Dieses Dokument ergänzt das Dokument „Start_Up_Tutorial.pdf“ der Vorarbeit. Die Thesis dieser Arbeit ist im GitHub-Ordner enthalten. Für Nachfolgende Arbeiten empfiehlt es sich gegebenenfalls diese Thesis und die Thesis der Vorarbeit zu lesen. Der Code ist kommentiert, um die Veränderungen gegenüber der Vorversion kenntlich zu machen und zu erklären. Für eine genaue Übersicht über alle Code-Änderungen empfiehlt es sich bei Bedarf z.B. in Visual Studio Code über einen File-Vergleich die Verschiedenen Versionen gegenüberzustellen.

Es empfiehlt sich genau nach der Anleitung der Vorarbeit vorzugehen, inklusive der Verwendung der gleichen Python-Version. Das Vorgehen nach dem Dokument „Start_Up_Tutorial.pdf“ der Vorarbeit bis einschließlich „Step 2“ sollte vor den Zusätzen in dieser Anleitung durchgeführt werden. Das im Folgenden beschriebene Vorgehen wird nach bestem Wissen und Gewissen beschrieben, Fehler können jedoch dennoch nicht gänzlich ausgeschlossen werden. Für das Coden wurde in dieser Arbeit Visual Studio Code mit Python-Extensions verwendet.

Abschnitt 0: Hauptunterschiede zur Vorversion

Relevante Dateien die gegenüber der Vorarbeit geändert wurden:

- in „\Implementation\files“: „journeys.xlsx“ enthält alle verwendeten Journeys für Training und Evaluierung. Die Auswahl der Journeys wird beim Training über den Code selbst gesteuert. Bei der Evaluierung werden die Journey-IDs 110, 111, 112, 113, 114, 115, 116, 117, 118 und 119 verwendet. Diese entsprechen in der Datei den Zeilen 112 bis 121. Diese Version von „journeys.xlsx“ ist zusätzlich im Ordner „journeysFinalOptuna“ gesichert. Diese Beschreibung gilt genauso für den Ordner „\ImplementationOhneExpert\files“.
- Die Code-Änderungen dieser Arbeit wurden fast ausschließlich in den Python-Files „main.py“ und „train_env.py“ vorgenommen. Das gilt sowohl für den Ordner „Implementation“ als auch für den Ordner „ImplementationOhneExpert“.
- Im Ordner „\Implementation\models“ sind die Dateien der Optuna Tunings und deren TensorBoard-Logs, sowie die erfolgreichsten Agents und deren Evaluierungen enthalten. Dabei ist zu beachten, dass auch bei identischem Seed auf anderen Computern abweichende Ergebnisse erzielt werden, falls versucht wird die Ergebnisse dieser Arbeit exakt zu reproduzieren.
- Im Ordner „\ImplementationOhneExpert \models“ sind beispielhafte Trainings- und Evaluierungsdaten für den QRDQN enthalten.

Um von der Code-Version mit Expert auf die Code-Version ohne Expert zu wechseln kann der Ordner „Implementation“ einfach z.B. in „ImplementationMitExpert“ umbenannt werden und der Ordner „ImplementationOhneExpert“ in „Implementation“. Der aktuelle Code ist immer für einen Ordner mit dem Namen „Implementation“ ausgelegt. Um von der Code-Version ohne Expert auf die Code-

Version mit Expert zurück zu wechseln kann dementsprechend umgekehrt vorgegangen werden.

Abschnitt 1: Neue Python-Bibliotheken

Nachfolgend sind die Installations-Kommandos für die im Zuge dieser Arbeit ergänzten Bibliotheken aufgeführt. Diese müssen in einem Terminal mit aktiviertem „virtual environment“ (siehe Start_Up_Tutorial.pdf der Vorarbeit) ausgeführt werden, um die Bibliotheken dem „virtual environment“ hinzuzufügen. Die pip-Kommandos können aus dieser Anleitung in das Terminal kopiert werden.

Bei Bibliotheken-Gruppe 1 handelt es sich um PyTorch-Versionen mit „cuda“, also mit Unterstützung für Berechnungen für die neuronalen Netzwerke mittels der Grafikkarte. Voraussetzung dafür ist eine „cuda“-fähige Grafikkarte. Eine Liste der „cuda“-fähigen Grafikkarten ist im Internet zu finden (Nvidia). Sollte keine derartige Grafikkarte verfügbar sein wird durch einen Code-Anteil der Vorarbeit automatisch eine Berechnung mittels der CPU vorgesehen.

Bei Bibliothek 2 handelt es sich um Stable-Baselines3 Contrib, also eine Bibliothek mit RL-Algortihmen.

Bei Bibliothek 3 handelt es sich um Optuna, also die bei dieser Arbeit verwendete Hyperparameter-Tuning-Bibliothek.

Bei Bibliothek 4 handelt es sich um Optuna Dashboard, also die Visualisierungs-Bibliothek für die Optuna-Tunings.

Installations-Kommandos:

1:

pip install torch==1.13.0+cu117 torchvision==0.14.0+cu117 torchaudio==0.13.0 --extra-index-url https://download.pytorch.org/whl/cu117

2:

pip install sb3-contrib==1.8.0

3:

pip install optuna

4:

pip install optuna-dashboard

Folgende Quellen sind für die neuen Bibliotheken (und verwendeten Algorithmen) gegebenenfalls hilfreich:

- https://stable-baselines3.readthedocs.io/en/master/index.html#
- https://sb3-contrib.readthedocs.io/en/master/index.html#
- https://github.com/optuna/optuna
- https://optuna.org/#code_examples

Abschnitt 2: Neue Argumente beim Ausführen des Programms über das Terminal

Argument, Kurzform	Beschreibung	Erwartete Werte
-algo, -a	angepasst gegenüber der Vorarbeit: „Which algorithm to use, either in training or evaluation. Should always be provided, even during evaluation.“	Für Version mit Expert: PPOmanualNN, DDQN, PPO, DQN, A2C, RecurrentPPO, QRDQN, TRPO Für Version ohne Expert: PPO, DQN, A2C, RecurrentPPO, QRDQN, TRPO
-hyperparameter_tuning, -ht	Hyperparameter-Tuning-Modus aktivieren.	-
-hyperparameter_tuning_path, -htp	Pfad zum Dateispeicherort, an dem die Hyperparameter-Tuning-Daten gespeichert werden sollen.	Beispiel: .\models\QRDQN_Agents\Testhyperparameters

Beispiele:

Das grundlegende Vorgehen inklusive Navigation zum Ordner „Implementation“ ist im „Start_Up_Tutorial.pdf“ der Vorarbeit beschrieben.

QRDQN-Training und Evaluierung auf einer durch die ID gewählten Journey:

mkdir .\models\QRDQN_Agents\Test

python.exe .\main.py -a QRDQN -t -tp .\models\QRDQN_Agents\Test

python.exe .\main.py -a QRDQN -ep .\models\QRDQN_Agents\Test -an agent4 -jm 2 -id 110

QRDQN-Hyperparameter-Tuning:

mkdir .\models\QRDQN_Agents\Testhyperparameters

python.exe .\main.py -a QRDQN -t -ht -htp .\models\QRDQN_Agents\
Testhyperparameters

In jeweils einem zusätzlichen Terminal können bei Bedarf folgende Kommandos verwendet werden, um Tensor-Board oder Optuna Dashboard zu starten. Diese sind noch entsprechend dem verwendeten Ordner jeweils anzupassen. Der im Web-Browser verwendbare Pfad wird jeweils im Terminal ausgegeben:

tensorboard.exe --logdir .\models\QRDQN_Agents

optuna-dashboard
sqlite:///.\models\QRDQN_Agents\Testhyperparameters\studyQRDQN.db

Abschnitt 3: Evaluierungs-Dateien

Bei der Auswertung von den bei der Evaluierung erstellten „.csv“-Dateien kann folgendermaßen vorgegangen werden (gegebenenfalls ergeben sich Abweichungen je nach verwendeter Excel-Version):

1. Datei Öffnen.
2. Datei speichern und bei der Meldung „Nein“ drücken. Dann ohne weitere Änderungen „Speichern“ drücken, um die Datei als „.xlsx“-Datei abzuspeichern.
3. Markieren der Spalte „A“.
4. In der oberen Leiste im Menü „Daten“ den Menüpunkt „Text in Spalten“ auswählen.
5. Als ursprünglichen Datentyp „Getrennt“ auswählen und „Weiter >“ drücken.
6. Bei „Trennzeichen“ die Option „Tabstopp“ abwählen und die Option „Komma“ wählen und „Weiter >“ drücken.
7. Auf der rechten oberen Seite „Weitere…“ anklicken und dort bei „Dezimaltrennzeichen“ von „,“ auf „.“ wechseln und bei „1000er-Trennzeichen“ von „.“ auf „,“ wechseln. Auf „OK“ drücken und dann „Fertig stellen“ drücken. Für die Titel der jeweiligen Spalten wird jeweils auf den Code der Datei-Erstellung verwiesen.