STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

Sofie – neuronová síť inspirovaná háďátkem obecným a možnosti její kontroly

Patrik Vácal
Plzeňský kraj

Plzeň 2021

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

Sofie – neuronová síť inspirovaná háďátkem obecným a možnosti její kontroly

Sofia – neural network inspired by Caenorhabditis elegans and possibilities of its control

Autoři: Patrik Vácal
Škola: Vyšší odborná škola a Střední průmyslová škola elektrotechnická Plzeň,
Koterovská 85, 326 00 Plzeň
Kraj: Plzeňský kraj
Konzultant: Mgr. Karel Vlna

Plzeň, 2021

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval samostatně a použil jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Plzni dne 15. března 2021 ……

Patrik Vácal

Poděkování

Rád bych poděkoval mému vedoucímu práce Mgr. Karlu Vlnovi za odborné vedení práce. Dále bych chtěl poděkovat Tereze Žatkovičové a Davidu Krásnému za jejich pomoc při stylystické úpravě práce. V poslední řadě bych rád poděkoval komunitě simulátoru PGDrive za jejich přátelský přístup při řešení problémů.

Anotace

Ve své práci SOČ jsem se zabýval možnostmi ovládání nového typu rekurentní neuronové vrstvy NCP inspirované biologickou neuronovou sítí háďátka obecného. V práci jsem navrhl a zrealizoval experiment na případu samořídících aut. Naučil jsem neuronovou síť držet se jízdního pruhu a v případě externího impulzu pruh změnit. Výsledkem experimentu je důkaz, že je vrstva NCP může být kontrolována vstupem rozkazu, který má vykonat. Následně jsem zmapoval vliv jednotlivých hyperparametrů vrstvy na výsledky sítě a navrhl možnosti navazujícího výzkumu.

Klíčová slova

Umělá inteligence; rekurentní neuronové sítě; samořídící auto

Annotation

In my Students` Professional Activity, I researched a possibility of controlling a new type of recurrent neural layer NCP inspired by the biological neural network of Caenorhabditis elegans. In this work I designed and realized an experiment in the case of self-driving cars. I trained the neural network to stick to the traffic lane and change the lane in case of an external impulse. The result of the experiment is proof that the NCP layer can be controlled by the input of the command to be executed. Subsequently, I mapped the influence of individual hyperparameters of the layer on the results and suggested follow-up research.

Keywords

Artificial intelligence; Recurrent neural networks; Self-driving car

Obsah

1 Úvod

Neuronové sítě se již od počátku hojně inspirují u jejich biologických příbuzných. Za jednu z prvních funkčních neuronových sítí schopných se učit můžeme považovat The Perceptron (Rosenblatt, 1957). Tato neuronová síť byla jednovrstvá a umožňovala pouze řešení lineárních problémů. Následující desetiletí se objevily hluboké neuronové sítě, které stavěly na vícevrstvé architektuře. Za další významný pokrok by se dal považovat vynález konvolučních neuronových sítí. (LeCun, a další, 1989) Tato konvoluční architektura byla inspirována pozorováním zrakové oblasti v mozku zvířete. (Hubel & Wiesel, 1962) V oblast strojového překladu, a ne jenom tam, nás velmi posunula inspirace krátkodobou pamětí pro vytvoření rekurentních neurnových sítí. V posledních letech také algoritmy věnování pozornosti určitým částem vstupu víc než jiným. (Vaswani, a další, 2017)

Tato práce se zabývá novým typem rekurentní neuronové sítě NCP (Lechner, a další, 2020) blízce inspirované biologickou neuronovou sítí háďátka obecného, u kterého je známa přesná konfigurace jeho neuronové sítě. (White, Southgate, Thomson, & Brenner, 1986) Vzhledem k předchozím skokům v oblasti neuronových sítí díky inspiraci jejich biologickým protějškem má tento typ sítě velký potenciál.

Cílem práce je potvrdit či vyvrátit hypotézu, že je možno této síti předávat rozkazy a ona podle nich uzpůsobí své chování. V případě potvrzení hypotézy bude dalším dílčím úkolem nalezení vhodné konfigurace této sítě pro daný úkol. Úkolem neuronové sítě je jízda v pruhu po tříproudé silnici. V případě, že dostane impulz pro změnu pruhu, tak by měla tento příkaz provést. Pokud bude schopno vozidlo zůstat na vozovce a změnit v případě impulzu pruh, bude experiment považován za úspěšný. V opačném případě bude vyvrácena platnost hypotézy.

Veškerý kód vytvořený během této práce lze nalézt na GitHubu na adrese: *https://github.com/gamecraftCZ/sofia-soc*

2 TEORETICKÝ ÚVOD

2.1 Klasifikace strojového učení

Vzhledem k zaměření této práce budou v následující části popsány pouze algoritmy a přístupy spojené se strojovým učením neuronových sítí.

2.1.1 Supervised learning

Metody strojového učení se dají rozdělit do tří kategorií. První z nich je takzvaný supervised learning neboli učení neuronové sítě na datech, které jsou označeny (velmi často ručně). Neuronová síť v tomto případě ví, k jakému vstupu patří jaký výstup, a snaží se tuto korelaci mezi vstupem a výstupem naučit. (IBM, 2020)

Příkladem supervised learning může být například detekce a určení objektů na videu, detekce spamu, či „předpověď" budoucnosti podle předchozích zkušeností. (IBM, 2020)

Ve své práci používám právě tento způsob učení. Dalo by se říci, že neuronová síť, stejně jako malé dítě, „vypozoruje", jakým způsobem řídím já, a poté můj způsob řízení napodobuje.

2.1.2 Unsupervised learning

Druhou kategorií je unsupervised learning. V tomto přístupu neuronová síť dostane pouze data, avšak na rozdíl od předchozího přístupu je nemá nijak označené. Jejím úkolem je najít korelaci mezi daty nebo najít části dat vybočující z normálu. Využívá se například k detekci anomálií, v doporučovacích algoritmech, či ke kategorizaci novinových článků. (IBM, 2020)

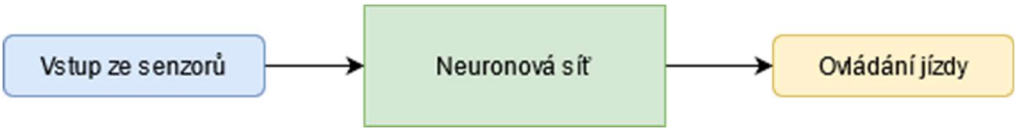
2.1.3 Reinforced learning

Poslední kategorií je reinforced learning neboli učení pomocí zpětné vazby. Tento přístup se velice podobá přístupu dítěte k doposud neprobádanému světu. Neuronová síť dostane k dispozici určité prostředí a vydá se jej objevovat. Za každý svůj pokus dostane určitou odměnu a tu se postupně snaží maximalizovat. (Hu, Hanlin, Carrasco, Lennox, & Arvin, 2020)

2.2 End-To-End neuronové sítě pro samořídící auta

Přístupy k samořídícím systémům jsou dvojího druhu. Prvním z nich je systém modulární. V tomto přístupu se dělí přístup k problému na více modulů. Příkladem mohou být moduly pro detekci objektů, plánování trasy, či modul bezpečnostní. Ve výsledku se výstupy z těchto modulů kombinují v ovládací příkazy pro vozidlo.

Druhým přístupem je „End-to-End", což v překladu znamená „od začátku do konce". V kontextu neuronových sítí to znamená trénování celého systému jako jednoho celku. V tomto přístupu není systém dělený do modulů, ale všechny moduly jsou zahrnuté do jedné neuronové sítě. Vstupem této sítě se pak stávají nezpracovaná data z čidel a senzorů a výstupem ovládací příkazy pro vozidlo. (Glasmachers, 2017) (**Obrázek** 1)



Obrázek 1: Návrh End-to-End sítě pro autonomní vozidlo. (vlastní tvorba)

V historii samořídících aut se můžeme vůbec poprvé setkat s End-to-End přístupem v roce 1989. Autonomous Land Vehicle in a neural network (ALVINN) se skládal ze tří zcela propojených vrstev a byl schopný autonomní jízdy za dobrého počasí. Na svou dobu byl ALVINN průkopnickou prací pro oblast End-to-End autonomních vozidel. (Pomerleau, 1989)

Na projekt ALVINN navázala v roce 2004 americká DARPA[1] s výzkumným projektem DAVE. Neuronová síť projektu DAVE již používala konvoluční neuronové sítě, a umožnila tak efektivnější zpracování obrazu ze vstupu. Průměrná ujetá vzdálenost v komplexních prostředích byla u projektu DAVE 20 metrů. Výstupem projektu byl především fakt, že nebyl v té době k dispozici dostatečně výkonný hardware pro praktické využití. (Lecun, Cosatto, Ben, Muller, & Flepp, 2004)

V posledních letech bych zmínil projekt od firmy Nvidia (Bojarski, a další, 2016) navazující na projekt DAVE a výzkum z Kolumbijské univerzity (Gu, Li, Di, & Shi, 2020) zkoumající význam rekurentní sítě LSTM v kontextu End-to-End neuronových sítí pro autonomní vozidla.

Do této kategorie se řadí neuronová síť navržená v rámci této práce.

2.3 Rekurentní neuronové sítě

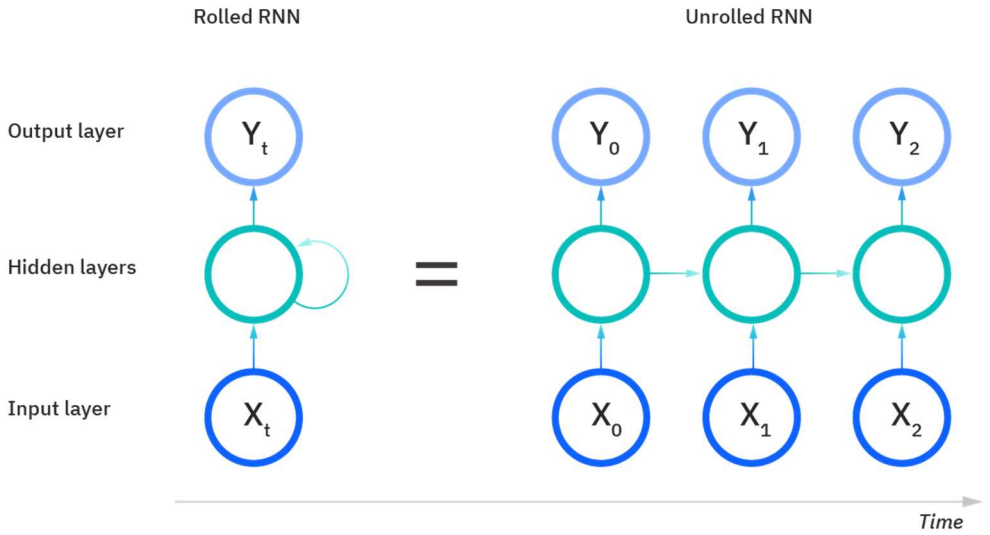
Pokud bychom se podívali na obrázek auta jedoucího po přerušované čáře, bude těžké říci, zda přejíždí z levého pruhu do pravého nebo naopak. U neuronových sítí je to stejné. K řešení tohoto problému se přistupuje několika různými způsoby. V prvním z nich se na vstup neuronové sítě přivedou i vstupy z minulosti. Toto triviální řešení je však neefektivní. S potřebou jít více do minulosti rychle roste velikost sítě. Dalším problémem je fakt, že pro každý bod v čase jsou samostatné váhy v neuronové síti, a tím se zhoršuje generalizace sítě. Druhým, efektivním řešením, se ukázaly být rekurentní sítě.

Rekurentní neuronové sítě se používají k práci se sekvenčními daty. Běžně se využívají ke zpracování psaného textu či rozpoznávání řeči. Rekurentní vrstva sdílí váhy mezi všemi časovými kroky, díky čemuž má síť lepší generalizační vlastnosti. (**Obrázek 2**) Nejjednodušší implementací rekurentní vrstvy je přivedení výstupu z minulého časového kroku na vstup kroku následujícího. Tento přístup je však velmi náchylný na vanishing gradient problem, kdy není neuronová síť schopna se efektivně či vůbec učit. Nejpopulárnější architekturou rekurentní vrstvy, která zmíněný problém řeší, je LSTM[2]. LSTM obsahuje vnitřní jednotku, která si „pamatuje“ důležité historické informace. Které informace si zapamatovat a které zapomenout

[1] Defense Advanced Research Projects Agency (česky: Agentura ministerstva obrany pro pokročilé výzkumné projekty)
[2] Long short-term memory

kontrolují tři vnitřní brány, které se během procesu trénování učí. Za zmínku stojí také používaná architektura GRU[3] podobná LSTM. Rozdílem je zde kombinace brány pro zapamatování nových informací a pro zapomenutí starých do jedné jediné. (IBM, 2020)

Mezi rekurentní neuronové vrstvy se řadí také vrstva NCP, kterou se tato práce zabývá.



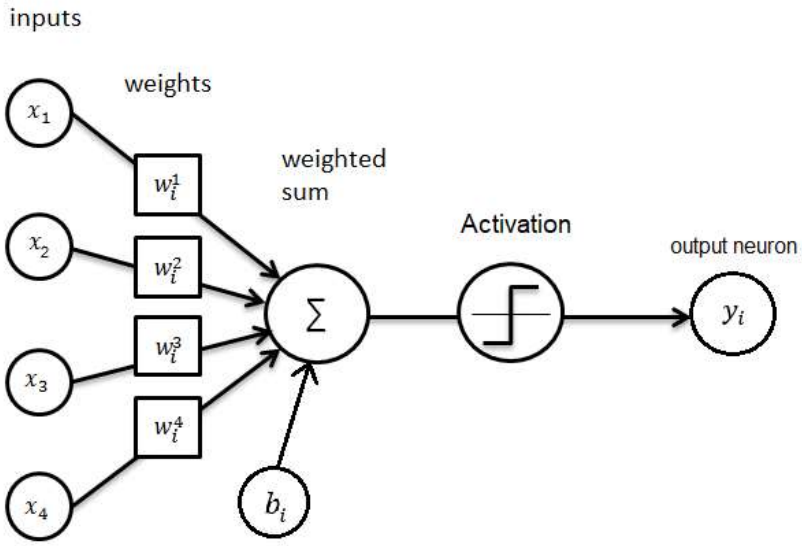
Obrázek 2: Rekurentní neuronová síť. Jednotlivé časové kroky sdílejí stejné aktivační váhy. Výtup každého kroku závisí na kroku předchozím. Převzato z (IBM, 2020)

2.4 Použité typy neuronových vrstev v experimentu

2.4.1 Zcela propojená vrstva

V této vrstvě je propojen každý neuron x_i ze vstupní vrstvy x s každým neuronem y_j z vrstvy y s váhou propojení W_i^j. Každý neuron y_i také může mít na vstupu bias b_i ovlivňující výslednou hodnotu neuronu. (**Obrázek 3**)

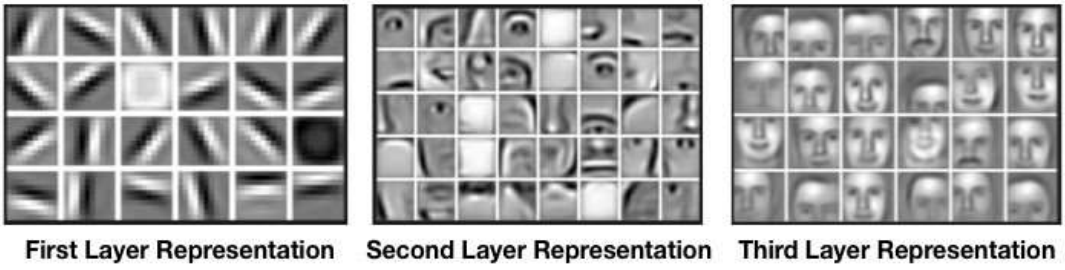
[3] Gated recurrent unit



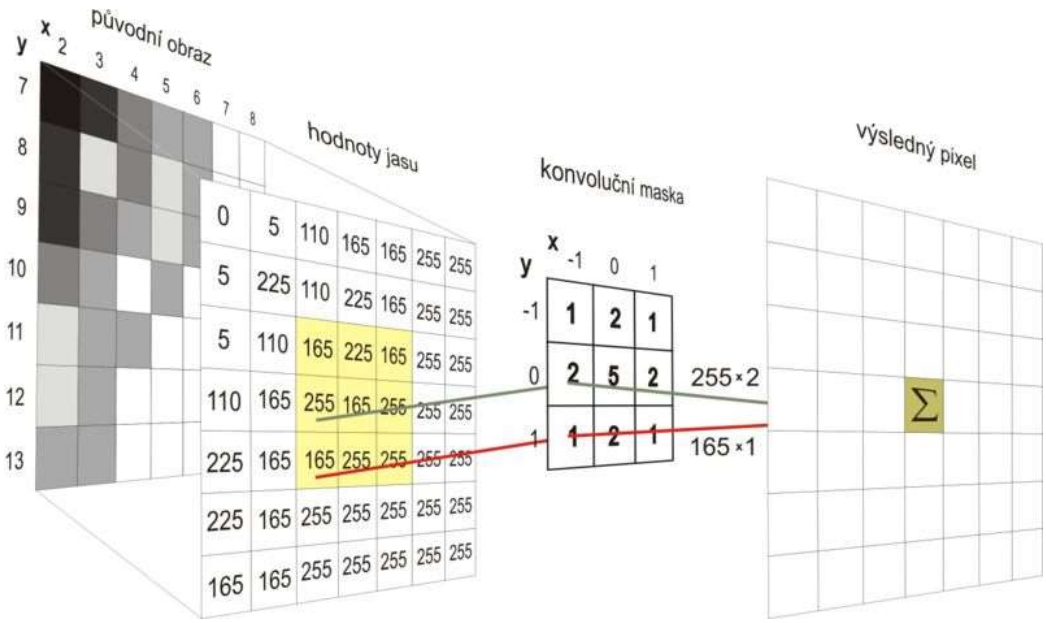
Obrázek 3: Grafické znázornění výpočtu hodnoty neuronu y_i zcela propojené vrstvy. Převzato z (Ognjanovski, 2020) upraveno.

2.4.2 Konvoluční vrstva

Zcela propojené vrstvy se ukázaly pro obrazové data jako velmi neefektivní. V praxi se proto používají vrstvy konvoluční. V architektuře této vrstvy jsou již částečně zakódovány některé informace o zpracování obrazové informace. První vrstva hledá vzory postupně v jednotlivých částech obrazu, následující konvoluční vrstvy tyto vzory skládají ve větší rysy. Během tréninku neuronové sítě se učí hodnoty konvoluční masky. (**Obrázek 4**) (**Obrázek 5**)



Obrázek 4: Příklad reprezentace vzorů detekovaných jednotlivými konvolučními vrstvami. Převzato z (Dwiyantoro, 2018)



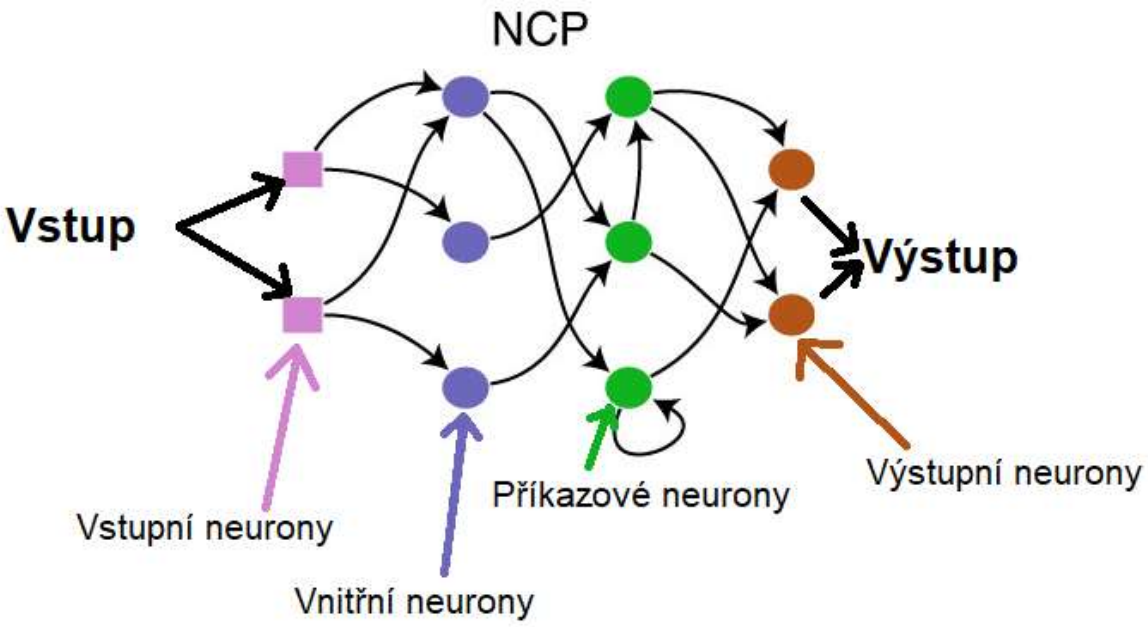
Obrázek 5: Konvoluce obrazových dat v jedné části vstupu. Konvoluční maska je naučená v průběhu trénování neuronové sítě. Převzato z (grt, 2006)

2.4.3 Rekurentní vrstva NCP

NCP, v překladu „politika neurální kontroly“, je svým typem vrstva rekurentní. Její stavba vychází z propojení v neuronové síti háďátka obecného. (White, Southgate, Thomson, & Brenner, 1986) Skládá se ze čtyř hlavních sad neuronů, a to neuronů vstupních, jejichž funkce je předávat vstup do dalších vrstev sítě, dále se skládá ze sady neuronů vnitřních zpracovávajících informace přijaté z neuronů vstupních a předávajících je do neuronů příkazových. Příkazové neurony hrají v síti největší roli, zaprvé jsou propojeny mezi sebou, zadruhé aktivují výstupní neurony, jež jsou posledním typem neuronů ve vrstvě. (Lechner, a další, 2020) (**Obrázek 6**)

Nastavitelnými hyperparametry u vrstvy NCP jsou:

- Počet vnitřních neuronů
- Počet příkazových neuronů
- Počet výstupních neuronů
- Počet výstupních propojení z každého vstupního neuronu
- Počet výstupních propojení z každého vnitřního neuronu
- Počet vstupních propojení každého výstupního neuronu
- Počet vracejících se propojení ve vrstvě příkazových neuronů



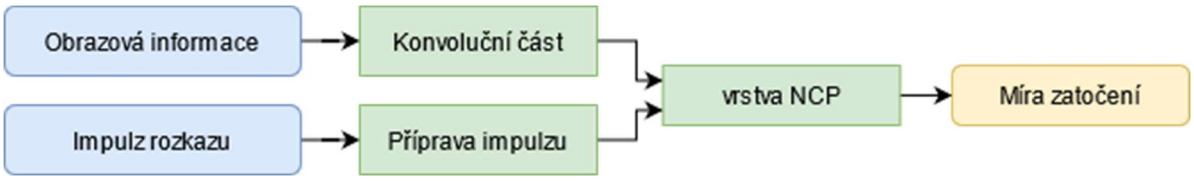
Obrázek 6: Vnitřní struktura vrstvy rekurentní vrstvy NCP. Převzato z (Lechner, a další, 2020) upraveno

3 NÁVRH EXPERIMENTU

Pro zjištění možnosti kontroly neuronové sítě pomocí rekurentní vrstvy NCP je v této práci navrženo množství experimentů. Zároveň je díky jejich množství možno zjistit, které parametry fungují pro tento úkol nejlépe. Vzhledem k počtu plánovaných experimentů je neuronová síť záměrně navržena jednoduše, abychom tak byli schopni v omezeném časovém horizontu všechny experimenty uskutečnit.

3.1 Struktura neuronové sítě

Neuronová síť, kterou jsem pro tento experimentu navrhl, se skládá ze tří hlavních částí. Tyto části dávají dohromady neuronovu síť jako celek. (**Obrázek 7**)



Obrázek 7: High level pohled na strukturu neuronové sítě použité v experimentu.

První část sítě je část konvoluční, sloužící ke předzpracování obrazového vstupu. Do této části bude vstupem obraz z kamery umístěné na voze v rozlišení 160x84px[4] a výstupem jeho redukovaná reprezentace. Tato reprezentace je následně použita jako vstup pro rekurentní NCP vrstvu. Jako konvoluční návrh jsem použil upravenou verzi modelu již využívaného k tomuto účelu. Konvoluční část zůstane v průběhu experimentu neměnná, měnit se bude pouze velikost poslední zcela propojené vrstvy. (Bojarski, a další, 2016) (**Obrázek 8**)

Druhá část se věnuje vstupu rozkazu do sítě. Na jejím vstupu budou příkazy „přejeď do levého pruhu“ a „přejeď do pravého pruhu“. Během experimentu bude měněna i tato část. Budou testovány tři možné návrhy, a to: jeden vstup v rozmezí -1 až 1 (-1 = doleva, 0 = zůstaň v pruhu, 1 = doprava), dva vstupy v rozmezí 0 až 1 (jeden pro každý příkaz) nebo využití plně propojené vrstvy o různých velikostech.

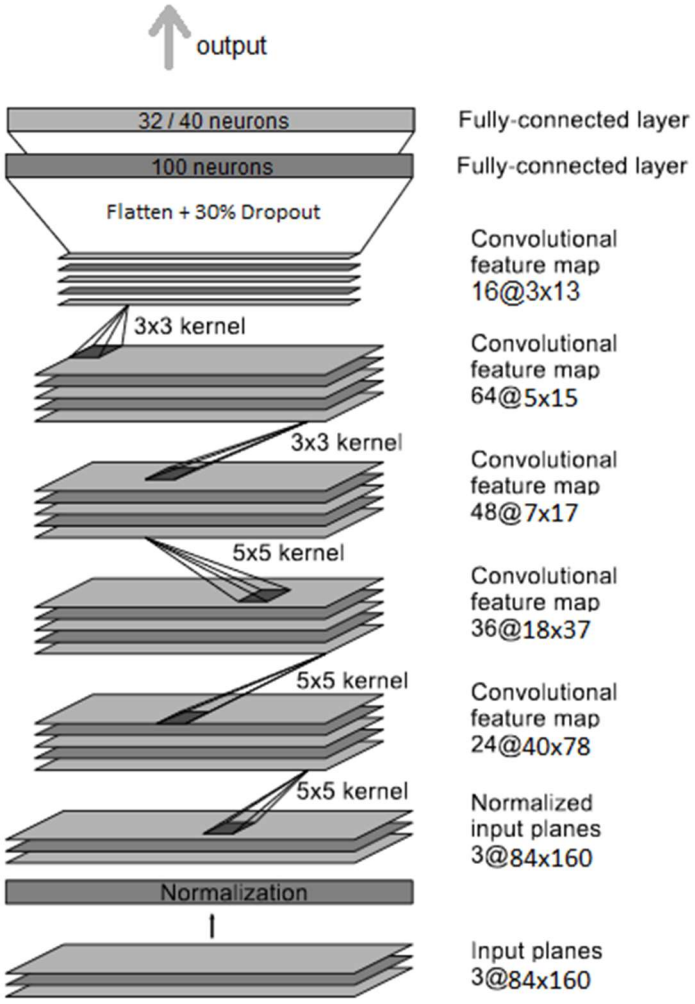
V poslední, třetí části, se výstup z konvoluční a rozkazové části spojují a jsou vstupem pro rekurentní NCP vrstvu. U této vrstvy je možno nastavit: počet vnitřních neuronů, počet ovládacích neuronů, počet výstupních neuronů, počet výstupních propojení vstupních neuronů, počet výstupních propojení vnitřního neuronu, počet rekurentních propojení a počet vstupních propojení do výstupního neuronu. (**Obrázek 6**)

Jako aktivační funkci pro jednotlivé vrstvy jsem zvolil Exponential Linear Unit (ELU). (Clavert, Unterthiner, & Hochreiter, 2016) Důvodem mé volby byl fakt, že tato funkce, oproti

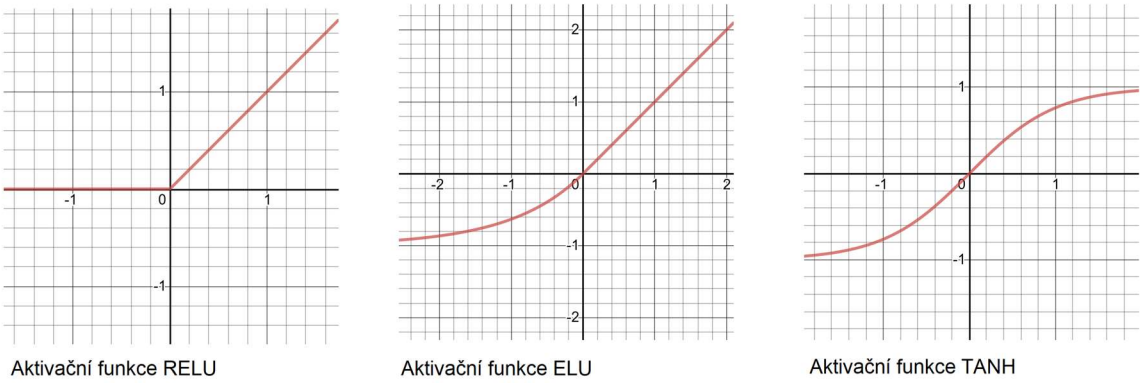
[4] Rozlišení 160x84px bylo zvoleno po manuálním prozkoumání tréninkových dat jako minimální, kde je možno rozpoznat jízdní pruhy.

například často používané RELU aktivaci, urychluje proces trénování a poskytuje lepší celkové výsledky trénované sítě. Dalším důvodem byl také fakt, že v případě hlubších sítí dokáže díky možnosti záporného výstupu efektivně předcházet problému mizejícího gradientu (vanishing gradient problem), díky čemuž bude experiment více vypovídající pro budoucí využití. Na výstupu by měla být hodnota zatočení v rozmezí -1 až 1. Proto jsem zvolil v poslední vrstvě sítě aktivační funkci Hyperbolický tangens (TANH). (**Obrázek 9**)

Z pozorování trénovacích dat vyplynulo, že změna pruhu netrvá nikdy déle než 3 sekundy, a tak bude při trénování postačovat využití 3sekundových částí.



Obrázek 8: Architektura obrazové části sítě. Převzato z (Bojarski, a další, 2016) upraveno.



Obrázek 9: Grafy funkce RELU: $y = \{x \leq 0: 0, x > 0: x\}$; funkce ELU: $y = \{x \leq 0: \exp(x), x > 0: x\}$; a funkce TANH: $y = \tanh(x)$; (vlastní tvorba)

3.2 Dílčí části experimentu

Vzhledem k ještě neprobádanému využití vrstvy NCP je vhodné zkusit více různých architektur a různé parametry a z nich vybrat tu nejlépe fungující kombinaci. Z různých možných obměn jsem vyzkoušel 16 kombinací, přičemž jsem se zaměřil především na vliv vnitřní konfigurace NCP vrstvy a vliv typu vstupu příkazu.

Různé obměny, které jsem ve své práci vyzkoušel, jsou:

- Změna vnitřních parametrů vrstvy NCP
  - Počet vnitřních neuronů 18 a 24
  - Počet vnitřních propojení vnitřních neuronů 5 a 9
  - Počet rekurentních propojení příkazových neuronů 3, 5 a 9
- Různé velikosti vstupu z konvoluční části, a to 32 a 40.
- Tři možné typy vstupu rozkazu
  - Vstup rozkazu jedním neuronem, kde hodnota
    - -1 = změna pruhu doleva
    - 0 = jízda v pruhu
    - 1 = změna pruhu doprava
  - Vstup rozkazu dvěma neurony, pro změnu pruhu doleva, respektive doprava
  - Použít na vstupu rozkazu plně propojenou vrstvu pro zvětšení velikosti vstupu do části NCP, a to o velikosti 12

4 PROVEDENÍ EXPERIMENTU

4.1 Použité nástroje

Experimenty byly implementovány v jazyce Python. Byly použity tyto nástroje:

- Keras – implementace vrstev a algoritmů učení pro neuronové sítě
- Keras-ncp – knihovna obsahující implementaci studované vrstvy NCP
- Google colab – projekt umožňující bezplatný běh experimentů na GPU v cloudu
- Matplotlib, Pydot, Seaborn – knihovny pro vizualizaci experimentu
- H5py – knihovna pro práci se soubory hdf5 pro efektivní ukládání trénovacích dat
- OpenCV – knihovna pro práci s obrázky

4.2 Sběr tréninkových dat

4.2.1 Existující dataset vs. Tvorba vlastního datasetu

Než jsem mohl začít s prací na neuronové síti, musel jsem nejdříve sehnat data, na kterých bych síť učil. Naskytly se dvě možnosti. První z nich bylo použití již existující datasetu. Z prostředí samořídících aut je jich velké množství. V našem případě jsem hledal takový, který má pohled z přední kamery, a informaci o míře zatočení. Takové podmínky splňovalo hned několik z nich, jmenovitě: nuScenes[5], comma2k19[6], Udacity dataset[7] či Waymo Open Dataset[8]. Jejich nespornou výhodou je, že není potřeba tvořit dataset vlastní, což je často časově velice náročné. Po prohlédnutí zmíněných datasetů jsem však usoudil, že by jejich využití přineslo přílišnou složitost, která by mohla negativně zkreslit výsledky experimentu, a neumožnila by mi vyzkoušet dostatečný počet navržených obměn. Pro experiment jsem tedy rozhodl vytvořit vlastní dataset jízdou v simulátoru. Vzhledem k relativně malému množství potřebných dat byl tento přístup vhodnější.

4.2.2 Volba simulátoru pro autonomní vozidla

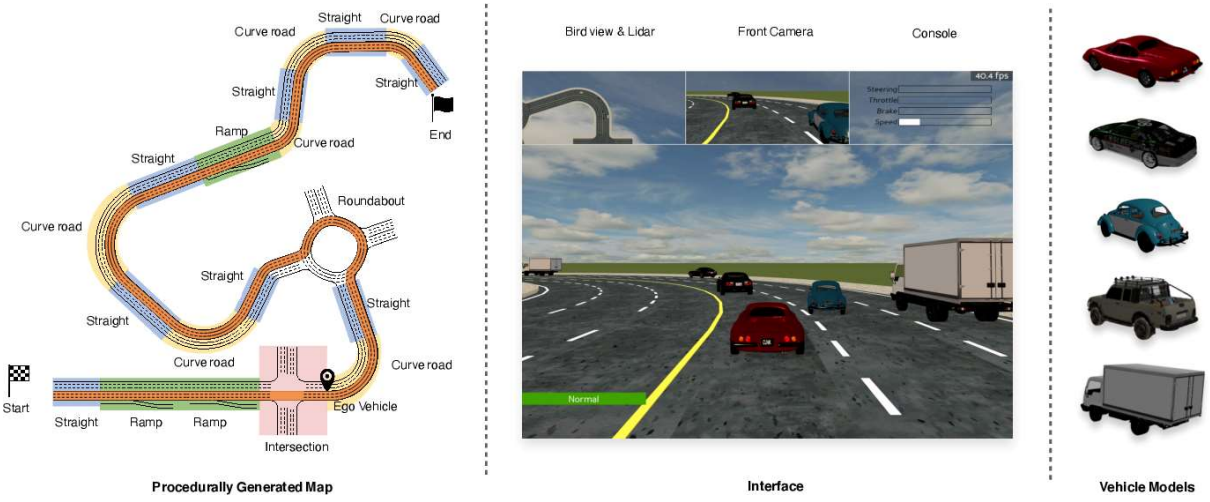
Pro zjištění funkčnosti v reálném prostředí, a v našem případě také pro sběr dat, byl použit simulátor. I zde bylo několik možností na výběr: CARLA[9] (Dosovitskiy, Ros, Codevilla, Lopez, & Koltun, 2017), LGSVL[10] (Rong, a další, 2020) či PGDrive[11] (Li, a další, 2020). Volba padla na simulátor PGDrive. CARLA a LGSVL umožňují fotorealistické prostředí a velké množství senzorů, avšak jsou velice náročné na výkon hardwaru. PGDrive má nároky na hardware minimální. Nepodporuje fotorealistické vykreslování, avšak to není pro experiment zásadní.

[5] https://www.nuscenes.org/nuscenes
[6] https://github.com/commaai/comma2k19
[7] https://github.com/udacity/self-driving-car/tree/master/datasets
[8] https://waymo.com/open
[9] https://github.com/carla-simulator/carla
[10] https://www.lgsvlsimulator.com
[11] https://github.com/decisionforce/pgdrive

Hlavním důvodem pro volbu PGDrive byla však jeho schopnost procedurálního generování prostředí, to znamená, že při každé jízdě je prostředí odlišné. (**Obrázek 10**)

4.2.3 Úpravy simulátoru PGDrive

Pro experiment bylo zapotřebí některých funkcí, které PGDrive k datu provedení experimentu nepodporuje. Prvními z nich byla změna výstupního rozlišení RGB kamery, používané jako vstup pro neurnovou síť, a změna maximální rychlosti vozidla. Druhou změnou bylo přidání vlastního typu zahnutí vozovky a změny pravděpodobností výskytu jednotlivých generovaných částí, které nebylo možno měnit přímo pomocí konfigurace prostředí. (**Obrázek 11**) Třetí provedenou změnou bylo umožnění přístupu k vnitřnímu stavu simulátoru, aby bylo možno efektivněji sbírat trénovací data a automaticky vyhodnocovat natrénované neuronové sítě a přiřadit jim číselné skóre úspěšnosti. Upravená verze PGDrive je dostupná z mého githubu.[12]

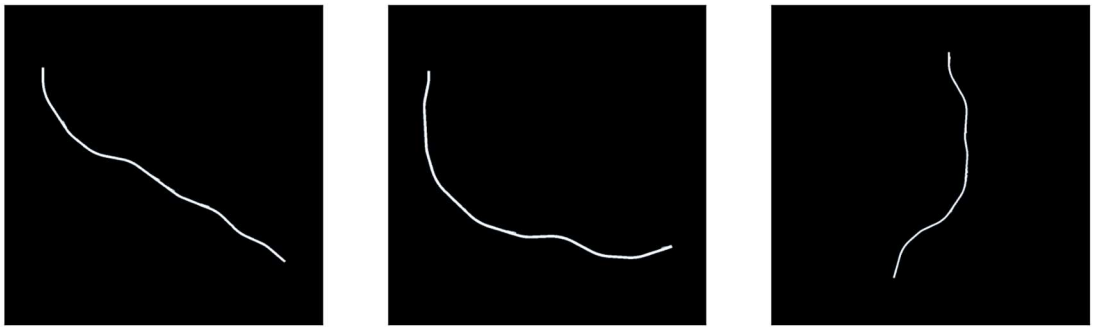


Obrázek 10: Ukázka simulátoru PGDrive a jeho procedurální generace. Převzato z (Li, a další, 2020)

4.2.4 Automatizace sběru dat

Manuální sběr dat by byl zdlouhavý proces. Díky informacím z vnitřního stavu simulátoru jsem proto vytvořil systém samostatné jízdy ve zvoleném pruhu. Pro lepší napodobení reálných dat jsem přidal do systému jízdy prvek náhody. Systém jsem nejdříve otestoval pro správnou funkčnost a vyladil ho tak, aby co nejvíce odpovídal reálné jízdě. Systém jsem následně spustil a po několika hodinách byl vytvořený dataset pro trénink sítě.

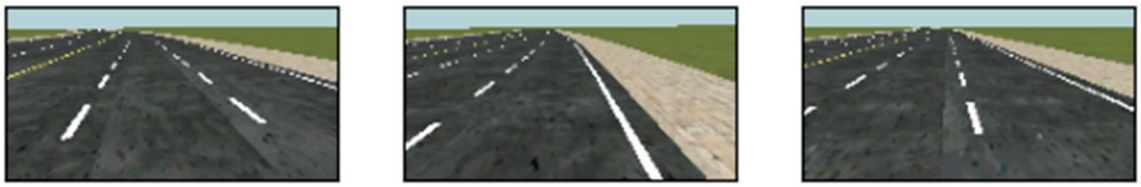
[12] https://github.com/gamecraftCZ/pgdrive



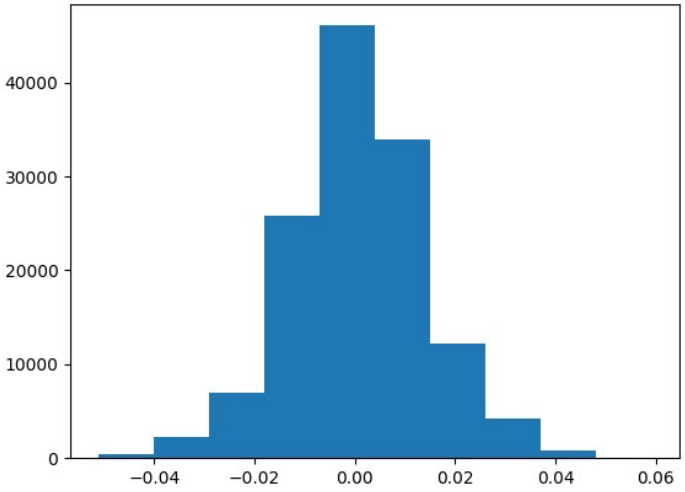
Obrázek 11: Ukázka některých procedurálně vygenerovaných silnic v simulátoru PGDrive. (vlastní tvorba)

4.3 Vhled do tréninkových dat

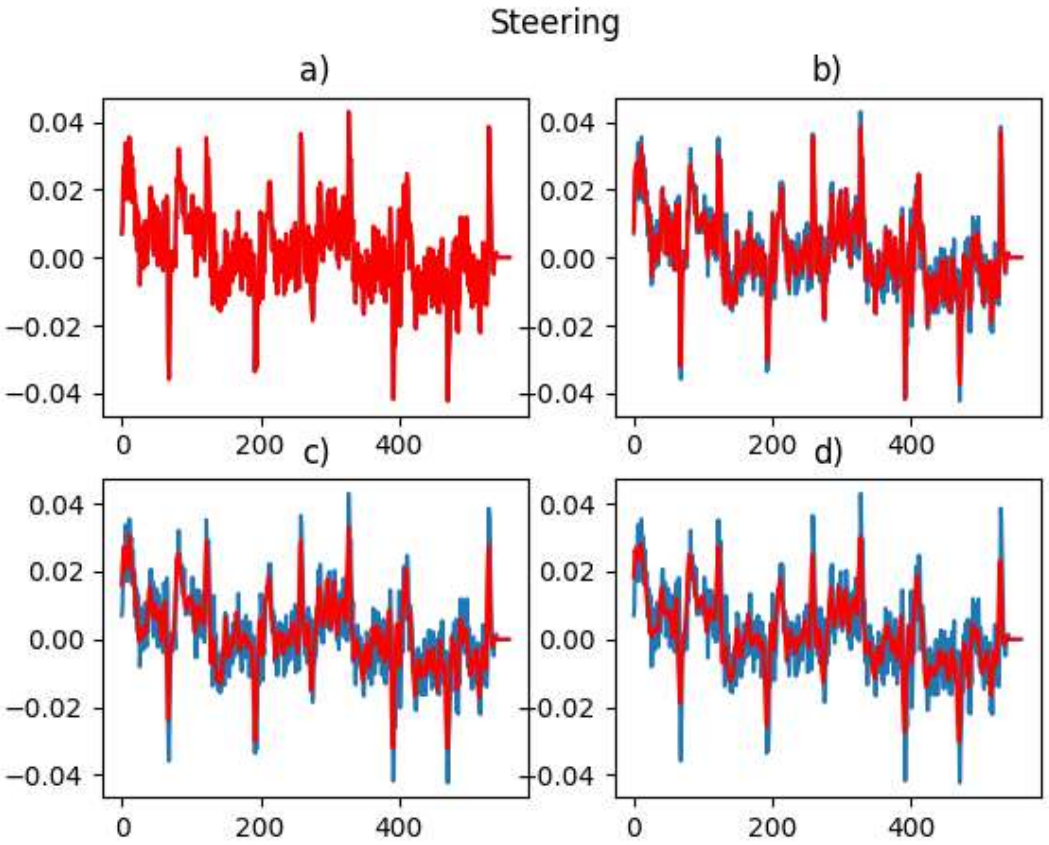
Před trénováním neuronové sítě je vhodné prozkoumat obsah trénovacích dat a přizpůsobit je pro trénování. Nejdříve jsem vizualizoval obrazový vstup. (**Obrázek 12**) Tento vstup nebylo potřeba upravovat, jelikož byl již připraven z fáze sběru dat. Další důležitou částí dat je míra zatočení. Na histogramu si můžeme všimnout rozsahu v pouze v rozmezí převážně -0.05 až 0.05. (**Obrázek 13**) Takto malé rozmezí by mohlo zpomalit a negativně ovlivnit trénování neuronové sítě, z tohoto důvodu byl vstup před trénováním zvětšen 20x a limitován do rozmezí -1 až 1. V grafovém zobrazení je vidět fakt, že zatáčení je roztěkané. Před vstupem jsem tuto hodnotu částečně vyhladil, a to pomocí jednoduchého klouzavého průměru, který je pro toto využití zcela dostačující. (**Obrázek 14**) Třetím vstupem je příkaz pro síť. Zde nebylo co vizualizovat, jelikož víme vše, co potřebujeme, z kroku sběru dat.



Obrázek 12: Náhled na obrazový vstup v různých časech jízdy. (vlastní tvorba)



Obrázek 13: Distribuce hodnot míry zatočení. (vlastní tvorba)



Obrázek 14: Graf míry zatočení pro jednu jízdu. Modrá čára značí původní míru zatočení. Červená čára značí míru zatočení vyhlazenou klouzavým průměrem o počtu započtených období **a)** 1, **b)** 2, **c)** 4, **d)** 5. (vlastní tvorba)

4.4 Realizace neuronové sítě

Realizace neuronové sítě proběhla v jazyce python. Pomocí knihovny Keras jsem převedl návrh neuronové sítě do funkčního řešení. (**Příloha 1**) Pro zpracování každého bodu v čase zvlášť jsem využil wrapper TimeDistributed, který časový úsek pro obalenou vrstvu „rozseká“ na jednotlivé časové body. Do rekurentní NCP vrstvy poté přijde časový úsek opět „složený“ jako celek. K rozdělení datasetu na 3sekundové části jsem vytvořil generátorovou funkci. Tato funkce vrací trénovacímu procesu data na vyžádání. Během trénování vytvořím trénovací úsek, až když je potřeba, a nemusím tak mít v paměti stovky GB předpřipravených úseků. ()

Pro optimalizaci vah sítě jsem využil algoritmus Adam, a to pro jeho jednoduchost a rychlost. (Kingma & Ba, 2014) Po více vyzkoušených rychlostech učení jsem zvolil jako základní rychlost 5×10^{-5} pro její přijatelnou časovou náročnost a schopnost dobře konvergovat. Jako funkci pro výpočet hodnoty loss, kterou se snažíme minimalizovat, jsem využil nejčastěji užívanou funkci pro tento typ problémů, a to funkci střední kvadratické chyby (mean squared error). Počet várek (batches) jsem zvolil 8, a to prvně podle dostupné paměti GPU a následně testováním drobných obměn. Počet kroků v jedné epoše a počet epoch jsem zvolil podle místa, kde se již výsledek z trénování nezlepšoval. Zvolil jsem tak 400 kroků za epochu a 30 epoch. Každé takovéto trénování zabralo na službě Google Colab v rozmezí 3–5 hodin podle přiřazeného stroje. Po dokončené epoše byl ukládán trénovaný model ve formátu h5 a hodnota loss v tomto bodě ve formátu tensorboard pro budoucí vizualizaci.

5 VÝSLEDKY

5.1 Způsob měření úspěšnosti

Pro porovnání jednotlivých konfigurací byly v práci zvoleny dvě metriky. První metrikou je hodnota loss validační části datasetu. Čím menší, tím lepší. Druhou metrikou je test v simulátoru. Zde byl měřen zaprvé počet počet přejetí pruhu bez příkazu a počet neuposlechnutí příkazu. Obě metriky jsou měřeny na modelech s nejlepší hodnotnou loss do 30. epochy.

Jako základní hodnota loss pro porovnávání byla zvolena průměrná hodnota loss v případě, že by neuronová síť vracela pouze průměrnou hodnotu míry zatočení. V případě, že testovaná síť má hodnotu loss menší nežli tato základní hodnota, znamená to, že se síť naučila alespoň částečně zatáčet.

5.2 Výsledky v číslech

Tabulka 1: Konfigurace neuronové sítě a jejich výkon; Zvýrazněné hodnoty představují nejlepší výsledky podle hodnoty loss za každý způsob vstupu rozkazu do sítě.

Konfigurace NCP[13]	**Počet parametrů NCP vrstvy**	**Hodnota loss**[14]	**Test v simulátoru**[15]
Baseline[16]	-	0.0387	-
32 / 18 / 5 / 3 / 2 / -	8 223	0.0066	6 / 35 / 27 / 22
40 / 24 / 9 / 9 / 2 / -	11 889	0.0046	0 / 33 / 7 / 2
32 / 18 / 5 / 9 / 2 / -	8 223	0.0078	10 / 32 / 34 / 27
32 / 24 / 9 / 9 / 2 / -	**10 689**	**0.0036**	**0 / 20 / 4 / 7**
40 / 18 / 9 / 9 / 2 / -	9 231	0.0051	13 / 29 / 18 / 12
40 / 24 / 5 / 9 / 2 / -	11 889	0.0041	1 / 23 / 6 / 3
40 / 24 / 9 / 7 / 2 / -	11 889	0.0040	2 / 17 / 3 / 5
40 / 24 / 9 / 5 / 2 / -	11 889	0.0038	9 / 20 / 13 / 8
40 / 24 / 9 / 3 / 2 / -	11 889	0.0042	0 / 30 / 4 / 4
32 / 18 / 5 / 9 / 2 / 12	12 189	0.0038	1 / 26 / 5 / 1
32 / 24 / 5 / 9 / 2 / 12	**9 483**	**0.0037**	**5 / 25 / 7 / 5**
32 / 24 / 5 / 3 / 2 / 12	12 189	0.0042	5 / 19 / 5 / 8
32 / 18 / 5 / 3 / 2 / 12	9 483	0.0054	9 / 34 / 18 / 17
40 / 24 / 9 / 7 / 1 / -	**11 739**	**0.0065**	**5 / 21 / 15 / 18**
32 / 18 / 9 / 7 / 1 / -	8 097	0.0096	16 / 24 / 29 / 27
32 / 18 / 5 / 3 / 1 / -	8 097	0.0091	9 / 32 / 35 / 30

[13] Počet neuronů na vstupu / Vnitřních neuronů / Počet výstupů každého vnitřního neuronu / Výstupů každého vstupního neuronu / Počet rekurentních propojení / Počet vstupů do výstupního neuronu sítě
[14] Nejlepší hodnota do 30. epochy menší = lepší
[15] Počet vyjetí ze silnice za 150 jízd / Změny pruhu bez příkazu za 30 jízd / Neuposlechnutí příkazu dlouhém jeden časový krok za 40 jízd / Neuposlechnutí příkazu dlouhém tři časové kroky za 40 jízd / Neuposlechnutí příkazu dlouhého pět časových kroků za 40 jízd – menší = lepší
[16] Model, kde je výstupem průměr hodnot zatočení

Z výsledků je vidět, že:

- Síť NCP funguje dobře pro problém autonomního řízení.
- Síť je možno bez kontrolovat i malým podílem vstupů rozkazu.
- Mezi výkonem sítě bez a se zcela propojenou vrstvou na vstupu není znatelný rozdíl. Zcela propojená vrstva je tedy v sítí přebytečná.
- Síť s jedním vstupem rozkazu vykazuje zhoršené výsledky.
- Velikost sítě neměla v tomto případě přílišný dopad na její výsledné schopnosti.

5.3 Zaznamenané problémy

Během experimentu se vyskytlo několik možných problémů, kterým by bylo dobré při praktickém nasazení předejít. Prvním problémem byl overfitting. Vzhledem k malé rozmanitosti prostředí (stále stejné počasí, žádné okolí, stejná textura silnice atd.) byly trénovací i testovací části datasetu velmi podobné. V případě tohoto experimentu to nebyl problém, jelikož dataset byl vypovídající o našem prostředí. V reálné aplikaci by však měl být dataset více rozmanitý a rozhodně by na něj měly být aplikovány metody pro rozšíření trénovacích dat.

Druhým možným problémem bylo provádění rozkazu v nenaučených prostředích. V případě prováděného experimentu se jednalo o případ, kdy jsme předali příkaz ke změně pruhu ve směru, kde se již žádný další pruh nevyskytoval. V tomto případě vyjelo auto mimo vozovku. Řešením tohoto problému je zahrnutí takovýchto případů do trénovacích dat.

6 ZÁVĚR

Cílem práce bylo navrhnout a zrealizovat neuronovou síť využívající rekurentní vrstvu NCP a ověřit hypotézu, zda lze vstupem rozkazu do sítě efektivně ovládat její chování. Experiment jsem provedl v prostředí samořídících aut, kde jsem testoval schopnost změny pruhu v odezvě na rozkaz. Svého cíle jsem dosáhl; zjistil jsem, že síť ovládat lze. Jako nejefektivnější architektura sítě se ukázala architektura s ovládáním jedním vstupem pro každý rozkaz (zatoč doleva, zatoč doprava). Vstup přes zcela propojenou vrstvu neměl žádný vliv na výkon sítě.

Tato práce sloužila především k ověření možností nové rekurentní vrstvy NCP pro kontrolované neuronové sítě. Díky potvrzení hypotézy o možnosti kontroly tohoto typu sítě se otevírají nové možnosti pro navazující výzkum, ve kterém bych chtěl po dokončení své práce SOČ pracovat. Jako nejvýznamnější kandidáti pro navazující výzkum se jeví:

- Vliv sériového zapojení vrstev NCP na rychlost trénování a schopnosti sítě.
- Využití v oblasti reinforced learningu a následné porovnání s již existujícími algoritmy jako jsou PPO[17], A3C[18], SAC[19] a další.
- Porovnání efektivity využití trénovacích dat u sítě NCP s dalšími řešeními pro rekurentní neuronové sítě jako jsou LSTM, GRU a další.
- Chování NCP v prostředích, kde je množství příkazů velmi velké (100+).
- Chování NCP v prostředích, kde se požadovaná akce vyžádaná jedním vstupem může lišit podle momentálního prostředí.
- Implementace do prostředí kvantového počítače

[17] Proximal Policy Optimization (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017)
[18] Asynchronous Actor Critic (Mnih, a další, 2016)
[19] Soft Actor Critic (Haarnoja, Zhou, Abbeel, & Levine, 2018)

7 POUŽITÁ LITERATURA

Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., . . . Zieba, K. (2016). *End to End Learning for Self-Driving Cars.*

Caesar, H., Bankiti, V. K., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., . . . Beijbom, O. (2020. Červen 2020). nuScenes: A Multimodal Dataset for Autonomous Driving., (stránky 11618-11628). doi:10.1109/CVPR42600.2020.01164

Clavert, D.-A., Unterthiner, T., & Hochreiter, S. (2016). Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv: Learning*.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (11 2017). CARLA: An Open Urban Driving Simulator. *ArXiv, abs/1711.03938*. Načteno z https://arxiv.org/abs/1711.03938

Dwiyantoro, A. P. (2018). *The Evolution of Computer Vision Techniques on Face Detection, Part 2*. Načteno z Medium: https://medium.com/nodeflux/the-evolution-of-computer-vision-techniques-on-face-detection-part-2-4af3b22df7c2

Glasmachers, T. (2017). Limits of End-to-End Learning. *ACML*.

grt. (2006). *Konvoluce 2rozm diskretni*. Načteno z Wikimedia commons: https://commons.wikimedia.org/wiki/File:Konvoluce_2rozm_diskretni.jpg

Gu, Z., Li, Z., Di, X., & Shi, R. (2020). An LSTM-Based Autonomous Driving Model Using Waymo Open Dataset. *Applied Sciences, 10*(6), 2046. doi:10.3390/app10062046

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. V J. D. Krause (Editor), *Proceedings of the 35th International Conference on Machine Learning. 80*, stránky 1861-1870. Stockholm Sweden: PMLR. Načteno z http://proceedings.mlr.press/v80/haarnoja18b.html

Hu, J., Hanlin, N., Carrasco, J., Lennox, B., & Arvin, F. (2020). Voronoi-Based Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning. *IEEE Transactions on Vehicular Technology, 69*(12), 14413-14423. doi:10.1109/TVT.2020.3034800

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology, 160*(1), 106-154. doi:10.1113/jphysiol.1962.sp006837

IBM. (2020). *Recurrent Neural Networks*. Načteno z IBM: https://www.ibm.com/cloud/learn/recurrent-neural-networks

IBM. (2020). *What is supervised learning?* Načteno z IBM: https://www.ibm.com/cloud/learn/supervised-learning

IBM. (2020). *What is unsupervised learning?* Načteno z IBM: https://www.ibm.com/cloud/learn/unsupervised-learning

Kingma, D., & Ba, J. (12 2014). Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*. Načteno z https://arxiv.org/abs/1412.6980

Lechner, M., Hasani, R., Amini, A., Henzinger, T. A., Rus, D., & Grosu, R. (2020). Neural circuit policies enabling auditable autonomy. *Nat Mach Intell*, 642-652. doi:10.1038/s42256-020-00237-3

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, E. R., Hubbard, W., & Jackel L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation, 1*(4), 541-551. doi:10.1162/neco.1989.1.4.541

Lecun, Y., Cosatto, E., Ben, J., Muller, U., & Flepp, B. (2004). *DAVE: Autonomous Off-Road Vehicle Control Using End-to-End Learning.* DARPA-IPTO Final Report. Načteno z http://net-scale.com/doc/net-scale-dave-report.pdf

Li, Q.-y., Peng, Z., Zhang, Q., Qiu, C., Liu, C., & Zhou, B. (2020). Improving the Generalization of End-to-End Driving through Procedural Generation. *ArXiv, abs/2012.13681*. Načteno z https://arxiv.org/abs/2012.13681

Mnih, V., Badia, A., Mirza, M., Graves, A., Lillicrap, T., Harley, T., . . . Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. *Proceedings of the 33rd International Conference on International Conference on Machine Learning. 48*, stránky 1928-1937. New York, NY, USA: JMLR.org. Načteno z https://arxiv.org/abs/1602.01783

Ognjanovski, G. (19. Únor 2020). *Everything you need to know about Neural Networks and Backpropagation*. Načteno z Towards Data Science: https://towardsdatascience.com/everything-you-need-to-know-about-neural-networks-and-backpropagation-machine-learning-made-easy-e5285bc2be3a

Pomerleau, D. A. (1989). *ALVINN: an autonomous land vehicle in a neural network.* Advances in neural information processing systems.

Rong, G., Shin, B., Tabatabaee, H., Qiang, L., Steve, L., Zelenkovsky, D., & Seonman, K. (05 2020). LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving. *ArXiv*. Načteno z https://arxiv.org/abs/2005.03778

Rosenblatt, F. (1957). *The Perceptron - a percieving and recognizing automaton.* Cornell Aeronautical Laboratory.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *ArXiv*. Načteno z https://arxiv.org/abs/1707.06347

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., . . . Polosukhin, I. (2017). Attention Is All You Need. Načteno z https://arxiv.org/abs/1706.03762

White, J. G., Southgate, E., Thomson, J. N., & Brenner, S. (1986). The structure of the nervous system of the nematode Caenorhabditis elegans. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences, 314 1165*, 1-340. doi:10.1098/rstb.1986.0056

8 SEZNAM OBRÁZKŮ A TABULEK

9 PŘÍLOHA 1: VIZUALIZACE JEDNÉ Z KONFIGURACÍ NAVRŽENÉ SÍTĚ