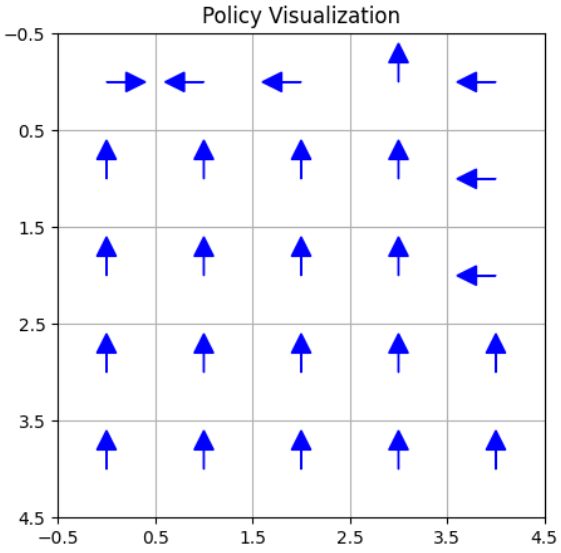
Deep Reinforcement Learning

Sheet 04

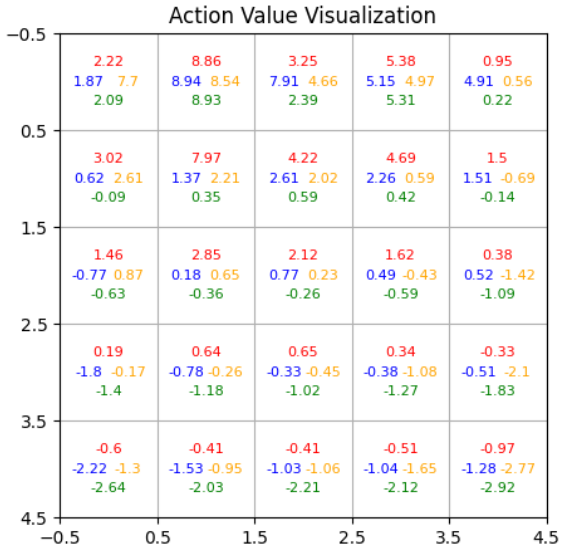
Sven Ullmann, Valentin Adam, Moritz Grunert

Exercise 1

(a)



(a) Optimal Policy computed with Off-Policy MC control algorithm.



(b) Computed Action Value Function Q.

Bei dem Off Policy Monte Carlo Control Ansatz betrachtet man eine Target Policy die wirklich optimiert wird und eine Hilfs-Behaviour Policy, die während des Algorithmusses nicht verbessert wird. Die Target Policy wird nie wirklich ausgeführt. Um nun eine Konvergenz/Verbesserung des während einer Episode gesammelten Rewards zu erhalten, mussten wir in jeder Episode die Target Policy noch einmal extra durchgehen (was in dem ursprünglichen Algorithmus gar nicht von Nöten ist). In Bild (a) sieht man die Optimale Target Policy. Diese führt immer wieder zu den Zuständen die einen positiven Reward ausspucken. Die Konvergenz zu dieser optimaen Policy sieht man in Bild (2). Man erkennt deutlich eine Konvergenz zur optimalen Policy mit Reward 100. Zu Beginn müssen erst genug zufällige Monte Carlo Episoden durchlaufen um die Action Values zu updaten und häufig genug der direkte Weg zu den guten States ausgewählt werden, damit die Target Policy entsprechend ausgewertet werden kann.

Unsere Implementierung von der Off Policy Monte Carlo Algorithmus ohne importance Sampling ergibt keine sinnvolle Konvergenz zu irgendeiner Policy (3) erkennt man sehr gut, dass die

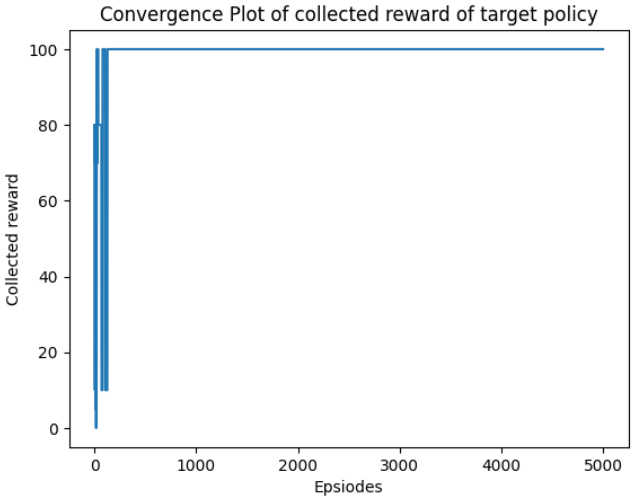


Figure 2: Convergence of the target policy.

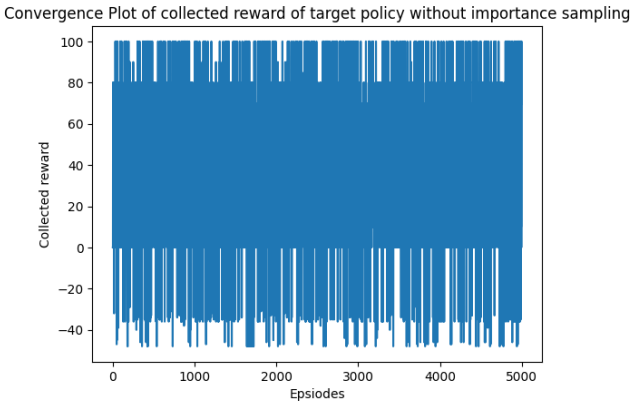
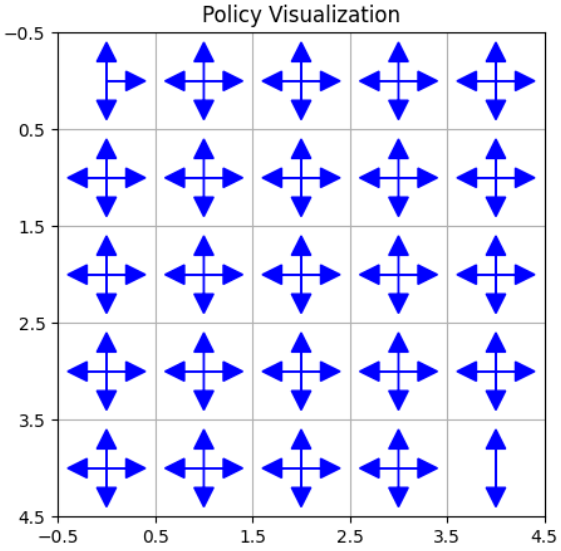


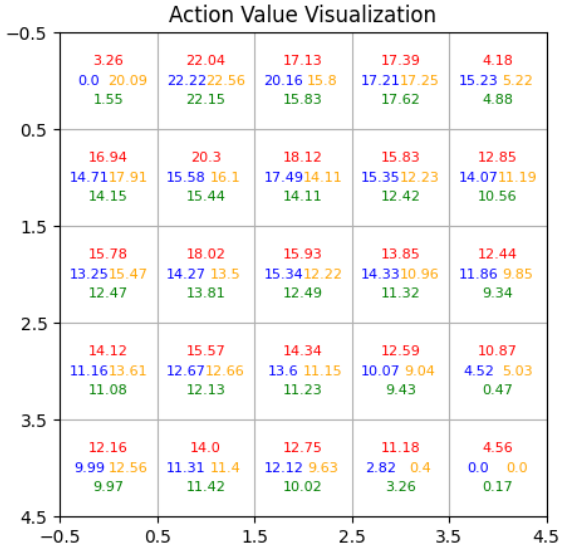
Figure 3: Caption

(b)

Unsere Implementierung des n-step SARSA scheint nicht richtig zu sein. Für keine sinnvollen n erhalten wir eine gute resultierende Policy. Dementsprechend können wir das Experiment nicht machen. Vgl. Bilder unten.



(a) Computed policy for $n = 1$ of n-step SARSA.

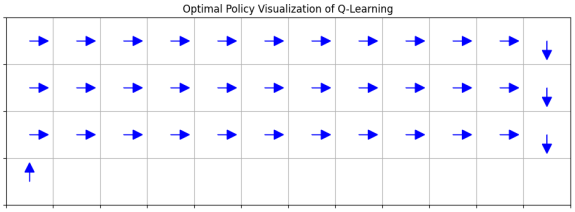


(b) Computed Q-values for $n = 1$ of n-step SARSA.

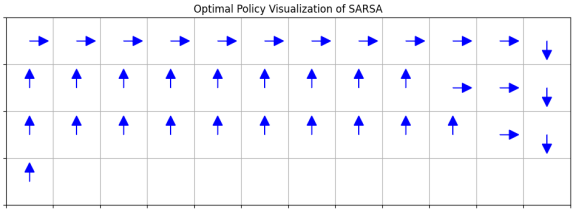
Exercise 2

a) Visualisierung und Reflexion

Die plots zeigen die optimalen Policies, welche wir mit SARSA und Q-learning erhalten haben



(a) Optimal Policy computed with Q-learning Algorithm.

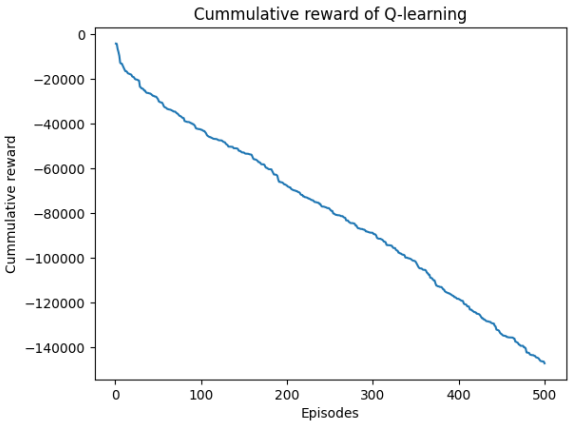


(b) Optimal Policy computed with SARSA-Algorithm.

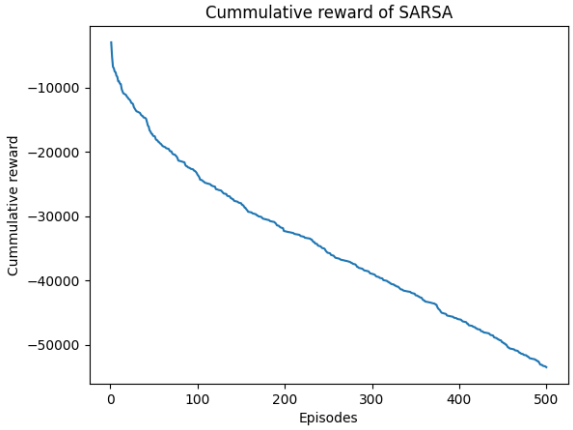
1) **Vergleich der Vor- und Nachteile von On & Off-Policy approaches**:

Bei On-Policy-Methoden, wie z. B. SARSA, wird die Policy, die Aktionen auswählt, direkt trainiert und evaluiert. Es handelt sich also um die gleiche Policy, die sowohl zum Exploren der Umgebung als auch zum Optimieren genutzt wird. Ein Vorteil von On-Policy ist ein realistischeres Verhalten, da die Policy, die optimiert wird, auch tatsächlich in der Umgebung verwendet wird. Zusätzlich neigen On-Policy approaches dazu, sicherere Entscheidungen zu treffen (z. B. Vermeidung riskanter Pfade). Nachteil von On-Policy: Oft langsamer bei der Konvergenz, da es nicht die theoretisch optimalen Aktionen verfolgt, sondern von der aktuellen Policy eingeschränkt wird.

Bei Off-Policy-Methoden, wie z. B. Q-Learning, wird die Policy, die zum Erkunden der Umgebung verwendet wird, von der Policy, die optimiert wird, getrennt. Man trainiert die optimale Policy basierend auf theoretisch besten Aktionen. Der große Vorteil ist, dass Off-Policy approaches theoretisch bessere Ergebnisse erzielen, da sie sich nicht auf die aktuell explorierte Policy beschränken. Dadurch ergibt sich aber auch der Nachteil, nämlich die erhöhte Instabilität durch mögliche Inkonsistenzen zwischen explorierter und trainierter Policy. Außerdem neigen Off-Policy approaches zu riskanten Aktionen.



(a) Cummulative reward computed with Q-learning Algorithm.



(b) Cummulative reward computed with SARSA-Algorithm.

Szenarien:

On-Policy bevorzugt: In sicherheitskritischen Umgebungen, wie z. B. autonome Fahrzeuge, bei denen risikoreiche Aktionen vermieden werden sollen.

Off-Policy bevorzugt: In simulierten oder virtuellen Szenarien, bei denen das Ziel ist, so schnell wie möglich eine optimale Policy zu finden, ohne Rücksicht auf Zwischenfehler (z. B. ein Computerspiel-Training).

(2) **SARSA vs. Q-Learning: Sichere vs. optimale Pfade**

SARSA berücksichtigt die aktuelle Policy und wählt basierend darauf die nächste Aktion. Dadurch passt SARSA seine Entscheidungen an die aktuelle Umgebung an, was bedeutet, dass es tendenziell vorsichtiger agiert. Beispiel: In einer Umgebung wie dem Cliff-Walking-Problem (wo ein Absturz an der Klippe bestraft wird), tendiert SARSA dazu, Pfade zu wählen, die ein Abstürzen vermeiden, auch wenn sie nicht den optimalen Pfad darstellen.

Q-Learning basiert auf einer greedy Policy, die nach der besten (theoretischen) Belohnung strebt, unabhängig davon, ob diese Aktion sicher ist oder nicht. Beispiel: Im Cliff-Walking-Problem könnte Q-Learning riskantere Pfade wählen, da es den optimalen Pfad entlang der Klippe favorisiert, was während der Exploration jedoch zu Abstürzen führen kann. Warum stürzt Q-Learning "von der Klippe"? Während der Exploration probiert Q-Learning Aktionen aus, die nicht immer mit der besten langfristigen Strategie übereinstimmen. Es kann dabei riskante Pfade erkunden (wie nahe an der Klippe), was zu hohen Strafen führt, bevor die optimale Policy vollständig gelernt wurde.

(3) **Niedrigere Gesamtbelohnungen pro Episode bei Q-Learning im Vergleich zu SARSA:**

Q-Learning sammelt niedrigere Gesamtbelohnungen, da es im Vergleich zu SARSA mit seiner Behaviour Policy den optimalen (aber eben auch riskanteren Pfad) sucht. Dies sieht man auch im plot der cummulative Rewards for Q-Learning: Hier ist der Reward wesentlich negativer als bei SARSA, denn der Agent stürzt öfters die Klippe herunter.

(4) **Rein greedy in Q-Learning:**

Nein, sie sind nicht identisch, aber sehr ähnlich. Sie wählen nicht exakt die gleichen Aktionen

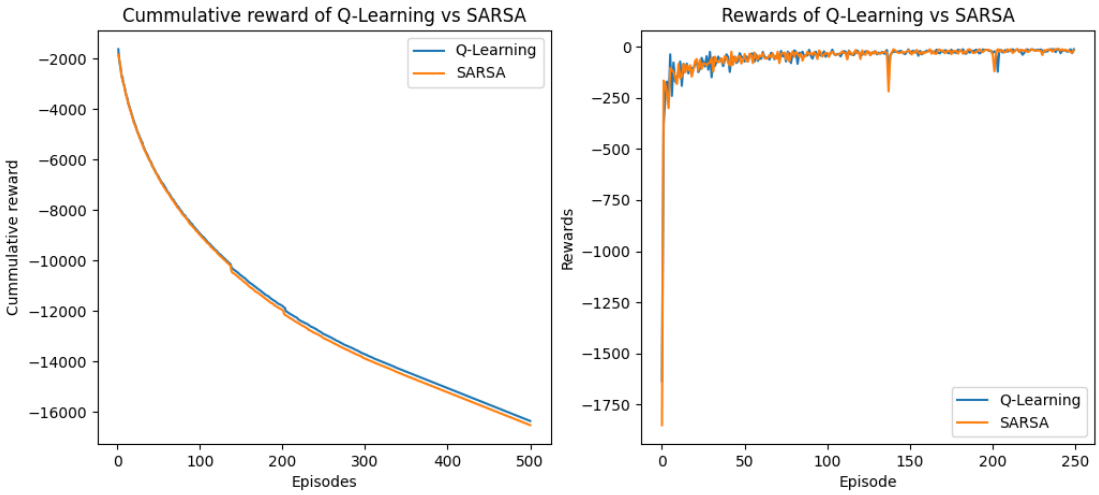
(b) Explore vs Exploit

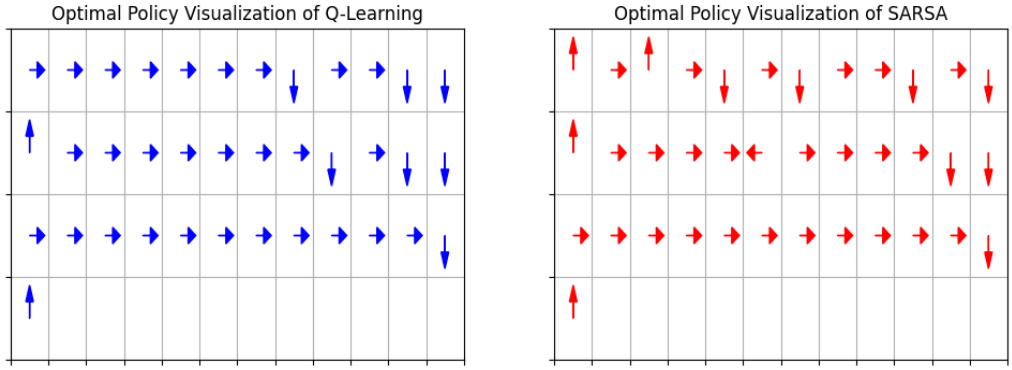
1) **Einfluss von Epsilon**:

ε	Q-Learning	SARSA
0.8	12659347.29	717446.21
0.5	504899.17	8439.84
0.2	19158.33	1170.39
0.05	2217.61	407.38
0.0	112.45	134.04

Table 1: Variance of rewards

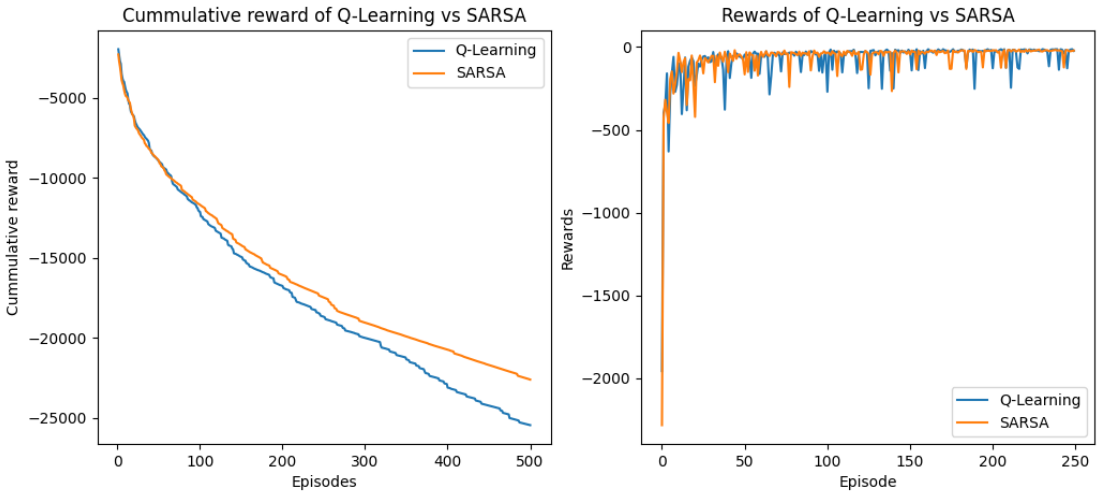
$\varepsilon = 0$**:**
SARSA policy ähnelt der Q-Learning policy. Ein niedriger ε-Wert bedeutet, dass wenig erkundet wird, somit ist die Belohnung und die Varianz niedrig, da fast nur die beste Aktion gewählt wird und der Agent somit fast gar nicht von der Klippe fällt. Aus dem Grund ähnelt SARSA auch Q-Learning, da der Agent nicht lernt, dass der Weg nahe der Klippe "gefährlich" ist.

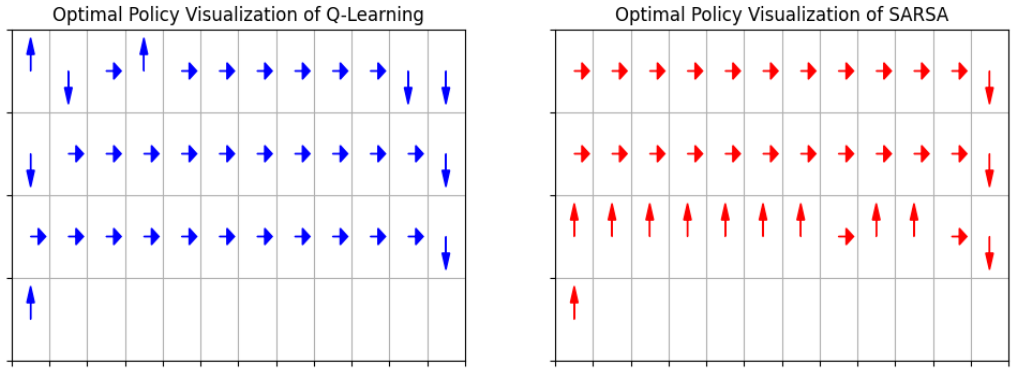




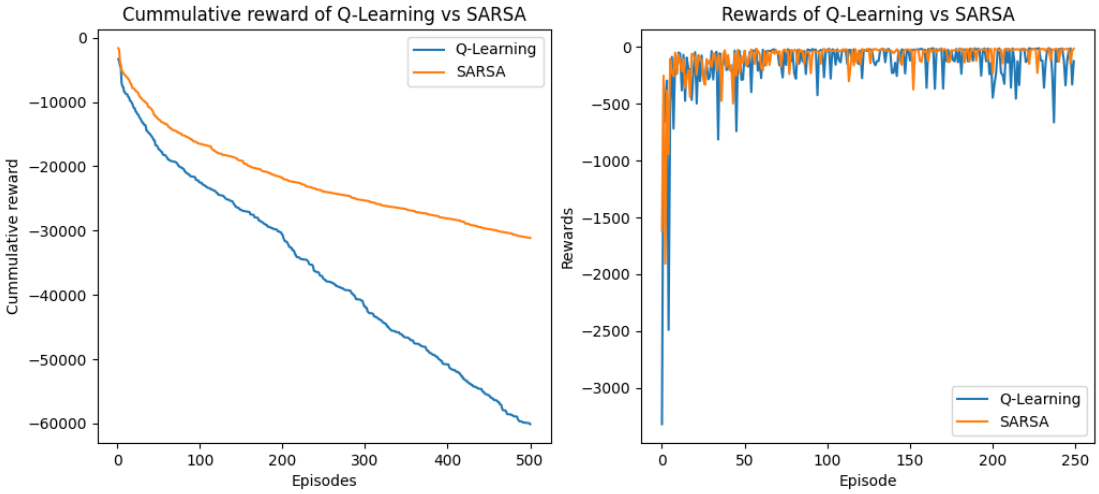
$\varepsilon = [0.05, 0.2, 0.5]$**:**
Beide Policies weisen normales Verhalten auf.

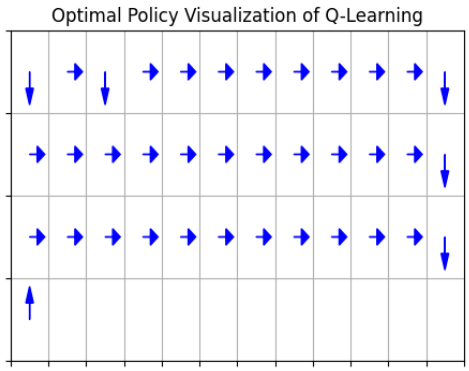
$\varepsilon = 0.05$**:**

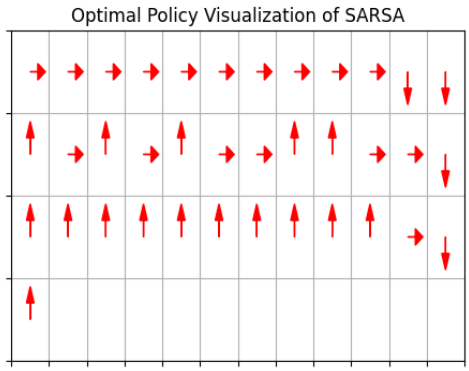




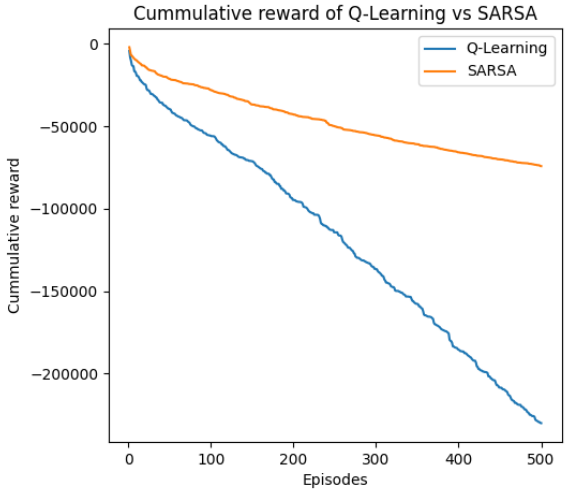
$\varepsilon = 0.2$**:**

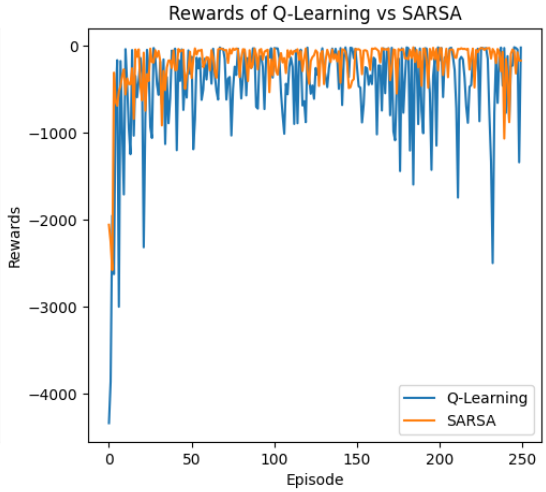


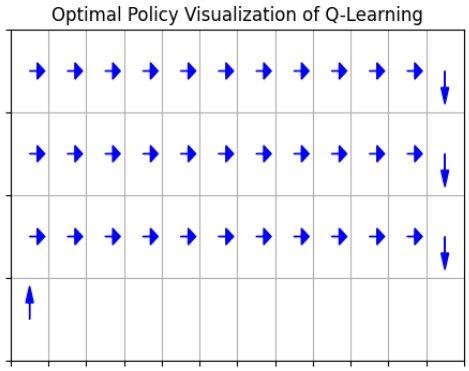


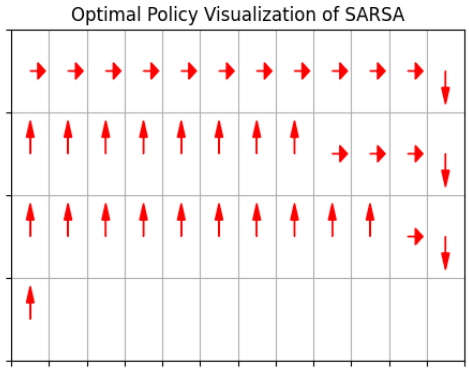


$\varepsilon = 0.5$**:**



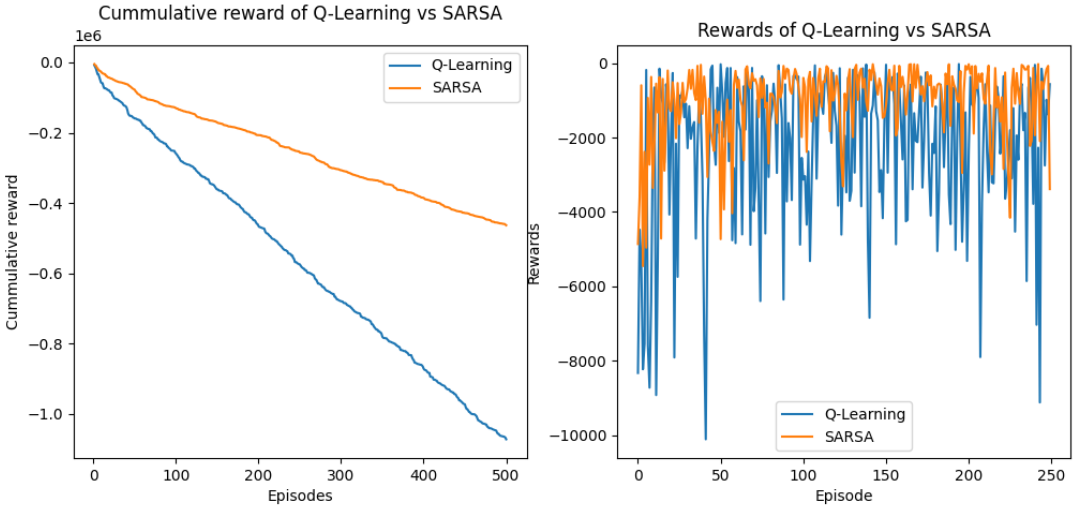


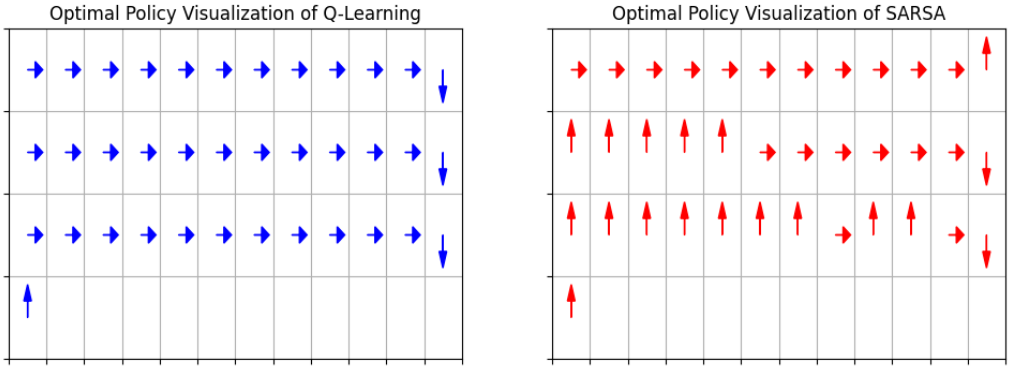




<u>$\varepsilon = 0.8$</u>**:**
In diesem Fall wird mehr erkundet, es werden also überwiegend zufällige Entscheidungen getroffen. Anhand der Varianz und der Belohnung sieht man, dass der Agent häufig herunterfällt.
Q-Learning: Man müsste davon ausgehen, dass der Q-Learning Algorithmus SARSA ähnelt, aber das ist nicht der Fall, da der Exploit überwiegt.
SARSA: Auch die SARSA policy wählt Wege nahe der Klippe, das kann mehrere Gründe haben. Zufällige Aktionen können dazu führen, dass der Agent häufiger an der Klippe landet, wodurch die

Q-Werte für diese Zustände angepasst werden. Der Agent könnte häufig nahe der Klippe agieren, ohne hinunterzufallen, dadurch könnten die Q-Werte diese Zustände ebenfalls attraktiv finden.





2) **Abnehmendes Epsilon**:

Wie schon in (1) erwähnt, ähnelt SARSA dem Q-Learning Algorithmus, wenn ε niedrig gewählt ist. Dadurch wird häufiger die beste Aktion gewählt und somit wird auch weniger die Klippe hinuntergefallen und somit wird nicht erkannt, dass der Weg "gefährlich" ist. Da weniger zufällige Bewegungen gemacht werden, wird der Weg nahe der Klippe gar nicht als "gefährlich" erkannt.

3) **Eliminierung von Exploration**:

Ein Vorteil ist, dass der Agent den schnellsten Pfad wählt, allerdings könnte ein Mangel an Exploration dazu führen, dass der Agent wichtige States verpassen könnte. Auch könnte der Agent sogar in einem lokalen Optimum stecken bleiben. Weitere Auswirkungen haben wir in (1) erläutert.

Modifikation der Exploration:
Epsilon-Decay Strategie: Man wählt zu Beginn einen hohen ε-Wert, damit zu Beginn eine umfassende Exploration ausgeführt wird. Im weiteren Verlauf wird ε minimiert, damit mehr Exploitation ermöglicht wird. Der Vorteil besteht darin, dass am Anfang eine hohe Exploration stattfindet, dadurch verschafft sich der Agent einen weitgehenden Überblick von der Umgebung. So können "verstecke" States gefunden werden. Später überwiegt die Exploitation, der Agent entweickelt somit eine stabiliere

und optimale Strategie.