Fachhochschule Nordwestschweiz
Hochschule für Technik

Sudoku

Schwierigkeitseinstufung und Generator

Projektarbeit IP-5

Studiengang Informatik

Betreuer: Manfred Vogel, Lucas Brönnimann

Auftraggeber: Rätsel Agentur AG

Verfasser: Matthias Keller, Simon Beck

Brugg, 23.01.2017

1 Abstract

Das Ziel dieser Arbeit ist die Erstellung eines Generators um Sudokus für den kommerziellen Gebrauch zu erstellen. Damit ein Generator gebraucht werden kann müssen zwei Aufgaben erfüllt werden. Erstens muss die Schwierigkeit von Sudokus evaluiert werden können und zweitens muss das Programm eine Vielzahl von verschiedenen Sudokus generieren.
Um diese Teilaufgaben zu lösen ist eine Repräsentation des Sudoku in Java programmiert worden. Zusätzlich wurden verschiedene Lösungsmethoden realisiert um die gegebenen Sudokus zu lösen.
Für die Schwierigkeitseinstufung werden bereits eingestufte Sudokus gelöst und dabei diverse Werte aufgenommen. Mithilfe dieser Werte wird ein neurales Netzwerk trainiert, welches anhand der Merkmale ein Sudoku einer Schwierigkeitsstufe zuweisen kann.
Bei der Generierung von neuen Sudokus wird ein Korpus von Sudokus mit 17 Startpositionen verwendet. Diese Sudokus werden transformiert und mit weiteren Zahlen ergänzt. Für das Ergänzen von Zahlen sind drei Techniken vorhanden, die jeweils andere Resultate hervorbringen.
Die Resultate unserer Arbeit werden vorgestellt und Erweiterungsmöglichkeiten sind aufgelistet.

2 Inhaltsverzeichnis

3 Einleitung

Unser Projekt ist ein Sudoku Generator, welcher neue Sudokus auf der Basis von gegebenen Sudokus mit 17 Ziffern generiert und in Schwierigkeitsstufen kategorisiert. Zur Generierung werden die 17er-Sudokus mehreren verschiedenen Permutationen unterzogen. Danach werden zusätzliche Ziffern aus der Lösung des Sudokus hinzugefügt, was zu einzigartigen Sudokus führt. Die für die Einstufung der Schwierigkeit werden die Sudokus mittels menschlichen Lösungsmethoden gelöst und danach durch ein neurales Netzwerk klassifiziert.

Der Kunde des Projektes ist die Rätsel Agentur AG, welche Rätsel für viele verschiedene Print- und Onlinemedien vertreibt. Unter ihren Kunden befinden sich «20 Minuten» und der «Blick am Abend».
Bisher hat die Rätsel Agentur AG ihre Rätsel von externen Anbietern eingekauft.
Das Ziel unseres Projektes ist ein Sudoku Generator, welcher auf Knopfdruck Sudokus in verschiedenen Schwierigkeitsstufen generiert.

Zuerst werden die Konzepte und Vorgehensweisen bezüglich der einzelnen Komponenten des Projekts erläutert, danach folgt die technische Umsetzung. Zum Schluss werden Resultate und mögliche Erweiterungen des Projektes diskutiert.

4 Konzepte

4.1 Anwendungsdomäne/Umfeld

Unsere Arbeit wird als Informatikprojekt 5 an der Fachhochschule Nordwestschweiz durchgeführt. Das Projekt befindet sich im Kontext eines KTI Projektes des Institutes für 4D Technologien. Das Gesamtprojekt hat ein Webportal zum Ziel, mit welchem sich verschiedene Rätselarten generieren lassen. Zurzeit ist ein weiteres Studentenprojekt im Gange, welches sich mit den Logikrätseln Nonogramm und Hashi auseinandersetzt.

4.1.1 Herkunft

Sudoku ist ein weltweit sehr beliebtes Zahlenrätsel. Da zur Lösung nur Logik und keine komplexen arithmetischen Berechnungen nötig sind, ist es für jeden verständlich.

In seiner Grundform basiert es auf den lateinischen Quadraten des schweizer Mathematikers Leonhard Euler. Das Heute als Sudoku bekannte Rätsel wurde Ende der 1970er Jahre im New Yorker «Dell Puzzle Magazine» unter dem Namen «Number Place» veröffentlicht. Das Rätsel verbreitete sich ins rätselbegeisterte Japan und wurde dort mit dem Namen «Sudoku» versehen, wobei «Su» Nummer und «Doku» einzeln bedeuten.
Nach mehreren Jahren hoher Popularität im Land der aufgehenden Sonne verbreiteten sich Sudokus mit der Zeit in der ganzen Welt.
Seit 2006 werden jährlich Sudoku-Weltmeisterschaften abgehalten, wobei im Jahr 2016 213 Rätsel-Liebhaber aus 33 verschiedenen Ländern teilgenommen haben.

4.1.2 Aufbau und Regeln

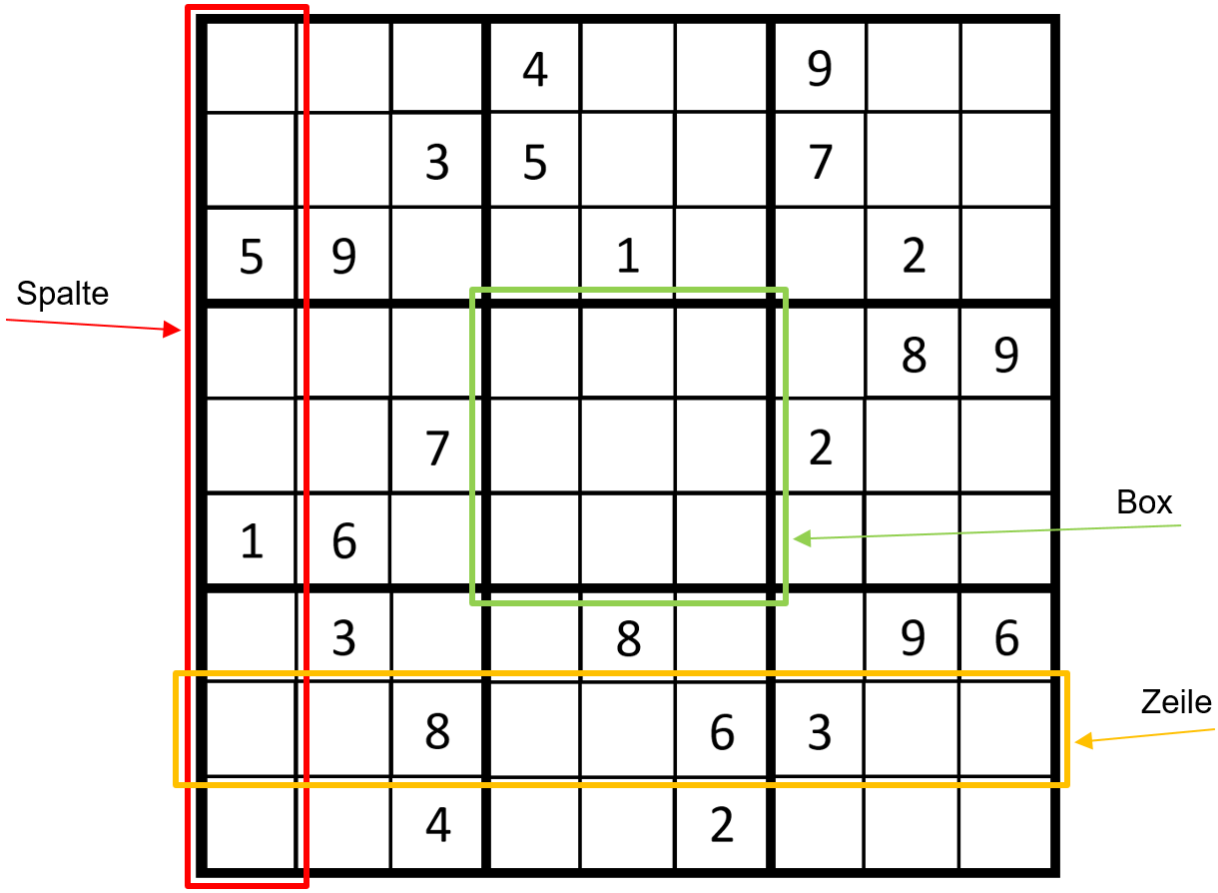

Abbildung 1: Aufbau des Sudoku

Sudokus bestehen meistens aus einem 9x9 Gitter, welches zum Teil mit Ziffern von 1 bis 9 gefüllt ist. Es gibt auch Sudokus in anderen Grössen. Für alle Grössen gelten jedoch dieselben Regeln. Ziel des Rätsels ist es, das gesamte Gitter mit Ziffern zu füllen, wobei nur eine einzige einfache Regel beachtet werden muss. In jeder Zeile, Spalte und jeder Box (3x3 Untergitter) darf jede Ziffer von 1 bis 9 nur genau ein einziges Mal vorkommen. 3 Boxen die horizontal oder vertikal nebeneinander stehen bezeichnen wir als horizontalen respektive vertikalen Block.

4.2 Lösen von Sudokus

Zur Lösung von Sudokus haben sich diverse Lösungsmethoden etabliert. Diese Lösungsmethoden sind sogenannte menschliche Lösungsmethoden. Diese erhalten ihren Namen vom Fakt, dass sie von Menschen zur Lösung des Rätsels verwendet werden.
Zum Lösen von Sudokus werden häufig Markierungen (engl. Pencilmarks) verwendet. Pencilmarks sind kleine Notationen pro Zelle, welche die noch möglichen Zahlen für diese Zelle repräsentieren. (Abbildung 2)

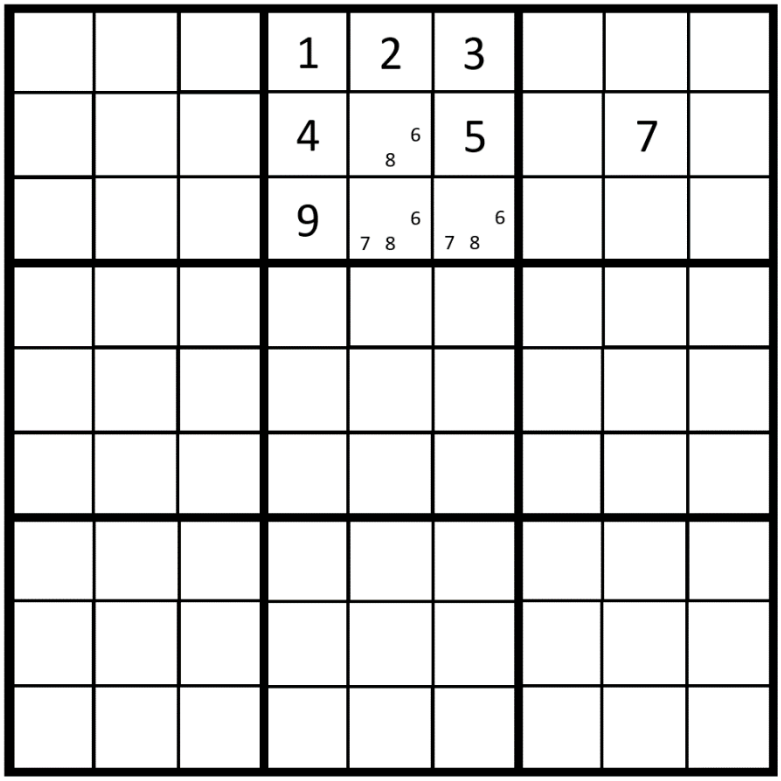

Abbildung 2: Pencilmarks

Pencilmarks werden durch das Anwenden diverser Lösungsmethoden immer weiter verringert, bis nur noch eine Markierung in einer Zelle vorhanden ist. Ist dies der Fall, kann der Zelle der Wert der übriggebliebenen Pencilmark zugewiesen werden, da diese Zahl die einzige ist, die in dieser Zelle noch möglich ist.

Lösungsmethoden können in zwei Kategorien unterteilt werden, Techniken die den Zellen einen Wert zuweisen und solche die die Pencilmarks diverser Zellen verringern. Die meisten Men-

schen wissen nicht, dass ihr Vorgehen beim Lösen eines Sudokus in spezielle Methoden eingeteilt werden kann. Folgend werden jene Lösungsmethoden beschrieben, welche wir in unserer Arbeit verwendet haben.

4.2.1 Naked Single

Ein Naked Single ist eine Zelle die nur noch eine einzige Pencilmark hat Somit kann der Wert der Zelle auf den der Pencilmark gesetzt werden.

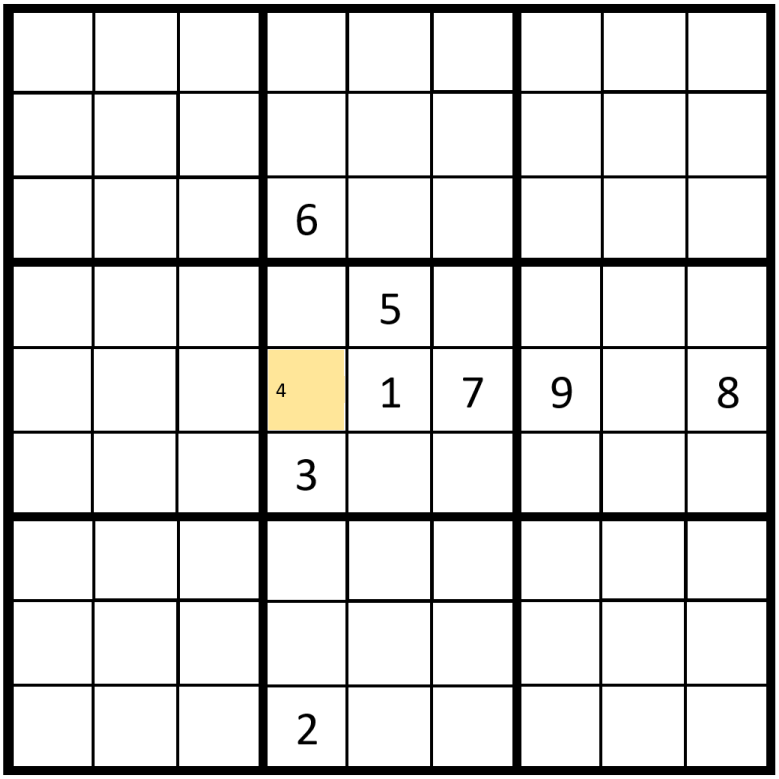

Abbildung 3: Naked Single

Im Beispiel hat die markierte Zelle nur noch eine einzelne Pencilmark (die Vier) übrig. Somit kann in diesem Feld die Vier gesetzt werden.

4.2.2 Hidden Single

Ein Hidden Single entsteht, falls in einem Container, d.h. in einer Zeile, Spalte oder einer Box ein Wert nur noch in einer Zelle möglich ist. In diesem Fall kann die Zelle auf diesen Wert gesetzt werden, unabhängig davon wie viele Pencilmarks die Zelle noch hatte.

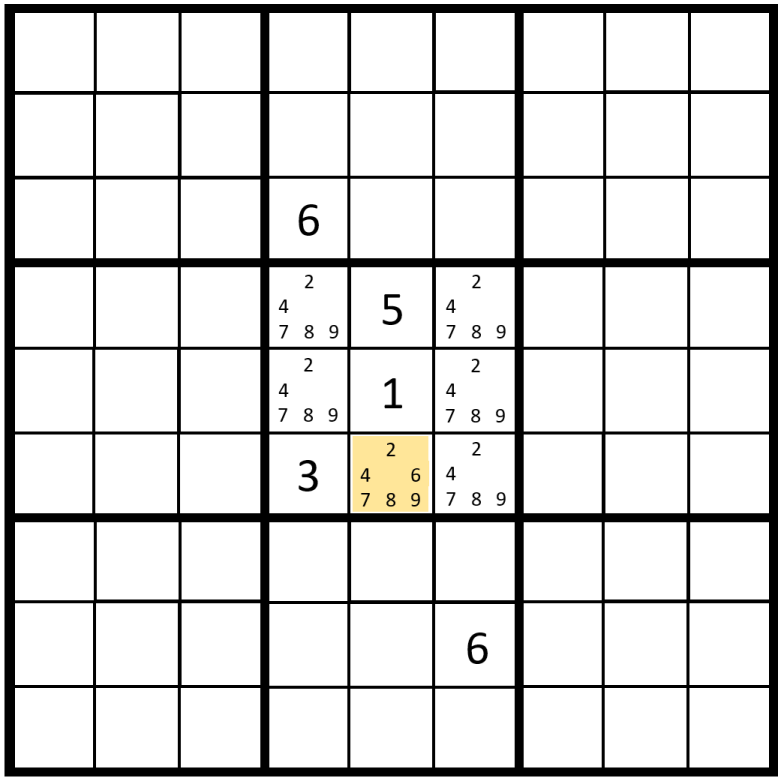

Abbildung 4: Hidden Single

In diesem Beispiel kann innerhalb der mittleren Box die Sechs nur in der markierten Zelle gesetzt werden. Folglich kann die Zelle mit der Sechs gefüllt werden.

4.2.3 Naked Subset

Ein Naked Subset beschreibt den Fall, wenn innerhalb einer Zeile, Spalte oder Box eine Anzahl von Werten nur in derselben Anzahl von Zellen gesetzt werden können und in diesen Zellen sonst keine Zahlen vorkommen können. Dies hat zur Folge, dass jene Werte in diesen Zellen vorkommen müssen und in allen anderen Zellen der Zeile, Spalte oder Box, jene Zahlen nicht erscheinen dürfen. Folglich können in den restlichen Zellen die Pencilmarks für diese Werte entfernt werden.

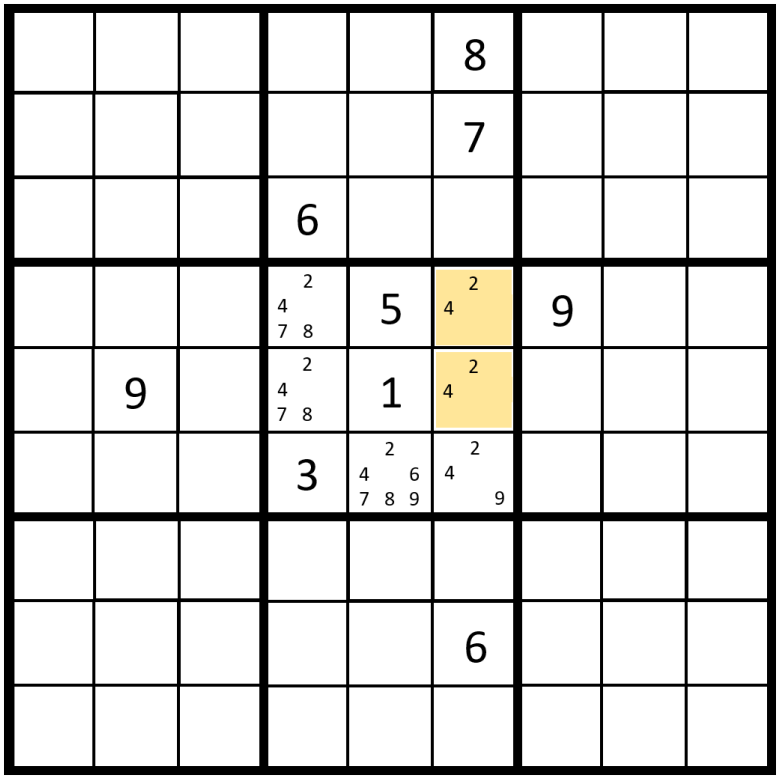

Abbildung 5: Naked Subset

Im Bild ist ersichtlich, dass die Ziffern Zwei und Vier in den markierten Zellen vorkommen müssen. Aus diesem Grund können die beiden Ziffern nirgends in der mittleren Box vorkommen, das heisst, die Pencilmarks für Zwei und Vier können bei den restlichen Zellen der Box entfernt werden.

4.2.4 Hidden Subset

Ein Hidden Subset liegt vor, falls innerhalb einer Zeile, Spalte oder Box eine Anzahl von Werten nur in genau derselben Anzahl Zellen vorkommen können. Daraus folgt, dass in diesen Zellen zwingend einer der Werte gesetzt werden muss, da diese Werte nirgends sonst im jeweiligen Container vorhanden sein können. Somit können für die betroffenen Zellen alle Pencilmarks, die nicht zum gefunden Subset an Werten gehören, entfernt werden.

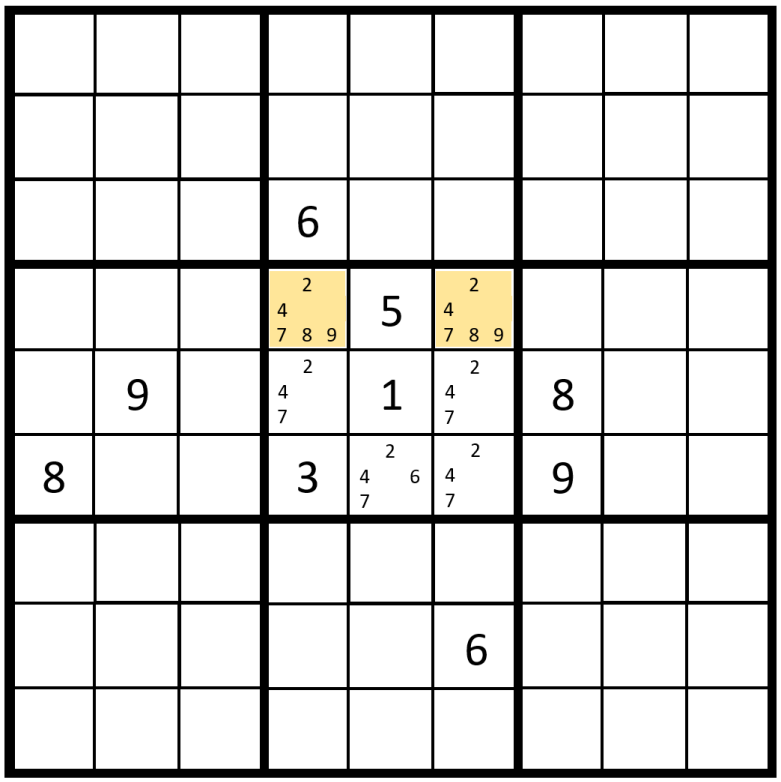

Abbildung 6: Hidden Subset

Die Werte Acht und Neun können in der mittleren Box nur in den markierten Zellen vorkommen. Dies impliziert, dass falls eine der beiden Zahlen in einer Zelle gesetzt wird, die andere Zahl in der übriggebliebenen Zelle vorkommen muss. Somit können die Pencilmarks für die anderen Zahlen in diesen beiden Zellen entfernt werden.

4.2.5 Block-Line Interactions

Block-Line Interactions, auch Locked Candidates genannt, ist eine Technik, bei der die Auswirkungen einer Box auf eine Zeile oder Spalte betrachtet werden. Kommen in einer Box alle Pencilmarks für einen Wert auf derselben Zeile oder Spalten vor, so muss dieser Wert in einer diesen Zellen vorkommen und deckt somit den Platz für diesen Wert in der Zeile oder Spalte ab. Deshalb können in allen anderen Zellen der Zeile oder Spalte (die nicht mit der Box überlappen) die Pencilmarks für diesen Wert entfernt werden.

					?			
					?			
					?			
			6	5	2 7 8 9			
	9		2 7 8	2 7 8	4			
			3	1	2 7 8 9			
					?			
					?			
					?			

Abbildung 7: Block-Line Interactions

In der mittleren Box kann die Zahl Neun nur in den beiden markierten Zellen vorkommen. Da die beiden Zellen beide Teil derselben Spalte sind, kann in den restlichen Feldern der Spalte (mit '?' markiert) die Neun nicht mehr vorkommen und kann deshalb aus den Pencilmarks entfernt werden.

4.2.6 X-Wing

Ein X-Wing entsteht, falls ein Wert in zwei unterschiedlichen Zeilen jeweils genau zweimal vorkommen kann und diese vier Zellen zusammen die Eckpunkte eines hypothetischen Rechteckes bilden. Diese Ausgangslage führt dazu, dass der Wert in diesen zwei Zeilen immer diagonal vorkommt und somit beide Spalten mit dem Wert gefüllt werden. Somit können für alle anderen Zellen, in den zwei Spalten, die nicht einer der vier Zellen entsprechen die Pencilmark für den gewählten Wert entfernt werden.

1	4	2 5 6 7 9	2 5	2 5	8	2 5 6 7	2 5 6 7 9	3
		?			7		?	
		?	6	9			?	
		?					?	
2 3 4 5 8	6	2 3 4 5 8 9	2 4 5 8	1	2 4 5	2 3 4 5 8	2 3 4 5 8 9	7
		?	3		9		?	
9		?					?	
		?				9	?	
		?					?	

Abbildung 8: X-Wing

Im Beispiel kann die Zahl Neun in den Zeilen Eins und Fünf jeweils genau zweimal vorkommen. Da die möglichen Vorkommnisse spaltenweise genau untereinander sind, bilden sie die Eckpunkte eines Rechteckes (markierte Zellen). Das heisst, dass die Neun entweder oben links und unten rechts oder oben rechts und unten links vorkommen kann. In beiden Fällen sind alle betroffenen Spalten mit einer Neun ergänzt worden. Deshalb können in den restlichen Zellen der beiden Spalten (mit '?' markiert) alle Pencilmarks für die Zahl Neun entfernt werden.

4.3 Schwierigkeitseinstufung

Die Rätsel Agentur AG hat ihre Sudokus in sieben verschiedene Schwierigkeitsklassen unterteilt. Die Schwierigkeitsstufen lauten folgendermassen:

- Very Easy
- Easy
- Medium
- Hard
- Very Hard
- Very Hard Expert
- Evil / Exotic

Die letzte Schwierigkeitsstufe wird nicht aktiv produziert und muss von uns auch nicht mit dem Generator erreicht werden.
Wir haben diese Schwierigkeitsstufen direkt übernommen und teilen schlussendlich unsere generierten Sudokus in diese Stufen ein.

Um Sudokus in eine Schwierigkeitsklasse einzustufen haben wir Machine Learning verwendet. Mithilfe eines neuralen Netzwerkes beurteilen wir anhand gewissen Eigenschaften, welcher Schwierigkeitsstufe ein gegebenes Sudoku entspricht.

Zu Beginn findet eine Anlernphase statt, in welcher dem neuralen Netzwerk einige Sudokus gezeigt werden, welche bereits in eine Schwierigkeitsklasse eingestuft wurden. Anhand dieser Daten lernt das Netzwerk, was ein Sudoku einer gewissen Schwierigkeit ausmacht.
Als nächstes wird mit weiteren bereits eingestuften Sudokus getestet, wie genau das Netzwerk neue Sudokus in die verschiedenen Schwierigkeitsstufen einteilt.
Sind nach der Testphase die gewünschten Werte erreicht, kann das Netzwerk zur Einstufung neuer Sudokus verwendet werden.

4.3.1 Modell

Das neurale Netzwerk beurteilt die Sudokus anhand mehrerer Features.

Da nur die Methoden Naked und Hidden Single Zahlen in das Feld einsetzen, werden diese nicht als absolute Werte verwendet. Die Anzahl der jeweils verwendeten Methoden wird mit der gesamten Anzahl der zu Beginn fehlenden Zahlen ins Verhältnis gesetzt.
Bei den Naked und Hidden Subsets betrachten wir nur die Subsets, welche eine Grösse von zwei, drei oder vier haben, da ein menschlicher Löser grössere Subsets nur extrem schwer finden kann.
Ebenfalls wird die Anzahl der Verwendungen der anderen beiden Lösungsmethoden Block Line Interactions und X-Wing als Feature im Modell verwendet.

Neben den Lösungsmethoden haben wir weitere Features gewählt, welche auf die Einstufung einen Einfluss haben. Hierzu gehört die Anzahl der vorgegebenen Zahlen, sowie die Anzahl der möglichen Startpositionen, an welcher mittels der Naked oder Hidden Single Methoden als erstes eine Zahl gesetzt werden kann. Für jede der verschiedenen Ziffern wird die Anzahl möglicher Startpunkte als einzelnes Feature verwendet. Als weiteres Feature wird die gesamte Anzahl der Pencilmarks im Sudoku ins Modell hinzugenommen. Die Markierungen zeigen für jede Zelle, welche Ziffern noch nicht ausgeschlossen wurden. Als letztes kommt hinzu, ob das Sudoku mit unseren Lösungsmethoden lösbar war (1) oder ob es mit Backtracking gelöst werden musste (0).

Diese von uns gewählten Features führen zu folgendem Modell:

- Anz. Naked Singles / Anz. fehlende Ziffern
- Anz. Hidden Singles / Anz. fehlende Ziffern
- Anz. Naked Subsets der Grösse zwei bis vier
- Anz. Hidden Subsets der Grösse zwei bis vier
- Anz. Block Line Interactions
- Anz. X-Wing
- Anz. gegebener Ziffern
- Anz. möglicher Startpositionen für jede Ziffer
- Anz. Pencilmarks
- Lösbar mit Lösungsmethoden

4.4 Generierung

Ein Sudoku muss eindeutig lösbar sein. Das heisst, es darf für ein Sudoku nur eine mögliche Lösung geben. Die Anzahl Lösungswege sind dabei jedoch nicht beschränkt. In einem Sudoku äussert sich diese Eigenschaft dadurch, dass die Reihenfolge von Lösungsmethoden die nicht voneinander abhängig sind keinen Einfluss auf die Lösung des Sudokus haben. Auch die Wahl der Lösungsmethode, sofern anwendbar, kann das Ergebnis nicht verändern.
Die minimale Anzahl Startpositionen, die ein Sudoku haben muss, damit es eindeutig lösbar ist, ist nicht offiziell bekannt. Es gibt jedoch eine Arbeit von Gary McGuire in welcher er durch eine erschöpfende Suche zeigt, dass es keine Sudokus mit 16 oder weniger Startpositionen gibt, die eindeutig lösbar sind. Diese Arbeit wurde jedoch bis dato nicht publiziert oder von anderen Forschern geprüft.

Damit wir bei der Generierung von neuen Sudokus nicht jeweils noch überprüfen müssen, ob ein Sudoku eine eindeutige Lösung hat, generieren wir unsere Sudokus nicht von Grund auf neu. Wir verwenden stattdessen bereits bekannte Sudokus mit 17 Startpositionen und eindeutiger Lösung (17er Sudoku) als Basis für die Generierung.

4.4.1 Transformationen

Aus einem 17er Sudoku können mithilfe von Transformationen eine grosse Menge weiterer 17er Sudokus generiert werden. Diese Sudokus sehen unter Umständen sehr verschieden aus, sind jedoch lösungstechnisch genau gleich lösbar wie das originale 17er Sudoku. Denn es werden keine Pencilmarks verändert beim transformieren des Sudokus. Die folgenden Transformationen verändern nur das Aussehen, jedoch nicht das Lösungsvorgehen des Sudokus.

- Transponieren
  Alle Zeilen mit allen Spalten des Sudokus tauschen, dies entspricht einer Spiegelung des Sudokus an einer Diagonalen.
  (2 Möglichkeiten)
- Permutationen der 9 verschiedenen Zahlen (9! Möglichkeiten)
  Die einzelnen Zahlengruppen untereinander vertauschen, z. B. werden alle Drei zu Vier, alle Vier zu Eins und alle Eins zu Drei.
  (9! Möglichkeiten)

- Zeilen innerhalb eines horizontalen Blocks vertauschen
  Die 3 Zeilen innerhalb eines horizontalen Blocks beliebig miteinander vertauschen.
  (3 x 6 Möglichkeiten)
- Spalten innerhalb eines vertikalen Blocks vertauschen
  Die 3 Spalten innerhalb eines vertikalen Blocks beliebig miteinander vertauschen.
  (3 x 6 Möglichkeiten)
- Horizontale Blocks vertauschen
  Die 3 Horizontalen Blocks beliebig miteinander vertauschen.
  (6 Möglichkeiten)
- Vertikale Blocks vertauschen
  Die 3 vertikalen Blocks beliebig miteinander vertauschen.
  (6 Möglichkeiten)

$$2 * 9! * 6^8 = 1'218'998'108'160$$

Kombiniert man diese Transformationen miteinander kann man aus einem Sudoku bis zu 1.2 Billionen verschiedene Sudokus erzeugen. Rotieren und Spiegeln des Sudokus können als Kombination der obenstehenden Transformationen realisiert werden und müssen deshalb nicht speziell aufgelistet werden.

4.4.2 Ergänzen

Da wir nicht nur 17er Sudokus generieren sollen, ergänzen wir die Sudokus in einem letzten Schritt mit weiteren Zahlen. Da das Sudoku eine eindeutige Lösung hat, ist es nicht möglich zufälligerweise Zahlen dem Feld hinzuzufügen, ohne in die Gefahr zu kommen das Sudoku unlösbar zu machen. Stattdessen wird das Sudoku gelöst und anschliessend aus der Lösung Zahlen ausgewählt und dem Feld hinzugefügt. Mit dieser Methode wird sichergestellt, dass keine falschen Zahlen dem Sudoku hinzugefügt werden.
Um für den Leser das Sudoku visuell ansprechend zu gestalten, achten wir darauf, dass das Sudoku eine möglichst symmetrische Anordnung der Startzahlen aufweist. Im Idealfall ist ein Sudoku horizontal, vertikal sowie über beide Diagonalen symmetrisch und zusätzlich noch Punktsymmetrisch.
Welche Schwierigkeit das Sudoku hat wird bei der Generierung nicht beachtet.

5 Technische Umsetzung

5.1 Technologien

Für die Realisierung unseres Projekts haben wir die Programmiersprache Java gewählt. Das Gesamtprojekt des Institutes für 4D Technologien baut auf der Java Virtual Machine auf und es wurde uns nahegelegt, eine Programmiersprache zu wählen, welche auch auf der JVM zu Hause ist. Wir haben schon viel mit Java gearbeitet und haben so die Einarbeitungszeit in die Grundfeatures der Sprache auslassen können.
Ein weiterer Punkt, welcher für die Wahl von Java spricht sind die vielen Ressourcen, welche im Netz erhältlich sind. Ebenfalls gibt es eine grosse Anzahl an Frameworks, welche Open Source verfügbar sind.
Zur Verwaltung und Versionierung unseres Quellcodes verwenden wir Git.
Das neurale Netzwerk wurde in Matlab trainiert und getestet. Für die Umsetzung des Netzwerkes in Java haben wir das Framework Neuroph verwendet.

5.2 Abbildung des Sudoku-Spielfelds

Für die Repräsentation des Sudoku-Spielfelds haben wir eine Board Klasse erstellt. Diese Klasse beinhaltet ein zweidimensionales Array mit den einzelnen Zellen sowie zusätzlich 3 Arrays für die Zeilen, Spalten und Boxen. Jede dieser Container-Instanzen besitzt jeweils ein Array in dem die einzelnen Zellen referenziert werden. Es kann somit über die Container als auch direkt über die Position auf eine Zelle zugegriffen werden.
Um Zellen mit einem Wert zu versehen haben wir eine Klasse Updater definiert, die mithilfe einer statischen Methode den Wert in die Zelle einsetzt und zusätzlich alle betroffenen Pencilmarks aktualisiert. Dabei wird sogleich in der Zelle gespeichert mit welcher Lösungsmethode der Wert gesetzt wurde.
Um unser Spielfeld persistent abzulegen, schreiben wir es in ein String, welcher in eine «.sudoku» Datei abgelegt wird. Der Feldstring ist folgendermassen aufgebaut:
[Feldhöhe] [Feldbreite] [Boxhöhe] [Boxbreite] [Alle Positionen] {Schwierigkeitsstufe}
Die Schwierigkeitsstufe ist nur im String enthalten, wenn diese bereits ermittelt wurde.

```
9 9 3 3 000000010400000000020000000000504070080003000010900003004002000501000000008060000
9 9 3 3 000000010400000000020000000000506040080003000010900003004002000501000000008070000
9 9 3 3 000000012000035000000600070700000300000400800100000000001200000800000400500006000
9 9 3 3 000000012003600000000007000410020000000500300700000600280000040000300500000000000
9 9 3 3 000000012008030000000000040120500000000004700060000000507000300000620000000100000
```

Abbildung 9: Beispiel des .sudoku Formats

5.3 Datensätze

Zum Anlernen des neuralen Netzwerkes haben wir zwei Datenpakete erhalten, welche jeweils aus ungefähr 600 Datensätzen bestehen. Das erste Paket besteht aus Sudokus, welche die Rätsel Agentur AG eingekauft hat, das zweite Paket stammt aus Eigenproduktion der Rätsel Agentur AG. Die erhaltenen Sudokus sind in unterschiedlichen Formaten (siehe Abbildung 10 und 11 abgespeichert. Wir haben die Sudokus mittels eines Parsers für das jeweilige Ursprungsformat in unser eigenes «.sudoku» Datenformat (Abbildung 9) umgewandelt und so zur weiteren Verwendung abgespeichert.

```
<Sudoku_Puzzle BuilderVersion="8.31.4716.39650" Profile="ClassicSudoku-9x9 - Medium"
  <Data Total="81">
    <Cell Index="0" Value="0" />
    <Cell Index="1" Value="4" />
    <Cell Index="2" Value="0" />
    <Cell Index="3" Value="0" />
    <Cell Index="77" Value="0" />
    <Cell Index="78" Value="0" />
    <Cell Index="79" Value="7" />
    <Cell Index="80" Value="0" />
  </Data>
</Sudoku Puzzle>
```

Abbildung 10: Format des ersten Datenpakets

```
20706;Standard Sudoku;Hard;9;3;3
-FC---BE-
H---G----
A--H-E--G
--IB--D--
-B-----H-
--A--FI--
B--G-I--E
----A---B
-AG---HF-
GFCAIDBEH
HEBFGCADI
AIDHBEFCG
EHIBCADGF
CBFIDGEHA
DGAEHFIBC
BDHGFICAE
FCEDAHGIB
IAGCEBHFD
```

Abbildung 11: Format des zweiten Datenpakets

5.3.1 Datenaufbereitung für Statistik

Um Statistiken für die Analyse und Klassifizierung zu erstellen, gehen wir folgendermassen vor. Wir lesen die gewünschten Sudokus in unser Programm ein. Danach wird jedes Sudoku gelöst, wobei die verwendeten Lösungsmethoden für jedes Sudoku gespeichert werden. Ebenfalls werden die anderen benötigten Daten wie die Anzahl der gegebenen Ziffern, die Anzahl der möglichen Startpositionen und die Anzahl der Pencilmarks aus dem Sudoku ausgelesen.
Die Daten aller Sudokus werden danach Komma-getrennt in die gewünschte Datei geschrieben.

5.3.2 Datenanalyse

Vergleicht man die beiden Datenpakete, welche wir erhalten haben, fallen einige gröbere Unterschiede auf.
Der erste Datensatz enthält keine Sudokus der Stufe Very Hard und der zweite Datensatz keine Sudokus der Stufe Evil / Exotic. Ebenfalls fällt auf, dass im zweiten Datensatz bereits ab der Schwierigkeitsstufe Easy gewisse Sudokus nicht mehr mit unseren Lösungsmethoden gelöst werden können und auf Backtracking zurückgegriffen werden muss. Die Abbildungen 12 und 13 zeigen dies anschaulich auf.

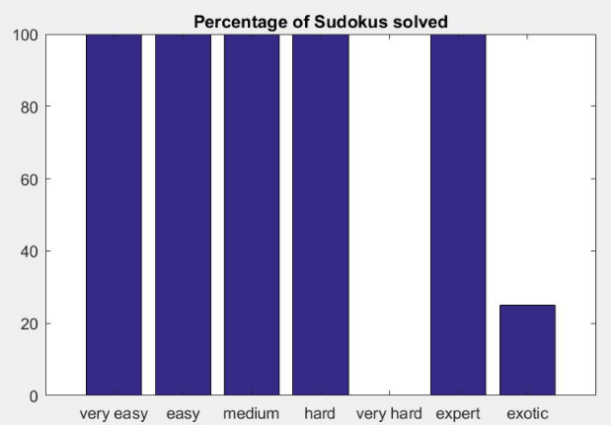

Abbildung 12: Datenpaket 1, prozentuale Anzahl gelöster Sudokus

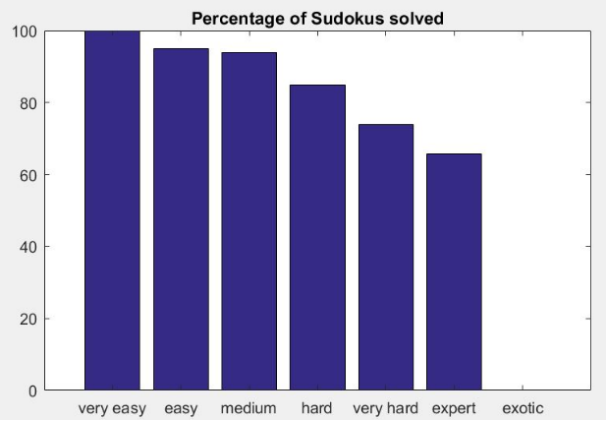

Abbildung 13: Datenpaket 2, prozentuale Anzahl gelöster Sudokus

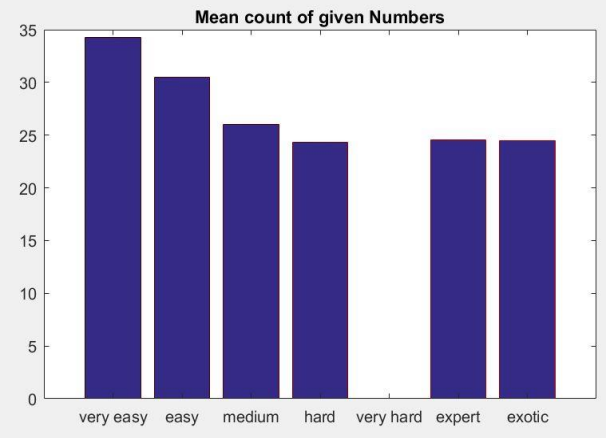

Abbildung 15: Datenpaket 1, durchschnittliche Anzahl vorgegebener Ziffern

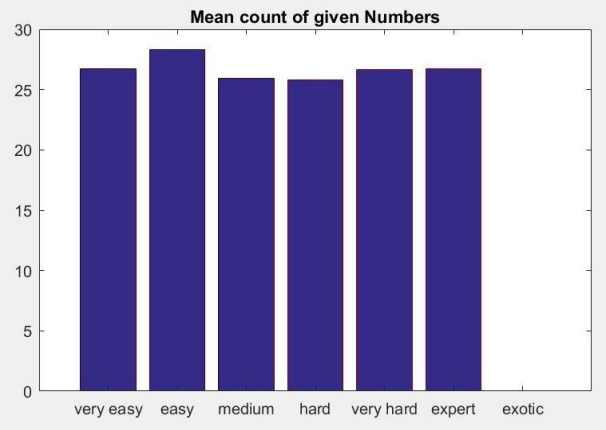

Abbildung 14: Datenpaket 2, durchschnittliche Anzahl vorgegebener Ziffern

Weiterhin sieht man auf Abbildung 15 gut, dass die durchschnittliche Anzahl der vorgegebenen Ziffern im ersten Datenpaket von Very Easy bis Hard abfällt. Im zweiten Datenpaket (Abbildung 14) sieht man jedoch gar keine solche Veränderungen.

Aufgrund dieser Ungereimtheiten, welche auch beim trainieren des Netzwerkes zu Schwierigkeiten führten, haben wir uns entschieden, nur mit dem ersten Datenpakt zu arbeiten.
Diese Entscheidung hat jedoch zur Folge, dass keine Sudokus der Schwierigkeitsstufe Very Hard generiert werden, da das erste Datenpaket keine solchen enthält.

5.4 Schwierigkeitseinstufung

5.4.1 Matlab

Die Daten, welche Matlab verwenden soll, müssen aus der generierten Statistik-Datei importiert werden. Weiter werden die importierten Daten zu unseren Features verarbeitet. Einige Features haben eine grosse Varianz, bei anderen Features unterscheiden sich die Wertebereiche stark. Um dies abzuschwächen werden die Features mittels Logarithmus in einen ähnlichen Wertebereich gebracht. Wir haben uns gegen eine Normalisierung entschieden, weil diese vom gesamten Datensatz abhängig wäre und wir aber die Möglichkeit bieten wollen, einzelne Sudokus zu klassifizieren.

Die Funktion zur Klassifizierung der Sudokus wurde in einer Anlernphase mit dem ersten Datenpaket mittels der Matlab-App «Neural Net Fitting» erstellt.
Es werden 80% der Sudokus verwendet, um diese Funktion zu erstellen.
In der App wird 5% als Validation und 10% als Test Set verwendet. Wir erstellen ein Netzwerk mit einem einzigen Hidden Layer, welcher aus 50 Hidden Neurons besteht. Als Trainingsalgorithmus haben wir Levenberg-Marquardt gewählt. Nach der Anlernphase führen wir mit den restlichen 20% der Sudokus einen eigenen manuellen Test durch, um die Genauigkeit zu überprüfen.

Die generierten Sudokus können mithilfe von Matlab klassifiziert werden. Sie müssen dafür wie zu Beginn des Kapitels beschrieben importiert werden.
Die generierten Sudokus werden danach vom trainierten Netzwerk klassifiziert. Die Zuordnung zwischen Sudoku und Schwierigkeitsstufe wird in einer Matrix (results) abgespeichert.

5.4.2 Neuroph

Es ist nicht benutzerfreundlich, wenn zur Klassifizierung der Sudokus immer eine Statistik-Datei generiert, danach Matlab gestartet und die Daten eingelesen werden müssen.
Aus diesem Grund haben wir unser neurales Netzwerk mithilfe des Frameworks Neuroph in Java abgebildet.
Da wir die genauen angelernten Werte des Netzwerkes nicht direkt aus Matlab übernehmen können, erstellen wir ein neues Netzwerk, welches auf den Erkenntnissen aus Matlab basiert.
Wir verwenden einen MultiLayerPerceptron, welcher auch aus einem Hidden Layer aus 50 Nodes besteht. Zum Trainieren des Netzwerkes werden 80% der Sudokus des ersten Datensatzes verwendet, zum Testen die restlichen 20%. Als Übergangsfunktion haben wir die Sigmoid-Funktion und als Lernalgorithmus MomentumBackpropagation verwendet.
Die Aufbereitung von der Statistik zum endgültigen Feature-Vektor wird auch komplett in Java durchgeführt.

5.5 Generierung

Für die Generierung eines Sudokus haben wir einen linearen Ablauf implementiert. Dabei wird ein Sudoku nach dem anderen generiert. Sudokus welche unser Programm nicht ohne Backtracking lösen kann werden verworfen, da sie der höchsten Schwierigkeit (Evil / Exotic) entsprechen würden, die von der Rätsel Agentur AG nicht verwendet werden. Bei der Generierung kann eingestellt werden, in welchem Bereich sich die Anzahl der gegebenen Startzahlen bewegen soll und wie viele Sudokus generiert werden sollen.

5.5.1 Basis

Um möglichst verschiedene Sudokus generieren zu können haben wir den Sudoku-Korpus von Gordon Royle als Basis verwendet. Dieser Korpus besteht aus einer Sammlung von Sudokus mit 17 Startpositionen und eindeutiger Lösung, die zudem unter den möglichen Transformationen verschieden sind. Das heisst, dass auch mithilfe der Transformationen kein 17er Sudoku aus dem Korpus auf ein anderes abgebildet werden kann. Der Korpus umfasst fast 50'000 solcher Sudokus.
Für die Generierung eines neuen Sudokus wird jeweils ein zufälliges Sudoku aus dem Korpus ausgewählt.

5.5.2 Transformieren

Das ausgewählte Sudoku aus dem Korpus wird anschliessend den verschiedenen Transformationen unterzogen. Alle Transformationen sind unter Zuhilfenahme eines Zufallsfaktors realisiert. Genauer gesagt werden die Permutationen für die einzelnen Transformationen zufällig gewählt. Beim Transponieren beschränkt sich der Zufallsfaktor auf die Entscheidung ob transponiert werden soll oder nicht.
Zuerst wird das Sudoku transponiert, danach werden die Zahlengruppen permutiert und zum Schluss werden die Zeilen, Spalten und Blocks vertauscht.

5.5.3 Ergänzen

Zum Schluss wird das transformierte Sudoku mit weiteren Zahlen ergänzt. Die Anzahl der wird zufällig ausgewählt, so dass die abschliessende Anzahl Startzahlen innerhalb des angegebenen Bereiches ist.
Für dieses Verfahren haben wir drei verschiedene Techniken implementiert:

- Technik 1: Zufällig Zahlen hinzufügen
  Alle Zellen, die im gegebenen Sudoku keinen Wert haben, in einer Liste sammeln. Diese Liste mischen und anschliessend entsprechend der gewünschten Anzahl minus die 17 Startzahlen Zahlenwerte hinzufügen. Dafür wird das Sudoku gelöst und die Zahl womit die Zelle gefüllt wird, wird aus der Lösung extrahiert.
- Technik 2: Gespiegelte Zahlen hinzufügen
  Für jede gegebene Zelle werden die (bis zu 7) gespiegelten Zellen extrahiert und in eine Liste gesammelt. Die Liste wird gemischt und wie bei der ersten Technik mit Zahlen aus der Lösung ergänzt.
- Technik 3: Zahlen entfernen
  Bei dieser Technik wird das 17er Sudoku zuerst gelöst. Von der Lösung werden anschliessend Zahlen entfernt, bis die gewünschte Anzahl erreicht ist. Dabei wird jeweils eine Zelle aus der Liste aller gesetzten Zellen ausgewählt. Ist die Zelle in der Mitte des Spielfeldes wird sie entfernt und eine neue Zelle wird gewählt. Ist die Zelle jedoch nicht in der Mitte so wird mithilfe eines Zufallswertes entschieden ob die Zelle und ihr punktgespiegeltes Gegenstück entfernt wird, oder nur die gewählte Zelle. Nach dem entfernen wird überprüft ob die gewünschte Anzahl erreicht wurde und falls nicht wird eine neue Zelle ausgewählt.

6 Resultate

6.1 Lösungsmethoden

Bei der Auswertung des Datenpaketes 1 (Abbildung 16) haben wir festgestellt, dass es eine klare Grenze zwischen den Schwierigkeitsstufen Medium und Hard existiert. Alle Sudokus, die als Medium oder tiefer eingestuft sind, können mit Naked und Hidden Singles komplett gelöst werden.
Weiterhin ist die Methode Block-Line Interactions im Datenpaket 1 nur von Sudokus der Stufe Evil / Exotic verwendet worden. Da jedoch Subsets der Grösse drei und höher vereinzelt in den Schwierigkeitsstufen Hard und Very Hard Expert verwendet wurden, stellt sich die Frage ob die Eingliederung der Block-Line Interactions zwischen den Subsets der Grösse zwei und drei richtig ist.

6.2 Schwierigkeitseinstufung

Die Rate, mit welcher beim Testen des neutralen Netzwerkes die Sudokus richtig eingeteilt werden zeigt auf, dass die Einstufung mittels menschlicher Lösungsmethoden ein guter Ansatz ist.

Abbildung 16 zeigt, dass die Schwierigkeitsstufen eine starke Auswirkung auf die durchschnittliche Anzahl der verwendeten Lösungsmethoden haben. Daraus folgt, dass die Lösungsmethoden auch für unsere generierten Sudokus ein guter Massstab für die Schwierigkeit sind.

Datenpaket 1	Naked Single	Hidden Single	Naked Subset (2)	Hidden Subset (2)	BlockLine-Interactions	Naked Subset (3)	Hidden Subset (3)
VeryEasy	46.49	0.25	0	0	0	0	0
Easy	49.32	1.17	0	0	0	0	0
Medium	47.58	7.43	0	0	0	0	0
Hard	42.68	13.99	1.35	0.27	0	0.01	0.02
VeryHardExpert	43.08	13.38	1.61	0.35	0	0	0.02
Evil/Exotic	13.91	12.485	1.64	0.805	0.165	0	0.13

Datenpaket 1	Naked Subset (4)	Hidden Subset (4)	X-Wing
VeryEasy	0	0	0
Easy	0	0	0
Medium	0	0	0
Hard	0	0	0
VeryHardExpert	0	0.01	0
Evil/Exotic	0	0.045	0.73

Abbildung 16: Durchschnittliche Anzahl Lösungsmethoden pro Schwierigkeitsstufe

6.3 Matlab und Neuroph

Wir haben uns für die Verwendung der Matlab App «Neural Net Fitting» entschieden, nachdem wir mehrmals viel Zeit damit verbracht hatten, unser Netzwerk an die verschiedenen Feature-Ideen anzupassen.

Diese App erleichtert das Feature Engineering und den daraus folgenden Umbau eines neuralen Netzes stark. Durch diese Applikation konnte bei der Anpassung der Netzwerke einiges an Aufwand eingespart werden.

Mit unserem Netzwerk erreichen wir auf dem ersten Datenpaket eine Genauigkeit von 75%. (Abbildung 17)

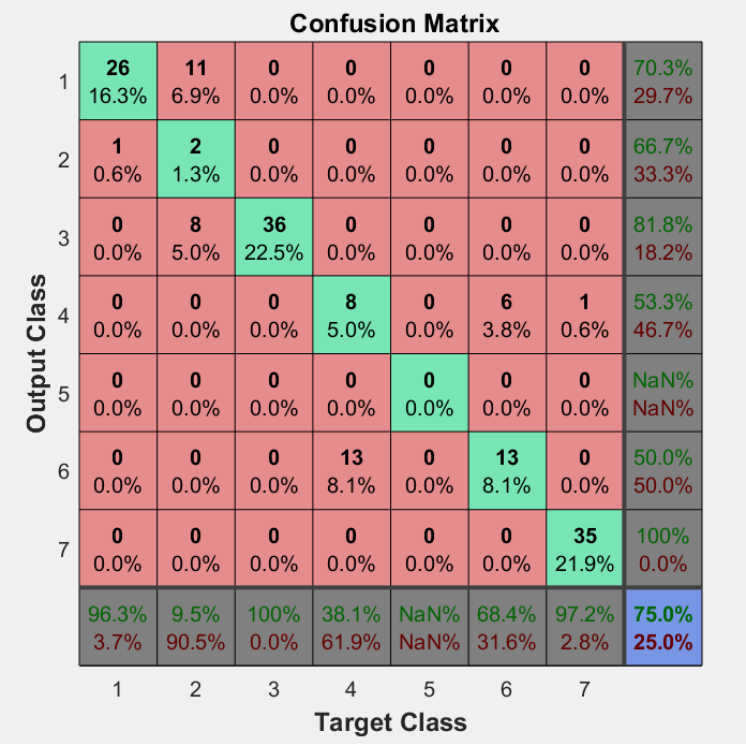

Abbildung 17: Konfusionsmatrix basierend auf erstem Datenpaket (Matlab)

Neuroph eignet sich gut, um ein einfaches neurales Netzwerk in Java zu implementieren, da sich schnell ein Netzwerk erstellen lässt. Dabei gilt es aber zu beachten, dass die genauere Konfiguration des Netzwerkes sehr komplex sein kann.
Das Netzwerk, welches wir in Java aufgebaut haben, erreicht aber nicht die Genauigkeit des Matlab-Netzwerkes. (Abbildung 18)
Zusätzlich variieren die Resultate in Neuroph, mit einer Genauigkeit von 61% - 66% und grösseren Abweichungen bei Fehleinschätzungen, stärker als in Matlab.

```
Result:
12      5       0       0       0       0       0
6       9       3       0       0       1       0
0       4       21      7       0       4       1
0       1       6       4       0       10      0
0       0       0       0       0       0       0
0       1       5       5       0       10      0
0       0       0       0       0       1       44
Total cases: 160.
Correctly predicted cases: 100.
Incorrectly predicted cases: 60.
Predicted correctly: 62.5%.
```

Abbildung 18: Konfusionsmatrix basierend auf erstem Datenpaket (Neuroph)

6.4 Generator

Die Resultate des Generators sind stark abhängig von der gewählten Technik um Zahlen zu ergänzen. Für die Tests wurden jeweils 2000 Sudokus generiert und anschliessend mit Neuroph einer Schwierigkeitsstufe zugeteilt. Zusätzlich wurde für jede Technik die Zeit gemessen, die gebraucht wurde um die Sudokus zu generieren.

Technik 1 generierte die 2000 Sudokus innerhalb 1454 Millisekunden. Die Verteilung der Schwierigkeitsstufen ist bei dieser Methode gleichmässig verteilt, wobei die mittleren Schwierigkeiten etwas häufiger generiert werden. Der Nachteil dieser Technik liegt daran, dass die Symmetrie des generierten Sudokus nicht beachtet wird.

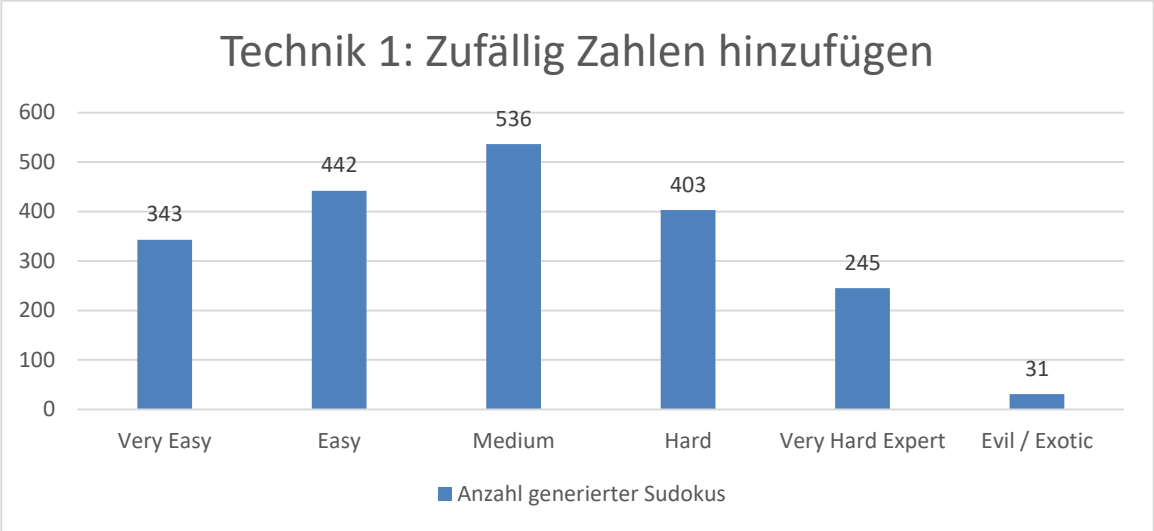

Abbildung 19: Verteilung der generierten Sudokus mit Technik 1

Technik 2 brauchte 1514 Millisekunden für die Resultate in Abbildung 20. Wie im Diagramm ersichtlich werden bei dieser Technik deutlich mehr Sudokus der Schwierigkeiten Medium und Hard generiert. Wie auch bei Technik 1 kann die Symmetrie der generierten Sudokus auch bei dieser Technik nicht garantiert werden, da die 17 Startzahlen keine Anforderungen an Symmetrie haben. Die Chance, dass ein Sudoku erstellt wird, das gewisse Symmetrien erfüllt ist jedoch höher als bei der ersten Technik.

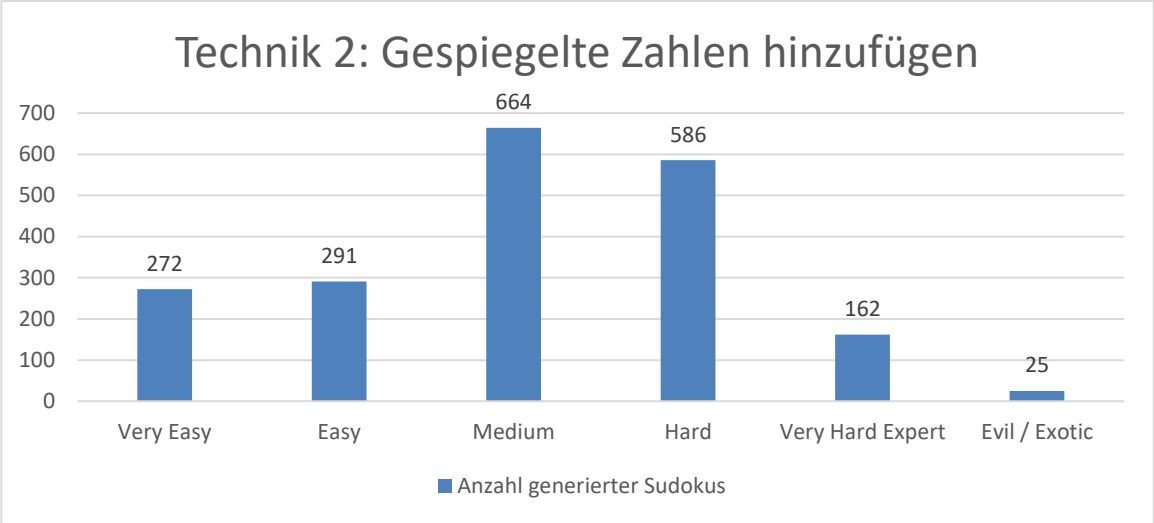

Abbildung 20: Verteilung der generierten Sudokus mit Technik 2

Technik 3 hat dadurch, dass die Zahlen in symmetrischen Paaren entfernt werden eine hohe Symmetrierate. Es werden jedoch fast ausschliesslich einfache Sudokus generiert. Zudem werden bei der Generierung viele Sudokus verworfen, weil sie unser Programm nicht lösen kann, dies zeigt sich auch in der langen Generierungszeit von 960 Sekunden für die 2000 Sudokus.

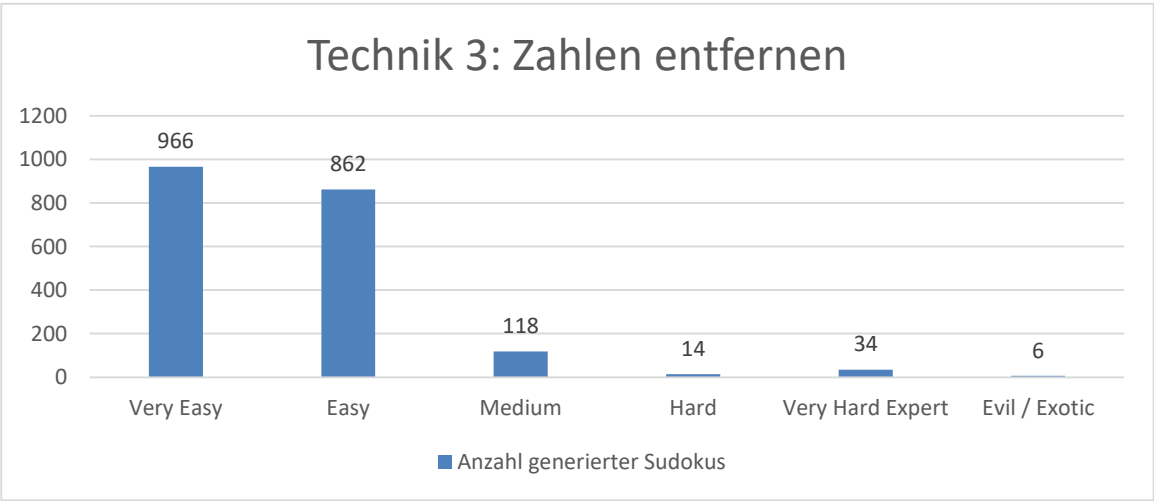

Abbildung 21: Verteilung der generierten Sudokus mit Technik 3

7 Erweiterungen

7.1 Verbesserung der Klassifizierung

Um bei der Klassifizierung bessere Resultate zu erzielen, können verschiedene Massnahmen ergriffen werden. Die Features, welche unser Modell ausmachen können weiter ausgearbeitet und falls nötig erweitert werden. Dies kann z.B. die Implementierung neuer Lösungsmethoden oder die Einbindung anderer Eigenheiten eines Sudokus sein.

Soll bei der Java Implementation die Genauigkeit erhöht und die Varianz der Resultate verringert werden, kann das Framework anders konfiguriert werden. Dies umfasst beispielsweise die Auswahl eines andern Lernalgorithmus oder einer anderen Übergangsfunktion.

7.2 Identifikation der Sudokus

Um Sudokus einfacher zu unterscheiden und zu identifizieren kann aus den Zellen des Sudokus ein Hashwert generiert werden, welcher dann zur Identifizierung der Sudokus verwendet wird.

7.3 Generierung gezielter Schwierigkeitsstufe

Sollen nur Sudokus einer bestimmten Schwierigkeit generiert werden, kann das Programm so erweitert werden, dass alle Sudokus, welche nicht der Schwierigkeit entsprechen wieder verworfen werden. Dadurch wird jedoch die Zeit, welche für das Generieren der Sudokus benötigt wird steigen, da insgesamt mehr Sudokus generiert werden müssen.

7.4 Zusatzinformationen in Zelle speichern

Soll die Performance der Updater Klasse erhöht werden, kann das Programm so erweitert werden, dass in jeder Zelle gespeichert wird, zu welcher Box, Spalte und Zeile sie gehört. Dies führt auch zu einer Vereinfachung der Implementation gewisser Lösungsmethoden.

7.5 Validierung der generierten Schwierigkeitsstufen

Um sicher zu gehen, dass die Einstufung der generierten Sudokus korrekt ist, gibt es mehrere mögliche Vorgehensweisen. Eine Idee ist eine zusätzliche Einordnung mit einem externen Klassifizierer, welcher bereits für die Einstufung von Sudokus verwendet wird.
Eine weitere Möglichkeit ist eine Feldstudie mit Personen, welche über möglichst unterschiedliche Sudoku-Kenntnisse verfügen. Diese Personen müssen einige Sudokus lösen und festhalten, wie schwer sie die gelösten Sudokus empfunden haben.

Mithilfe der auf diese Art klassifizierten Sudokus kann das neurale Netzwerk neu trainiert werden um die Genauigkeit zu erhöhen.

7.6 Visualisierung der generierten Sudokus

Um die Sudokus für Print- und Onlinemedien darzustellen muss eine Erweiterung für das Programm implementiert werden, welche aus dem Feldstring oder sogar direkt aus dem Board Objekt eine visuelle Darstellung des Sudokus generiert.

8 Schluss

Die Einteilung von Sudokus in bestimmte Schwierigkeitsstufen aufgrund menschlicher Lösungsmethoden ist ein guter Ansatz. Dabei muss jedoch notiert werden, dass die komplexeren Lösungsmethoden für unsere generierten Sudokus nur sehr selten verwendet werden. Dazu kommt, dass nicht nur die verwendeten Lösungsmethoden Einfluss auf die Klassifizierung von Sudokus haben.
Es ist sehr wichtig, dass die Klassifizierung der Testdaten von hoher Qualität ist, ansonsten folgt daraus eine schlechte Einteilung der neu generierten Sudokus. Ein grosses Trainingsset ist ebenfalls wichtig, wobei darauf geachtet werde muss, dass alle Sudokus vom selben Klassifizierer eingestuft wurden.
Die drei implementierten Generatormethoden weisen eine unterschiedliche Verteilung der Schwierigkeitsstufen der erstellten Sudokus auf. Weiterhin unterscheiden sich die generierten Sudokus im Symmetriegrad. So kann weiterführend die gewünschte Schwierigkeit und Symmetrie die Wahl des Generators beeinflussen.

Unsere Rolle in diesem Projekt ist nun abgeschlossen und das Projektziel der Implementation eines Sudoku Generators, welcher auf Basis von 17er Sudokus neue Rätsel verschiedener Schwierigkeiten erstellt, wurde erreicht.
Unser Projekt wird nun in das KTI-Projekt eingebunden und dort als Teil der gesamten Software weiterentwickelt.

9 Literaturverzeichnis

Smith, David (2005): So you thought Sudoku came from the Land of the Rising Sun ... https://www.theguardian.com/media/2005/may/15/pressandpublishing.usnews (abgerufen am 03.01.2017).

Royle, Gordon: Minimum Sudoku; University of Western Australia, Perth. http://staff-home.ecm.uwa.edu.au/~00013890/sudokumin.php (abgerufen am 19.10.2016).

Bräunlein, Michael (2014): Klassifikation der Schwierigkeitsgrade von Sudokus mit Methoden des maschinellen Lernens; Bachelor-Thesis, Technische Universität Darmstadt.

Stuart, Andrew C. (2007, Updated 2012): Sudoku Creation and Grading.

Urban, Wolfgang (2006): SUDOKU – Strategien zur Lösung; HIB Wien.

Ercsey-Ravasz, Mária; Toroczkai, Zoltán (2012): The Chaos Within Sudoku; Faculty of Physics, Babes-Bolyai University, Cluj-Napoca.

McGuire, Gary (2013): There is no 16-Clue Sudoku: Solving the Sudoku Minimum Number of Clues Problem via Hitting Set Enumeration; School of Mathematical Sciences, University College Dublin.

Neuroph, Java Neural Network Framework: http://neuroph.sourceforge.net/ (abgerufen am 19.10.2016).

10 Abbildungsverzeichnis

11 Ehrlichkeitserklärung

Hiermit erkläre ich, die vorliegende Projektarbeit selbständig, ohne Hilfe Dritter und nur unter Benutzung der angegebenen Quellen verfasst zu haben.

Brugg, Januar 2017

Matthias Keller

Simon Beck