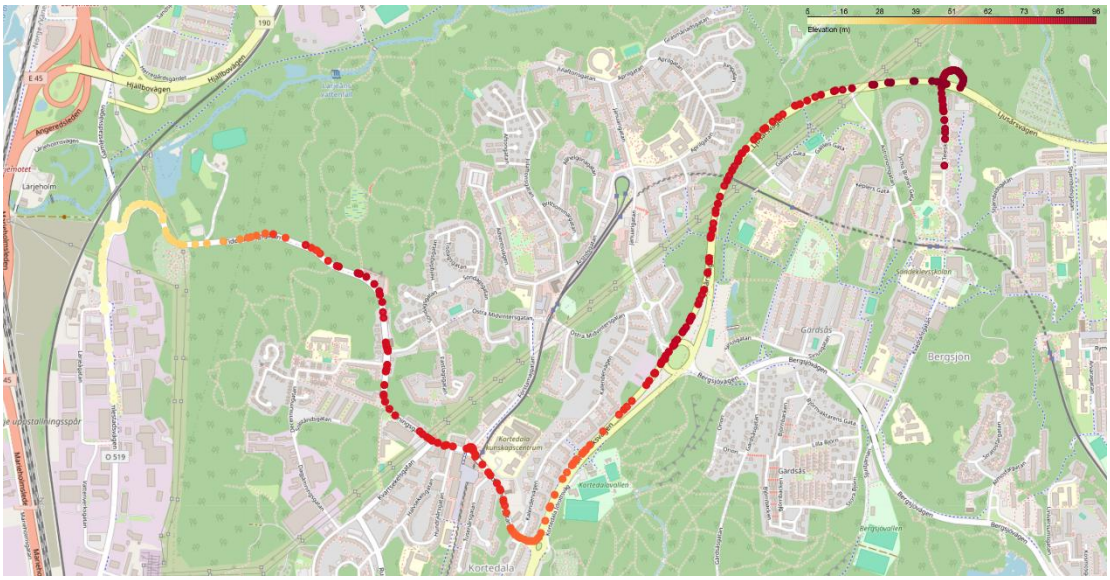
CHALMERS



AI-baserad ruttplanering för effektivisering av fordons energiförbrukning

Ruttoptimering med AI och geodata

Examensarbete inom högskoleingenjörsprogrammet Mekatronik

Nermin Karabegovic
Dorian Rozic

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK

CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2025
www.chalmers.se

Förord

Examensarbetet har genomförts av Dorian Rozic och Nermin Karabegovic vid institutionen för data- och informationsteknik på Chalmers tekniska högskola, under våren 2025.

Vi vill tacka vår handledare Sakib Sistek för värdefull vägledning, konstruktiv feedback och stöd under hela arbetet. Ditt kunnande och engagering i vårt arbete har varit ovärderlig för oss.

Vi vill även rikta ett stort tack till institutionen för möjligheten att genomföra arbetet hos er samt för den praktiska insyn och expertis ni har delat med er.

Göteborg, Juni 2025

Dorian Rozic & Nermin Karabegovic

Sammanfattning

Detta projekt syftar till att undersöka och testa hur inkorporering av höjddata, med hjälp från Google Elevation API till ett ruttplaneringssystem, påverkar energiförbrukningen. Ruttplaneringssystemet integrerar GPS-data i realtid, OSM-resurser och AI-algoritmer för bättre övergripande prestanda. Det primära målet är att få en uppskattning och förbättring av energiförbrukningen av en rutt med hjälp av AI. Resultaten visar att med hjälp av vägdata från Google API och OSM kan det skapas ett ruttplaneringssystem som genererar en rutt. Koordinater markeras längs vägsträckan för att sedan användas som utgångspunkter för datainsamling som AI algoritmen ska använda för beräkningar. Agenten beräknar sedan energikostnaden genom elevations- och hastighetsdata med hjälp av fysikaliska formler och reinforcement learning AI. Detta påvisar i sin tur att energieffektiviteten för ett fordon ökar med en signifikant mängd. Inkluderingen av ytterligare data såsom trafikflöde, farthinder och rondeller skulle kunna leda till en ännu lägre energikonsumtion och förbättra ruttplanerarens prestanda och noggrannhet ytterligare. Dessa faktorer påverkar körning och innebar även problem under testning och verifiering, då oförutsedda trafikförhållanden ej kunde inkluderas i mätning. Samtliga mål i projektet har uppnåtts. Ett fungerande system för energibaserad optimering av körning med hjälp av förstärkningsinlärning har utvecklats och testats, och resultaten visar att metoden har potential för framtida tillämpningar.

Nyckelord: Artificiell intelligens (AI), Google API, OpenStreetMap (OSM), GPS, Ruttplanering, Energieffektivitet, Geodata, maskininlärning, Reinforcement learning, Bränsleförbrukning

Abstract

This project aims to investigate and test how incorporating elevation data, using the Google Elevation API, into a route planning system that affects energy consumption. The route planning system integrates real-time GPS data, OSM resources, and AI algorithms for better overall performance. The primary goal is to estimate and improve the energy consumption of a route using AI. The results show that using road data from Google API and OSM, a route planning system can be created that generates a route. Coordinates are marked along the road section and then used as starting points for data collection that the AI algorithm will use for calculations. The agent then calculates the energy cost from the elevation and speed data using physical formulas and reinforcement learning AI. This in turn demonstrates that the energy efficiency of a vehicle increases by a significant amount. The inclusion of additional data such as traffic flow, speed bumps and roundabouts could lead to even lower energy consumption and further improve the performance and accuracy of the route planner. These factors affect driving, which has an impact on fuel consumption, and cause problems during testing and verification, as unforeseen traffic conditions could not be included in the measurement. All goals of the project have been achieved. A functioning system for energy-based optimization of driving using reinforcement learning has been developed and tested, and the results show that the method has potential for future applications.

Keywords: Artificial Intelligence (AI), Google API, OpenStreetMap (OSM), GPS, Route Planning, Energy Efficiency, Geodata, Machine Learning, Reinforcement Learning, Gas Consumption

Figurförteckning

Förkortningar, begrepp och variabler

Förkortning	**Förklaring**
RL	Reinforcement Learning (Förstärkningsinlärning)
API	Application Programming Interface
MDP	Markov Decision Process
GPS	Global Positioning System
OSM	OpenStreetMap
AI	Artificiell Intelligens
Agent	Den autonoma beslutsfattaren i ett system

Variabel	**Förklaring**
$Q(s, a)$	Aktuellt värde för tillstånd *s* och handling *a*
α	Inlärningshastigheten
r	Erhållna belöningen efter utfört handling
γ	Diskonteringsfaktor som avgör hur mycket framtidbelöning väger
$\max Q(s', a)$	Högsta Q värdet för alla handlingar i nästkommande tillstånd
m	Fordonets massa
g	Tyngdaccelerationen
v	Hastighet
v_1	Gamla hastigheten

v_2	Nya hastigheten
θ	Lutning i radianer
ρ	Luftens densitet
C_d	Luftmotståndkoefficient
C_r	Rullmotståndskoefficient
A	Projekterad frontyta
d	Avstånd mellan två punkter
R	Jordens radie
ϕ_1	Latitud i radianer för första punkten
ϕ_2	Latitud i radianer för andra punkten
$\Delta\lambda$	Skillnaden i longituder mellan två punkter

Innehållsförteckning

1. Inledning

Transportsektorn förblir ännu idag ett växande område vilket även innebär en ökad energiförbrukning och frågor när det kommer till de miljömässiga konsekvenserna skapade av ineffektiv ruttplanering. Kamal Puri argumenterar i en artikel [1] att effektiv ruttplanering kan spara företag upp till 20% på bränslekostnader och att detta även minskar underhållskostnader vilket ökar fordonets livslängd och sänker den totala kostnaden. Detta tyder på att fordon inom logistik och kollektivtrafik åker längs suboptimala rutter vilket leder till högre kostnader och negativ påverkan på miljön.

Traditionella ruttplaneringsalgoritmer tar ofta hänsyn till trafik och sträcka för att avgöra den mest effektiva och snabba rutten, men genom användning av data för lutning kan en algoritm skapas som är optimerad för att identifiera de mest energieffektiva vägarna.

1.1. Syfte

Projektets huvudsakliga syfte är att skapa en ruttplanerare som med hänsyn till sträcka och lutning kan identifiera en rutts optimala körstrategi för besparing av energi. Genom att använda positionsbaserade källor för höjddata, som Google Maps Elevation API, strävar projektet efter optimering av energiförbrukningen i dagens fordon.

1.2. Mål

Projektets övergripande mål är att utveckla en AI-modell som validerar rutter baserat på energikostnad och snabbhet. Ruttvalideringen innefattar även realtidspårning som indikerar var användaren befinner sig med hjälp av en GPS-modul, vart användaren vill åka samt meddelar den optimala hastigheten baserat på användarens position och rutt.

För att uppnå detta bestämdes följande delmål:

- Att skapa ett system som kan hämta in höjd- och väginformation via Google Maps API
- Att modellera belöningsfunktionen baserat på energikostnader med hjälp av lutning, luftmotstånd, rullmotstånd och hastighet.
- Att utveckla, testa och utvärdera en förstärkt inlärningsalgoritm för energieffektiv körning längs en rutt med hjälp av verkliga geografiska data från Google Maps.
- Att implementera en Q-learning baserad algoritm som kan fatta beslut om att accelerera, bromsa, rulla samt bibehålla hastigheten.
- Att analysera och jämföra den förstärkta inlärningsagentens optimerade körstrategi med en traditionell kör stil för utvärdering av energieffektiviteten.

1.3. Avgränsningar

- Projektet fokuserar enbart på energikostnader relaterad till själva fordonsrörelsen (acceleration, inbromsning, rullning och bibehållen hastighet)
- Faktorer som motorverkningsgrad, batteristatus och växellåda inkluderas inte i beräkningar.
- Trafikljus, korsningar, fotgängare och annan trafik beaktas inte – algoritmen antar att bilen kör ostört genom hela rutten.
- Hastighetsbegränsningar antas vara tillgängliga via OpenStreetMap och betraktas som statiska och korrekta.
- Tillfälliga ändringar i hastigheter, till exempel på grund av vägarbeten eller väderförhållanden, tas inte in i modellen.
- Bilens tjänstevikt tas med i beräkningar, men ytterligare vikt på passagerare och eventuell last beaktas inte.

2. Teori

I detta kapitel presenteras teorin som ligger till grund för examensarbetet. Syftet med kapitlet är att förklara de begrepp, metoder och tekniker som använts vid utvecklingen av den artificiella intelligensmodell som optimerar fordonsrörelse längs verkliga vägar. Fokuset i projektet ligger på förstärkningsinlärning, energikostnadsmodeller samt de externa tjänster som används för att hämta geografiska data.

2.1. Förstärkningsinlärning

Förstärkningsinlärning, även känt på engelska som Reinforcement Learning, är en gren inom maskininlärning. I förstärkningsinlärning ingår den så kallade agenten som lär sig fatta beslut genom att integrera med en miljö för att maximera en kumulativ belöning. Till skillnad från övervakad inlärning, där modellen får en uppsättning av indata och motsvarande utdata, lär sig agenten i RL genom att utforska och utnyttja miljön [2]. Det innebär att den måste balansera mellan att prova nya handlingar och att använda kända handlingar som ger hög belöning [2][3].

I detta projekt baserades RL på Markovs beslutprocess (MDP) där miljön representeras av tillstånd (states), handlingar (actions), utforskningshastighet och belöningsfunktioner. Agentens mål är att hitta en strategi eller ett sätt för att välja handlingar baserat på tillstånden, som maximerar den förväntade kumulativa belöningen över tid.

En vanlig metod för att implementera förstärkningsinlärning i praktiken är Q-learning – en modellfri algoritm som inte kräver kännedom om miljöns utforskningsgrad. Q-learning bygger på att agenten successivt uppdaterar ett så kallat Q-värde för varje kombination av tillstånd och handling [3]. Q-värdet representerar den förväntade framtida belöningen vid val av en viss handling i ett visst tillstånd, följt av optimal policy.

Uppdateringen av Q-värdet sker med hjälp av följande formel:

$$Q(s,a) = Q(s,a) + \alpha[\, r + \gamma \max Q(s',a) - Q(s,a)] \qquad (1)$$

där:

- $Q(s,a)$ är det aktuella värdet för tillståndet s och handling a.
- α är inlärningshastigheten (learning rate)
- r är den erhållna belöningen efter att ha utfört handlingen
- γ är diskonteringsfaktorn (discount factor), som avgör hur mycket framtida belöningar väger

- $\max Q(s', a)$ är det högsta Q-värdet för alla möjliga handlingar i nästa tillstånd s'.

Formeln används löpande för att uppdatera varje tillstånds kumulativa belöning beroende på handlingen som väljs [2]. Detta är särskilt användbart i problem där modellen lär sig direkt genom erfarenhet och gör det lämpligt att inkludera i projekt där agenten upptäcker nya strategier för att integrera med en simulerad ruttmiljö.

2.2. Belöningsfunktionen

I förstärkningsinlärning utgör belöningsfunktionen en central del av problembeskrivningen. Den definierar vad agenten ska uppnå. Enligt Sutton och Barto är belöningssignalen den enda källan till feedback som agenten får om hur bra den presterar i miljön [3]. Det är genom att observera belöningar som agenten kan lära sig att skilja mellan önskvärde och icke önskvärda tillstånd eller handlingar.

Belöningsfunktionen tilldelar ett numeriskt värde till varje tillstånd eller (tillstånd, handling) -par och ger agenten information om omedelbar nytta. Detta värde används sedan för att beräkna ett långsiktigt mått på förväntad framtida belöning – ofta kallat ”return” [3]. Det gör att målet för agenten blir att maximera detta förväntade mått över tid.

I artikeln [3] skriver författarna även att belöningsfunktionen inte behöver (eller bör) instruera agenten direkt om rätt beteende. I stället bör den spegla det önskade resultatet av beteende och därmed själv lära sig en effektiv strategi genom interaktion med miljön [3].

2.3. Energikostnad vid körning

För att säkerställa att agenten kan fullfölja projektets mål behöver agentens belöningsfunktioner anpassas för att effektivisera bränsle- eller energikostnad längs en given rutt. Detta gjordes genom att konstruera en belöningsfunktion som baseras bara på fysikaliska principer för där energiförlusten speglas vid körning. De centrala komponenterna i denna modell inkluderar rörelseenergi, potentiell energi samt energiförlust av luftmotstånd och friktionskrafter. Rörelse och potentiell energi antas vara energi som bilen förlorar eller behöver tillsätta för att komma upp i hastighet eller komma över en lutande väg, medan rullmotstånd och luftmotstånd beräknas utifrån arbetet eller energiförlusten som konstant sker när bilen är i rörelse.

Rörelseenergi (E_k) beräknas enligt formeln:

$$E_k = \frac{1}{2} m(v_2 - v_1)^2, \tag{2}$$

där m är fordonets massa och v är dess hastighet [4]. Detta speglar det energi som krävs för att accelerera fordonet från en viss fart till en annan, och är därför en viktig del i bedömningen av energibehov vid hastighetsförändringar.

Potentiell energi (E_p) är relevant vid körning i kuperad terräng och beräknas:

$$E_p = mg * \sin(\theta) * d \tag{3}$$

där g är tyngdaccelerationen, θ är lutningen i radianer mellan två punkter och d är sträckan mellan koordinaterna. I det här fallet används $\sin(\theta) * d$ i stället för h för att infoga lutning förändring och anta att varje par av vägsegmenten är en ramp [5]. Denna komponent påverkar agentens beslut för hastighet med hänsyn till upp- och nedförsbackar, där exempelvis körning uppför en backe ökar energikostnaden medan nedförsbackar kan bidra till energibesparing.

Luftmotståndet beräknas utifrån den aerodynamiska kraften (F_a):

$$F_a = \frac{1}{2}\rho C_d A v^2, \tag{4}$$

$$E_a = F_d * d, \tag{5}$$

där ρ är luftens densitet, C_d är luftmotståndkoefficienten, A är frontaarean på fordonet, v är hastigheten [4]. Omvandlingen till energi sker genom att multiplicera kraften med sträckan för att ta reda på hur mycket energi som försvinner på grund av luftmotståndet [6]. Eftersom denna kraft ökar med kvadraten på hastigheten har den stor påverkan på energikostnader vid högre farter. Genom att multiplicera kraften med sträckan får motsvarande energiförlust i form av arbete. Det resulterar i följande energiekvation:

$$E_a = \frac{1}{2}\rho C_d A v^2 * d \tag{6}$$

Lika så med rullningsmotståndet kan energiförlusten under rullning modelleras som:

$$F_r = C_r mg \tag{7}$$

$$E_r = F_r * d \tag{8}$$

$$E_r = C_r mg * d \tag{9}$$

där C_r är rullmotståndkoefficientet, m är fordons massa, g är tyngdaccelerationen och d är avståndet mellan startpunkten och slutpunkten. Energiförlusten beräknas likadant som för luftmotståndet där energin beräknas i form av arbete enligt ekvation (9). Rullmotstånd är en

basnivå av energiåtgång som alltid finns när fordonet rör sig, oavsett acceleration eller lutning.

2.3.1. Avståndberäkning

För att beräkna det horisontella avståndet mellan två punkter kan Haversine-formeln användas. Den tar hänsyn till jordens krökning och definieras enligt:

$$d = 2R * \arcsin\left(\sqrt{sin^2\left(\frac{\Delta\phi}{2}\right) + cos(\phi_1) * cos(\phi_2) * sin^2\left(\frac{\Delta\lambda}{2}\right)}\right) \quad (10)$$

Där:

- d är avståndet mellan två punkter på jordens yta
- r är jordens radie (approximativt 6378 km)
- ϕ_1 och ϕ_2 är latituderna i radianer
- $\Delta\phi$ är skillnaden i latitud
- $\Delta\lambda$ är skillnaden i longitud

2.4. Google Maps API

Inom detta projekt användes olika API-tjänster som tillhandahålls av Google Maps-plattformen för att möjliggöra ruttoptimering. Google Maps API erbjuder kraftfulla verktyg för att hämta detaljerad geografisk och vägbaserad information i realtid [7]. Detta gör det möjligt att simulera och analysera verkliga vägförhållanden på ett strukturerat och automatiserat sätt. I och med detta har Google API en central roll i att koppla den teoretiska modellen till en realistisk och praktiskt användbar kontext.

I detta avsnitt presenteras de olika API tjänster som använts i projektet och deras respektive funktion och betydelse i systemets uppbyggnad.

2.4.1. Directions API

Directions API [8] används för att hämta vägbeskrivningar mellan två eller flera geografiska punkter. Tjänsten returnerar detaljerad ruttinformation såsom avstånd, uppskattad restid samt koordinater för varje delsegment av rutten. Denna information utgjorde grunden för att bygga upp agentens väg och analysera en sträcka som optimeras, exempelvis genom att kombinera vägsegment med lutning och hastighetsdata från andra API:er.

2.4.2. Elevations API

Elevations API [9] tillhandahåller exakt höjddata för geografiska punkter över hela världen. Genom att skicka förfrågningar till API:et kan utvecklare få höjdinformation för enskilda punkter eller längs en väg/rutt. Detta är användbart för projekt, likt detta, som kräver terränganalyser.

I detta projekt användes Elevation API för att beräkna lutningen mellan intilliggande ruttpunkter, vilket i sin tur användes för att modellera lägesenergi och påverkan av gravitationskrafter vid körning i uppförs eller nedförsbackar. Genom att kombinera elevation med positioner från Directions API kunde en höjdprofil för hela rutten konstrueras. Detta utgjorde en viktig komponent i agentens beslutsunderlag för energieffektiv körning.

Tjänsten möjliggör både enskilda punktförfrågningar och batchförfrågningar längs en väg, vilket gör den flexibel för att integrera höjddata i realtid eller som en del av en förbearbetning av ruttinformation [9].

2.4.3. Roads API

Roads API [10] används för att identifiera och koppla koordinater till faktiska vägar, vilket är särskilt användbart när GPS-koordinater från exempelvis Directions API inte exakt matchar vägbanan. Genom att använda funktionen ”Snap to Roads” kan man justera en följd av punkter så att de följder en riktig väg och med ”Speed limits” funktionen kan man även hämta hastighetsbegränsning för varje punkt [10].

2.5. OpenStreetMap

OpenStreetMap (OSM) [11] är en ’open-source’ geodatabas som förser användaren med ett grafbaserat nätverk som innehåller väginformation. I detta projekt används OSM bland annat för hastighetsbegränsningar mellan två väggränssnitt, som med hjälp av Python biblioteket OSMnx hämtar vägnätet inom ett specificerat geografiskt område som i sin tur möjliggör identifiering av rutter och deras data.

3. Metod

Projektet bygger på en simuleringsbaserad metodik med målsättningen att undersöka hur förstärkningsinlärning kan användas för att optimera ett fordons energiförbrukning längs en given rutt. Arbetet genomförs i flera steg, där varje steg utformas för att möjliggöra konstruktionen och utvärderingen av en inlärningsmodell i realistisk uppbyggd miljö.

3.1. Modellval och inlärningsstrategi

Som grund för hela systemet används en förstärkningsinlärningsmodell som kallas för Q-learning som gör att agenten själv lär sig utveckla en körstrategi som maximerar belöningsfunktionen kopplad till energibesparing. Metoden valdes för dess förmåga att hantera sekventiella beslut i dynamiska miljöer som är särskilt bra för projekt där optimal handling beror på tidigare val och framtida konsekvenser.

Agentens handlingsutrymme definieras på två möjliga sätt:

1. Diskret handlingsval – Där agenten väljer mellan fyra fördefinierade handlingar (Accelerera, bromsa, bibehåll hastighet eller rulla)
2. Kontinuerlig handlingsval – Där agenten fritt väljer en hastighet inom ett angivet intervall.

3.2. Datainsamling

För uppbyggnad av miljön som agenten skall tränas i behövs detaljerad information om vägsträckningen. Informationen ska bestå av geografiska koordinater, höjddata samt hastighetsbegränsningar.

I detta fall valdes Google Cloud för hämtningen av nödvändiga data som ruttens koordinater i en sekvens, samt höjd- och hastighetsdata för dessa koordinater. Framför allt är det Directions API, Elevations API och Roads API som skall användas. Projektet utgår dessutom från att gratis versionen av Google Cloud används vilket innebär att vissa tjänster inte är tillgängliga. För eventuella problem kommer projektet luta sig åt användning av open-source navigationsdata vid namnet OpenStreetMap.

Kombinationen av dessa möjliggör uppbyggnaden av ett tillståndsrum där varje punkt representerar ett unikt segment med egenskaper som position, lutning och hastighetsgräns.

3.3. Belöningsfunktion

Utformning av belöningsfunktionen görs med fokus på att minimera energikostnader, där kostnaderna modelleras utifrån fysikaliska formler. Rullmotstånd, luftmotstånd,

rörelseenergi och potentiell energi kommer användas för att approximera energin som går åt vid varje tillstånd.

3.4. Implementering och träning

Algoritmen implementeras i Python och en simulerad miljö byggs upp utifrån föregående exempel. Agenten tränas iterativt genom ett antal simuleringar av rutten, där strategin förbättras över tid. Inlärningsparametrar som inlärningshastighet, diskonteringsfaktor, utforskningsgrad samt antal episoder justeras för identifiering av konvergens.

3.5. Utvärdering

Agentens prestanda utvärderas genom att analysera konvergens med optimal strategi samt jämförelser av energikostnader mellan olika körsätt. Resultatet visualiseras med bilder på bränsleförbrukningen efter varje körning. Projektet bedöms som framgångsrikt om den inlärda körstrategin resulterar i en lägre bränsleförbrukning jämfört med en traditionell körning utan optimering.

4. Genomförande

I detta kapitel beskrivs hur metoder och tekniker nämnda i metoden användes för att skapa programmet, samla in data, bygga upp tillståndsrummet (state space), samt träna och testa modellen för energieffektiv körning.

4.1. Datainsamling och miljöuppbyggnad

För att kunna optimera energikostnader längs en given rutt utvecklades en modell baserad på förstärkningsinlärning. Miljön för modellen byggdes upp genom att varje tillstånd representerade ett segment av vägen med associerad information om geografisk position, vägsegmentens lutning samt tillhörande hastighetsbegränsning.

Data för tillståndsdefinitionen samlades in via Google Maps API-tjänster och OSM. Koordinater för en given rutt hämtades genom Directions API, och höjddata för varje koordinat extraherades via Elevations API. Höjdskillnaderna användes därefter för att beräkna lutningsvinkeln mellan varje efterföljande punkt-par. Roads API användes för att koppla givna koordinaterna från Directions API direkt till verkliga vägar. På grund av begränsningarna i den kostnadsfria versionen av Google Cloud kunde funktionen för hastigheten i Roads API inte användas. I stället definierades hastighetsbegränsningarna manuellt i en separat vektor, där varje värde motsvarade ett segment i den uppbyggda rutten. Detta gjordes med hjälp av OSM som användes för att ta reda på var på rutten hastigheten ändras. Därefter användes en ”egen-skapad” funktion för att automatiskt tilldela hastighetsbegränsningar till varje tillstånd i tillståndsrummet.

För beräkning av lutning användes Haversine-formeln (Se avsnitt 2.3.1). När det horisontella avståndet mellan punkterna har beräknats kan lutningen beräknas i radianer enligt:

$$Lutning(radianer) = \left(\frac{\Delta h}{d}\right) \tag{11}$$

På så sätt kunde varje segment i tillståndsrummet tilldelas en lutningsvinkel som senare användes vid beslutstaganden i agentens miljö.

4.2. Tillstånd och handlingar

Tillståndsrummet som användes i förstärkningsinlärning-algoritmen bestod av en sekvens av vägsegment som inkluderade:

- Latitud och longitud
- Beräknad lutning i grader
- Tillhörande hastighetsgräns

Agentens möjliga handlingar delades upp i:

- Accelerering
- Bromsning
- Bibehåll hastighet
- Rullning

Varje handling resulterade i en förändring av hastigheten enligt fördefinierade regler som syns i tabellen nedan.

Tabell 1. Fördefinierade regler för hastighets ändring beroende på handling.

Handling	**Hastighet**		
Acceleration	+3 km/h		
Bromsning	-3 km/h		
Bibehåll hastighet	Förblir samma		
Rullning	Lutning < -3°	Lutning > 3°	-3° < Lutning < 3°
	+1 km/h	-4 km/h	-2 km/h

Ett förenklat flödesschema över koden som skapar tillståndsrummet syns i bilaga A.

4.3. Skapandet av belöningsfunktionen

Vid varje steg i rutten beräknades energikostnaden baserat på följande fysikaliska ekvationer:

- Rörelseenergi (E_k): *Se ekvation (2)*
- Lägesenergi (E_p): *Se ekvation (3)*
- Rullmotstånd (E_r): *Se ekvation (6)*
- Luftmotstånd (E_a): *Se ekvation (9)*

För att driva inlärningen mot energieffektiv körning definierades summeringen av ekvationerna som den negativa summan av den beräknade energin:

$$Reward = -\left(E_k + E_p + E_r + E_a\right) \tag{12}$$

Detta uppmuntrar agenten att använda så lite energi som möjligt. För att säkerställa att belöningen endast påverkas av faktisk energikostnad, och inte av inbromsningar eller andra situationer där energi ”sparas”, räknas bara det positiva kinetiska samt lägesenergin. Det innebär att energibidrag ignoreras i den summerade energin för att säkerställa att total

energikostnad inte underskattas. Med andra ord straffas energianvändning, men energibesparingar räknas som neutrala snarare än belönande. Detta val gjordes för att agenten inte ska uppmuntras till ineffektiva beteenden som enbart får agenten att bromsa in för att spara energi.

4.4. Extra belöning och straff

För att ytterligare styra inlärningen mot energieffektiv körning implementerades en belöningsfunktion som inte bara byggde på energiåtgång utan även förstärktes med extra straff och belöningar beroende på om agenten följde hastighetsgränser eller anpassade sig till lutningen på ett smart sätt. Exempelvis:

- +30: om agenten accelererade när hastigheten låg under tillåten miniminivå
- -10: om agenten överskred maxhastigheten vid acceleration
- +10: om hastigheten hölls inom tillåtet intervall
- +4 + abs(lutning): vid rullning nedförsbacke
- -4 – abs(lutning): vid rullning uppförsbacke
- -20: om agenten bromsade inför uppförslut utan att det fanns behov
- -10: om agenten rullade med låg hastighet

Syftet med att införa extra belöningar och straff var för att säkerställa att agenten följer hastighetsbegränsningarna, där den lägsta tillåtna hastigheten definierades som:

$$Min\ hastighet = Max\ hastighet - 10\ (km/h)$$

Ett ytterligare syfte var att uppmuntra agenten att använda rullning i större utsträckning. Även om de fysikaliska modellerna för energikostnaden inkluderade grundläggande krafter som luftmotstånd och rullmotstånd, baserades dessa på antaganden snarare än exakta värden för till exempel vägens friktion eller luftens densitet. Därför infördes kompletterande belöningar och straff för att styra agentens beteende bort från åtgärder som i praktiken sannolikt skulle vara energikrävande, även om det fullt ut inte kunde modelleras i energiekvationerna.

4.5. Inlärningsalgoritm

Träningen utfördes med hjälp av en Q-Learning-algoritm, där varje (tillstånd, handling)-par uppdaterades iterativt baserat på den erhållna belöningen och nästa tillstånd i enlighet med formel (1) i avsnitt om förstärkningsinlärning. I bilaga B syns ett flödesschema över hur inlärningsalgoritmen strukturerades.

Parametrar som inlärningshastighet, diskonteringsfaktor och antal episoder justerades experimentellt för att uppnå stabil konvergens.

4.6. Testning och Verifiering av Modellen

Efter att modellen tränats färdigt testades den för att kunna utvärdera dess förmåga att fatta energieffektiva beslut längs en fördefinierad rutt. Testningen utfördes genom att testa agentens beslut längs samma rutt som användes under träningen.

För att kunna utföra testningen kopplades en GPS-modul för hämtning av position i realtid samt jämföra den med koordinaterna extraherade i den optimala rutten. Genom att jämföra realtidspositionen med koordinaterna i optimala rutten kunde körningen utföras i liknande stil.

Till följd av långsam respons i programvaran och jämförelse med optimala körstrategin från agenten behövde testningen göras på ett annat sätt, där en person kör bilen och den andra läser agentens beslut ur den plottade kartan som syns i Bilaga E och utskriften i Bilaga D.

Resultaten utvärderades genom att köra rutten ett par gånger, där mätning för bränsleförbrukning återställs innan körning för att mäta hur mycket bränsle som har förbrukats:

- En vanlig körning utan agentens påverkan.
- En körning enligt agentens optimala rutt.

Syftet med traditionella körningen var inte att köra maximalt bränslesnålt, utan snarare efterlikna en realistisk och medveten körning i vardaglig trafik.

Under testningen kördes det enligt följande plan:

- Traditionell körning för rutt 1
- Traditionell körning för rutt 2
- Körning enligt agentens rekommendation för rutt 1
- Körning enligt agentens rekommendation för rutt 2

För varje utfört steg återställdes bränsleförbrukningsmätaren i bilen och sedan kördes nästa rutt. För rutt 3 och 4 följdes samma princip, där det först kördes fram och tillbaka genom traditionell körning, därefter fram och tillbaka med agentens rekommendationer.

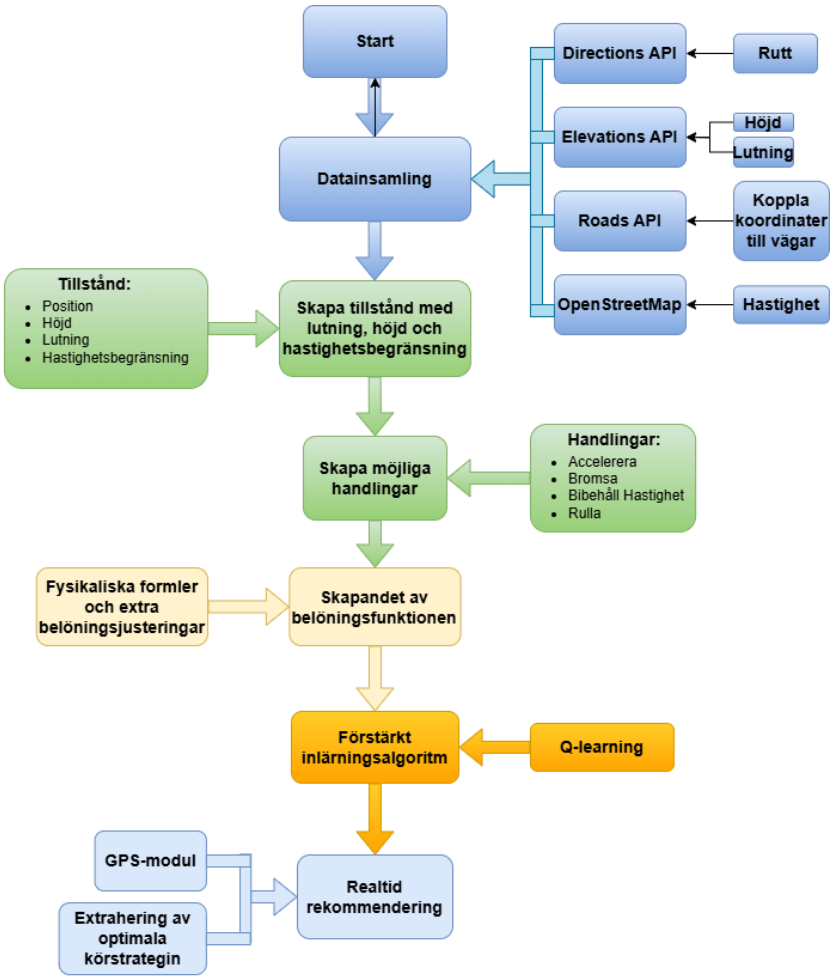
Efter testningen jämfördes bränsleförbrukningen för de olika körningar och validerades genom att undersöka om agenten sparar mer bränsle än traditionella körningen.

5. Systemkonstruktion

I detta avsnitt diskuteras teknisk beskrivning av de delar i projektet som har utvecklats och hur dessa samverkar för att möjliggöra agentens optimala körstrategi.

5.1. Översiktlig systemarkitektur

Systemet är uppbyggt i moduler som tillsammans möjliggör hämtning av rutt- och höjddata, genererar tillstånden, beräknar belöning för agenten, tar beslut via en förstärkningsalgoritm samt extraherar optimala körstrategin ur algoritmens resultat. Figur 1 nedan visar en förenklad översikt över systemets komponenter och deras inbördes relation.



Figur 1. En överblick på systemets arkitektur

Ett förenklat koncept av systemet syns i bilaga C.

5.2. Mjukvarustruktur

Systemet har, som nämnt i metodavsnittet, utvecklats i PyCharm med programmeringsspråket Python. Systemet består dessutom av ett antal logisk separerade moduler, där varje modul ansvarar för ett steg i processen. De centrala skripten är:

- Datainsamling:
  - **get_route_polyline()** – Hämtar rutten
  - **snap_to_roads()** – Hämtade rutten kopplas till vägar
  - **get_elevation()** – Hämtar höjddata
  - **haversine_distance()** – Beräknar avståndet mellan två punkter
  - **generate_speed_limits()** – Genererar lista med hastighetsbegränsningar
- Skapa tillstånd:
  - **compute_state_space_with_ta_sl()** – Skapar tillståndsrummet med lutning och hastighetsbegränsning
- Belöningsfunktion:
  - **reward_function()** – Beräknar belöningen beroende på handlingen
  - **Car** – En class function som innehåller viktiga parametrar för energiberäkning
- Q-learning
  - **run_rl()** – Funktion som sätter styr över agentens träning.
- Optimal körstrategi
  - **get_optimal_path_with_speeds()** – Extraherar optimala körstrategin ur Q-tabellen från run_rl()
- Rekommendering till användaren
  - **get_current_gps_position()** – Hämtar koordinater för användarens nuvarande position
  - **find_closest_path_point()** – Jämför nuvarande position med den extraherade körstrategin och hitta närmaste punkten
  - **plot_optimal_path_on_map()** – Plottar optimala körstrategin på en karta.

5.3. Hårdvarustruktur

För utvärdering och testning av strukturen har systemet huvudsakligen körts och testats på en bärbar dator i kombination med en extern GPS-modul. Syftet med GPS-modulen var att möjliggöra framtida utvärdering av systemet i en verklig rörlig miljö, där livepositionering kan kopplas till tränade agentens beslut.

Den använda hårdvaran:

- Laptop
    - Modell: Lenovo Yoga Slim 7 Pro
    - Processor: AMD Ryzen 7 5800H
    - RAM: 16 GB
    - Operativsystem: Windows 11 Pro
- GPS-modul:
    - Modell: UBLOX NEO-7M
    - Kommunikation: Seriell anslutning via USB
    - Positioneringsfrekvens: Upp till 10 Hz
    - Noggrannhet: 2.5m
    - Gränssnitt: NMEA-data via seriell kommunikation (9600 baud)

Den seriella kommunikationen med GPS-modulen hanterades i Python via *pyserial,* där realtidsdata togs emot och tolkades.

6. Resultat

I detta avsnitt presenteras resultaten från implementationen av förstärkningsinlärning som utförts med Q-learning som inlärningsalgoritm och hur systemet för beräkning av energikostnad, med hjälp av Google API, fungerade i helhet. Utifrån den teoretiska bakgrunden har fokus legat på hur agenten, genom interaktion med sin omgivning och med hjälp av en definierad belöningsfunktion, lär sig att välja lämpliga hastigheter beroende på vägavsnittens lutning och hastighetsbegränsningar.

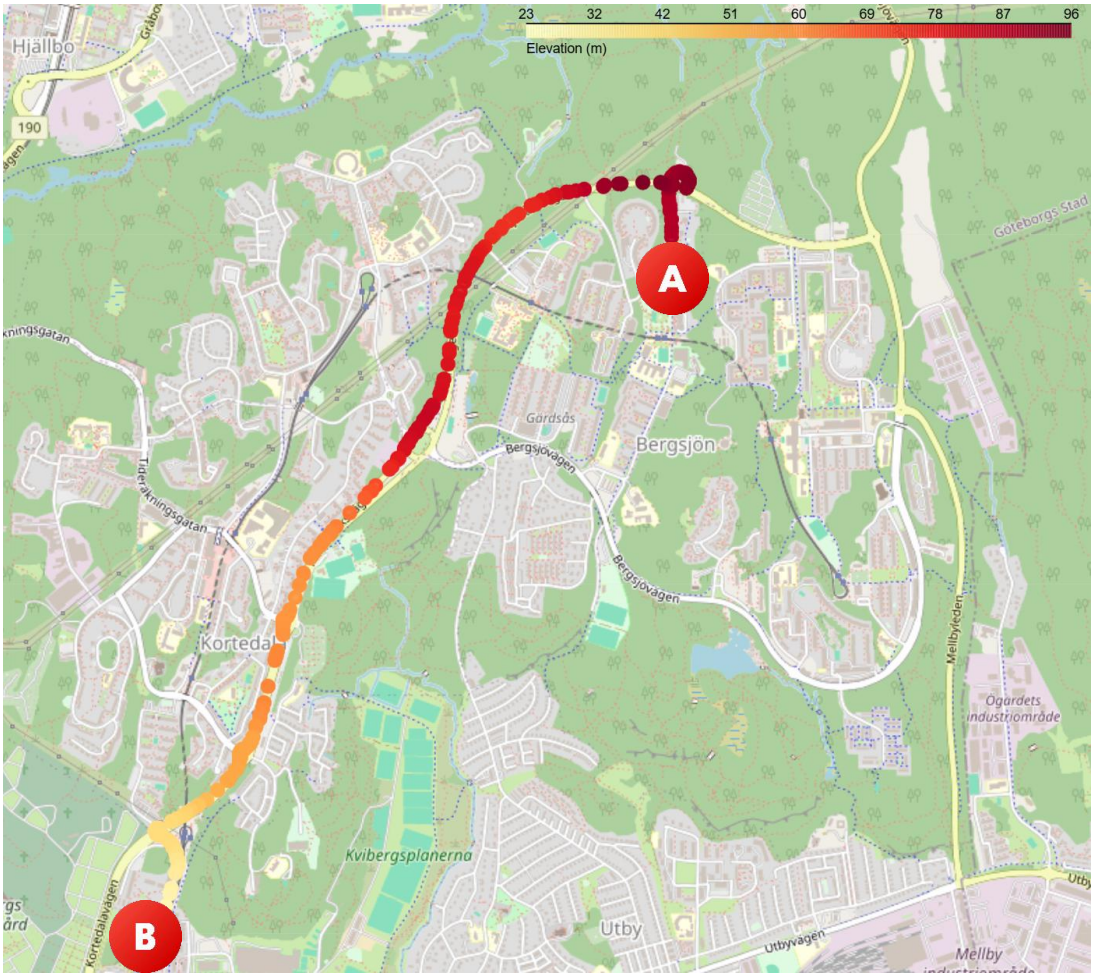
Resultaten syftar till att visa hur agentens beteende utvecklas över tid och fyra olika rutter, hur belöningsfunktionen påverkar inlärningen samt hur den tillämpade metoden presterar i olika typer av miljöer. Utvärderingen baseras bland annat på den totala ackumulerade belöningen, energikostnaden, valda hastigheter längs vägen och hur väl agentens beslut överensstämmer med det önskade energieffektiva beteendet.

6.1. System för höjd- och väginformation via Google Maps API

Fyra olika rutter testades för att utvärdera agentens förmåga att välja energieffektiva hastigheter. Dessa rutter valdes med avsikt att inkludera variation i lutning och hastighetsbegränsningar.

På grund av begränsad projekttid valdes totalt fyra rutter varav två rutt-par utgörs av fram- och tillbakaresor mellan samma punkter. Den första ruttgruppen går från Teleskopgatan 8 till Beväringsgatan 21 och tillbaka. Den andra från Teleskopgatan 8 till Gamlestadsvägen 311 och tillbaka. Detta möjliggjorde att olika lutningsprofiler och hastighetsförhållande kunde testas i båda riktningar, utan att antalet testningar blev för omfattande.

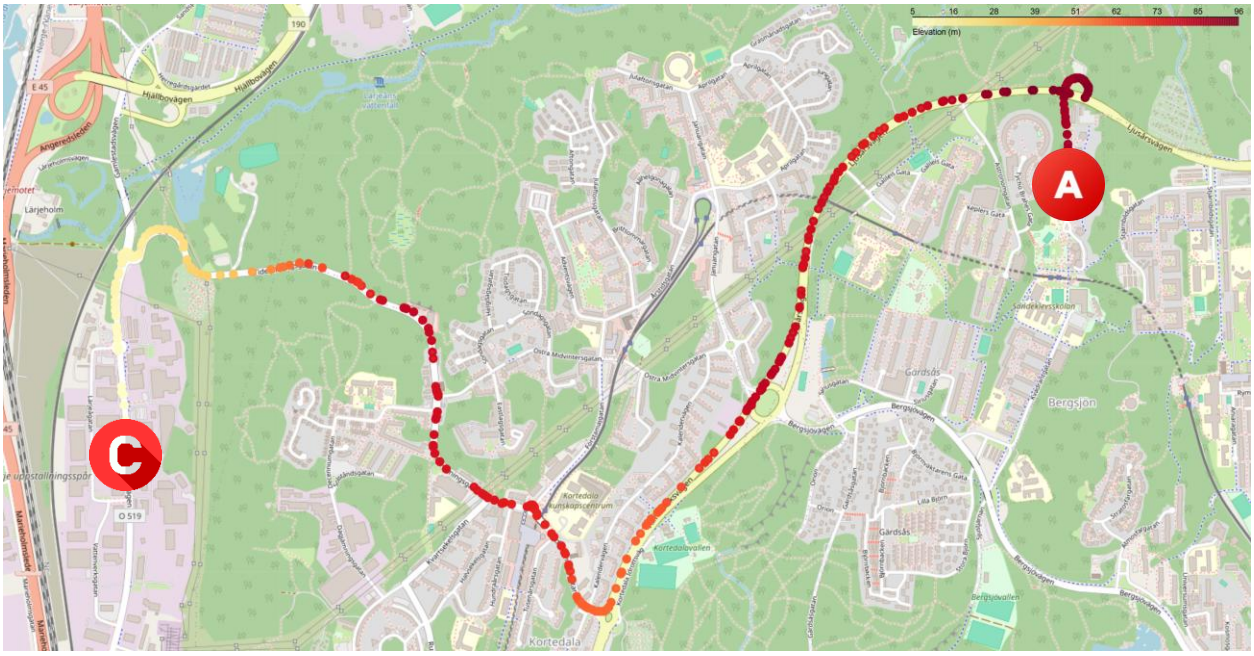
För varje rutt samlades data in om agentens valda hastigheter, ackumulerad belöning och energikostnad. Figur 2 visar ett exempel på höjdprofil för rutt 1 och rutt 2, där vägavsnitten är färgkodade efter höjd.



Figur 2. Höjddata för rutt 1 och 2. Höjden går från 23m till 96m över havsnivån. A – Teleskopgatan 8, B – Beväringsgatan 21

Gul innebär låg höjd medan rött innebär hög höjd. Lutningen längs sträckan påverkades av dessa höjdskillnader och används som en del av tillståndsinformationen i inlärningen.

I figur 3 nedan syns samma princip fast för rutt 3 och 4.



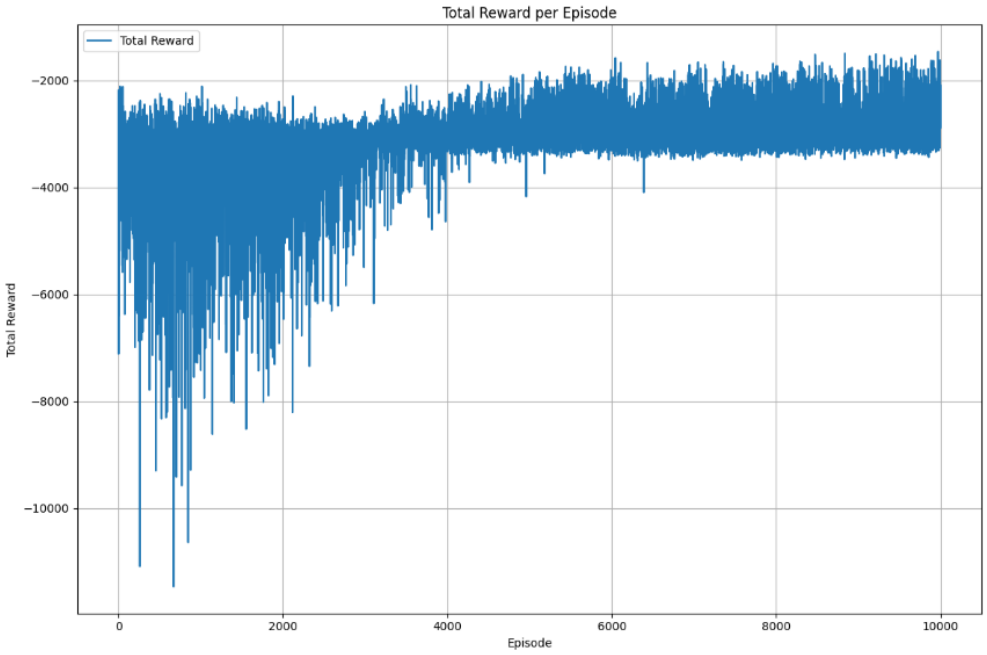
Figur 3. Höjddata för rutt 3 och 4. Höjden går från 5 m till 96 m över havsnivån. A – Teleskopgatan 8, C – Gamlestadsvägen 311.

6.2. Inlärningsprocess och konvergens

För varje testrutt tränades agenten 10 000 episoder där ca 2000 av de första episoderna inte visade något tydligt mönster utan utforskade tillståndsrummet slumpmässigt. Anledningen till detta är för att ϵ-greedystrategin användes för att styra agentens beslutsfattande. I början sattes utforskningsfaktorn till ett högt värde ($\epsilon = 1.0$) vilket innebar att alla handlingar i första episoden var slumpmässigt valda. Detta uppmuntrade agenten till utforskning och säkerställde att den fick erfarenhet av olika scenarier innan den började maximera optimala handlingar.

Efter varje episod reducerades diskonteringsfaktorn gradvis med 0,5% vilket ledde till att agenten i högre utsträckning började utnyttja kunskapen som den hade byggt upp. För varje episod som gick reducerades faktorn till ännu lägre värde där den som lägst når 0.050 efter ca 600 episoder.

Efter 5000 episoder börjar belöningen stabiliseras och det blir tydligt att agenten har hittat ett bra beteendemönster. I figur 4 syns den ackumulerade belöningen per episod gradvis fram till ett läge där variationer mellan episoder blev minimal.



Figur 4. Graf över total belöning per episod/ruttkörning (Rutt 1).

I grafen syns totala belöningen för varje ruttkörning eller episod och konvergensen syns vid episod 4000 där agenten långsamt börjar klättra i belöning i stället för att falla längre ner i straffar. Anledningen till att belöningen fortfarande är negativ är för att belöningsfunktionen baseras på energin som går åt under rutten och de extra belöningar till belöningsfunktionen är inte tillräckligt stora för att höja belöningen över till positiva y-axeln.

För att uppnå konvergens, ändrades fler variabler än bara utforskningsgraden. Parametrarna som; antal episoder, inlärningshastighet och diskonteringsfaktorn ändrades också för att hitta en bra balans. Bästa resultat för de fyra valda rutter var:

- Episoder = 10 000
- Inlärningshastighet = 0.01
- Diskonteringsfaktorn = 0.8

6.3. Effekten av belöningfunktionens utformning

Under projektets gång testades olika belöningsfunktioner. Den första byggde strikt på beräkningar av energikostnader enligt fysikaliska formler, diskuterade i genomförandet (avsnitt 4.4). Resultaten från första versionen av belöningsfunktionen indikerade att agenten nästan alltid föredrog att köra så långsamt som möjligt – ibland till och med stillastående. Detta eftersom låg fart innebar minimal energiåtgång, även om det ledde till ineffektiv körning i praktiken.

För att motverka detta beteende infördes en justering i belöningsfunktionen enligt genomförandet (avsnitt 4.5). Resultatet efter dessa förändringar visades vara mycket mer effektiva, där agenten följde vägens hastighetsbegränsningar och även tog hänsyn till lutningar på vägen för att spara mer energi.

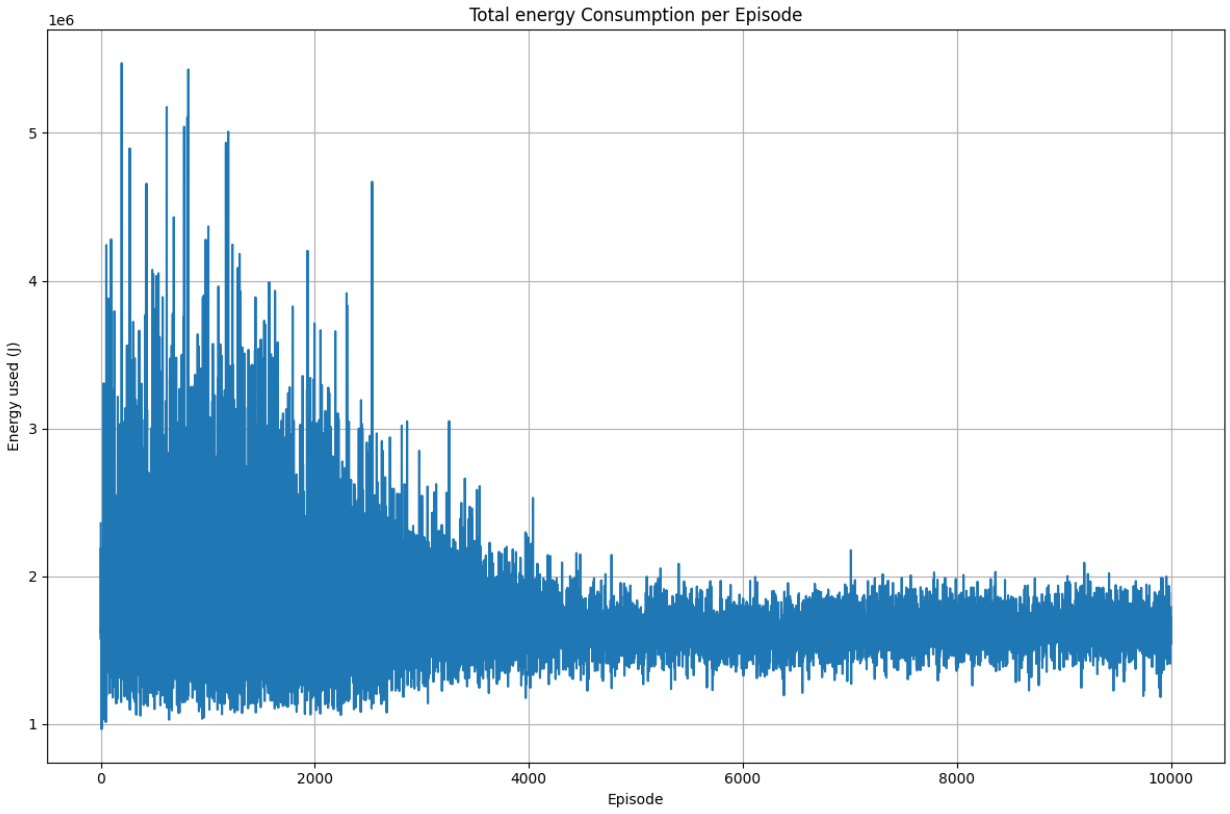
6.4. Resultat från testade rutter och uppställning

För att möjliggöra en jämförelse mellan agenten och den traditionella kör stilen genomfördes referenskörningar. Dessa körningar utfördes med en normal men sparsam körning, där hastighetsbegränsningarna följdes exakt och accelerationen hölls på en nivå som motsvarar hur man vanligtvis kör i verkliga situationer.

6.4.1. Rutt 1: Teleskopgatan 8 (A) till Beväringsgatan 21 (B)

För att illustrera agentens påverkan på energikostnaden presenteras här en jämförelse mellan två körningar av rutt 1, en utan agentens inblandning och en där körningen baseras på agentens beslut.

I figur 5 nedan presenteras den total energikostnaden per episod under träningsförloppet, vilket åskådliggör hur agentens körstrategi successivt blir mer energieffektiv över tid.



Figur 5. Ändringar i total energikostnad per episod för agentens simulerade körning på rutt 1.

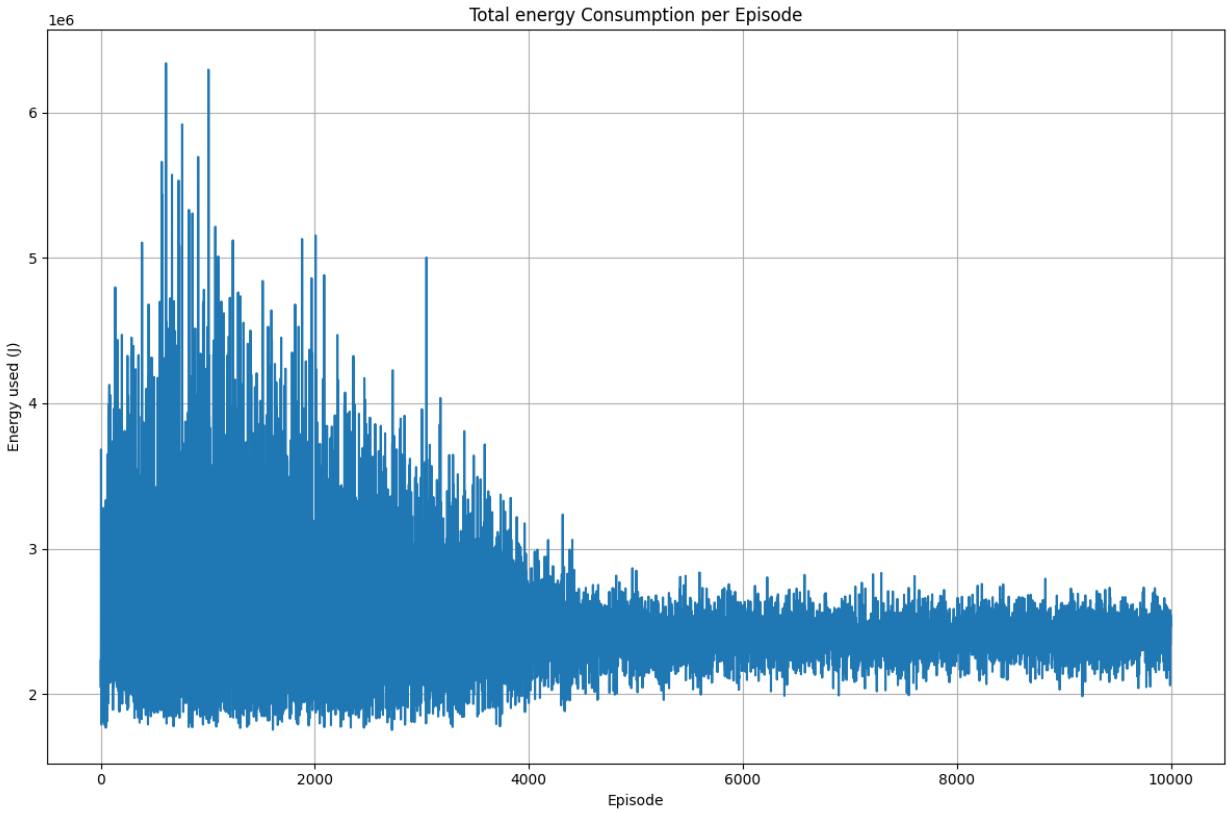
I figurer 6 och 7 syns jämförelsen i bränsleförbrukningen mellan en traditionell körning och en körning baserat på agentens beslut.

Figur 7. Inringat i gult visas bränsleförbrukning efter referenskörning för rutt 1.

Figur 6. Inringat i gult visas bränsleförbrukningen med agentens beslut för rutt 1.

6.4.2. Rutt 2: Beväringsgatan 21 (B) till Teleskopgatan 8 (A)

Resultatet för totala energikostnaden under träningsförloppet för rutt 2 syns i figur 9.



Figur 8. Ändringar i total energikostnad per episod för agentens simulerade körning på rutt 2.

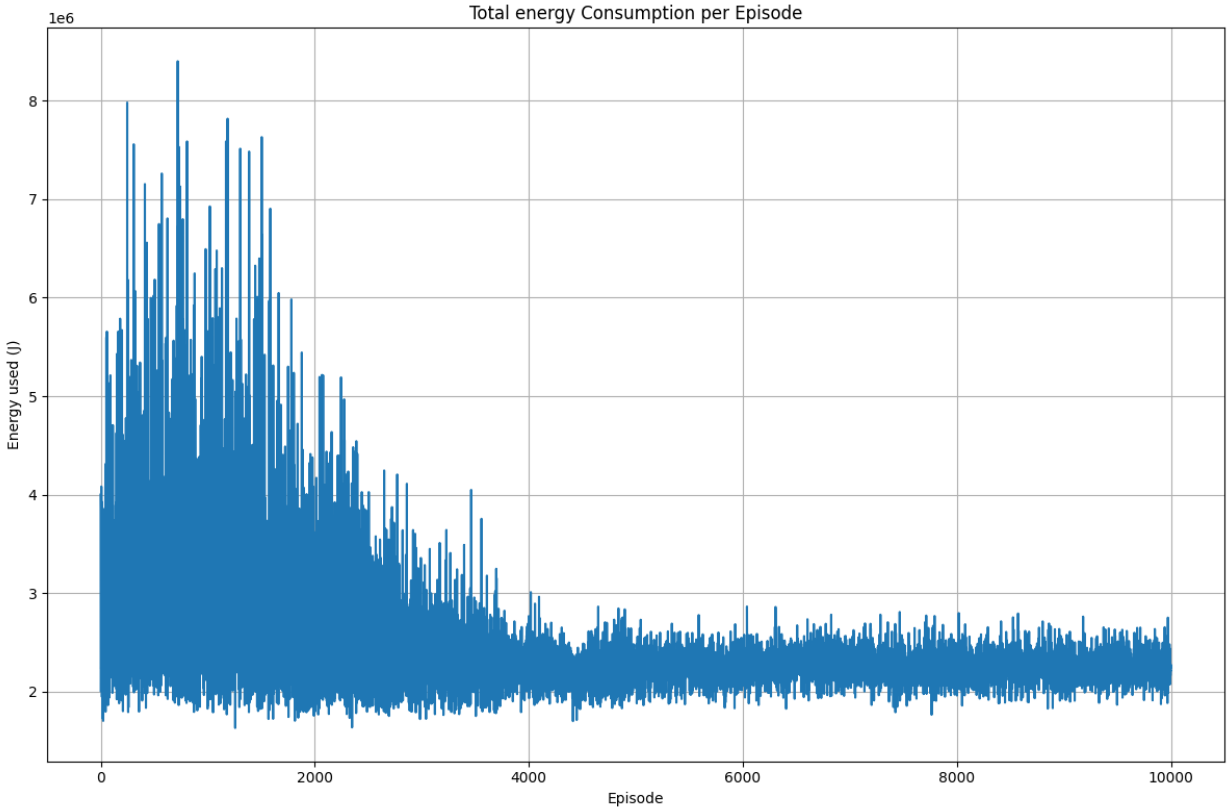
Likadant för rutt 2 följdes samma princip, där första körningen var en referenskörning och nästa var baserat på agentens beslut. Resultat visas i figur 9 och 10 nedan.

Figur 10. Inringat i gult visas bränsleförbrukning efter referenskörning för rutt 2.

Figur 9. Inringat i gult visas bränsleförbrukning med agentens beslut för rutt 2.

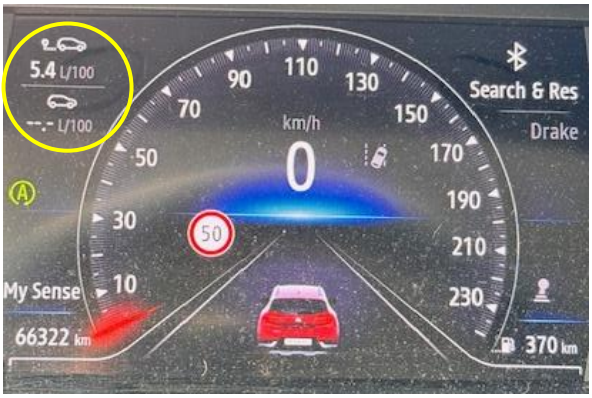
6.4.3. Rutt 3: Teleskopgatan 8 (A) till Gamlestadsvägen 311 (C)

För rutt 3 illustreras den ackumulerade energikostnaden under samtliga träningsepisoder i figur 11.



Figur 11. Ändringar i total energikostnad per episod för agentens simulerade körning på rutt 3.

Jämförelse resultatet för rutt 3 visas i figurerna nedan där den inringade cirkeln på figuren visar bränsleförbrukningen beroende på körstilen.

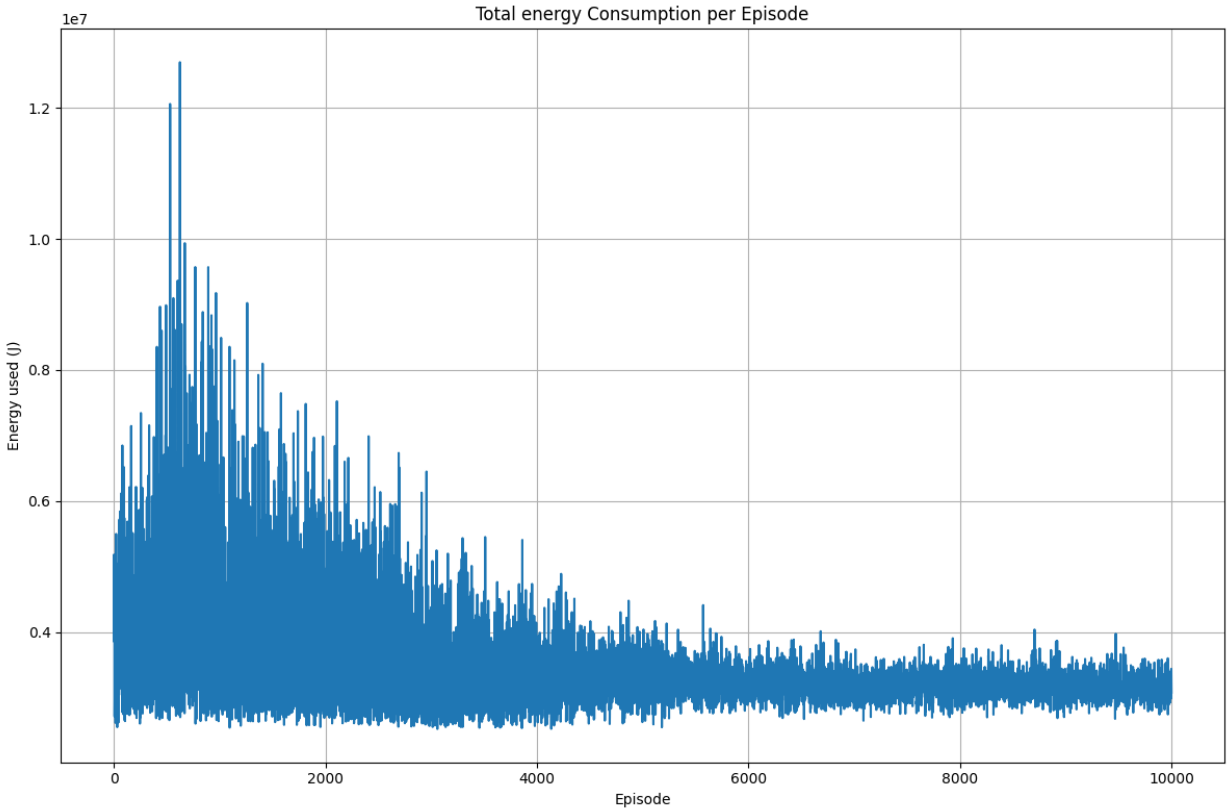


Figur 12. Inringat i gult visas bränsleförbrukning efter referenskörning för rutt 3.

Figur 13. Inringat i gult visas bränsleförbrukningen med agentens beslut för rutt 3.

6.4.4. Rutt 4: Gamlestadsvägen 311 (C) till Teleskopgatan 8 (A)

Resultatet för totala energikostnaden syns i figur 14 nedan.



Figur 14. Ändringar i total energikostnad per episod för agentens simulerade körning på rutt 4.

Resultatet för jämförelsen med traditionella körningen syns i figurerna 15 och 16.

Figur 15. Inringat i gult visas bränsleförbrukning efter referenskörningen för rutt 4.

Figur 16. Inringat i gult visas bränsleförbrukningen med agentens beslut för rutt 4.

7. Diskussion

Syftet med detta arbete var att utveckla ett system som kombinerar geografiska data med förstärkt inlärning för optimering av energiförbrukningen vid körning.

Ett av målen för projektet var att utveckla ett system som kan hämta in höjd- och väginformation via Google Maps API och genom en integrering av Directions och Elevations API kunde systemet samla in koordinater längs rutter och extrahera motsvarande lutningsdata för varje segment. Google Roads API begränsade projektet genom att inte ge möjligheten för extrahering av hastighetsbegränsningar på grund av att tjänsten bara är tillgänglig till användare som är prenumererade. Det innebar att användning av OSM var nödvändig för att kunna extrahera exakt vilken hastighetsbegränsning som gäller, från vilket väg segment, samt var ändringarna skedde. Utifrån detta implementerades en funktion som manuellt skrev in hastighetsbegränsningarna till varje segment på vägen så att tillståndsrummet korrekt kunde avläsas av agenten.

Agenten hade inte heller någon information om trafik, farthinder eller rondeller vilket påverkade realismen av simuleringen, särskilt vid utvärderingen av laglig körning och jämförelse med verkliga scenarier. Simulerade rutten hade samma hastighet som på verkliga vägar när rondellerna förekom men eftersom agenten inte tog hänsyn till trafik eller centrifugalkraftens påverkan på hastighet kunde dess rekommendation inte följas. Det ledde till att föraren istället behövde improvisera.

Belöningsfunktionen baserades på en energimodell som innehöll fyra fysikaliska komponenter: luftmotstånd, rullmotstånd, kinetisk energi och potentiell energi. Trots att energimodellen är förenklad, visar resultaten att agenten konsekvent lyckas minska energikostnaden jämfört med traditionell körning. Faktorer som trafik, väder och majoriteten av andra trafik- och externa förhållanden togs ej i åtanke när agenten bestämde rutt, vilket innebär att energivärdena inte är helt tillförlitliga i praktiken. Detta gör att de absoluta energivärdena i simuleringen inte kan jämföras direkt med faktisk energikostnad i ett fordon än, men ger en relevant referens när det kommer till energieffektiv körning. Fler upprepade tester på andra rutter, eller samma rutter hade kunnat ge ett bättre medelvärde och minskat felmarginalen. På grund av bland annat tidsbrist utgick vi endast utifrån ett fåtal tester. Ett annat litet problem med belöningsfunktionen var att det var en förenklad energimodell och inte tog hänsyn till exakt hur mycket energi som går åt när motorn verkar för att uppnå en viss hastighet.

Målet att implementera och testa en Q-learning algoritm har uppnåtts. Resultatet visade att agenten konsekvent nådde lägre bränsleförbrukning än traditionell körning för samtliga testade rutter. Detta stärker måluppfyllelsen om utveckling av en energieffektiv strategi.

Samtidigt som målet har uppfyllts, uppstod ett antal motgångar där agenten körde långsammare än en genomsnittlig förare och i vissa fall understeg den tillåtna hastighetsgränsen. Exempelvis körde den 58 km/h på en väg med 70 km/h som hastighetsgräns. Detta kan bero på att agenten prioriterade energi framför tidsoptimering men också på att extra tillagda straffar i belöningsfunktionen inte var tillräckligt anpassade till energimodellen.

En annan begränsning identifierades under testningen i form av GPS-kommunikationens responsivitet där uppdatering av vår befintliga position inte uppdaterades i realtid. Även om GPS-modulen kontinuerligt levererade positionsdata så skedde uppdateringar med betydande fördröjning. Det gjorde att vår aktuella position inte stämde överens med vår faktiska plats på rutten, vilket ytterligare orsakade att handlingar och hastigheten baserades på tidigare vägsegment i stället för den vi befann oss på. Det innebar att agenten agerade enligt instruktioner för platser den redan passerat, vilket försämrade precisionen i realtidsstyrningen. Detta ledde till att vi istället behövde använda agentens rekommenderade körstrategi som en referens, och manuellt utföra rekommenderad handling, vilket innebär att den mänskliga faktorn som fördröjd reaktion ingick och påverkade resultaten negativt. Om agenten istället var del av ett integrerat system skulle denna faktor upphäva att ha någon effekt vilket skulle ledda till bättre resultat.

Intressant nog lyckas agenten prestera bättre än motsvarande traditionella körning. I resultatet syns de olika figurerna som jämför bränsleförbrukningen. För alla rutter i resultatet syns ett tydligt mönster där agentens optimala körstrategi ledde till en märkbar minskning i bränsleförbrukningen. Detta tyder på att den inlärda strategin inte bara teoretiskt optimerar energianvändningen utan även praktiskt ger resultat i testningen.

Det största förbättringen observerades i rutt 1, där den manuella körningen resulterade i en bränsleförbrukning på 7,1 l/100km, medan den agenten påverkade körningen reducerade förbrukningen till 5,7 l/100km. Medan det minsta ändringen hände förvånansvärt på två av rutterna, rutt 2 och rutt 4, och motsvarade ändring från 8,1 l/100km till 7,4 l/100km.

I enlighet med graferna som syns i resultatet syns dessutom att den högsta totala energikostnaden var på rutt 4 där det låg på en ungefärlig 0.3e7 Joule medan de andra låg på något lägre. Skillnaden i energikostnaden är på grund av olika höjddata och hastighetsbegränsningar som rutterna behåller och exakta siffror på totala energiförbrukning speglar ingen större mening utan att det egentligen förstärker något av målen för projektet. Det är snarare den faktiska bränsleförbrukningen som ger en tydlig indikation på algoritmens effektivitet. Att minimera energiförbrukningen är ett centralt mål, men det är den direkta kopplingen mellan bränsleförbrukningen och energikostnaden som algoritmens funktion kan verifieras i praktiken.

Graferna i resultatet som visar total energikostnad per episod användes främst för att visa hur agenten ändrar sin körstrategi under träningen till att optimera mer till energibesparing. Detta tyder på att målet att utveckla, testa och utvärdera en förstärkt inlärningsalgoritm för energieffektiv körning har uppnåtts. Även med en enkel energimodell, har maskininlärning potential att identifiera mönster som är svåra att uppfatta intuitivt för en mänsklig förare. Särskilt för komplexa miljöer med varierande lutning och hastighetsbegränsning.

8. Miljö

I detta avsnitt diskuteras inverkan AI-optimerad ruttplanering har på miljön, men också hur det påverkar samhället på en etisk och social nivå.

8.1. Miljöpåverkan

AI-drivna ruttplanerings system, speciellt de som är optimerade för energieffektiv körning har stort potential att reducera miljöpåverkan som transport tillför. Den högsta förbättringen i bränsleförbrukningen var på en skillnad upp emot 20%. 20% mindre bränsle för en körd rutt är ett optimalt exempel som framtida utvecklingar kan sikta på. Det innebär 20% mindre pengar för bränsle men även 20% mindre utsläpp från transportfordon. Resultaten i detta projekt påvisade att detta koncept faktiskt funkar, men framtida uppgraderingar till systemet kan innebära ännu bättre resultat. Ett system där optimering av rutter också påverkas av trafikstockningar, ineffektiva start- och stopmoment under körning och val av väg, kan utvecklas och skulle innebära stora förändringar i trafiken.

8.2. Etisk inverkan

Den etiska inverkan av AI-drivna ruttplanerings system är kopplade till hur man hanterar data från användare, som till exempel geolokalisering av privat personer. Dessa problem förekommer och blir ännu större eftersom man inte tydligt avgör vad för policyer som används för datadelning.

Visa regioner kan också underrepresenteras då algoritmen föredrar mer komplett och mer uppdaterade data, vilket inte alltid finns tillgängligt, speciellt i socioekonomiskt drabbade områden.

Det är alltså viktigt att integritetspolicyer inkorporeras i ett ruttplaneringssystem som vill bevara en god etisk standard. Transparens i hur beslut till rutter tas är också viktigt och avgörande för att upprätthålla tillit och förtroende till allmänheten.

8.3. Samhällsmässiga och infrastrukturella konsekvenser

Ur ett samhälleligt perspektiv har AI-drivna ruttplaneringssystem potential att förbättra trivseln och den generella levnadsstandarden. Till exempel genom att utöka ruttplanerarens förmågor till att kunna planera rutter för cyklister eller fotgängare, så att allmänheten kan ta sig till platser utan att behöva oroa sig över att stötta på olika medel av transport där de inte tillhör. Men de utan tillgång till smartphones eller denna typ av AI-tjänst kan i sin tur drabbas och uteslutas från dessa fördelar vilket skapar ojämlikhet mellan olika samhällsgrupper.

Ett system som denna, som ger förslag på energieffektivt körsätt påverkar inte bara energiförbrukningen utan även förarens körbeteende över tid. Genom kontinuerlig rekommendation av specifika körstrategier kan föraren gradvis anpassa sitt beteende för att uppnå målen som systemet förespråkar. Detta kan ses som positivt men samtidigt kan det uppstå en risk där användaren blir alltför beroende av tekniken och därmed tappar sin egen förmåga att fatta snabba och korrekta beslut i komplexa trafiksituationer.

Utöver detta kan ett system som denna även uppfattas som påtvingande eller överkontrollerande, särskilt om det används i yrkestrafik där prestationsmätning kan kopplas till energieffektivitet. I sådana fall kan systemet upplevas som att den tar över förarens ansvar och initiativ, vilket påverkar arbetsmiljön och säkerhetskänsla negativt. Det är därför viktigt att systemet utformas med hänsyn till användarnas självbestämmande och att det fungerar som stöd, snarare än ersättning av mänskligt omdöme.

AI-driven ruttplanering har också stor utvecklingspotential och kan skalas upp till något som används av majoriteten av förare i ett samhälle. Även fast detta gynnar samhället av den orsaken att trafiken kan bli mer effektiv och miljöpåverkan kan minskas, kan det i andra hand oavsiktligt skapa ojämlikheter mellan olika områden när det kommer till regleringen av trafik. Med detta menas hur algoritmen väljer att optimera förarnas rutter så att trafikflödet delas upp jämnt, och inte till exempel styra bort trafik från mer centrala delar och istället mot mindre orter. AI-driven ruttplanering är alltså något som behöver hanteras på ett bra sätt så att det ska kunna bidra till ett mer hållbart och rättvist samhälle, vilket är något som kräver tid, ansvar och diskussion.

9. Slutsats

Samtliga mål i projektet har uppnåtts. Ett fungerande system för energibaserad optimering av körning med hjälp av förstärkningsinlärning har utvecklats och testats, och resultaten visar att metoden har potential för framtida tillämpningar.

För framtida projekt hade belöningsfunktionen kunnat baseras på mer detaljerade energi formler som ger agenten förståelse på hur mycket energi förloras men även identifierar möjligheter att aktivt spara energi genom strategiska körbeslut. Ett konkret exempel på en sådan förbättring skulle vara att agenten inför varje beslut kan analysera den kommande vägens lutning och anpassa sin strategi därefter. Detta gjordes i detta projekt men i form av extra belöningar och straffar vilket inte är visar tillräckliga fakta. Om samma princip appliceras i form av fysiska formler eller beräkningar kan agenten i stället välja att utnyttja kinetisk energi för att vinna över en backe med minimal bränsleförbrukning. Denna förbättring skulle innebära att agenten lär sig en strategi där energioptimering inte bara handlar om förluster utan även tillfällen för att utnyttja hastigheten till sin fördel.

Hastighetsändringar som sker när agenten väljer en handling hade dessutom kunnat göras mer matematiskt och exakt. Möjligheten för agenten att själv välja vilken hastighet den vill accelerera till skulle kunna förbättra energiåtgången under en rutt. I detta projekt är den begränsad till att hastighetsändringar är fasta, där varje handling leder till en förbestämd ökning i hastighet.

När det kom till testningen och verifieringen förekom en del problem där uppdateringsfrekvensen av agentens beslut inte var tillräckligt snabba, som också nämndes i kapitel 4.6. Detta var förväntat eftersom hårdvaran som användes inte var helt optimerad för att hantera programmet, speciellt när snabba uppdateringar behövdes. För framtida tester och implementationer rekommenderas att agenten integreras i ett dedikerat system i fordon eller körs på en mer optimerad hårdvaruplattform. Detta kommer sannolikt förbättra både prestanda och tillförlitlighet i den praktiska tillämpningen.

En annan möjlig förbättring eller projekt kan vara att utforska mer kring hur feedbacken till användaren kan byggas upp. På vilket sätt är det visuellt bäst för användaren att ta in information under körningen. Ett tillägg på det är även att vidare utveckla själva uppdateringen och se över exakta problemet kring varför användarens befintliga position tar längre tid att jämföra med optimala rutten. Kan det bero på hårdvaran eller mjukvaran?

Användning av informella källor och AI-verktyg

Under projektet har informella källor och AI-baserade verktyg använts som stöd i de inledande faserna av informationssökning och begreppsförståelse. Utöver detta har AI-verktyg som ChatGPT använts för att verifiera vissa tekniska förklaringar, få feedback på struktur och korrektur av formuleringar. All text i rapporten är dock skriven av rapportförfattarna själva.

10. Källförteckning

[1] K. Puri, "Route Optimization: An Eco-Friendly Logistics Solution," FarEye, [Online]. Tillgänglig: https://fareye.com/resources/blogs/route-optimization-eco-friendly-logistics, Hämtad: 2025-05-22.

[2] R. S. Sutton, *Reinforcement Learning*. Springer Nature, 1992. doi: https://doi.org/10.1007/978-1-4615-3618-5.

[3] R. S. Sutton, *Introduction to reinforcement learning*. Cambridge, Mass: Mit Press, 04-98, 1998.

[4] J. Wang, I. Besselink, and H. Nijmeijer, "Battery electric vehicle energy consumption modelling for range estimation," *International Journal of Electric and Hybrid Vehicles*, vol. 9, no. 2, p. 79, 2017, doi: https://doi.org/10.1504/ijehv.2017.085336.

[5] "Motion Along Inclined Planes | Brilliant Math & Science Wiki," *Brilliant.org*, 2016. [Online]. Tillgänglig https://brilliant.org/wiki/motion-along-inclined-planes/ (hämtad: 2025-03-25).

[6] "3.5: Drag Using Conversion of Energy," *Engineering LibreTexts*, 2020. [Online]. Tillgänglig: https://eng.libretexts.org/@go/page/24097 (hämtad: 2025-03-25).

[7] "Google Cloud APIs," *Google Cloud*, 2025. https://cloud.google.com/apis/docs/overview (hämtad: 2025-03-02).

[8] Google for Developers "Directions API overview," 2025. [Online]. Tillgänglig: https://developers.google.com/maps/documentation/directions/overview (hämtad: 2025-03-02).

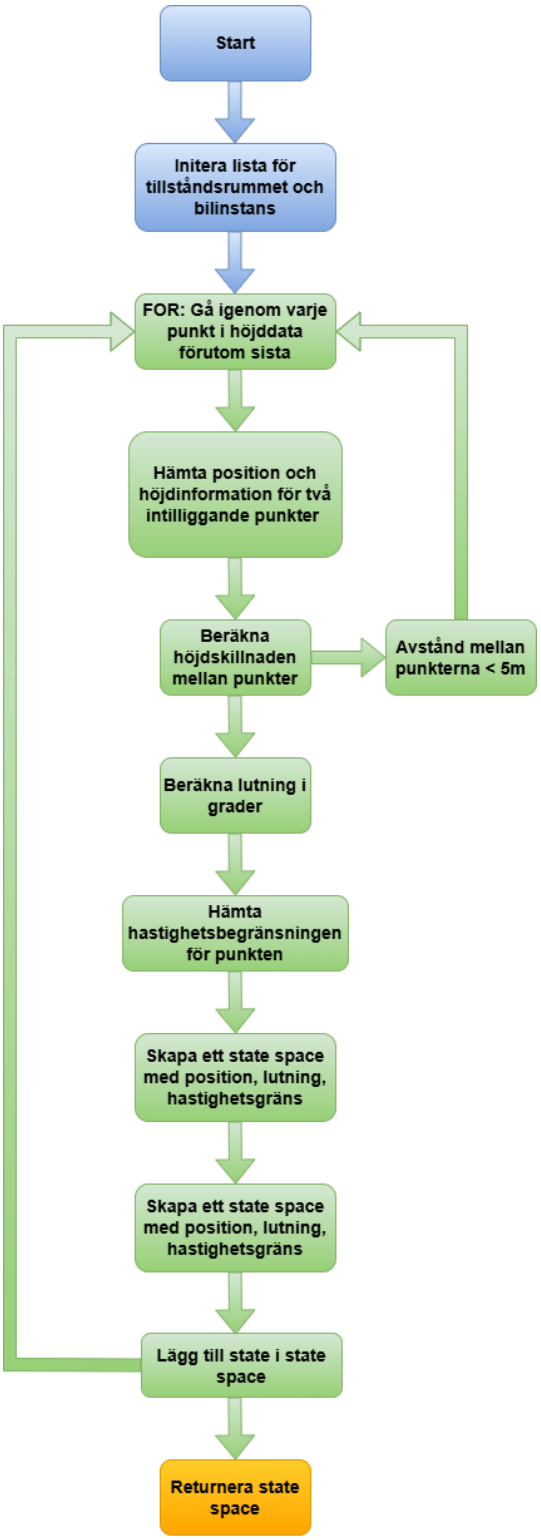
[9] Google for Developers "Elevations API overview," 2025. [Online]. Tillgänglig: https://developers.google.com/maps/documentation/elevation/overview (hämtad: 2025-03-02).

[10] Google for Developers "Roads API overview," 2025. [Online]. Tillgänglig: https://developers.google.com/maps/documentation/roads/overview (hämtad: 2025-03-18).

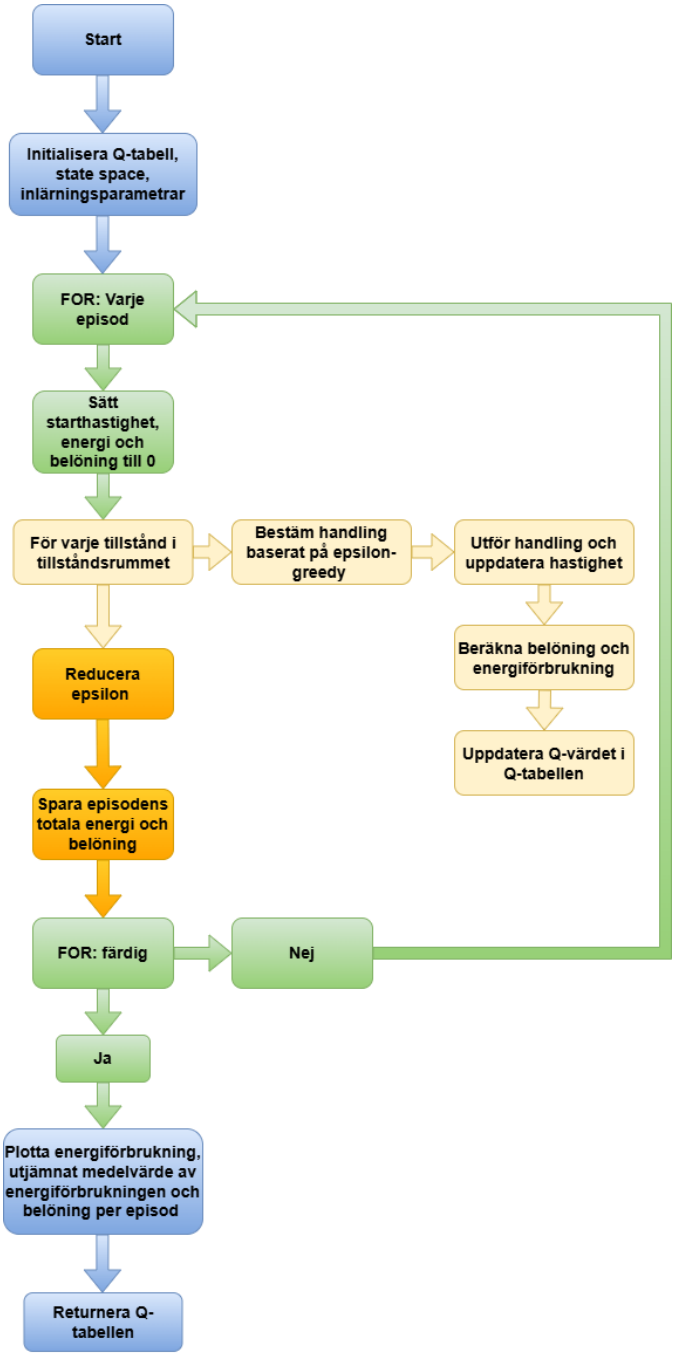
[11] OpenStreetMap, "OpenStreetMap", 2025. [Online]. Tillgänglig: https://www.openstreetmap.org/about (hämtad: 2025-04-06).

11. Bilagor

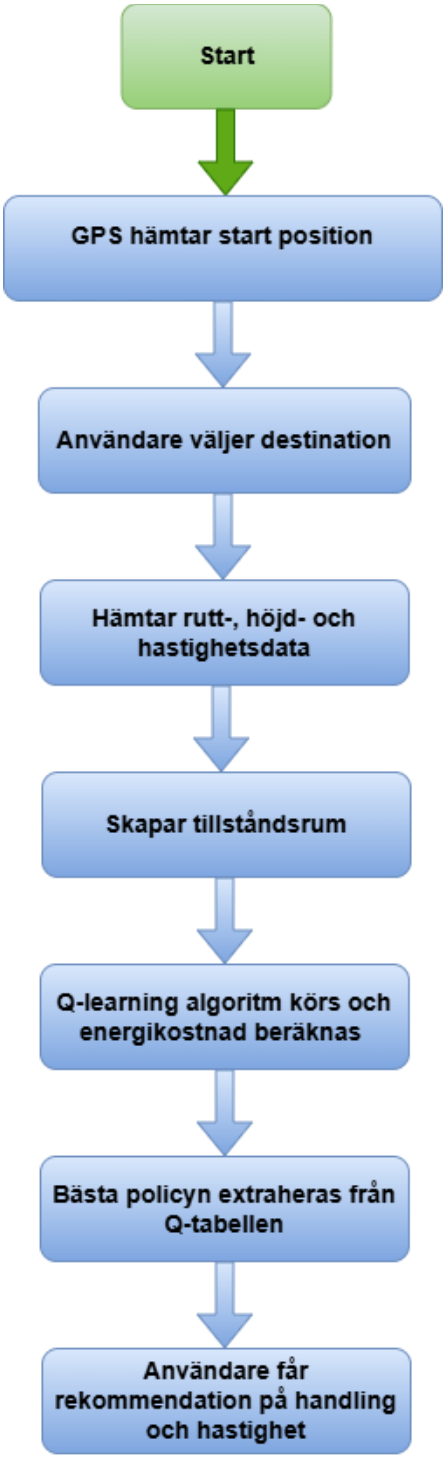
Bilaga A. Flödesschema för utveckling av tillståndsrummet



Bilaga B. Flödesschema för inlärningsalgoritmen



Bilaga C. Översiktlig flödesschema över hela koden



Bilaga D. Exempel på utskriften för optimala körningen

State 89: Pos: (57.75562066329401, 12.04368770220632), Tilt: -2.26°, Speed Limit: 70 km/h → Current Speed: 49 km/h → Best Action: Accelerate

State 90: Pos: (57.75556820000001, 12.0436345), Tilt: -3.71°, Speed Limit: 70 km/h → Current Speed: 52 km/h → Best Action: Accelerate

State 91: Pos: (57.75525829999999, 12.0432564), Tilt: -3.24°, Speed Limit: 70 km/h → Current Speed: 55 km/h → Best Action: Accelerate

State 92: Pos: (57.75503029999999, 12.0429282), Tilt: -3.63°, Speed Limit: 70 km/h → Current Speed: 58 km/h → Best Action: Accelerate

State 93: Pos: (57.7549113, 12.0427566), Tilt: -4.33°, Speed Limit: 70 km/h → Current Speed: 61 km/h → Best Action: Accelerate

State 94: Pos: (57.75438740000001, 12.0419151), Tilt: -2.67°, Speed Limit: 70 km/h → Current Speed: 64 km/h → Best Action: Maintain

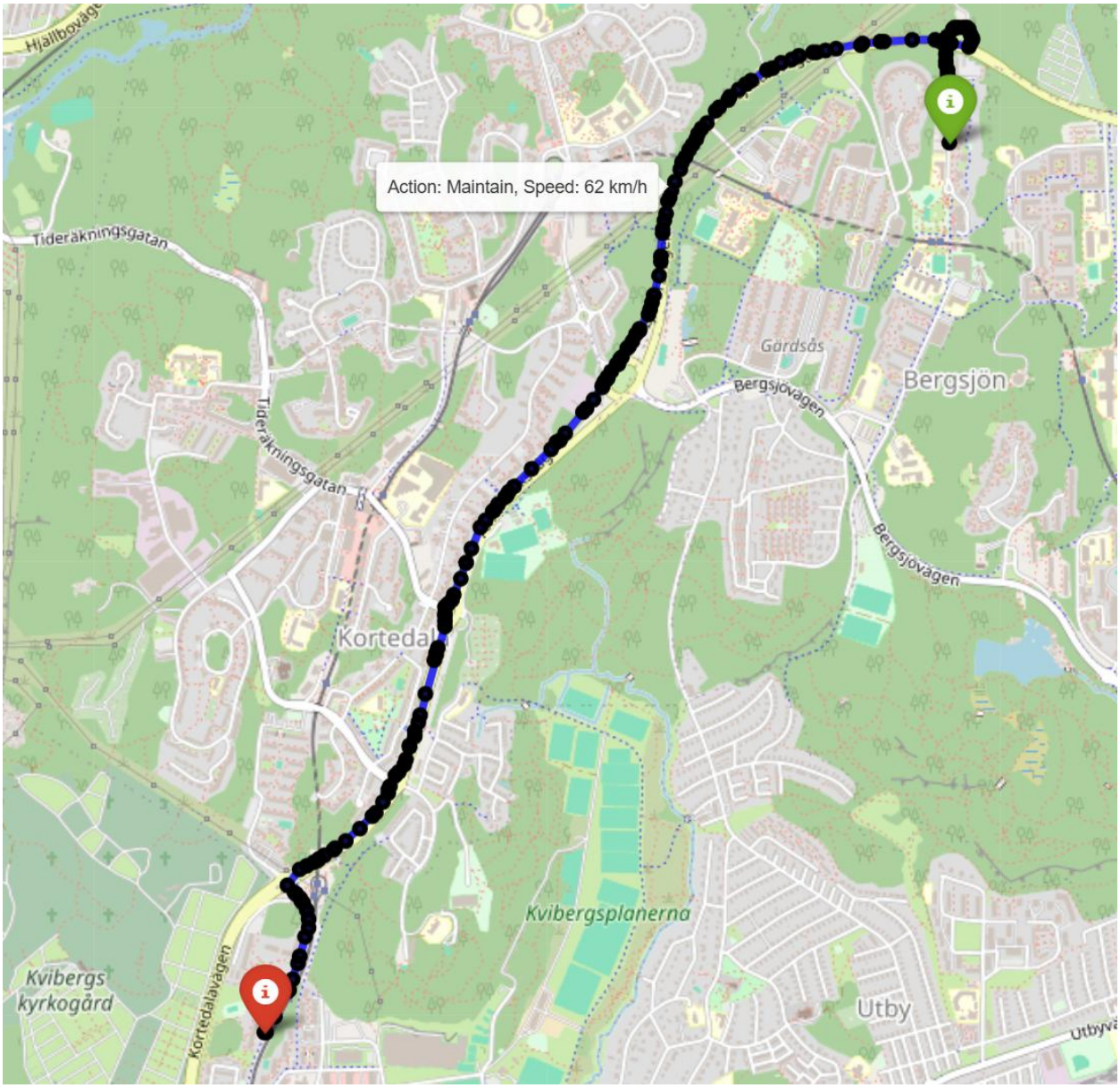
State 95: Pos: (57.75421220009541, 12.04160562479445), Tilt: -1.21°, Speed Limit: 70 km/h → Current Speed: 64 km/h → Best Action: Maintain

State 96: Pos: (57.7541314, 12.0414629), Tilt: -2.66°, Speed Limit: 70 km/h → Current Speed: 64 km/h → Best Action: Maintain

State 97: Pos: (57.7539592, 12.0411589), Tilt: -2.17°, Speed Limit: 70 km/h → Current Speed: 64 km/h → Best Action: Maintain

State 98: Pos: (57.75343837886848, 12.04021313588834), Tilt: -1.54°, Speed Limit: 70 km/h → Current Speed: 64 km/h → Best Action: Maintain

Bilaga E. Karta över agentens rekommenderade handlingar och hastighet



INSTITUTIONEN FÖR DATA- OCH
INFORMATIONSTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2025
www.chalmers.se

CHALMERS