Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

Hochschule für
Technik und Wirtschaft
Fachbereich 4

Dokumentation der Chess-AI

„Chessify“

Methoden der Wissensverarbeitungen

Projektarbeit

Projektteilnehmer
Tachmyrat Annayev
Hassan Alhawari,
Krisjanis Spridis,

Inhaltsverzeichnis

1. Einleitung

Unser Schach-KI-Projekt wurde mit dem Ziel entwickelt, eine fortschrittliche Schach-Engine zu erschaffen, die in der Lage ist, Schachspiele zu spielen. Wir haben hierfür ein leistungsstarkes Schach-Modell mithilfe von TensorFlow implementiert, das auf neuronalen Netzen basiert. Die Schach-Engine ermöglicht es uns, den aktuellen Spielzustand zu verfolgen, gültige Züge zu generieren und diese auf dem Schachbrett anzuzeigen. Dabei nutzt sie das Schach-Modell, das mit TensorFlow trainiert wurde, um fundierte Entscheidungen zu treffen und die besten Züge zu berechnen. Um die Interaktion mit unserer KI zu erleichtern, haben wir außerdem eine intuitive grafische Benutzeroberfläche entwickelt. Diese ermöglicht es Benutzern, mühelos gegen die KI zu spielen, Züge zu machen und den Spielverlauf zu verfolgen.

1.1. Architektur

Die Architektur des Chess-Bots wird durch vier Hauptdateien beschrieben. Die Datei main.py ist die Hauptklasse, die für die Verarbeitung der Benutzereingaben und die Anzeige des aktuellen GameState-Objekts verantwortlich ist. Die Funktionen zur Speicherung aller Informationen über den aktuellen Zustand des Schachspiels und zur Ermittlung gültiger Züge im aktuellen Zustand, einschließlich eines Zugprotokolls, sind in der Datei Engine.py enthalten. Für die künstliche Intelligenz, die auf TensorFlow basiert und die Schachzüge optimiert, haben wir das Modul Model.py entwickelt. Abschließend bietet die Datei ChessAI.py Funktionen zur Verbesserung der Züge mithilfe wichtiger Algorithmen.

Unser Schach-Bot wurde hauptsächlich in Python implementiert, wobei die Schachregeln und -logik sorgfältig mithilfe von Klassen und Methoden umgesetzt wurden. Zur Erstellung der grafischen Benutzeroberfläche haben wir eine externe Bibliothek verwendet.

2. Funktionen und Features

Die Schach-KI bietet eine Reihe leistungsstarker Funktionen, die ein anspruchsvolles und interaktives Spielerlebnis ermöglichen. Im Folgenden werden einige der herausragenden Features näher erläutert:

2.1. TensorFlow-basierte KI

Die Schach-KI basiert auf TensorFlow, einem leistungsstarken Tool für maschinelles Lernen. Ihr Modell verwendet Convolutional und Dense-Layer, die die Schachposition analysieren und optimierte Züge berechnen. Die Zugauswahl erfolgt mit Principal Variation Search (PVS), negamax und Alpha-Beta-Pruning. Die KI interagiert mit dem Stockfish-Schachmotor und bietet eine formelle und sachliche Umgebung für anspruchsvolles Schachspiel. Die genaue Dokumentation von Tensorflow befindet sich auf der offiziellen Seite unter folgendem Link: https://www.tensorflow.org/api_docs

2.1.1. Das TensorFlow-Modell erstellen:

Die Klasse Model aus der TensorFlow-Bibliothek wurde verwendet, um ein neuronales Netzwerk für die Schach-KI zu erstellen. Das Modell besteht aus mehreren Schichten von Convolutional und Dense-Layern, die dazu dienen, die Schachposition zu analysieren und eine Bewertung für den besten Zug zu erstellen.

2.1.2. Die Eingabeform des Modells:

Das Modell erwartet eine Eingabe mit der Form (8, 8, 16), die dem Schachbrett entspricht. Das Schachbrett wird in eine dreidimensionale Tensorstruktur umgewandelt, wobei die erste Dimension die Zeilen, die zweite Dimension die Spalten und die dritte Dimension die verschiedenen Kanäle repräsentiert. Insgesamt gibt es 16 Kanäle, die die verschiedenen Schachfiguren sowie die Zugmöglichkeiten für den nächsten Zug darstellen.

2.1.3. Das Modell trainieren:

Das Modell wird mit Schachdaten trainiert, um seine Gewichtungen zu optimieren und eine bessere Bewertung der Züge zu erlernen. Dazu werden Schachpositionen in Form von FEN-Strings in Tensorstrukturen umgewandelt und dazu passende Bewertungen als Trainingsdaten verwendet.

2.1.4. Zugauswahl:

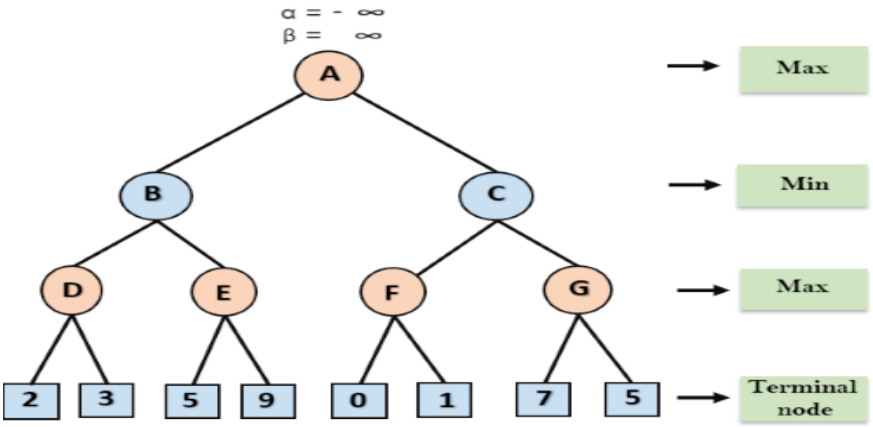
Die Klasse choose_move verwendet eine Variante des Principal Variation Search (PVS)-Algorithmus, um den besten Zug für die gegebene Schachposition auszuwählen. Dabei werden die negamax- und Alpha-Beta-Pruning-Techniken verwendet, um die Suche zu beschleunigen und effizientere Bewertungen zu erhalten.

2.1.5. Interaktion mit Stockfish:

Die Schach-KI kann gegen den Stockfish-Schachmotor spielen. Die KI wählt ihren Zug basierend auf dem trainierten TensorFlow-Modell aus, während Stockfish seine Züge mithilfe eines leistungsstarken Schachalgorithmus berechnet.

2.2. Optimierung des Modells

Die Schach-KI basiert auf dem NegaMax-Algorithmus mit Alpha-Beta-Pruning und hat den Hauptzweck, den besten Zug in einer gegebenen Schachposition zu ermitteln. Sie ermöglicht es dem KI-Spieler (Weiß) optimale Aktionen zu berechnen, indem verschiedene Züge analysiert und der beste Zug anhand der Minimax-Strategie und des Alpha-Beta-Pruning ausgewählt wird.



Tree Search von Alpha Beta Pruning, Quelle

2.2.1. NegaMax-Technik

Der NegaMax-Algorithmus ermöglicht es, Züge im Schachbaum effizient zu evaluieren, indem er die Bewertung der besten Aktion für den aktuellen Spieler mit der Bewertung der schlechtesten Aktion für den Gegner verknüpft. Dieser Ansatz

vereinfacht die Suche nach der besten Aktion und reduziert den Rechenaufwand im Vergleich zum Minimax-Algorithmus.

2.2.2. Alpha-Beta-Pruning

Das Alpha-Beta-Pruning hilft der Schach-KI, unnötige Berechnungen zu vermeiden und schneller zum besten Zug zu gelangen. Durch Verfolgung der besten Bewertungswerte für sich selbst und den Gegner überspringt die KI bestimmte Teile des Suchbaums, die keine Auswirkungen auf das Endergebnis haben. Dadurch spart sie Zeit und kann effizienter vielversprechende Züge evaluieren.

2.3. Spielerauswahl: Main-Funktion

Die main()-Funktion ist das Herzstück des Schachprogramms und ermöglicht verschiedene Spielmodi, in denen entweder zwei menschliche Spieler, zwei KIs oder ein Spieler gegen eine KI antreten können. Dies wird durch die Variablen playerOne und playerTwo gesteuert.

Die Variable playerOne entscheidet, ob der erste Spieler ein menschlicher Spieler (True) oder eine KI (False) ist. Ähnlich gilt für die Variable playerTwo, die den zweiten Spieler als menschlichen Spieler (True) oder KI (False) kennzeichnet.

- Wenn beide Variablen auf True gesetzt sind, spielen zwei menschliche Spieler gegeneinander.
- Wenn beide Variablen auf False gesetzt sind, treten zwei KIs gegeneinander an.
- Wenn playerOne auf True und playerTwo auf False gesetzt ist, spielt ein menschlicher Spieler gegen die KI.
- Wenn playerOne auf False und playerTwo auf True gesetzt ist, spielt die KI gegen den menschlichen Spieler.

Die main()-Funktion verarbeitet die Benutzereingaben und führt die Spielzüge aus, abhängig davon, welcher Spieler an der Reihe ist und ob es sich um einen menschlichen Spieler oder eine KI handelt. Sie ermöglicht so verschiedene Spielkonfigurationen und bietet ein flexibles Spielerlebnis, in dem Spieler die Möglichkeit haben, gegen menschliche Gegner oder KIs anzutreten, je nachdem, wie die Variablen playerOne und playerTwo eingestellt sind.

2.4. Züge validieren

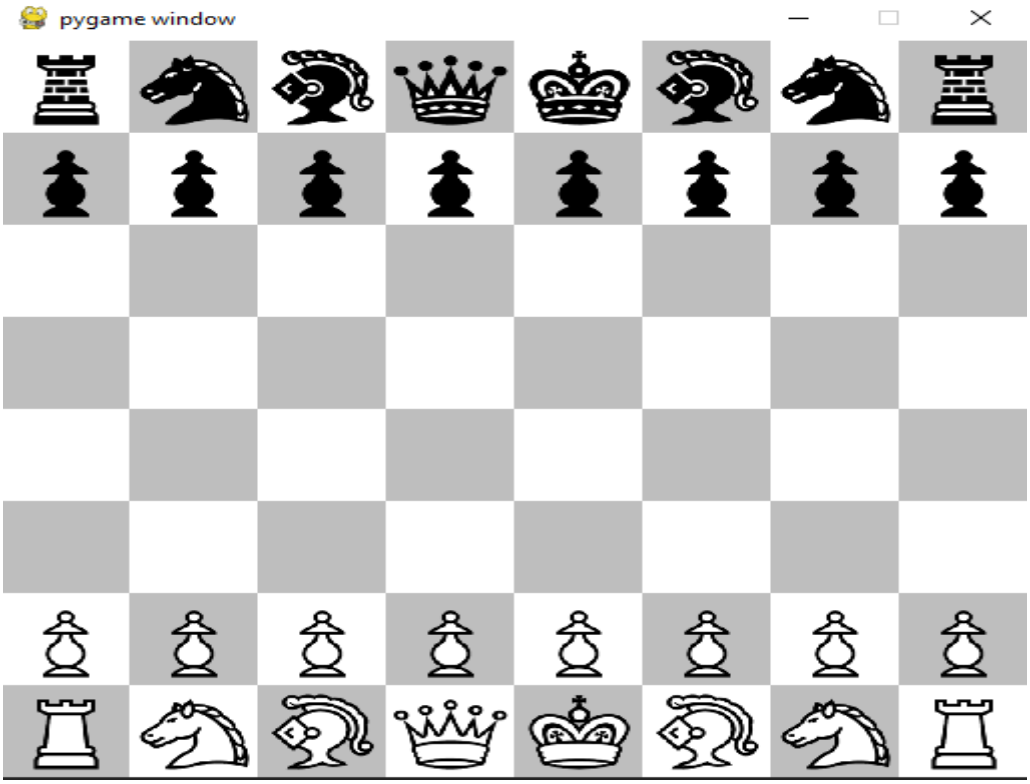
Die Engine-Klasse validiert Züge, indem sie alle möglichen Züge für den aktuellen Spieler ermittelt und dabei die grundlegenden Schachregeln beachtet. Dabei werden mögliche Bedrohungen (Pins) und Schachstellungen berücksichtigt, um ungültige Züge auszuschließen. Für jeden Zug wird überprüft, ob er den eigenen König im Schach stehen lassen würde. Wenn dies der Fall ist, wird der Zug als ungültig markiert und nicht zugelassen. Nachdem alle Züge ermittelt und validiert wurden, wird überprüft, ob der König des aktuellen Spielers im Schach steht. Wenn ja, wird geprüft, ob es noch gültige Züge gibt. Ist dies nicht der Fall, handelt es sich um ein Schachmatt. Wenn der König nicht im Schach steht und es keine gültigen Züge gibt, handelt es sich um ein Patt.Die Liste der gültigen Züge wird dann zurückgegeben, und sie kann verwendet werden, um die besten Züge für eine Schach-KI zu ermitteln oder den Spieler bei einem Schachspiel zu führen.

Die Kernfunktionen der Klasse sind:

- **makeMove**(move): Führt einen Zug aus, indem die Schachfigur von der Startposition zur Endposition bewegt wird. Dabei werden die Schachregeln berücksichtigt, einschließlich der Rochade, des En-passant-Zugs und der Umwandlung eines Bauern in eine andere Schachfigur, wenn dieser die gegnerische Grundreihe erreicht.
- **undoMove**(): Macht den letzten Zug rückgängig, um das Schachbrett in den vorherigen Zustand zurückzusetzen.
- **getValidMoves**(): Ermittelt alle möglichen gültigen Züge für den aktuellen Spieler unter Berücksichtigung der Schachregeln. Dabei werden auch Bedrohungen (Pins) und Schach überprüft.
- **checkForPinsAndChecks**(): Überprüft, ob der König im Schach steht oder ob es Bedrohungen (Pins) gibt, bei denen ein Stück den König schützt.
- **isInCheck**(): Prüft, ob der König im Schach steht, indem überprüft wird, ob das Königsfeld von einem gegnerischen Stück bedroht wird.

3. Schach-KI: Die graphische Oberfläche

Die graphische Benutzeroberfläche (GUI) dieses Schachspiels wird mithilfe der Pygame-Bibliothek erstellt. Pygame ist eine leistungsstarke Python-Bibliothek, die häufig für die Entwicklung von 2D-Spielen und Anwendungen verwendet wird. Die GUI verwendet ein Tile-based Grid-Design, bei dem das Schachbrett in ein festes Raster aus quadratischen Zellen (Tiles) unterteilt wird. Jede Zelle repräsentiert eine Position auf dem Schachbrett, und die Schachfiguren werden innerhalb dieser Zellen platziert, um ihre aktuellen Positionen darzustellen. Die Schachfiguren werden als Bilder geladen und in einem globalen Dictionary IMAGES gespeichert. Die GUI reagiert auf Benutzerinteraktionen, wie Mausklicks und Tastatureingaben, um die Schachzüge zu steuern. Bei einem Mausklick auf eine Schachfigur oder ein Schachfeld werden die entsprechenden Aktionen ausgelöst. Die Funktionen drawBoard() und drawPieces() zeichnen das Schachbrett und die Schachfiguren auf dem Bildschirm. highlightSquares() hebt ausgewählte Zellen und mögliche Züge hervor, während drawGameState() den aktuellen Spielzustand aktualisiert. drawText() wird verwendet, um Textnachrichten zu präsentieren.



Screenshot der graphischen Oberfläche – Quelle: Chessify-GUI

3.1. Grid-Koordinaten

Bei jedem Mausklick auf das Schachbrett wird die Position des Klicks in Pixelkoordinaten erfasst. Diese Koordinaten werden dann in Zeilen- und Spaltenkoordinaten des Schachbretts umgerechnet, um die Schachbrettzelle zu identifizieren, auf die der Benutzer geklickt hat. Beim Klicken auf eine Schachfigur wird die Position der entsprechenden Zelle, in der sich die Figur befindet, als Startkoordinate gespeichert. Klickt der Benutzer danach auf ein anderes Schachfeld, wird die Position dieses Feldes als Zielkoordinate gespeichert. Die Start- und Zielkoordinaten werden verwendet, um einen Schachzug zu erstellen und zu überprüfen, ob er gültig ist. Wenn der Zug gültig ist, wird er ausgeführt, und die Schachfigur wird entsprechend animiert, um sich von der Startposition zur Zielposition zu bewegen.

4. Benutzerhandbuch

Bevor man das Schach-Programm nutzen kannst, ist es wichtig sicherzustellen, dass Python 3.0 oder eine neuere Version auf deinem Computer installiert ist. Python wird als Programmiersprache für das Schach-Programm verwendet und ist erforderlich, um das Programm auszuführen. Sollte Python noch nicht auf deinem Computer sein, kannst man es einfach von der offiziellen Python-Website herunterladen und den Installationsanweisungen folgen.

4.1. Installation der benötigten Packages:

Damit das Schach-Programm einwandfrei funktioniert, benötigt es zusätzliche Python-Pakete. Die folgenden Packages sind erforderlich:

- pygame: Dieses Package ist eine Bibliothek, die die Erstellung von Spielen und grafischen Anwendungen vereinfacht. Es ermöglicht dem Schach-Programm, das Schachbrett und die Schachfiguren in einer visuell ansprechenden Benutzeroberfläche darzustellen.
- numpy: Dieses Package ist eine leistungsstarke Bibliothek für numerische Berechnungen und die Verarbeitung von Arrays. Numpy wird vom Schach-Programm verwendet, um das Schachbrett und die Spielzustände effizient zu verwalten.
- pandas: Dieses Package ist eine umfangreiche Bibliothek zur Datenanalyse und Datenmanipulation. Es organisiert Daten in tabellarischer Form und wird

vom Schach-Programm genutzt, um das Zuggesuchsprotokoll und andere Informationen zum Spielstatus zu verwalten.

- tensorflow: Dieses Package ist eine leistungsstarke Open-Source-Bibliothek für maschinelles Lernen und neuronale Netzwerke. Es bildet die Grundlage für die künstliche Intelligenz des Schach-Programms und optimiert die Schachzüge.

Um diese Packages zu installieren, öffne einfach die Kommandozeile oder das Terminal und gib die folgenden Befehle ein:

Pip install pygames; pip install numpy; pip install pandas; pip install tensorflow

4.2. Start des Schach-Programms:

Nachdem du Python und die benötigten Packages erfolgreich installiert hast, navigiere zum Verzeichnis, in dem sich das Schach-Programm befindet. Öffne dann die Kommandozeile oder das Terminal und führe die Datei "Main.py" aus, um das Schach-Programm zu starten. Du kannst dazu den folgenden Befehl verwenden:

python Main.py

4.3. Benutzeroberfläche:

Sobald das Schach-Programm gestartet ist, wird die grafische Benutzeroberfläche angezeigt. Hier siehst du das Schachbrett mit den Schachfiguren. Du kannst Züge auf dem Brett auswählen, indem du einfach mit der Maus auf die Startposition der gewünschten Schachfigur klickst und dann auf das Zielfeld ziehst.

4.4. Spielablauf:

Der Spielablauf ist einfach und intuitiv. Wähle die gewünschte Schachfigur aus, indem du auf ihre Startposition klickst, und wähle dann das Zielfeld aus, um den Zug auszuführen. Das Schach-Programm überprüft automatisch die Gültigkeit des Zugs und aktualisiert den Spielzustand.

4.5. Spielstatus:

Die Benutzeroberfläche zeigt Informationen zum Spielstatus an, darunter den aktuellen Spieler, ob sich das Spiel in einer Schach- oder Schachmatt-Situation befindet, sowie das Zuggesuchsprotokoll, das die bisherigen Züge und deren Reihenfolge enthält.

Fazit

Unser Schach-Bot ist ein beeindruckendes Projekt, auf das wir stolz sein können. Die intuitive Benutzeroberfläche, die herausfordernde künstliche Intelligenz und das modulare Design machen es zu einem unterhaltsamen und ansprechenden Schachspiel für Spieler aller Niveaus.

Dennoch bleibt Raum für weitere Verbesserungen und Weiterentwicklungen. Durch das offene Ohr für das Feedback der Nutzer und die Bereitschaft, auf ihre Bedürfnisse einzugehen, können wir das Projekt kontinuierlich optimieren..

Insgesamt ist unser Schach-Bot ein Erfolg, der zeigt, wie die Verbindung von Programmierung und Gaming zu einem ansprechenden und lehrreichen Spielerlebnis führen kann. Wir sind zuversichtlich, dass das Projekt weiterhin wachsen und verbessert werden kann, um Spieler auf der ganzen Welt zu begeistern und die Freude am Schachspiel zu fördern.