELM außerdem benötigte dieser Aufbau weniger Zeit für die Berechnungen [10].

3.2 Extreme Learning Machine mit einem Layer

Eine Extreme Learning Machine mit einer Schicht wird wie folgt aufgebaut und trainiert [8, 4]:

- K versteckten Neuronen
- N Beispiele (x_i, t_i) mit $x_i = [x_{i1}, x_{i2}, ..., x_{in}]^T \in R^n$ und $t_i = [t_{i1}, t_{i2}, ..., t_{im}]^T \in R^m$
- einer N x K Ausgabematrix der versteckten Schichten $H = \{h_{ij}\}$ mit $i = 1, ..., N$ und $j = 1, ..., K$
- einer N x m Matrix $T = [t_1, t_2, ..., t_N]^T$ mit den gewünschten Ausgabewerten
- einer K x m Gewichtsmatrix $\beta_j = [\beta_{j1}, \beta_{j2}, ..., \beta_{jm}]^T$, welcher den j-ten versteckten Knoten mit den Ausgangsneuronen verbindet
- einer n x K Gewichtsmatrix $w_j = [w_{j1}, w_{j2}, ..., w_{jn}]^T$, welcher den j-ten versteckten Knoten mit den Eingangsneuronen verbindet
- einer Aktivierungsfunktion $g(x)$
- einem Bias b_j des j-ten versteckten Neurons

Mit K Neuronen in der versteckten Schicht können $N = K$ Beispiele exakt erlernt werden. Es reichen allerdings auch weniger Neuronen, falls ein gewisser Lernfehler erlaubt ist. Da bei der Extreme Learning Machine die Gewichte und der Bias der verborgenen Schicht zufällig initialisiert werden, handelt es sich um ein lineares System:

$$H\beta = T,$$

mit $H = \{h_{ij}\}$ und $h_{ij} = g(w_j * x_i + b_j)$ als Ausgabe des j-ten versteckten Neurons und dem i-ten Datenpunkt. Mit der Moore-Penrose Inverse $H^\dagger$ können nun auf einfache Art und Weise die Ausgangsgewichte β berechnet werden:

$$\beta = H^\dagger T,$$

mit $H^\dagger = (H^T H)^{-1} H^T$. Durch diese einfache Berechnung, welche einer reinen Matrizenmultiplikation entspricht, erfolgt das Training einer Extreme Learning Machine deutlich schneller, als das Training anderer Lernalgorithmen, wie beispielsweise des Backpropagation-Algorithmus.

3.3 Extreme Learning Machine mit zwei Layern

Eine Extreme Learning Machine mit zwei versteckten Schichten (TELM) bietet den Vorteil, dass sie weniger Neuronen benötigt um eine gewünschte Funktion zu erzielen [2].
Dabei wird eine Two-Hidden-Layer Extreme Learning Machine (siehe 3.1) wie folgt trainiert und aufgebaut:

- N Beispiele mit $X = [x_1, x_2, ..., x_N]^T, T = [t_1, t_2, ..., t_N]^T$
- L Neuronen je Schicht, $2L$ Neuronen insgesamt
- einer Aktivierungsfunktion $g(x)$
- einer Gewichtsmatrix W, welche die Gewichte zwischen der Eingangsschicht und der ersten versteckten Schicht bestimmt
- einer Gewichtsmatrix W_H, welche die Gewichte zwischen der ersten und der zweiten versteckten Schicht bestimmt
- einer Gewichtsmatrix β zwischen der zweiten versteckten Schicht und der Ausgangsschicht
- einen Biasvektors B der ersten Schicht und B_1 der zweiten Schicht
- einer Ausgabematrix H der ersten Schicht
- einer erwarteten Ausgabematrix H_1 der zweiten Schicht und der tatsächlichen Ausgabematrix H_2 der zweiten Schicht

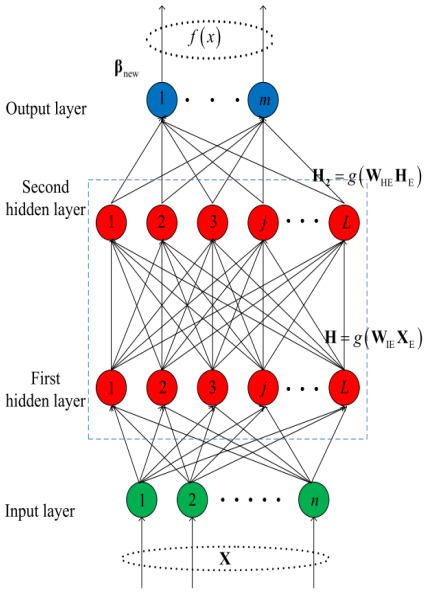


Abbildung 3.1: Beispiel einer ELM mit zwei Schichten (von [2])

Laut Qu et al. [2] ist die Vorgehensweise eine TELM zu berechnen wie folgt:
Als ersten Schritt wird eine Gewichtsmatrix W zwischen der Eingabeschicht und der ersten Schicht und der Bias B der ersten Schicht zufällig initialisiert.
Danach kann die Ausgabe dieser Schicht mit

$$H = g(W_{IE}X_E) = g(B + WX)$$

berechnet werden, wobei $W_{IE} = [B\ W]$ und $X_E = [1\ X]^T$. Anschließend wird auch die Gewichtsmatrix β zwischen der zweiten Schicht und der Ausgabeschicht mithilfe der Formel

$$\beta = H^{\dagger}T$$

berechnet. Als nächstes kann die erwartete Ausgabe der zweiten Schicht mittels

$$H_1 = T\beta^{\dagger}$$

bestimmt werden. Mit diesem Ergebnis können nun die Parameter des zweiten Layers, wie die Gewichtsmatrix zwischen dem ersten und zweiten Layer sowie dem Bias der zweiten Schicht, bestimmt werden. Die Gewichtsmatrix und der Bias werden durch $W_{HE} = [B_1\ W_H]$ festgelegt und werden berechnet mit

$$W_{HE} = g^{-1}(H_1)H_E^{\dagger}, \text{ mit } H_E = [1\ H]^T.$$

Die Funktion $g^{-1}(x)$ bezeichnet die Inverse der Aktivierungsfunkton $g(x)$. Dabei ist zu beachten, dass die Parameter von H_1 im Bereich zwischen -0.9 und 0.9 normiert werden müssen, um die Durchführbarkeit der Inversion zu garantieren. Ist eine Normierung notwendig, müssen die Parameter von H_2 am Schluss wieder entsprechend umgerechnet werden. Nun kann die tatsächliche Ausgabe der zweiten Schicht mit der Formel

$$H_2 = g(W_{HE}H_E) = g(B_1 + W_H H)$$

ermittelt werden. Als letzter Schritt wird dann die neue Gewichtsmatrix dieser Schicht berechnet mit

$$\beta_{neu} = H_2^{\dagger}T$$

und anschließend die endgültige Ausgabe der TELM

$$f(x) = H_2\beta_{neu} = g(W_{HE}H_E)\beta_{neu} = g(B_1 + W_H g(WX + B))\beta_{neu}.$$

4 Betrachtete Architekturen

4.1 Addierte Extreme Learning Machine

Es ist leicht vorstellbar, dass je mehr Kenntnisse über eine Funktion vorliegen, welche vorhergesagt werden soll, desto besser wird die Vorhersage. Deshalb sollte dies auch für die Struktur der Funktion gelten. Baut man nun auf dem Wissen auf, dass es sich um eine Funktion der Form $f(x, y) = g(x) + h(y)$ handelt, kann man je eine ELM mit der Formel $g(x)$ trainieren und eine weitere mit der Formel $h(y)$. Die beiden Ergebnisse werden linear zusammengesetzt und ergeben dann $f(x, y)$. Der Aufbau der zwei separaten Extreme Learning Machines, deren Ausgaben linear-verknüpft die gesamte Ausgabe von $f(x, y)$ ergeben, wird in dieser Arbeit als addierte ELM benannt.

Der Aufbau des Reservoirs sieht dabei wie folgt aus:

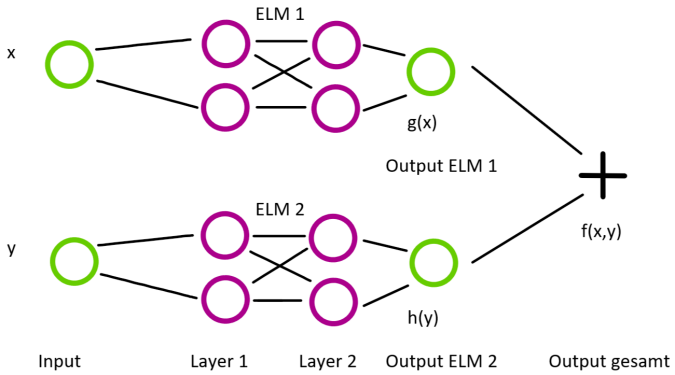


Abbildung 4.1: Beispiel addierte ELM

Die Genauigkeit dieses Aufbaus wird über den mittleren quadratischen Fehler errechnet. Dabei ist anzumerken, dass der gesamte mittlere quadratische Fehler (siehe Kapitel 5.6) sich berechnet aus der Summe des Outputs der Funktion $g(x)$ und des Outputs der Funktion $h(y)$.

Für die in Kapitel 5.3 Trainings- und Testdaten beschriebenen Formeln sind die Teilfunktionen $g(x)$ und $h(y)$ bekannt. Somit kann dieses Modell der Extreme Learning Machine berechnet und die Ergebnisse anschließend verglichen werden.

4.2 Modulare Extreme Learning Machine

Einen sehr ähnlichen Aufbau zur addierten Extreme Learning Machine besitzt die modulare ELM. Die beiden Modelle unterscheiden sich lediglich darin, dass bei der modularen ELM die Zwischenausgaben $g(x)$ und $h(y)$ nicht bekannt sind. Das heißt es ist nur der Output f(x,y) bekannt und mit diesem werden die Extreme Learning Machines trainiert.
Eine modulare ELM bezeichnet hierbei zwei separate Extreme Learning Machines, welche je ein Input Neuron besitzen. Eine ELM wird mit dem Input x und eine andere ELM wird mit dem Input y trainiert und getestet.

Der Aufbau des Reservoirs der modularen ELM sieht dabei wie folgt aus:

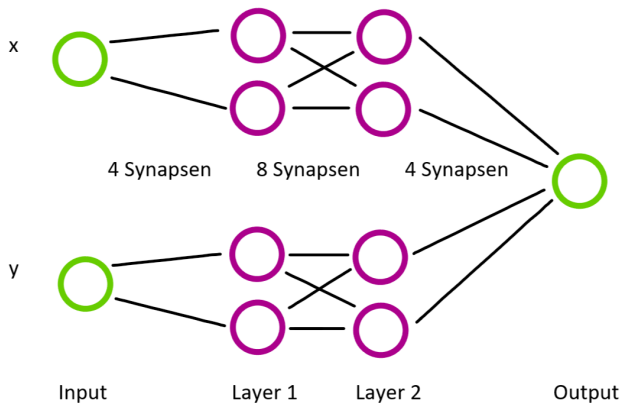


Abbildung 4.2: Beispiel modulare ELM

Dabei wird aus den Ausgaben der beiden Extreme Learning Machines ein gemeinsamer Merkmalsvektor erstellt und mit den vorgegebenen Ausgabedaten trainiert. Hierbei wird bei der Fehlerberechnung der mittlere quadratische Fehler der vorhergesagten Ausgabewerte zu den erwarteten Werten ermittelt.

4.3 Nicht modulare Extreme Learning Machine

Die nicht modulare ELM beschreibt in dieser Arbeit eine gewöhnliche Extreme Learning Machine mit zwei Inputneuronen x und y und einem Output $f(x, y)$. In diesem Aufbau wird die Struktur des Problems nicht berücksichtigt und wird genutzt um die Ergebnisse mit der modularen ELM und der addierten ELM zu vergleichen.

Eine nicht modulare ELM ist dabei wie folgt aufgebaut:

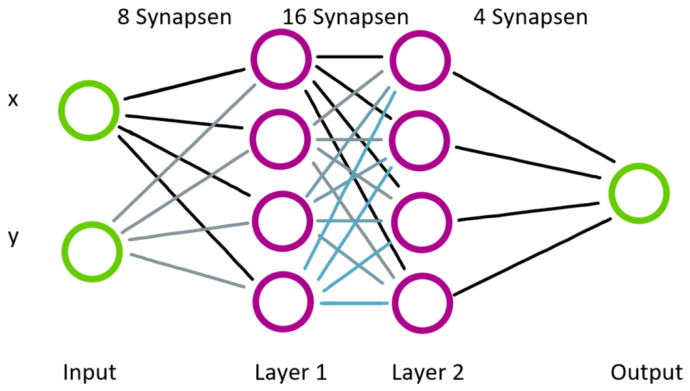


Abbildung 4.3: Beispiel nicht modulare ELM

Auch bei der nicht modularen ELM wird die Genauigkeit des Aufbaus mithilfe des mittleren quadratischen Fehlers der Ausgabewerte ermittelt.

5 Experimentdesign

Das Ziel dieser Bachelorarbeit ist es zu überprüfen ob ein Reservoir mit modularer Struktur bessere Ergebnisse bei Problemen erzielt, die eine modulare Struktur aufweisen. Dabei wird eine Formel der Form $f(x, y) = g(x) + h(y)$, als Problem mit modularer Struktur verstanden. Zur Überprüfung dieser These wurden unterschiedlich aufgebaute Extreme Learning Machines in Phython (Version 3.6) mithilfe von Tensorflow und Keras implementiert. Die verschiedenen Architekturen die umgesetzt wurden, wurden in Kapitel 4 vorgestellt. Die Implementierung der Grundstruktur einer nicht modularen ELM, mithilfe von Keras, wurde dabei von meinem Betreuer, Herrn Kissel, zur Verfügung gestellt.

5.1 Reservoir-Architektur

Der Reservoir-Aufbau spielt beim Vergleich und zum besseren Verständnis der Ergebnisse eine wichtige Rolle. Eine **nicht modulare ELM** (Abbildung 4.3) ist eine Extreme Learning Machine, welche zwei Input Neuronen, ein Reservoir mit N Neuronen in zwei Schichten und ein Output Neuron besitzt. Unter einer **modularen ELM** (Abbildung 4.2) hingegen werden zwei Extreme Learning Machines verstanden, welche jeweils einen Input, ein Reservoir mit $\frac{N}{2}$ Neuronen besitzen und einen gemeinsamen Output ansprechen. Für die **addierte Extreme Learning Machine** (Abbildung 4.1) gilt ähnliches. Jede ELM besitzt einen Input und ein Ausgabeneuron, diese Outputs werden dann addiert und ergeben das finale Ergebnis. Die gesamte Anzahl der Neuronen in beiden Systemen ist somit identisch. Allerdings, wie man in den beiden Abbildungen 4.3 und 4.2 sehen kann, besitzt die modulare ELM weniger Synapsen, als die nicht modulare ELM. Synapsen sind ein sehr wichtiger Bestandteil des neuronalen Netzes, denn durch deren Modifikation wird das Lernen eines Netzwerkes ermöglicht [11]. Somit haben sie einen nicht vernachlässigbaren Einfluss auf das Lernverhalten des Netzes. Durch ihre geringere Synapsenanzahl ist die modulare ELM benachteiligt.

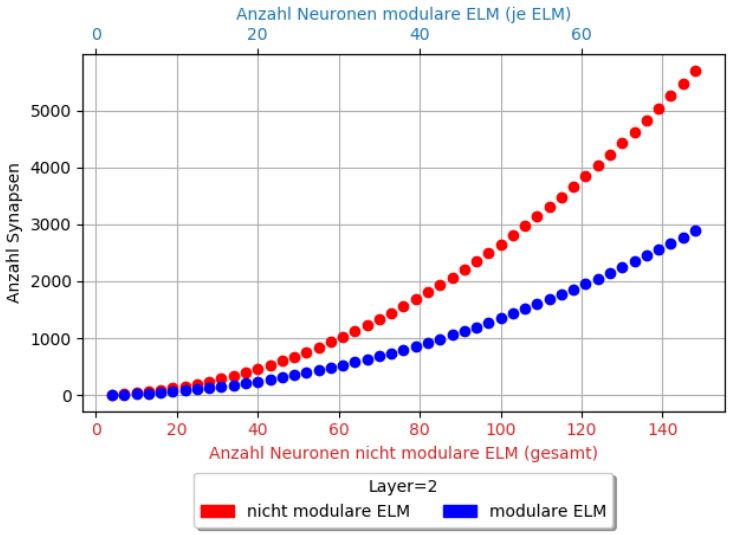


Abbildung 5.1: Anzahl an Synapsen bei steigender Neuronenzahl

Mithilfe der Abbildung 5.1 wird erkennbar, dass der Unterschied in der Anzahl der Synapsen steigt je mehr Neuronen verwendet werden. Diese Angabe ist für die Evaluierung des Fehlers der erzielten Ergebnisse wichtig.

5.2 Supervised Learning

Der Begriff Supervised Learning beschreibt ein überwachtes Training eines neuronalen Netzes [11]. Das heißt, es existieren für vorgegebene Trainings- und Testdaten bereits die gewünschten Ausgaben. So kann das Gewicht der Neuronen solange angepasst werden, bis der Fehler von gewünschter Ausgabe zu tatsächlicher Ausgabe des neuronalen Netzes minimiert wird.
In dieser Bachelorarbeit werden mithilfe der Funktionen in 5.3 die richtigen Ergebnisse für die zufällig initialisierten Trainingsdaten der Inputs x und y berechnet. Diese Werte verwendet man dann, um das Training der ELM zu überwachen und mithilfe des Fehlers zwischen den vorhergesagten Werten und den richtigen Ergebnissen die Gewichte des Netzes anzupassen. Damit nicht nur Beispielfunktionen getestet werden, wird ein realer Datensatz verwendet. Dieser wird vom UC Irvine Machine Learning Repository [12] zur Verfügung gestellt und ist ein Datensatz, welcher zur Approximation von Schülerleistung verwendet wird. Auch dieses Datenset des UCI Repository beinhaltet bereits die korrekten Endnoten aller Befragten.

5.3 Trainings- und Testdaten

Zur Überprüfung der These, dass bei Problemen der Struktur $f(x, y) = g(x) + h(y)$ eine modulare ELM bessere Lernergebnisse erzielt, als eine nicht modulare ELM werden folgende drei Beispielfunktionen verwendet:

- $f(x, y) = 5,2 * \sin(x) + \frac{1}{\exp(y)}$ (exponentiell-modulare Funktion)
- $f(x, y) = \frac{3}{2} * x + 4,17 * y$ (linear-modulare Funktion)
- $f(x, y) = 3,7 * \tan(x) - 1,2 * \cos(y)$ (trigonometrisch-modulare Funktion)

Es werden drei einfache Funktionen verwendet um zu beweisen, dass es sich bei den erzielten Ergebnissen nicht um einen Ausnahmefall handelt. Um anschließend zu zeigen wie sich die Ergebnisse für Probleme ohne modulare Struktur bei einer modularen ELM verhalten, wird als Gegenbeispiel folgende Funktion verwendet:

- $f(x, y) = 5,2 * \exp(x * y) + y^2$

Die Berechnung der Trainings- und Testdaten der beiden Inputs x und y erfolgt mithilfe von zufällig initialisierten Arrays. Das Training der Extreme Learning Machines findet anschließend mit unterschiedlichen Anzahlen an Test- und Trainingsdatenpunkten statt.

Um die These weiter zu überprüfen und ihre Aussage nicht nur an Beispielfunktionen zu testen, wird sie mithilfe eines Datensatzes des UC Irvine Machine Learning Repositorys überprüft ([12], [13]). Das Datenset besteht aus 649 Datensätzen, von denen etwa zehn Prozent als Testdaten verwendet werden und die übrigen 90 Prozent als Trainingsdaten. Der Datensatz beinhaltet 32 Merkmale, die die Schüler und ihre familiäre und schulische Situation beschreiben. Mithilfe der Merkmale wird die Endnote der Schüler, im Fach Portugiesisch, aus zwei portugiesischen Schulen vorhergesagt werden.

5.4 Aktivierungsfunktion

Eine Aktivierungsfunktion bestimmt, ob ein Neuron feuert und ein Signal übermittelt oder nicht [3]. Die Funktion bestimmt dabei den Ausgabewert des Neurons, welcher aus der Summe der Eingangssignale dieses Neurons berechnet wird.
In dieser Arbeit wird bei allen Experimenten eine Rectified Linear Unit (ReLU) als Aktivierungsfunktion der Neuronen des Reservoirs verwendet (Abbildung 5.2). Das Ausgabeneuron ist im Gegensatz dazu linear. Gegenüber der Sigmoidfunktion bietet ReLU den Vorteil, dass nicht nur Werte im Bereich $[0, 1]$ berücksichtigt werden, sondern Werte im Bereich von $[0, \infty]$.

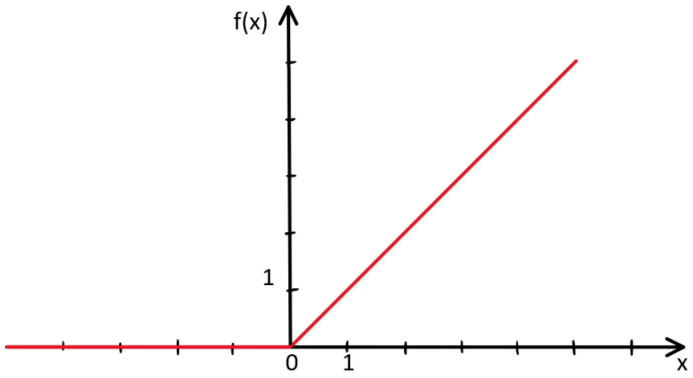


Abbildung 5.2: Rectified Linear Unit (ReLU)

5.5 Hyperparameter

Die veränderbaren Parameter in dieser Arbeit sind die Anzahl an Schichten, in der die Neuronen angeordnet sind, sowie die Anzahl an Neuronen. Wobei es sich bei der Gesamtzahl an Neuronen um alle Neuronen im Reservoir der verschiedenen Architekturen handelt. Das heißt eine gewöhnliche Extreme Learning Machine mit beispielsweise 2 Schichten und 16 Neuronen insgesamt besitzt je Layer 8 Neuronen. Das additive und modulare Modell besitzen je ELM 8 und je Schicht 4 Neuronen, was insgesamt wieder 16 Neuronen ergibt. Da die Ergebnisse für eine unterschiedliche Anzahl an Neuronen interessant für die Auswertung sind, wird mithilfe einer Schleife diese Anzahl um eine einstellbare Sprungweite erhöht. Ein weiterer Parameter der verändert werden kann ist die Anzahl an Trainings- und Testdaten, die generiert werden sollen.

5.6 Auswertung der Ergebnisse

Mehrere Durchläufe des gleichen Experiments vermeiden einen zu großen Einfluss von Ausreißern auf das Ergebnis und ermöglichen so zuverlässige Ergebnisse.

Bei jedem Durchlauf wird der mittlere quadratische Fehler (MSE) mithilfe der Formel

$$mse(y, \hat{y}) = \frac{1}{n_{Beispiele}} * \sum_{i=0}^{n_{Beispiele}-1} (y_i - \hat{y}_i)^2$$

berechnet. Dabei entspricht y den tatsächlichen Werten, $\hat{y}$ den geschätzten Werten der ELM und $n_{Beispiele}$ der Anzahl an Testdaten. Je kleiner dieser Fehler ist, desto höher ist die Genauigkeit der Extreme Learning Machine. Um nicht nur die Ergebnisse eines

einzelnen Durchlaufs zu ermitteln wird über zehn Schleifen der Mittelwert und die Standardabweichung über die Werte des MSE berechnet. Somit verringert sich der Einfluss von Ausreißern auf die Evaluierung. Der Mittelwert berechnet sich mit der Formel:

$$Mittelwert : \bar{x} = \frac{1}{n} * \sum_{i=0}^{n_{Beispiele}-1} mse_i$$

und die Standardabweichung berechnet sich mithilfe der Formel:

$$Standardabweichung : \sigma = \sqrt{\frac{1}{n} * \sum_{i=0}^{n-1} (x_i - \bar{x})^2}$$

Die Standardabweichung (STD) berechnet die Abweichung der Werte um den Mittelwert. Je größer diese ist, desto größer ist die Streuung der Werte.

6 Ergebnisse und Interpretation

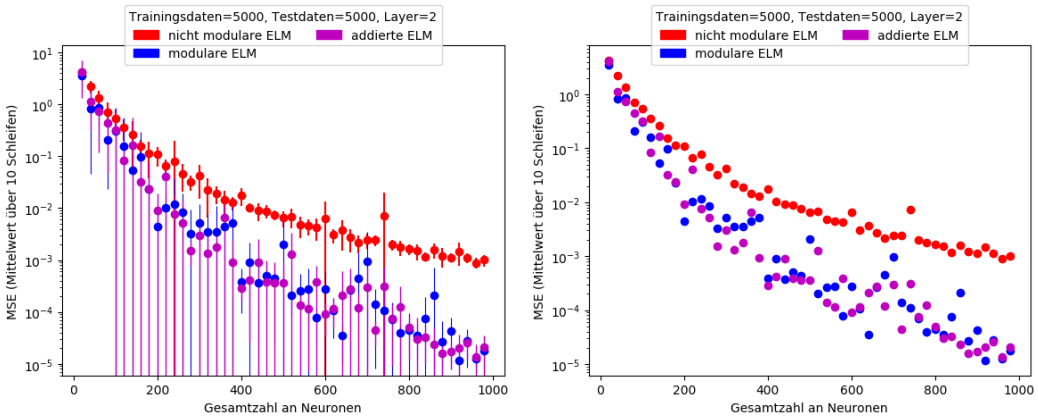
In diesem Kapitel werden die erzielten Ergebnisse dieser Arbeit dargestellt und anschließend interpretiert. Eine gleiche Gesamtzahl an Neuronen in den verschiedenen Modellen sorgt für relativ faire Bedingungen, lediglich die geringere Synapsenzahl benachteiligt die additive und modulare Architektur im Gegensatz zum nicht modularen Modell.
In den Grafiken sind die Resultate der Modelle mit unterschiedlicher Anzahl an Neuronen, Layern und Trainingsdaten veranschaulicht. Es wurden weiterhin unterschiedliche, ausgedachte Funktionen getestet und ein Datenset des UC Irvine Machine Learning Repository mithilfe der modularen und nicht modularen Extreme Learning Machine approximiert.
In den Plots sind jeweils die Performanz der modularen ELM, der addierten ELM und der nicht modularen ELM in unterschiedlichen Farben dargestellt, damit ein direkter Vergleich der Modelle möglich ist. In einigen Grafiken ist die y-Achse logarithmisch skaliert, außerdem ist jeweils die Standardabweichung mittels Fehlerbalken um den MSE angegeben.

6.1 Unterschiedliche Anzahl an Neuronen

Die Anzahl an Neuronen im System beeinflusst die Ergebnisse, da sie den Unterschied der Synapsen bestimmt (siehe Abbildung 5.1). Ein Neuron ändert Signale ab und leitet diese an andere Neuronen weiter [3]. Mithilfe der Neuronen und der damit verbundenen Gewichte lernt das neuronale Netz.
Zur Erzeugung unterschiedlicher Neuronenzahlen wird die Gesamtzahl der Neuronen mit einer einstellbaren Schrittweite erhöht. Die Anzahl an Schichten ist dabei für die folgenden Versuche auf zwei festgelegt und die Anzahl an generierten Trainings- und Testdaten auf 5.000 Datenpunkte. Die y-Achse ist logarithmisch skaliert, damit auch sehr kleine Ergebnisse besser dargestellt werden können.

6.1.1 Exponentiell-modulare Funktion

Betrachtet man die Ergebnisse der Formel $f(x, y) = 5, 2 * \sin(x) + \frac{1}{\exp(y)}$, siehe Abbildung 6.1, ist gut erkennbar, dass die nicht modulare ELM überwiegend schlechtere Ergebnisse erzielt, als die beiden anderen Modelle. Zu Beginn, mit sehr wenig Neuronen, ist der Fehler aller Modelle noch relativ groß, jedoch sinkt er, je mehr Neuronen das neuronale Netz enthält. Die Ergebnisse der addierten und modularen ELM sind großteils ähnlich präzise.



(a) MSE und STD

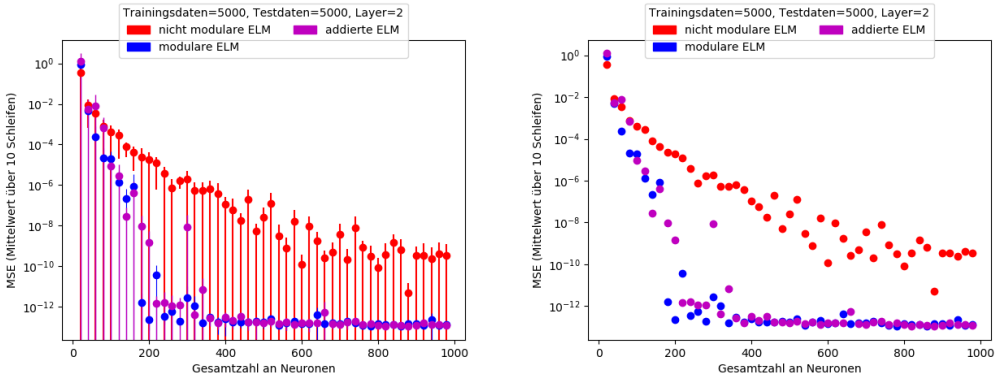
(b) MSE

Abbildung 6.1: Ergebnisse für unterschiedliche Neuronenzahlen der exponentiell-modularen Funktion

Mithilfe dieser Funktion wird die These der Arbeit gestützt, dass zusätzliche Informationen über die Struktur des Problems zu einem besseren Lernerfolg führen, wenn diese im Reservoir-Design berücksichtigt werden. Zu erwarten wäre gewesen, dass die additive ELM deutlich bessere Ergebnisse erzielt als das modulare Modell, da noch mehr Wissen über die Funktion in das Design einfließt. Dieser Widerspruch lässt sich allerdings aufheben wenn man bedenkt, dass bei diesem Modell die Fehler der beiden Extreme Learning Machines aufaddiert werden. Der Formelteil $g(x) = 5,2 * \sin(x)$ erzielt einen deutlich größeren Fehler als der Teil $h(y) = \frac{1}{\exp(y)}$, da dessen Werte sich generell nur in einem sehr kleinen Bereich bewegen. Bei der modularen ELM können sich diese beiden Fehler jedoch gegenseitig ausgleichen. Die Differenz zwischen den Fehlern der modularen und nicht modularen ELM nimmt mit steigender Neuronenzahl ab. Ein Grund dafür könnte der in Abbildung 6.4 dargestellte Unterschied in der Synapsenzahl sein, welcher mit steigender Anzahl an Neuronen größer wird. Das heißt mit sehr vielen Neuronen wird der Vorteil der nicht modularen Extreme Learning Machine ausgebaut, da sie eine viel größere Anzahl an Synapsen besitzt als das modulare Modell und somit bessere Lernerfolge erzielt.

6.1.2 Linear-modulare Funktion

Die Formel $f(x, y) = \frac{3}{2} * x + 4,17 * y$ ist ebenfalls wieder modular aufgebaut und liefert, wie auch die exponentiell-modulare Funktion 6.1.1, das Resultat, dass die modulare und additive ELM geringere Fehler erzielt als die nicht modulare ELM. Des Weiteren sind auch die Ergebnisse des additiven Modells wieder meist ähnlich gut wie die der modularen ELM.



(a) MSE und STD

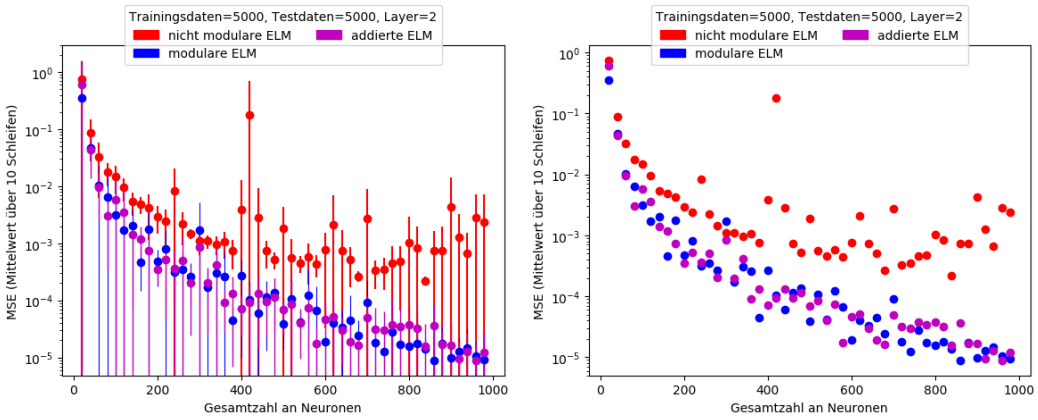
(b) MSE

Abbildung 6.2: Ergebnisse für unterschiedliche Neuronenzahlen der linear-modularen Funktion

Die deutlich geringeren Fehlerzahlen liegen vermutlich an der sehr einfachen, linearen Funktion, welche gut vorhersagbar ist. Für die Performance der additiven ELM gilt wie in 6.1.1 beschrieben, das Gleiche. Der Teil $h(y) = 4,17 * y$ produziert einen mehr als doppelt so großen Fehler, als der Teil $g(x) = \frac{3}{2} * x$, welche sich nicht ausgleichen können. Die Abstände zwischen den Fehlern der modularen und nicht modularen ELM nehmen, wie in 6.1.1 bereits beschrieben, mit steigender Neuronenzahl ab, was am Unterschied der Synapsenzahlen liegen könnte.

6.1.3 Trigonometrisch-modulare Funktion

Eine weitere modular aufgebaute Formel ist $f(x, y) = 3,7 * \tan(x) - 1,2 * \cos(y)$. Auch hier tritt das erwartete Ergebnis ein. Die modulare und additive ELM erzielen wieder einen deutlich geringeren Fehler, als die nicht modulare ELM. Wie auch schon bei den vorangegangen Funktionen diskutiert, erzielt das additive Modell ähnliche Fehler, als das modulare.
Anzumerken ist, dass der MSE der nicht modularen ELM bei etwa 700 Neuronen leicht ansteigt, während der Fehler des modularen und additiven Modells weiter abnimmt (siehe Abbildung 6.3). Dieses Phänomen genauer zu untersuchen liegt allerdings außerhalb des Rahmens dieser Arbeit. Deshalb wird im Folgenden angenommen, dass es sich hierbei um Rauschen in den Ergebnissen handelt.



(a) MSE und STD

(b) MSE

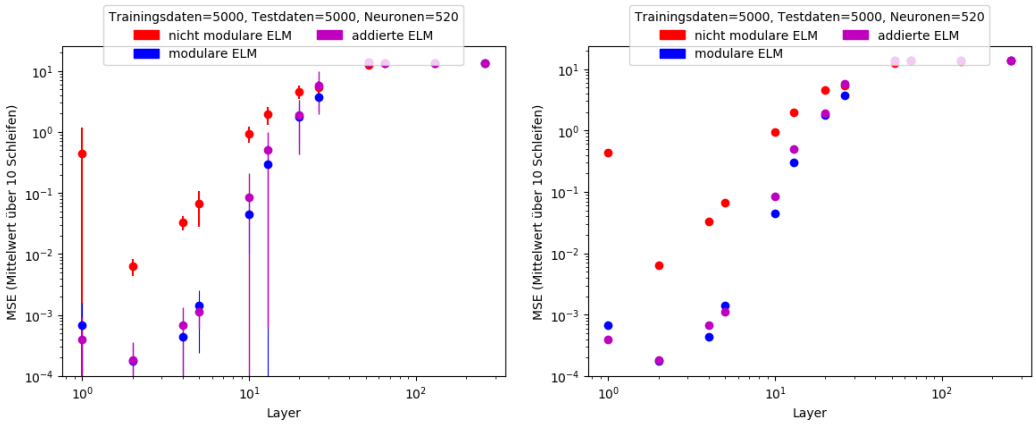
Abbildung 6.3: Ergebnisse für unterschiedliche Neuronenzahlen der trigonometrisch-modularen Funktion

6.2 Unterschiedliche Anzahl an Layern

Die Anzahl an versteckten Schichten wird in folgendem Abschnitt verändert, um die These weiter zu stützen. Die erzielten Ergebnisse der exponentiell-modularen Funktion in 5.3 mit je 5.000 Trainings- und Testdatenpunkten und insgesamt 520 Neuronen sind in den folgenden Grafiken visualisiert.
Der MSE wurde für die folgenden Anzahlen an Schichten berechnet:

$$Layer \in [1, 2, 4, 5, 10, 13, 20, 26, 52, 65, 130, 260]$$

In der Grafik 6.4 sind aufgrund besserer Übersichtlichkeit beide Achsen logarithmisch skaliert. Der Fehler aller Modelle erscheint bei zwei Schichten am besten, deshalb wurde für alle weiteren Versuche zwei Layer als Eingabeparameter verwendet. Steigt die Anzahl an Schichten so verschlechtert sich auch die Performanz aller Extreme Learning Machines. Dies kann man zum einen wieder auf die Anzahl an Synapsen zurückführen. Befindet sich beispielsweise in jedem Layer nur ein Neuron ergeben sich bei einer gewöhnlichen Extreme Learning Machine mit 520 Neuronen nur 522 Synapsen, sind diese jedoch in zwei Layern angeordnet befinden sich in jeder Schicht 260 Neuronen und es ergeben sich 68.380 Synapsen. Dieser Unterschied ist enorm und erklärt den schlechten Lernerfolg mit sehr vielen Schichten. Mit nur einem Layer ergeben sich nur 1.560 Synapsen und somit ist nachvollziehbar, dass auch diese ELMs schlechter performen. Es fließen aber natürlich auch andere Effekte in das Ergebnis ein. Beispielsweise nimmt mit zunehmender Anzahl an Schichten auch die Nichtlinearität des Systems zu.



(a) MSE und STD

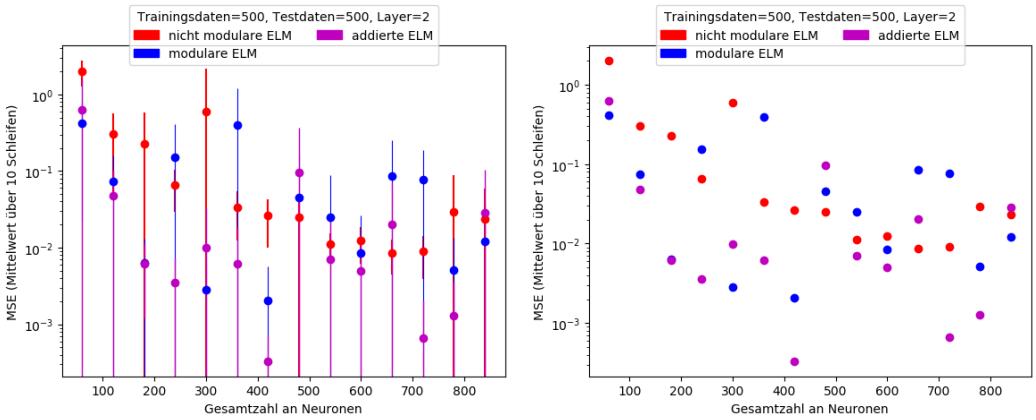
(b) MSE

Abbildung 6.4: Ergebnisse der exponentiell-modularen Funktion mit unterschiedlichen Anzahlen an Schichten

6.3 Unterschiedliche Anzahl an Trainingsdaten

Interessant erscheint auch die Auswirkung der Trainings- und Testdaten auf den Lernerfolg. Deshalb zeigen die Plots in folgendem Abschnitt die Auswirkungen für drei unterschiedliche Trainings- und Testdatenzahlen. Dabei wird wieder die exponentiell-modulare Funktion in 5.3 verwendet. Außerdem erfolgt die graphische Darstellung über unterschiedliche Neuronenzahlen für zwei versteckte Schichten.

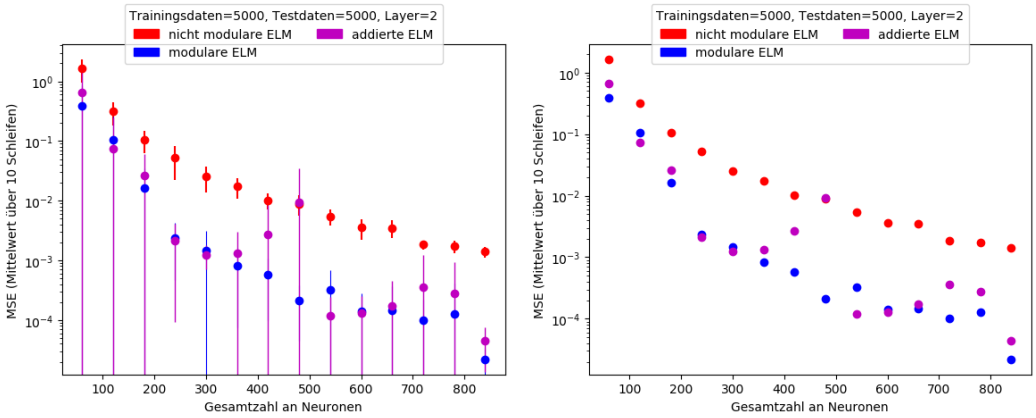
Zu Beginn werden 500 Datensätze für Training und Test verwendet (Abbildung 6.5). Das Ergebnis ist hierbei nicht eindeutig. Es ist nicht erkennbar, dass ein Modell qualitativ höherwertige Ergebnisse erzielt, als die anderen. Dies liegt vermutlich daran, dass die Zahl der Trainingsbeispiele zu gering gewählt ist. Die durch die geringere Synapsenzahl schon benachteiligte modulare ELM, lässt so keine Tendenz erkennen, niedrigere Fehler zu erzielen als eine gewöhnliche ELM.



(a) MSE und STD (b) MSE

Abbildung 6.5: Ergebnisse der exponentiell-modularen Funktion mit 500 Trainings- und Testdatenpunkten

Bei 5.000 Trainings- und Testdatensätzen ergeben sich die Grafiken in 6.6.

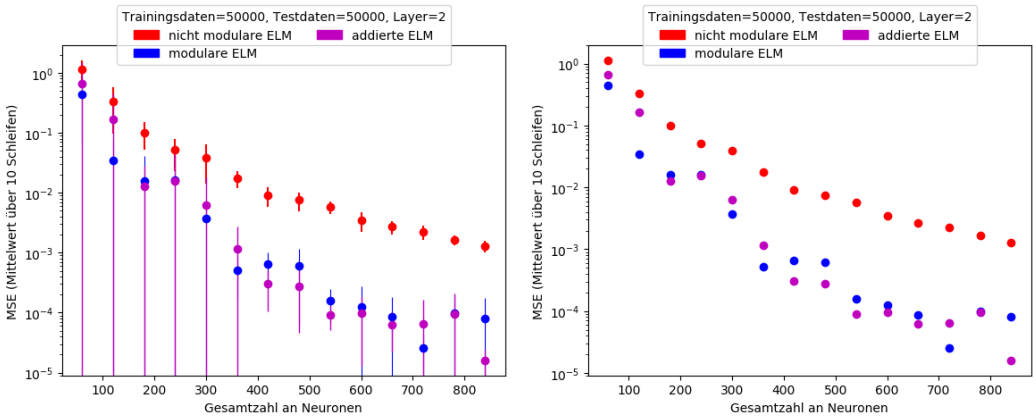


(a) MSE und STD (b) MSE

Abbildung 6.6: Ergebnisse der exponentiell-modularen Funktion mit 5.000 Trainings- und Testdatenpunkten

Mit dieser Anzahl an Datenpunkten erkennt man die Vorteile der modularen ELM, welche durchgehend deutlich geringere Fehler erzielt, als die nicht modulare ELM. Da die exponentiell-modulare Funktion mit 5.000 Test- und Trainingsdatensätzen bereits ausführlich in 6.1.1 diskutiert wurde, wird hier nicht genauer darauf eingegangen.
Betrachtet man die Grafik mit 50.000 Trainings- und Testbeispielen (siehe Abbildung 6.7)

stellt man fest, dass die Modelle ähnlich gute Ergebnisse erzielen, als mit 5.000 Datensätzen.



(a) MSE und STD

(b) MSE

Abbildung 6.7: Ergebnisse der exponentiell-modularen Funktion mit 50.000 Trainings- und Testdatenpunkten

Der mittlere quadratische Fehler bleibt bei den Modellen etwa gleich, es sind keine großartigen Verbesserungen von 5.000 Datenpunkten (Abbildung 6.6) zu 50.000 (Abbildung 6.7) erkennbar. Dies lässt darauf schließen, dass die 45.000 zusätzlichen Trainingsdaten zu keinem deutlich besseren Lernerfolg führen.

6.4 Gegenbeispiel

Das Gegenbeispiel soll zeigen, dass ein modularer Aufbau nicht immer von Vorteil ist. Hat man keinerlei Kenntnisse über die Struktur des Problems kann es sein, dass der Aufbau der modularen ELM schlechtere Ergebnisse erzielt. Dies wird mit Hilfe der Formel $f(x, y) = 5,2 * \exp(x * y) + y^2$ gezeigt. In Abbildung 6.8 wird deutlich, dass eine gewöhnliche Extreme Learning Machine einen viel kleineren MSE erzielt, als die modulare oder addierte ELM. Die ähnlichen Ergebnisse des addierten und des modularen Modells liegen vermutlich an ihrem sehr ähnlichen Aufbau. Diese Architektur wirkt sich in diesem Beispiel sehr negativ auf das Ergebnis aus, da der erste Teil der Funktion $g(x, y) = 5,2 * \exp(x * y)$ jetzt von x und y abhängt und nicht mehr nur von x. Dies stellt ein Problem dar, da als Input in diese separate ELM nur die Daten für x gegeben werden und keine Daten für y. Auch mit zunehmender Neuronenzahl ändert sich an dieser Tatsache nichts. Deshalb ist auch keine Verbesserung der Performanz, der modularen oder additiven ELM mit mehr Neuronen erkennbar.

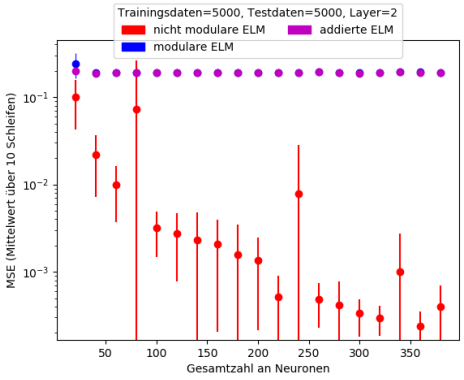


Abbildung 6.8: Gegenbeispiel mit unterschiedlicher Anzahl an Neuronen

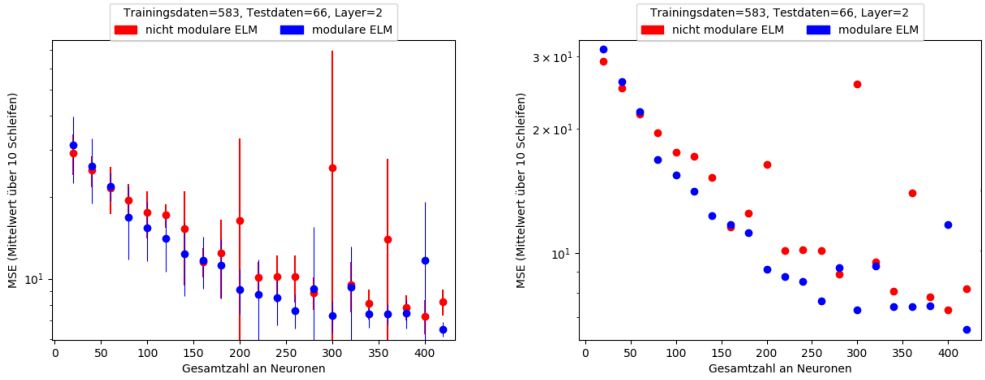
6.5 Approximierte Schülerleistung

Um einen Ausblick zugeben, welche tatsächlichen Anwendungen die These neben der ausgedachten Funktionen bietet, wurde sie an einem realen Datenset getestet.
Das Datenset stammt von UCI, einem Machine Learning Repository [12], dabei handelt es sich um die Leistung von Schülern an zwei portugiesischen Schulen. Mit Hilfe verschiedener Features wird die Endnote der Schüler im Fach Portugiesisch berechnet, die zwischen null und zwanzig Punkten liegt. Es handelt sich um 649 Datenpunkte. Mit 583 Datenpunkten werden die Modelle trainiert und anschließend mit 66 Testdaten die Genauigkeit der Architekturen ermittelt. Der Datensatz besteht aus 32 Merkmalen und dem Output. Um zu zeigen, dass eine modulare ELM bessere Ergebnisse aufweist, sollte eine modulare Struktur dem Problem zugrunde liegen. Im Fall dieses Datensets sind einige der Features in schulische und familiäre Aspekte unterteilbar. Jedoch ist die zugrunde liegende Funktion nicht bekannt und eine Modularität wird nur vermutet. Die für Training und Test verwendeten Merkmale sind je zwölf schulische und familiäre Aspekte.

- Schulische Aspekte
  - Schule an der der Schüler unterrichtet wurde
  - Grund für die Entscheidung der Schule
  - Wöchentliche Lernzeit
  - Anzahl an in der Vergangenheit durchgefallener Kurse
  - Schulische Bildungsförderung
  - Zusätzlich bezahlte Kurse im Fach Portugiesisch
  - Außerschulische Aktivitäten
  - Besuch eines Kindergartens

  - Studium wird angestrebt
  - Anzahl an Fehltagen
  - Note des ersten Drittels des Schuljahres
  - Note des zweiten Drittels des Schuljahres
- Familiäre Aspekte
  - Geschlecht des Schülers
  - Alter des Schülers
  - Wohngegend der Familie
  - Familiengröße
  - Beziehungsstatus der Eltern
  - schulische Bildung der Mutter
  - schulische Bildung des Vaters
  - Beruf der Mutter
  - Beruf des Vaters
  - Familiäre Bildungsförderung
  - Zuhause mit Internetanschluss
  - Qualität familiärer Beziehungen

Die unterschiedlichen Aspekte des Datensets sind sowohl numerisch als auch mit Zeichenfolge beschrieben. Die Vorbereitung der Daten für das Training erfolgt mittels Umwandlung der Zeichenfolgen in numerische Zahlen und einer anschließenden Skalierung zwischen null und eins. Das Training und der anschließende Test des modularen und nicht modularen Modells erfolgt mithilfe dieser beiden Inputs.
Es ist deutlich erkennbar, dass auch in Abbildung 6.9 die modulare ELM ab etwa 80 Neuronen großteils besser abschneidet als die nicht modulare ELM.
Im Artikel „Using data mining to predict secondary school student performance“ [13] wurden unterschiedliche Data-Mining Modelle mit allen Features trainiert und das beste Modell, ein Random Forest, erzielte einen RMSE (Wurzel des mittleren quadratischen Fehlers) von $1,32$. Das Ziel dieser Arbeit war es die Noten möglichst genau vorherzusagen mit den unterschiedlichen Modellen. Vergleicht man nun die Ergebnisse der modularen ELM mit denen in [13] wird ersichtlich, dass der Fehler größer ist, als der des Random Forest Modells. Der niedrigste MSE wurde mit 420 Neuronen erzielt und liegt bei $6,51$ was einem RMSE von $2,55$ entspricht. Das schlechtere Ergebnis kann auf mehrere Ursachen zurückgeführt werden, zum Beispiel wurden für das Training der ELMs nicht alle Merkmale verwendet, sondern nur 24 speziell ausgewählte. Ebenfalls lag das Hauptaugenmerk in dieser Arbeit nicht auf der absoluten Fehlerminimierung, sondern hauptsächlich darin zu



(a) MSE und STD (b) MSE

Abbildung 6.9: Ergebnisse der approximierten Schülerleistung

zeigen, dass eine modulare ELM auch bei realen Datensets, bei denen man die zugrunde liegende Funktion nicht genau kennt aber eine modulare Struktur vermutet, bessere Ergebnisse erzielt als die nicht modulare ELM. Das heißt die absoluten Ergebnisse haben keine wichtige Rolle in dieser Arbeit gespielt, sondern der Schwerpunkt lag auf dem relativen Vergleich der beiden Methoden.

7 Resümee

Betrachtet man nun alle im Kapitel Ergebnisse und Interpretation (Kapitel 6) evaluierten Grafiken erkennt man, dass zusätzliche Informationen über die Struktur eines Problems, das Lernergebnis positiv beeinflussen können. Für die Übergruppe der neuronalen Netze wird diese Tatsache bereits in [6] erwähnt (siehe Kapitel 2). Der Forschungsbereich der modularen neuronalen Netze sagt aus, dass ein Aufteilen der Aufgabe in kleinere und einfachere Teilaufgaben, die von unterschiedlichen neuronalen Netzen gelöst werden können, einen geringeren Fehler erzielen, als ein einziges neuronales Netz [7]. Da eine Extreme Learning Machine eine Unterart eines neuronalen Netzes ist, sollten auch dort die in [6] und [7] ermittelten Vorteile des modularen Aufbaus eines neuronalen Netzes gelten. Dies geht auch eindeutig aus den Ergebnissen dieser Bachelorarbeit hervor. Die Gesamtaufgabe des Netzwerkes war die Berechnung einer Funktion der Form $f(x, y) = g(x) + h(y)$. Diese Aufgabe wurde, wie bei MNN vorgeschlagen, in zwei einfachere Teilaufgaben aufgeteilt, nämlich der Berechnung von $g(x)$ und $h(y)$. Das Training der beiden Module erfolgt dann durch zwei, mit gleichen Daten initialisierten ELMs, deren Vorhersagen bezüglich der Trainingsdaten für x und y kombiniert werden und mit Unterstützung des vorgegebenen Outputs $f(x, y)$ die Gewichte angepasst werden. Durch diese Aufteilung mithilfe der modularen ELM können die Resultate der Teilaufgaben gemeinsam das Gesamtergebnis $f(x, y)$ bestimmen. Die Ergebnisse aus Kapitel 6 bestätigen die These, dass Modelle mit modularem Reservoir Vorteile bei Problemen mit modularer Struktur bieten.

Im Kapitel 3 werden bereits zwei Arbeiten vorgestellt, welche eine modulare Struktur einer ELM nutzen und durch die Unterteilung in einfachere Teilaufgaben bessere Ergebnisse erzielen. Allerdings gibt es im Teilgebiet des Extreme Learning Algorithmus noch nicht viele Arbeiten, die von der Modularität eines Problems profitieren. Will man mithilfe dieses Algorithmus ein neues Problem vorhersagen, sollte man sich zuerst Gedanken über die zugrunde liegende Struktur machen und überlegen, ob diese Aufgabe durch unterteilen in Teilaufgaben vereinfacht werden kann. Die Modularität könnte bei Klassifizierungsaufgaben, der Annäherung an Funktionen und bei Filtern helfen, bessere Ergebnisse zu erzielen [7].

Limitiert man ein Modell bereits im Vorfeld auf die selbe Anzahl an Neuronen, schränkt man dieses in ihrer Flexibilität ein [6]. In dieser Arbeit wurden die beiden ELMs mit modularen Aufbau, mit der gleichen Anzahl an Neuronen initialisiert, um möglichst faire Bedingungen zu schaffen. Man könnte deshalb untersuchen, ob eine andere Aufteilung

der Neuronen bei den unterschiedlichen Aufgaben zu einem noch besseren Ergebnis führt. Interessant wäre auch der Unterschied in den Resultaten, wenn komplett faire Bedingungen zwischen den einzelnen Systemen geschaffen werden. In dieser Arbeit ist das modulare System aufgrund der insgesamt geringeren Anzahl an Synapsen benachteiligt. Betrachtet man nun die in der Einleitung erwähnte Bachelorarbeit von Herrn Liebald [5], wäre es interessant zu wissen, ob die erneute Beurteilung der Architekturen bei einem modularen Problem die gleichen Ergebnisse liefert oder ob die unterschiedlichen Designs möglicherweise doch einen positiven Einfluss auf ein Problem mit modularer Struktur haben. Da ein Echo State Netzwerk sehr ähnlich wie eine Extreme Learning Machine aufgebaut ist, nur mit zusätzlichen rekurrenten Verbindungen, wäre es legitim anzunehmen, dass dies eine Verbesserung erzielt.

Literaturverzeichnis

1. D. Fogel, D. Liu und J. M Keller. *Fundamentals of Computational Intelligence: Neural Networks, Fuzzy Systems, and Evolutionary Computation*, Kapitel Introduction and Single-Layer Neural Networks, S. 7–33. Wiley, Hoboken, 2016. ISBN 9781119214342.

2. B. Qu, B. Lang, J. Liang, A. Qin und O. Crisalle. Two-hidden-layer extreme learning machine for regression and classification. In *Neurocomputing*, 175, S. 826 – 834, 2016. doi:https://doi.org/10.1016/j.neucom.2015.11.009.

3. T. Rashid und F. Langenau. *Neuronale Netze selbst programmieren: Ein verständlicher Einstieg mit Python*, Kapitel Neuronen - Die Rechenmaschinen der Natur, S. 30 – 39. Dpunkt.Verlag GmbH, Heidelberg, 2017. ISBN 9783960090434.

4. Q.Y. Zhu, A. Qin, P. Suganthan und G.B. Huang. Evolutionary extreme learning machine. In *Pattern Recognition*, 38(10), S. 1759 – 1763, 2005. doi:https://doi.org/10.1016/j.patcog.2005.03.028.

5. B. Liebald. *Exploration of effects of different network topologies on the ESN signal crosscorrelation matrix spectrum*. Bachelorarbeit, International University Bremen, 2004.

6. K. Chen. *Springer Handbook of Computational Intelligence*, Kapitel Deep and Modular Neural Networks, S. 473–494. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015. ISBN 978-3-662-43504-5.

7. G. Auda und M.S. Kamel. Modular neural networks a survey. In *International journal of neural systems*, 9, S. 129 – 151, 1999. doi:10.1142/S0129065799000125.

8. G. Huang, Q. Zhu und C.K. Siew. Extreme learning machine: Theory and applications. In *Neurocomputing*, 70(1-3), S. 489 – 501, 2006. doi:10.1016/j.neucom.2005.12.126.

9. H.J. Rong, Y.X. Jia und G.S. Zhao. Aircraft recognition using modular extreme learning machine. In *Neurocomputing*, 128, S. 166 – 174, 2013. doi:10.1016/j.neucom.2012.12.064.

10. Zedong Tang, Maoguo Gong und Mingyang Zhang. Evolutionary multi-task learning for modular extremal learning machine. In *IEEE Congress on Evolutionary Computation (CEC)*, S. 474 – 479. 2017. doi:10.1109/CEC.2017.7969349.

11. K. Mainzer. *Gehirn, Computer, Komplexität*, Kapitel Komplexität neuronaler Netze, S. 143–161. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997. ISBN 978-3-540-61598-9.

12. D. Dua und C. Graff. UCI machine learning repository. 2019. URL `http://archive.ics.uci.edu/ml`.

13. P. Cortez und A. Silva. Using data mining to predict secondary school student performance. In *A. Brito and J. Teixeira Eds.*, Ostend, S. 5 – 12. EUROSIS, 2008. ISBN 978-9077381-39-7.