Prezentace: Učení AI modelů pro řízení autíček na trati

Tým Zero

Slide 1. Úvod

Naším úkolem je naučit agenta projet trať za použití 9 paprskových senzorů a informace o rychlosti. Ovládání zajišťují čtyři akce: plyn, brzda, vlevo, vpravo. Hlavním cílem je, aby algoritmus dělal smysluplná rozhodnutí v různorodých situacích a zvládal se adaptovat.

Slide 2. Prostředí a data

Na vstupu máme 9 čísel ze senzorů a normalizovanou rychlost. Výstupní akce mají hodnoty od 0 do 1, přičemž akce je provedena reálně až při překročení hodnoty 0.5. Klíčové je zohlednění bezpečnosti: pokud agent nebere v úvahu vzdálenost od stěn, velmi rychle se „zničí“. V nových verzích jsme přidali dynamickou detekci „bezpečné zóny“, abychom snížili počet havárií – i když to zvyšuje složitost rozhodování.

Slide 3. Model č. 1 – Behavior Tree (omezení klasické logiky)

Behavior Tree (`AI_engines/AIbrain_BehaviorTree.py`) představuje plně transparentní a interpretovatelnou logiku. Výhoda: snadná úprava prahů nebo podmínek. Nevýhoda: vyžaduje spoustu ručního ladění pro každou trať/fyziku; i malé změny v mapě rozbijí scénáře. Dále má nízkou flexibilitu a velmi rychle narazí na strop – zlepšení je možné jen výrazným zvyšováním složitosti stromu. Proto jsme téměř okamžitě přešli na Q-learning, abychom se vyhnuli ručnímu ladění.

Slide 4. Model č. 2 – Q-learning (zásadní limity a vylepšení)

Q-learning (`AI_engines/AIbrain_QLearning.py`) staví Q-tabuli a agent se učí online. Zpočátku metoda vypadala slibně, ale rychle jsme narazili na problém exponenciálního růstu paměťových nároků – soubor s tabulkou Q-values narostl do obřích rozměrů, což zpomalovalo ukládání i načítání dat. Často docházelo k chybám při čtení, záznamy se někdy poškodily nebo jsme ztratili velké množství epizod. Po přechodu na jednodušší formát se situace ještě zhoršila a docházelo k dlouhým zaseknutím systému. Snažili jsme se problém zmírnit pravidelnou aktualizací vah a předáváním zkušeností mezi mapami a epochami, ale nikdy se nám nepodařilo potíže zcela odstranit.
Celkově byla integrace mechanismu rychlého ukládání a načítání velmi bolestivá, protože se soubor často přepisoval od začátku a museli jsme zásadně upravit kód samotné hry.

Slide 5. Model č. 3 – Neuronová síť (MLP: funkční řešení a speciální optimalizace)

Třetí přístup – vícevrstvý perceptron (MLP, `AI_engines/AIbrain_Zero.py`). Vstup: 9 paprskových senzorů a rychlost; první skrytá vrstva má 24 neuronů (ReLU), druhá 16 (ReLU). Výstup: 4 hodnoty (plyn, brzda, vlevo, vpravo) se sigmoid aktivací. Akce je aktivní při >0.5, ale pomocná logika zajišťuje, že plyn a brzda, případně levý a pravý směr, nemohou být aktivované současně.

Pro optimalizaci ovládání a zvládání zatáček jsme implementovali několik mechanismů přímo v `AIbrain_Zero.py`:

- **Dynamické určení "otáčkového režimu"** – síť zpracovává informace o průměrné vzdálenosti vlevo/vpravo, porovná rozdíl ("balance") a podle prahu pozná, kdy je potřeba výrazněji zatočit. Pokud agent detekuje ostrý rozdíl, sníží se cílová rychlost a priorita přechází na plynulé bezpečné projetí, nikoli maximální rychlost ("target speed turn" vs. "target speed straight").
- **Prevence srážky s hranou** – pokud je vzdálenost od stěny menší než určený práh, aktivuje se brzda nebo se diferencuje řízení na opačnou stranu. Při detekci kritické blízkosti agent omezí plyn a nuceně zatočí od překážky.
- **Automatické škálování paprsků** – normalizace hodnot paprsků podle reálného maxima na mapě, aby síť nezávisela na konkrétní scéně a lépe rozeznávala situace v různých prostředích.
- **Bezpečnostní "coasting"** – pokud je vpředu volno, plyn se zvýší, při hrozbě srážky je rychle aktivována brzda a vypnut plyn.
- **Policy gradient pro efektivní učení** – akce nejsou pouze mutované, ale učí se pomocí zpětné vazby z dosažených odměn, což výrazně urychluje adaptaci.
- **Oddělení plynu a brzdy, rozdělení směrových akcí** – plyn s brzdou a levý/pravý nesmí být současně → síť je k tomu vedená speciální maskovací/mutující funkcí v rozhodovací logice.

Díky těmto mechanizmům se agent naučil nejen rychle projíždět rovinky, ale hlavně adaptovat své chování v ostrých zatáčkách a vyhýbat se překážkám bez zbytečných srážek.

Slide 6. Inference přes MLP (ukázka algoritmu)

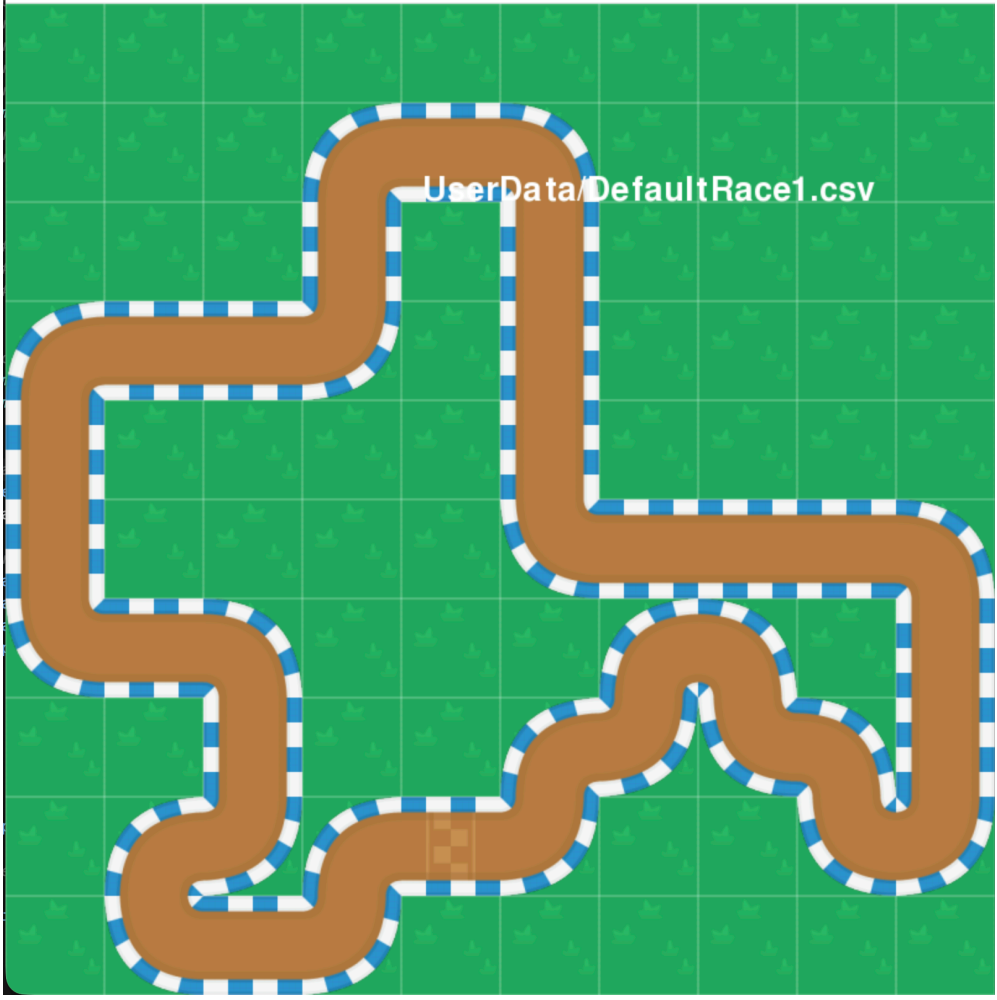
Vstupní data se násobí vahami první vrstvy, přičte se bias, následuje ReLU; totéž se opakuje na druhé vrstvě. Výstup proženeme sigmoidem. Poté rozhodovací funkce převede hodnoty na diskretní akce podle prahu >0.5. Navíc je tam speciální kontrola, která nedovolí, aby byly současně aktivní plyn a brzda, nebo vlevo a vpravo – tím minimalizujeme chyby řízení a podporujeme plynulý pohyb po trati.

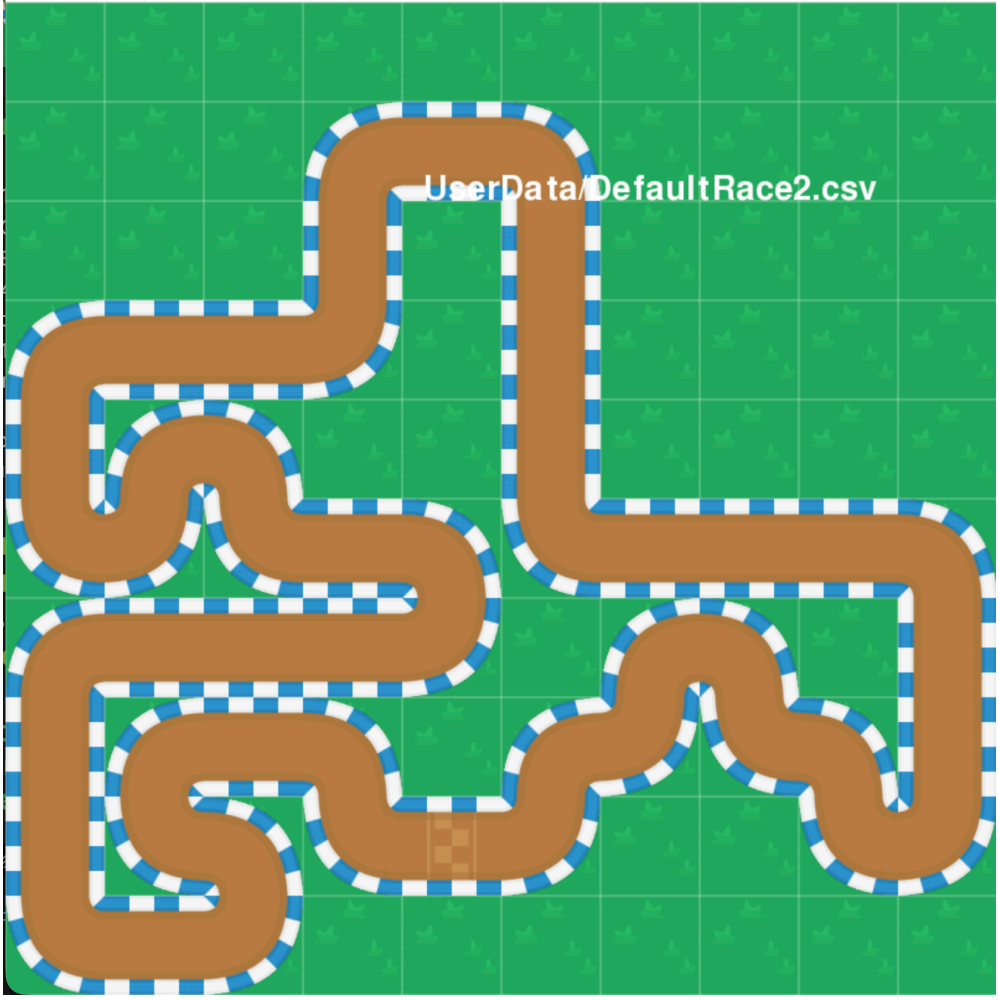
Slide 7. Proč byl MLP lepší než ostatní

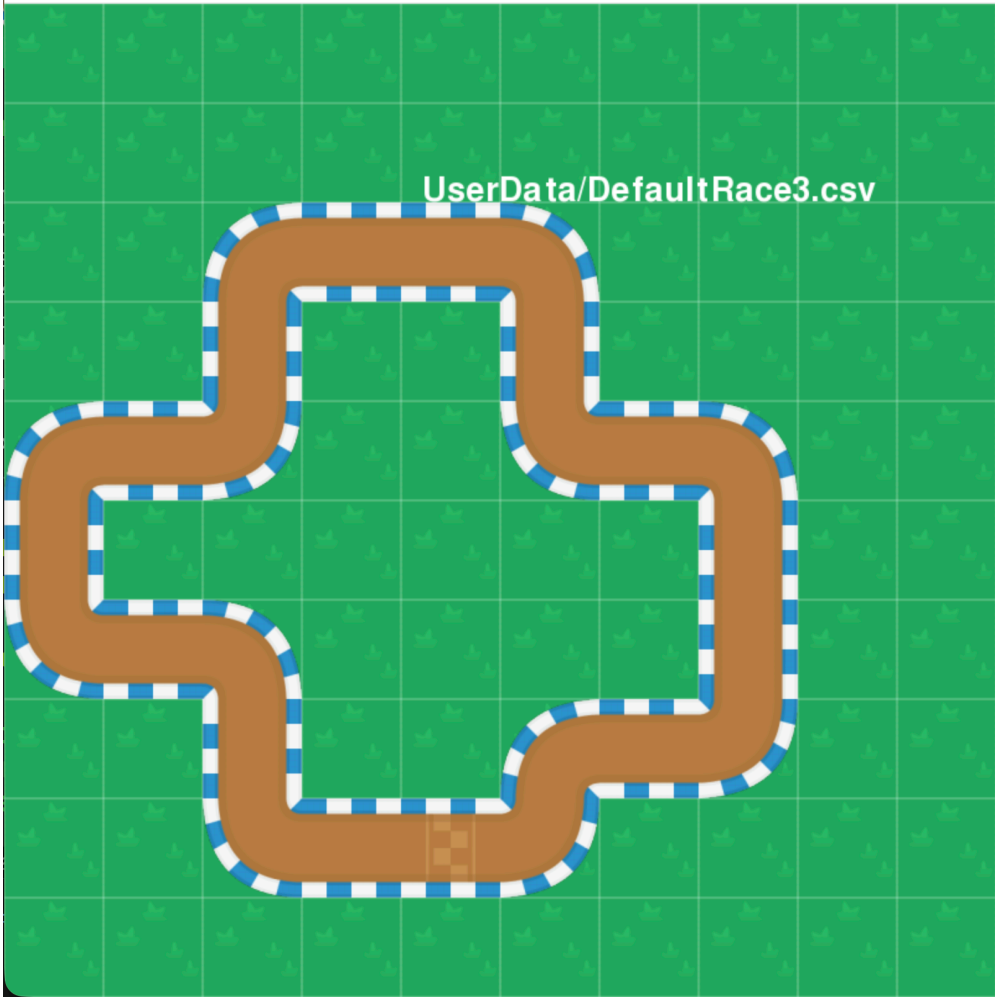
Hlavní výhody: kompaktní velikost váh (nedochází k „výbuchu“ paměťových nároků), MLP se dobře škáluje a lze jej učit mutacemi i gradientově (policy gradients). Síť dobře přenáší znalosti mezi různými tratěmi. Díky policy gradient dosahujeme rychlé adaptace na nové mapy a samotný model zůstává malý – jen pár desítek kilobajtů.

Slide 8. Použité tratě a mapa učení

Trénování probíhalo na třech mapách – `Map1.png`, `Map2.png`, `Map3.png`. V nových testech jsme přidali další mapy pro ověření obecnosti učení.


UserData/DefaultRace1.csv


UserData/DefaultRace2.csv



Slide 9. Závěry

Behavior Tree je dobrý a rychlý začátek, ale není škálovatelný ani dobře přenositelný. Q-learning přidal flexibilitu, ale narazil na technologické limity velikosti a ukládání. MLP (neuronová síť) je střední zlatá cesta: stabilní, škálovatelná, s velkým potenciálem dalšího učení a adaptace i v omezených prostředích. Poslední vylepšení výrazně zvýšila generalizaci na nových tratích a odolnost vůči náhodným chybám.

Slide 10. Další kroky

Momentálně pracujeme na:

- automatickém určování a adaptaci „bezpečné zóny“
- tréninku na nových a složitějších mapách
- integraci fyziky zatáčení a kontroly úhlu
- experimentech s různými typy odměňování a trénovacích algoritmů (policy gradient, evoluce, hybridní přístupy)
- rozvoji monitoringu a vizualizace výuky pro rychlou analýzu chyb