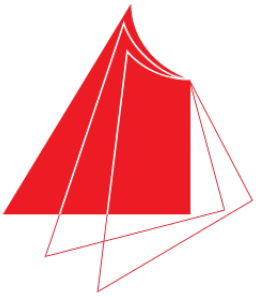


Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Genetische Algorithmen zur Optimierung von Hyperparametern eines künstlichen neuronalen Netzes

Fakultät für Maschinenbau und Mechatronik
der Hochschule Karlsruhe
Technik und Wirtschaft

Bachelorarbeit

vom 01.03.2018 bis zum 31.08.2018
vorgelegt von

Christian Heinzmann
geboren am 18.02.1995 in Heilbronn
Martrikelnummer: 52550

Winter Semester 2019

Professor	Prof. Dr.-Ing. habil. Burghart
Co-Professor	Prof. Dr.-Ing. Olawsky
Betreuer FZI:	M. Sc. Kohout

Eigenständigkeitserklärung und Hinweis auf verwendete Hilfsmittel

Eigenständigkeitserklärung und Hinweis auf verwendete Hilfsmittel. Hiermit bestätige ich, dass ich den vorliegenden Praxissemesterbericht selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen des Berichts, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht.

Datum: _____ Unterschrift: _____

Ausschreibung

BACHELORARBEIT

Genetische Algorithmen zur Optimierung von Hyperparametern eines künstlichen neuronalen Netzes

Zur Vermeidung der weiteren Ausbreitung von multiresistenten Keimen in Einrichtungen des Gesundheitswesens (insb. Krankenhäuser) werden am FZI Systeme und Methoden entwickelt, welche helfen sollen die Händehygiene-Compliance von Mitarbeitern dort zu erhöhen. Durch technische Unterstützung bei häufig wiederkehrenden Maßnahmen, soll mehr Zeit geschaffen und gleichzeitig auf die Wichtigkeit von Desinfektionsmaßnahmen aufmerksam gemacht werden. Dies soll mit Hilfe von Augmented-Reality umgesetzt werden. Dabei ist das Ziel, bekannte Prozesse, wie beispielsweise den Wechsel von postoperativen Wundverbänden, visuell zu unterstützen und automatisch zu dokumentieren.

AUFGABEN

Im Kontext der Detektion von Aktionen und Objekten, soll die Optimierung von Neuronalen Netzen mit Genetischen Algorithmen durchgeführt werden. Das Ziel hierbei ist es, ein Framework zu entwickeln und zu implementieren, in welchem künstliche neuronale Netze automatisiert trainiert werden. Unter anderem sollen die Hyperparameter mit Hilfe von Genetischen Algorithmen inteligent angepasst und das trainierte Netz anschließend ausgewertet werden. Diese Aufgaben sollen voll automatisiert ablaufen. Daraus ergeben sich folgende Aufgaben:

- Literaturrecherche über aktuelle Genetische Algorithmen und aktuelle Neuronale Netze
- Einarbeiten in vorhandene Frameworks für Genetische Algorithmen und Neuronale Netze
- Konzeptionierung und Implementierung des ausgewählten Ansatzes zur Optimierung von Hyperparameter des Neuronalen Netzes
- Evaluation und Auswertung speziell unter der Beachtung geringer Datenmengen
- Wissenschaftliche Aufbereitung und Dokumentation des Projekts

WIR BIETEN

- Aktuelle Softwaretools im täglichen wissenschaftlichen Einsatz
- eine angenehme Arbeitsatmosphäre
- konstruktive Zusammenarbeit

WIR ERWARTEN

- Grundkenntnisse in maschinellem Lernen
- Kenntnisse in folgenden Bereichen sind von Vorteil: Python, Tensorflow, C/C++
- selbständiges Denken und Arbeiten
- sehr gute Deutsch- oder Englischkenntnisse
- Motivation und Engagement

ERFORDERLICHE UNTERLAGEN

Wir freuen uns auf Ihre PDF-Bewerbung an Herrn Lukas Kohout, kohout@fzi.de, mit folgenden Unterlagen:

- aktueller Notenauszug
- tabellarischer Lebenslauf

WEITERE INFORMATIONEN

- Start: ab sofort

FZI Forschungszentrum Informatik | Embedded Systems and Sensors Engineering (ESS)
Lukas Kohout | kohout@fzi.de | Tel. +49 721 9654 - 167
Mehr Informationen unter www.fzi.de/karriere

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

1 Einleitung

1.1 Motivation

Künstliche Neuronale Netze dominieren das Feld des Maschinellen Lernens, aber ihr training und ihr erfolg hängt immer noch von sensibel empiriisch ausgesuchten Hyperparametern. Zu diesen Hyperparametern können Modelachitekur, Verlustfunktion und optimierungs Algorithmen. [2]

Momentan werden diese meist nach groben ermessen des Entwicklers ausgewählt. Das Auswählen und Testen beansprucht sehr viel Zeit und Mühe. Es gibt ansätze wie Random Search und Grid Search welche

1.2 Aufgabenstellung

Ziel der Arbeit ist es, zunächst die Optimierung von Hyperparametern zu vereinfachen. Dazu ist eine automatiesierter Trainings- und Auswerte-vorgang nötig. Anschließend sollen die Hyperparameter mit Hilfe von Genetischen Algorithmen noch verbessert werden, um schneller bessere Ergebnisse zu erhalten. Diese Ergebnisse sollen dann einer klassichen Grid Search gegenübergestellt werden.

Um dies zu vereinfachen soll ein Konzept geschaffen werden, welches die Vorgänge automatisiert und Optimiert. Dabei geht es hauptsächlich um den Vorgang der Auswahl von Hyperparameter und die Auswahl der Dimension eines Künstlichen Neuronalen Netzes. Diese berechneten Werte sollen gespeichert und anschließend übersichtlich anzeigt werden, wodurch sich die idealen Parameter herausbilden. Diese Ergebnisse sollen dem momentanen Ansatz gegenübergestellt werden.

(Mit diesem Ansatz kann die Dimensionierung eines Netzes einfacher umgesetzt werden.) Ein weiter Anwendungsfall ist die Hyperparameterauswahl. Mit Hilfe dieses Werkzeugs soll eine einfachere und bessere Auswahl der Hyperparameter erfolgen. Diese berechneten Werte sollen gespeicher und anschließend übersichtlich und intuitiv anzeigt werden, wodurch sich die idealen Parameter herausbilden. Mit diesem Ansatz soll die Richtigkeit des Netzes erhöht werden, sodass es bessere Ergebnisse liefesert. Dieses Werkzeug soll Konzeptioniert und Implementiert werden. Anschließend soll eine Evaluation und Auswertung über die mögliche Verbessung durchgeführt werden.

1.3 Aufbau der Arbeit

Zunächst wird im zweiten Kaptiel auf die verwendeten Grundlagen eingegangen. Zunächst wird im zweiten Kaptiel auf die Grundlagen zu Genetischen Algorithmen und Künstlichen Neuronalen Netzen eingegangen.

Welche Algorythmen bei dieser Arbeit verwendet werden. Und mit welchen Künstlichen Neuronalen Netzen diese Optimierungs Algorithmen getestet werde. Außderdem wird in Abschnitt 3 auf den Momentanen Stand der Technik und Forschung eingegangen dort werden auch einige Anwendungbeispiele der Genetischen Algorithmen genannt. Nun folgt in Kapitel 4 die ausarbeitung des Konzeptes mit erklärungen der einzelnen Ideen. Darauf aufbauend kommt Implementierung in Kapitel 5 in welcher mit psyodocode erklärt wird wie die Arbeit umgesetzt wurde. Anschließend wird das Implementierte system Evaluiert und getestet. Zum Schluss in Kapitel 7 gibt es eine Zusammenfassung

2 Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen, welche zum Verständnis der vorliegende Arbeit wichtig sind, beschrieben. Zu Beginn erfolgt ein kurzer Einstieg in die Optimierungsgrundlagen. Dann folgt die Einführung in die Grundlagen der Genetischen Algorithmen. Anschließend werden die wichtigen Grundlagen der Künstlichen Neuronalen Netzen erklärt. Zum Schluss wird kurz auf die Hyperparameter und ihre Besonderheiten eingegangen.

2.1 Optimierungsgrundlagen

Angenommen es soll ein Künstliches Neuronales Netz mit k Layern und L Neuronen zur Klassifizierung von einfachen handgeschriebenen Zahlen erstellt werden. Der Entwickler entscheidet sich für ein 3 Layern Netz mit jeweils 3 Neuronen. Nach dem Training hat es die Genauigkeit von 85 Prozent. Nun kann man nicht sicher sagen, ob für k = 3 und l = 3 die optimale Lösung gefunden wurde. Um dies beurteilen zu können, müssen viele Experimente durchgeführt werden. Die Frage ist, wie kann man die besten Werte für k und j finden, um die Klassifizierung zu maximieren? Dieser Vorgang, speziell im Zusammenhang mit Künstlichen Neuronalen Netzen, wird als Hyperparameter-Optimierung bezeichnet. Bei der Optimierung wird mit einem Initialwert gestartet, dieser ist in den seltensten Fällen die exakte Lösung. Dieser Initialwert muss einige Male verändert werden, um auf ein Optimum zu kommen. Manchmal ist dieses Optimieren so komplex, dass es durch eine Funktion ersetzt werden muss. In diese Arbeit ist dafür der Genetische Algorithmus zuständig.

Ractical Computer Cision Apllica ss.130

2.2 Genetische Algorithmen

Die Inhalte des folgenden Abschnittes sind, sofern nicht anderweitig angeführt, aus den Grundlagenbüchern xxxx und xxxx übernommen. Genetische Algorithmen sind heuristische Suchansätze. Im Wesentlichen zeichnet sie eine probabilistische Eltern Selektion als primären Suchoperator aus. Als weitern Suchoperator kann noch auf die Mutation zurückgegriffen werden. Dieser garantiert eine Erreichbarkeit aller Punkte im Suchraum und erhält die Grunddiversität in der Population. Es gibt zwei verschiedene Algorithmen der Standart-GA tauscht nach einer Generation die komplette Elternpopulation durch die Kinderpopulation aus. Und bestehen in der Regel immer aus fünf gleichen Schritten wie in Abb. 10 zusehen ist. Im Gegensatz dazu gibt es den Steady-State-GA, welcher durch seine überlappende Population auszeichnet, dieser Algorithmus wird in der Arbeit

nicht verwendet und wird deswegen nicht weiter erklärt.

Evolutionäre Algo – s.128 und Seite - 11 Genetic Algorithm Essentials

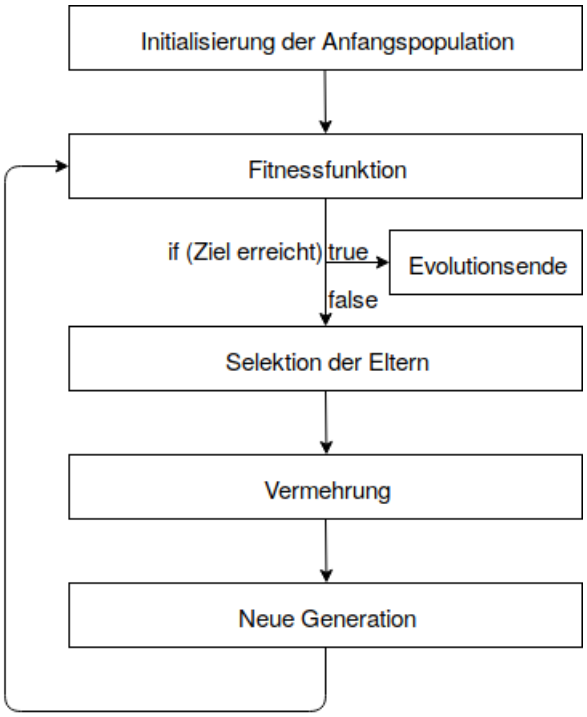


Abbildung 1: Ablaufdiagramm eines Genetischen Algorithmuses mit 5 Schritten

Der Standard genetische Algorithmus 6 besteht aus folgenden 5 Schritte:
Schritt 1, Initialisieren der Anfangspopulation.
Schritt 2, Fitness berechnen mit Hilfe der Fitnessfunktion.
Schritt 3, Selektieren der Eltern.
Schritt 4, Vermehren durch Crossover und Mutation.
Schritt 5, Austausch der Populationen.
In den nachfolgenden Unterkapiteln werden auf die einzelnen Schritte genauer eingegangen.

Im folgenden Kapitel werden auf diese 5 Schritte des Genetischen Algorithmus 6 näher eingegangen. Zuerst wird auf Aufbau und die Initialisierung eingegangen. Anschließend folgt eine Zusammenfassung der Fitnessfunktion. Darauffolgend werden Verschiedene Möglichkeiten für die Eltern Selektion dargelegt. Anschließend wird die Vermehrung durch Crossover und Mutation erklärt. Zum Schluss wird noch auf den Austausch der

Algorithm 1 Basic Genetischer Algorithm

```
1: Initialisieren der Anfangspopulation                ▷ put some comments here
2: while Fitness ≤ Abbruchbedingung do
3:     Fitness aller Individuen Berechnen
4:     Selektieren der Eltern
5:     Vermehren durch Cross-over und Mutation
6:     Austausch der Populationen
```

Populationen eingegangen.

2.2.1 Aufbau und Initialisierung der Population

Der klassische genetische Algorithmus basiert auf einer Reihe von Kandidatenlösungen. Die Größe der Population ist somit auch die Anzahl der Lösungen. Jede Lösung kann als einzelnes Individuum gesehen werden und wird durch ein Chromosomenstrang repräsentiert. Ein Chromosom besteht wiederum aus vielen Genen, welche die Hyperparameter repräsentieren. Der Aufbau ist grafisch in Abbildung 2 dargestellt. Es gibt verschiedene Möglichkeiten, diese Gene dazustellen. Um die Grundlagen nahe des später folgenden Konzepts zuhalten, wird der Ablauf per Dezimal-Genen erklärt.

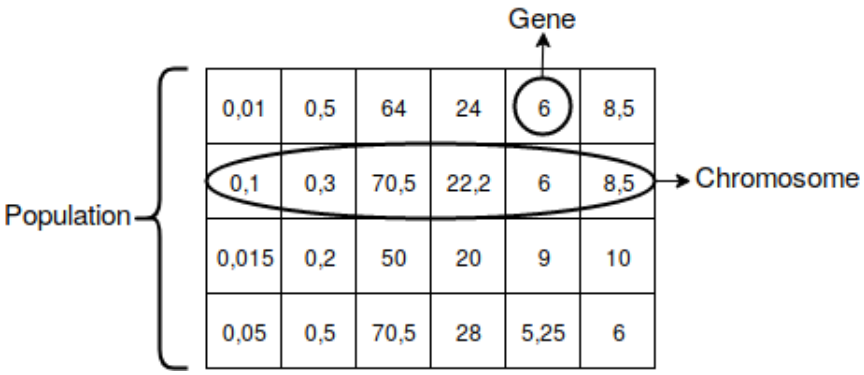


Abbildung 2: Beispiel einer Population mit 4 Induviduen (Chromsomen) welche 6 dezimale Gene besitzen

Diese Anfangspopulation (Generation 0) wird zufällig initialisiert, um die größt mögliche Abdeckung des Suchraums zu gewähren. Die erste Generation besitzt eine sehr geringe Fitness, welche im verlauf des Trainings stetig steigert bis sie das Maximum erreicht. (XXXFachbegriffxx)

2.2.2 Fitnessfunktion

Die Fitnessfunktion (engl. Fitnessfunction) bewertet das Individuum anhand seiner Funktionstauglichkeit, bezogen auf die vorhandene Aufgabe. Dabei werden nicht einzelnen Gene bewertet, sondern das ganze Chromosom. Es gibt keine universelle Fitnessfunktion, diese muss also für jede Anwendung speziell geschrieben werden. Es wird nicht berücksichtigt welches Gene sich positiv bzw. negativ auswirken. Als Rückgabewert gibt die Fitnessfunktion uns einen dezimalen/float Fitnesswert. Dabei steht ein höherer Fitnesswert stehst für eine höher Qualität an Individuum sprich bessere Lösung.

Practical Computer Vison Apllica s.134

2.2.3 Selektion der Eltern

Bei dem Schritt Selektion (engl. Select Parents) geht es darum einen Elternpool zu generieren, aus welchem die neue Generation erstellt wird. Deshalb ist es wichtig, nur die Besten, geeignetesten Individuen auszuwählen. Es gibt verschiedene Ansätze bei der Selektion, die elementar wichtigsten werden genannt und erläutert.

Informationen wurden aus dem Paper [3] entnommen.

- **Auswahl proportional zur Fitness (engl. Fitness Proportonal Selction(FPS))** Die Eltern werden nach ihrer Fitness proportional ausgewählt und zum Elternpool hinzugefügt. Wenn $f(a_i)$ die Fitness des Individuell a_i in der Population ist, dann ist die Wahrscheinlichkeit selektiert/ausgewählt zu werden:

  $$ps(a_i) = \frac{f(a_i)}{\sum_{i=1}^{n} f(a_j)}; j \in 1, 2, ..., n \tag{1}$$

  wobei n die Anzahl der Individuen einer Population ist. Diese Wahrscheinlichkeit ps kann man sich als Anteil auf einem Rouletterad, wie in Abblidung 4 vorstellen. Auf dem zufällig die Eltern aus den Individuen $a1, .., an$ „ausgedreht "werden. Dieser Ansatz hat leider das Problem, dass Individuen die am Anfang sich als gut beweisen, schnell die ganze Population übernehmmen. Das kann dazuführen, dass eine mögliche bessere Lösung durch den Algorithmus im Suchraum nicht gefunden wird.

  introduction to evolutionary comp s80

- **Ranking Selektion** Diese Selektion wurde von Backer als Verbesserung der Fitness Proportonal Selection entwickelt [4]. Dabei werden die Eltern nicht direkt nach

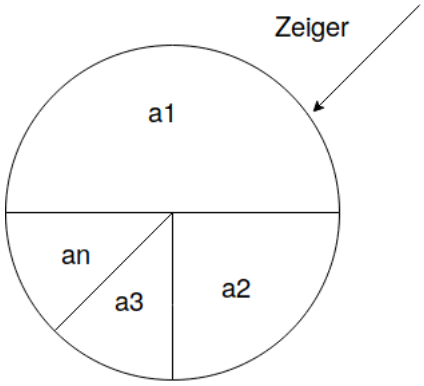


Abbildung 3: Rouletterad mit proportinalen Anteil der Individuen anhand ihrere Fitness

ihrer Fitness ausgewählt. Die Fitness dient nur zum Einteilen in eine Rangliste. Anhand dieser Rangliste wird dann wieder mit Hilfe des Rouletterades ausgewählt. Dabei gibt es verschiedene Verfahren wie diese Verteilung aussehen kann:

Das lineare Ranking:

$$p_i = \frac{1}{N}(n^- + (n^+ - n^-)\frac{i-1}{N-1}; i \in 1, ..., N \tag{2}$$

Wobei p_i die Wahrscheinlichkeit des Individuums ist selektiert zu werden. $\frac{n^-}{N}$ ist die Wahrscheinlichkeit des schlechtesten Individuums selektiert zu werden und $\frac{n^+}{N}$ ist die Wahrscheinlichkeit des besten Individuums selektiert zu werden.

Das expontienelle Ranking:

$$p_i = \frac{c^{N-i}}{\sum_{j=1}^{N} c^{N-j}}; i \in 1, ..., N \tag{3}$$

die Summe $\sum_{j=1}^{N} c^{N-j}$ normalisiert die Wahrscheilichkeit um Sicherzustellen das $\sum_{i=1}^{N} p_i = 1$. Wobei die Berechnungen 2 und 3 nur den Anteil eines Individuums auf dem Rouletterades verändern.

- **Tunier Selektion** In diesem Verfahren werden zufällig k Induviduuen der Population ausgewählt. Diese k Individuen treten wie in einem Tunier gegeneinander an. Der Gewinner ist das Individuum mit dem besten Fitnesswert, dieser wird dann auch als Elternteil ausgewählt. Hierbei wird auf den Elternpool verzichtet und direkt ein Kind aus zwei Gewinnern erstellt. Eingesetzt wird dies bei kleineren Populationen mit weniger als 20 Individuen.

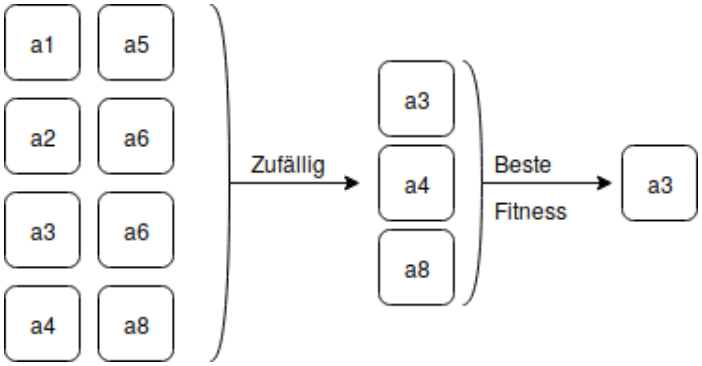


Abbildung 4: Tunier Selektion mit k = 3 Individuen und dem Gewinner Individuum 3

2.2.4 Vermehrung

Aus dem Elternpool werden nun Nachkommen (neue Individuuen) geschaffen. Alleine durch die Paarung(engl. Crossover) von qualitativ hochwertigen Individuen wird erwartet, dass die Nachkommen eine bessere Qualität besitzen als die ihrer Eltern. Als zweite Verbesserung wird noch die Mutation einzelner Gene angewendet. Für Crossover und Mutation gibt es verschiedene Ansätze, die in diesem Abschnitt genauer erklärt werden.

Crossover nennt man die Operation, bei der die Chromostränge der Kinder Individuen zusammengesetzt werden. Beim Crossover gibt es mehrer Varianten, die Ein-Punkt-Crossover (engl. One-Point-Crossover), in welchem zufällig ein Punkt im Chromsomenstrang festgelegt wird. Ab diesem Punkt wird der Chromosomenstrang dann aufgeteilt und anschließend mit dem Crossover des anderen Elternteils wieder zusammen gesetzt. Ein einfaches Beispiel ist im Oberenteil der Abbildung 5 zu sehen.

Eine Abwandlung des Ein-Punkt-Crossover ist das Zwei-Punkt-Crossover oder k-Punkt-Crossover. Hier wird der Chromsomenstrang an k Punkten aufgeteilt und anschließend mit dem Anteil des zweiten Elternteil wieder zusammengesetzt. In mittleren Teil der Abbildung 5 ist ein k = 2 Crossover oder auch zwei-punkt-crossover (engl. two-point-crossover) zu sehen.

Eine weitere grundlegende Operation beim Crossover ist die Uniform-crossover [5] in welcher es keine festgelegte Anzahl an Punkten gibt. Hier wird für jedes Gen zufällig entschieden aus welchem Elternteil das Gen entnommen wird. Dies im unteren Teil der Abbildung 5 noch einmal veranschaulicht.

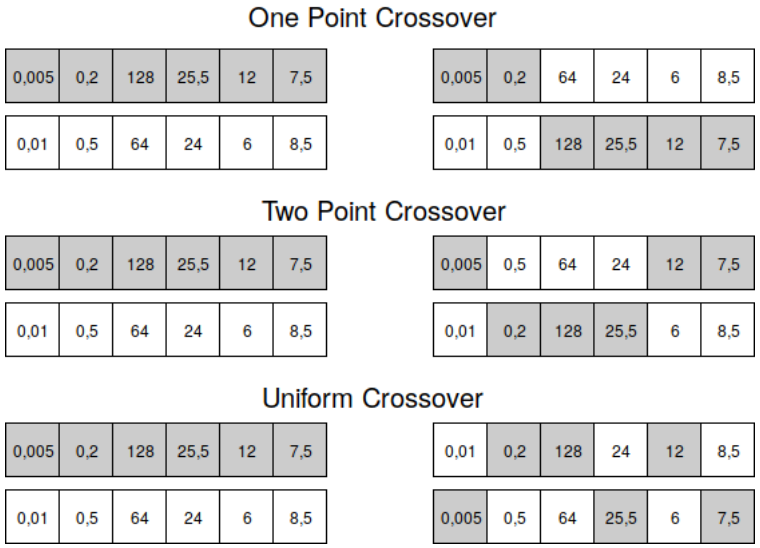


Abbildung 5: Die Wichtigsten drei Crossover Operationen. Oben die one-point Crossover, mitte two-point Crossover, unten Uniform Crossover. Auf der linken Seite sind Eltern abgebildet und auf der Rechtenseite die neu erzeugten Kinder

Crossover nach dem Paper [6].

Mutation Hierbei wird jedes Gen des Individuums zufällig mit einer zufälligen Mutation versehen. Durch diese Mutation wird eine höhere Diversität in die nachfolgende Generation übergeben. Diese Mutation macht es möglich einen größeren Suchraum abzudecken und somit die Werte genauer anzupassen, um so auf die optimale Lösung zu kommen. Hier wird die Gauss-Mutation verwendet. Es wird ein Zufallswert aus der Normal- oder Gauß-verteilung hinzu addiert. Durch diese Wahrscheinlichkeitsverteilung werden viele Mutationen nur kleine Veränderungen, aber auch größere Sprünge sind möglich. Um die vorher mit Crossover neu bestimmten Individuen, welche schon eine hohe Grundfitness haben, nicht zu stark zu verändern wird ein Gen nur um wenige Prozent verändert. Dies ist in Abbildung 6 zusehen. Die Mutationen in der Zeichnung wurden zufällig gewählt.

Evolutionäre Algorithmen S.60

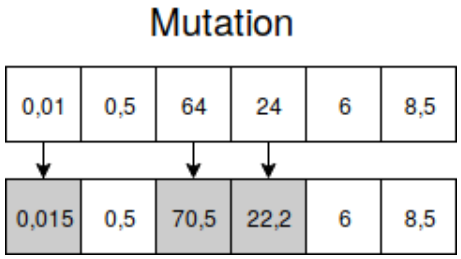


Abbildung 6: Muation von Genen um eine höhere Diversität zu erhalten

Practical Computer visionaplica ss.140

2.2.5 Neue Generation

Der letzte Schritt des Genetischen Algorithmuses besteht aus dem Austausch der Generationen. Die neue Kind Generation tauscht nun die alte Eltern Generation aus. Anschließend folgen die gleichen 4 Schritte so lange bis die gewünschte Fitnesswert erreicht ist. Nach dem erreichen der Abbruchbedingung, kann aus der letzten Generation das qualitativ hochwertigste Individuum ausgesucht werden und als Lösung verwendet werden.

2.3 Künstliche Neuronale Netze

Künstliche Neuronale Netze sind dem natürlichen Vorbild der neuronalen Netze im Gehirn nachempfunden. Beide Netze setzen sich aus einzelnen Neuronen zusammen, welche miteinandern verbunden sind und somit ein großes Netz entstehen lassen. Wie man in Figure 8 sieht ist jede Schicht aus einzelnen Neuronen aufgebaut, welche mit den Neuronen der nächsten Schicht verbunden sind. Diese Verbindungen repräsentieren die Gewichte, über diese kann einem Netz verschiedene Zusammenhänge von Input und Output antrainiert bzw. angelernt werden.

Im folgenden Kapitel wird zuerst der Aufbau eines Neurons/Perseptron erklärt. Anschließend wird auf den strukturellen Aufbau eines Künstlichen Neuronalen Netzes nähergebracht. Zum Schluss werden noch wichtige Eigenschaften wie die Verlustfunktion und der Gradientenabstieg eingegangen, sowie auf die Hyperparameter, welche für die Arbeit essenziell sind.

Grundlagen aus dem Buch ArificalNeuroalNetworks s.11

2.3.1 Aufbau eines Neurons

Ein Neuron besteht immer aus dem gleichen Aufbau: Eingänge, Gewichte, Schwellwert, Aktivierungsfunktion und einem Ausgang. Nachfolgenden Unterkapitel werden diese Ausführlich erklärt.

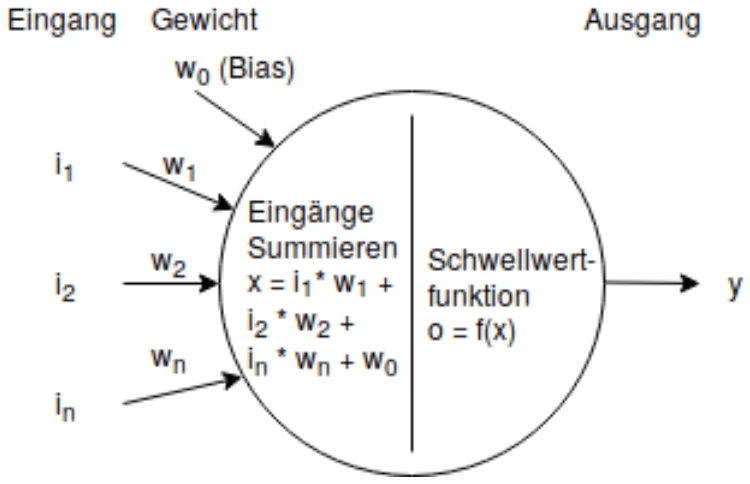


Abbildung 7: Aufbau eines Neurons

Eingang Bei den Eingangswerten $i_1, ..., i_n$ handelt es sich um einfache xxxFloatwertxxx, diese werden mit den Gewichten $w_1, ..., w_n$ verrechnet. Ein Neuron hat meist mehrere Eingangsgrößen, welche alle zusammen mit den Gewichten und dem Schwellwert aufsummiert werden. 4 Diese Werte werden zufällig initialisiert und per Training verbessert, somit handelt es sich um einen angelernten Werte, welche durch die Fehlerrückführung (engl. Backproagation) verbessert werden.

Schwellwert Auf dieses Aufsummiertes Ergebniss wird anschließend ein Schwellwert (engl. Bias) w_0 gerechnet, dieser führt zu einem besseren Verhalten beim Trainieren. Bei diesen Werten handelt es sich auch um angelernte Werte und helfen die Flexibitlität der Netze zu erhöhen.

$$x = \sum_{k=1}^{n} i_k * w_k + w_0 \tag{4}$$

Aktivierungsfunktion Die Aktivierungsfunktion kann man sich als Schwellwertfunktion vorstellen, ab wann das Neuron den Eingang weiter gibt. Die entstandenen Summe x wird Aktivierung genannt und anschließend über eine Aktivierungsfunktion transformiert:

$$o = f(x) \tag{5}$$

Die Aktivierungsfunktion kann dabei verschiedene Formen haben. Für einfache Aufgaben kann Beispielsweise eine Sprungfunktion verwendet werden:

$$\sigma(t) = \begin{cases} 1, & \text{wenn } t \geq 0 \quad (6a) \\ 0, & \text{sonst } t < 0 \quad (6b) \end{cases}$$

Für das approximieren von wertkontinuierlichen Funktionen wird die Sigmoid Funktion verwendet.

$$sig(t) = \frac{1}{1 + e^{-t}} \tag{7}$$

Bei der Klassifikation werden hingeben, der ReLU-Layer 8b 8b oder der Leaky-ReLU Layer 9b benutzt, diese verhindern das Explodieren bzw. Verschwinden des Grandienten beim Training:

$$R(z) = \begin{cases} 1, & \text{wenn } z \geq 0 \quad (8a) \\ 0, & \text{sonst } z < 0 \quad (8b) \end{cases}$$

$$R(z) = \begin{cases} 1, & \text{wenn } z \geq 0 \quad (9a) \\ \alpha z, & \text{sonst } z < 0 \quad (9b) \end{cases}$$

Ausgang Wenn die Schwellwertfunktion aktiviert wird, wird am Ausgang des Neurons ein Wert geschalten. Dieser Ausgangswert kann dann entweder an andere Neuronen weitergeben werden oder als finales Ergebnis verwendet werden.

2.3.2 Struktureller Aufbau eines Künstlichen Neuronalen Netzes

Aus schlauem zusammen schließen solcher Neuronen entsteht ein Künstliches Neuronales Netz welches auch Multi-Layer-Perseptron genannt wird. Dabei sind grundsätzliche jegliche strukturelle Anordnung der Neuronen möglich. Besonders verbreitet ist das sogenannte Feedforward Netz (FFW). Bei den FFW Netzen sind die Verbindungen nur in eine Richtung gerichtet. Dies wird beispielhaft in Abbildung 8 gezeigt. Hier ist gut zu sehen, dass diese Netze in drei Schichten unterteilt werden können.

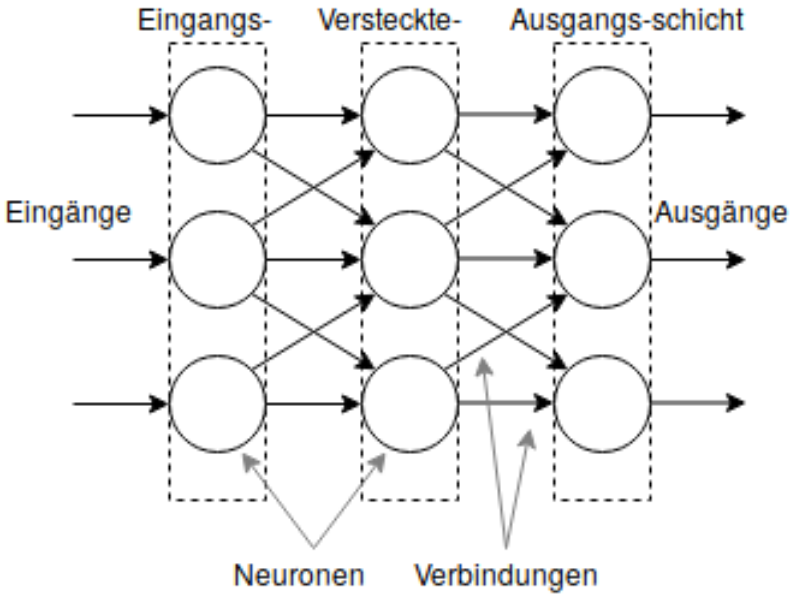


Abbildung 8: Künstliches Neuronales Netz mit drei Schichten je drei Neuronen

- **Eingangsschicht** Die Neuronen der Eingangsschicht sind nicht wie die in 2.3.1 beschriebenen Neuronen aufgebaut. Die einzige Aufgabe der Eingangsneuronen ist das verteilen der Eingagsinformationen an die Verstecktеschicht, weshalb es immer genau soviele Eingangsneuronen wie Eingangssignale geben muss.

- **Verstecktеschicht** Die Verstecktesicht besteht aus den in 2.3.1 erläuterten Neuronen. Sie kann aus einer Beliebigen Anzahl dieser aufgebaut sein. Es kann auch beliebig viele Verstecktеschichten geben. In der Literatur bezeichnet man Neuronale Netze mit mehr als einer Verstecktenschicht als Deep Neural Networks.

- **Ausgangsschicht** Die Ausgangsschicht beinhaltet genau soviele Neuronen wie Ausgangsgrößen gewünscht. Aus dieser können dann die Klassifizierungswerte entnommen werden.

2.3.3 Verlustfunktion

Die Verlustfunktion(engl. Lossfunction) stellt ein ausgesuchtes Maß der Diskrepanz zwischen den beobachteten und den vorhergesagten Daten dar. Sie bestimmt die Leistungsfähigkeit des Künstlichen Neuronalen Netzes während des Trainings. Ziel ist es, im laufenden Prozess der Modellanpassung, die Verlustfunktion zu minimieren.

2.3.4 Gradientenabstieg

Um die Fehlerfunktion zu minimieren wird als Werkzeug der Gradientenabstieg benutzt. Diese ist nur möglich, da ein Künstliches Neuronales Netz aus verketteten differenzierbaren Gewichte der Neuronen(Tensoroperationen) aufgebaut ist, die es erlauben duch Anwendung der Kettenregel die Gradientenfunktion zu finden, die den aktuellen Parametern des Datenstapels Werte des Gradienten zuordnet. Es gibt auch hier verschiedene Ansätze von Optimierern, welche die genauen Regeln wie der Gradient der Verlustfunktion zu Aktualisierung der Parameter verwendet wird hier könnte Beispielweise den RMSProp-Optimierer, der die trägheit des Gradientenabstiegsverfahren berücksichtet.

Seite 83 - Deep Learning chollet

2.3.5 Hyperparameter

Als Hyperparameter werden, in Bezug auf Künstliche Neuronale Netze, werdem meist die Anfangsbedingungen bezeichnet. Für diese Hyperparameter gelten keine universellen Werte sondern müssen je nach Daten und Funktion bzw. künstliches Neuroales Netz speziell angepasst und verändert werden. Deshalb gibt es nur einige Regeln und grobe Abschätzungen in welchen Grenzen sich diese Hyperparameter befinden. Zu diesen Hyperparameter gehören folgdende:

- **Learningrate**,blabalaa
- **Dropout**
- **Lossfunktion**

- **Optimizer**
- **Model Achitektur**

xxxwird noch angepasst wenn ich weis welche Parameterxxx

2.4 Zusammenfassung

3 Stand der Forschung und Technik

Durch die gestiegene Rechenleistung der heutigen Computer ist auch die anwendungshäufigkeit deutlich gestiegen. Es werden nicht nur neue Ansätze von großen Unternehmmen wie Google Deepmind entwickelt. Sondern auch kleiner anwendungen die nicht nur im IT-Bereich ihre anwendungen finden.

3.1 Deep Mind

Den größten Fortschitt der letzen Jahre verzeichnete Googles Deepmind. Google setzt dabie auch auf einen Populations bassierenden Algorithmus welcher auf den Genetischen Algorithmen aufbaut. Zum selbst Algortihmus später mehr. Goolge nutz ihre PopulationBasedTraining vorallem im Bereich der Künstlichen Inteligzenz. So konnten sie bei verschiedene Knns wie AlphaGo(Go)[7] und AlphaStar(Starcraft2)[8] deutliche verbesserungen durch das PBT erreichen. Google verwendet PBT nicht nur für Brett- bzw. Computer-spiele sodnern auch für Anwendungen im Bereich des autonomen fahrens in zusammenarbeit mit der Google-Tochter Waymo.

3.2 PopulationBasedTraining

PBT ist eine weiter entwicklung der klassischen Grid- und Randomsearch und wird von den Genetischen Grundlagen stark beinflusst. Dies ist zusehen daran das PBT auch eine Population verwendet, auch werden hier die übernahmme des Fitesten und Muationen verwendet. Zudem ist das Training Asynchron und Parallel möglich. Desweitern wurde ein online anpassung der Hyperparameter wärend des Trainings implementiert. Durch dieses Feature, wird viel Rechenleistung eingespaart bzw. das berechnen der Hyperparameter stark beschleunigt. [2]

3.3 Software Testing with Ga

Die Softwarebewertung spielt eine entscheidende Rolle im Lebenszyklus eines Software-Produktionssystems. Die Erzeugung geeigneter Daten zum Testen des Verhaltens der Software ist Gegenstand vieler Forschungen im Software-Engineering. In diesem Beitrag wird die Qualitätskontrolle mit Kriterien zur Abdeckung von Anwendungspfaden betrachtet und ein neues Verfahren auf der Grundlage eines genetischen Algorithmus zur Erzeugung optimaler Testdaten vorgeschlagen. [9]

3.4 Travelling Salesman Problem

Einer der bekanntesten anwendungen für GA ist das Travelling Salesman Problem, in welchem die kürzeste Route beim Austragen von Briefen für einen Postboten berechnet werden soll. Es ist ein Optimales Problem zum Lösen für GA, durch die zurückgelegte steckte kann leichte eine Fitnessfunktion aufgestellt werden. Desweitern wird der Suchbereich mit jedem auszutragenden Brief um einen Expotonenten größer. Und ist somit für gänige Algorothmen nur mit sehr viel Rechnleistung zu bewältigen.

Doch je mehr Briefe der Postbote austragen soll umso mehr Variablen gibt es, sprich es wird wesentlich schwerer für ein fest geschrieben (eng. Hardcoded) Algorithmus den kürzesten weg zu finden. Für den Ga ist dies kein Problem da mit der richtigen Fitnessfunktion eine einfacher Rückgabewert der Funktion zu bekommen ist. Dementsprechend kann die Route einfach optimiert werden.

3.5 Temperatur Schätzungen

–Fällt weg– wegen GP Es gibt auch Forschungen zur berechnung von Temperaturverläufen der Erde [10]. Vom gleichen Author gibt es auch eine abschätzungen von Wärmeflusses zwischen Athmosphäre und Meereies in Polarregeionen [**?**, **?**]. All diese Vorraussagen wurden mithilfe von Genetischen Programming ausgerechnet.

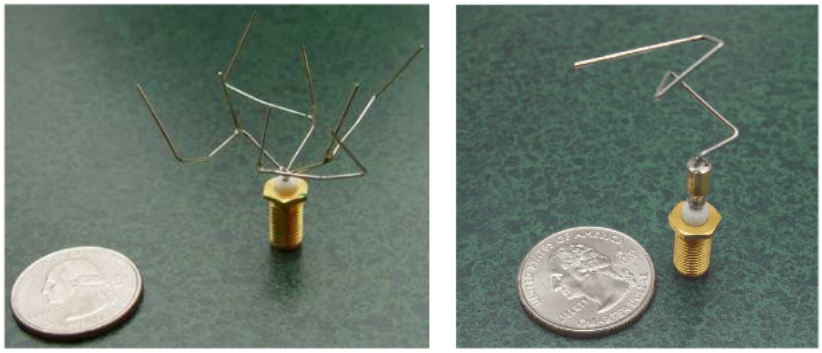
In dem die Aufzeichnungen der Temperaturverläufe als Trainingsdaten verwendet werden konnten Lösungen gefunden werden, welche den rellen daten sehr nahe kommen.

3.6 Generativ Design

Heute gibt es in viele Computer-aided design(CAD) Programme in welchen es implementierungen von Generativen Design Werkzeugen gibt. In dennen über Iterationen neue mögliche Designs berechnet werden deis passiert auch auf Basis der Genetischen Evolution. Sie bauen nicht auf den Genetischen Algorithmen auf sind aber nahe verwante und sollten nicht unterschätzt werden. Mit ihnen ist es möglich Bionische Stukturen für addetive Vertigung zu designen. Und sie speziell auf die Anwendung anpassen. So kann aus einem einfachen Frästeil ein wesentlich Leichtes und Material spaarenderes Model entwickelt werden.

Abbildung 9: Additives Design über mehrer Iterationen

NASA - Antenne DIe Nasa hat 2006 eine Weltraumantenne mit hilfe evolutionären Algorithmen entwickelt. Die Entwicklung der Antenne von hand ist sehr zeitaufwändig und arbeitsintehnsiv, zusätzlich braucht man großes wissen in der entsprächenden Domain. Deshalb nutzen die Forscher einen Populations bezogenen Suchansatz, um Umgebungsstrukturen und elektromagnetischer mit einzubeziehen. Die entwickelte Antenne wurde produziert und auch auf Space Technology 5 mission genutzt. [1]



(a) (b)

Abbildung 10: Foto der Prototypen für unterschiedliche Anforderungen. [1]

3.7 GLEAM

General Learning Evolutionary Algorthm and Method ist eine vom Kit entwickelte Methode um Aktionsketen zu berechnen. Dazu gehörtz zum beispiel das Aufeinandern abstimmen der Maschinen in einem Maschinenpark um so genannte totzeiten der Maschinen zu verringern also die gesamt auslastung zu erhöhen.

Mit GLEAM wurde auch versucht die Stuerung von 6-Achsigen Robotorarmen zu verbessern. Es konnte gezeigt werden das die Steuerung mit Gleam funktioniert, dies wurde

aber leider nie der neue Industie Standart. Es wird immer noch mit der klasschischen xxxSteuerungxxx gearbeitet.

3.8 Reainforcment learning with GA

Reinforcment learning ist möglichweise einer der größten Anwendungsgebiete der GA. Hierbei werden Neuronale Netze nicht mit hilfe von Gradienstieg training, wie im Gundlagen Kapitel besprochen. Sondern mit hilfe von Genetischen Algorithmen. Dabei wird das Neuronale Netz nicht

3.9 Zusammenfassung

4 Konzept

4.1 Anforderungsanalyse

4.2 Genetischer Algorithmus

4.2.1 Eltern auswahl

Ranking Selction mit 50 übernahme als Eltern. Zusätzlich werden die besten 4 Eltern ganz übernommen. das die Generationen sich nicht zu sehr ähnlichsind wird hoh mutation bei den Kindern. Zusätzlich werden bei jeder neuen Generation immer ein paar neu und damit zufällige Individuen hinzu gefügt.

Best 50 prozent,heißt aus der oberen hälfte der alten Generation werden alle Induviduen dem Elternpool hinzugefügt. Aus welchen dann zufällig die einzelnen Elternteile ausgewählt werden. Es mssen natürlich nicht immer 50 prozent sein, es kann sich auch um einen anderen Prozentsatz handeln.

4.3 Zusammenfassung

5 Implementierung

5.1 Systemaufbau

5.2 Zusammenfassung

6 Evaluation und Tests

6.1 Einleitung

6.2 Testszenarien

6.3 Evaluation

6.4 Ergebniss und Interpretation

6.5 Zusammenfassung

7 Zusammenfassung und Ausblick

7.1 Einleitung

7.2 Zusammenfassung

7.3 Bedeutung der Arbeit

7.4 Ausblick

Literatur

[1] Gregory Hornby, Al Globus, Derek Linden, and Jason Lohn. *Automated Antenna Design with Evolutionary Algorithms.*

[2] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks. *CoRR*, abs/1711.09846, 2017.

[3] Anupriya Shukla, Hari Pandey, and Deepti Mehrotra. Comparative review of selection techniques in genetic algorithm. 02 2015.

[4] James Edward Baker. Adaptive selection methods for genetic algorithms. In *Proceedings of an International Conference on Genetic Algorithms and their applications*, pages 101–111. Hillsdale, New Jersey, 1985.

[5] Gilbert Syswerda. Uniform crossover in genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 2–9, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[6] Dr. Anantkumar Umbarkar and P Sheth. Crossover operators in genetic algorithms: A review. *ICTACT Journal on Soft Computing (Volume: 6 , Issue: 1)*, 6, 10 2015.

[7] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.

[8] Kai Arulkumaran, Antoine Cully, and Julian Togelius. Alphastar: An evolutionary computation perspective. *CoRR*, abs/1902.01724, 2019.

[9] Sina Keshavarz and Reza Javidan. Software quality control based on genetic algorithm. *International Journal of Computer Theory and Engineering*, pages 579–584, 01 2011.

[10] Karolina Stanislawska, Krzysztof Krawiec, and Zbigniew W. Kundzewicz. Modeling global temperature changes with genetic programming. *Comput. Math. Appl.*, 64(12):3717–3728, December 2012.