Integrácia moderných prostriedkov umelej inteligencie do mobilného robota

Andrej Lúčny

Katedra aplikovanej informatiky, Fakulta matematiky, fyziky a informatiky, Univerzita Komenského
KAI FMFI UK, Mlynská dolina, 842 48 Bratislava
lucny@fmph.uniba.sk

Abstrakt

Úspechy umelej inteligencie spočívajúce v rozvoji výpočtových prostriedkov, hlbokého učenia a cloudových služieb zásadne zmenili nároky na výkon, ktorý očakávame od mobilných robotov. Na príklade lacného robota s kamerou, od ktorého by sme v minulosti čakali vyhýbanie sa prekážkam, ukazujeme ako je možné zapojiť spomínané nové technológie a zlepšiť inteligenciu robota. Vybavíme ho hlasovým ovládaním (pomocou cloudovej služby) a dáme mu možnosť rozprávať (rečový syntetizátor). Vybavíme ho schopnosťou porozumieť jazyku využitím technológie tvorby chatbot-ov. Vložíme do neho schopnosť rozpoznať ľudí vo svojom okolí na základe detekcie ich tvárí.

Zabudovanie týchto modulov prináša taktiež zvýšené nároky na riadiaci systém robota. Tento musí zvládnuť kombináciu viacerých procesov, pričom niektoré z nich sú pomalé, iné zase musia byť vykonané rýchlo. To je oblasť, v ktorej môžeme zúžitkovať vlastné architektonické riešenia vytvorené v minulosti.

Záverom diskutujeme, čo by bolo potrebné robotovi ešte pridať, ale nie je to zatiaľ jednoduché. Napríklad schopnosť lokalizácie hovoriaceho človeka.

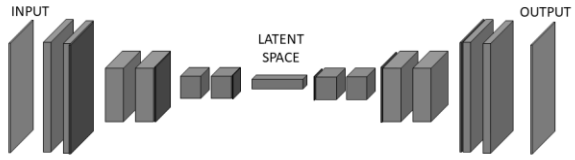
1 Vývoj umelej inteligencie v poslednom období

Od roku 2012, keď AlexNet porazila v súťaži ImageNet ostatné riešenia rozdielom triedy sme svedkami veľkého boomu konekcionistického prístupu v umelej inteligencii v podobe tzv. hlbokých neurónových sietí či hlbokého učenia. Vskutku táto technológia dosiahla stav, kedy kvalitou prevýšila všetko, čo bolo doteraz k dispozícii.

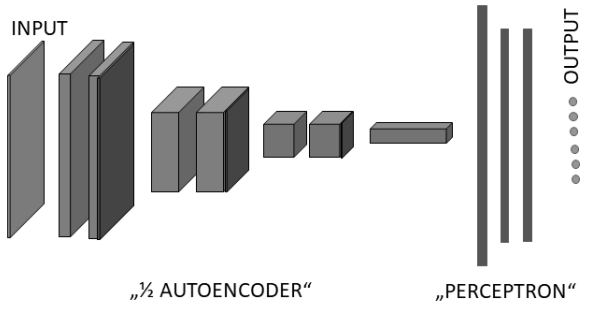
Ako uvádza monografia (Goodfellow, Bengio, Courville 2016), za týmto úspechom stojí viacero vynálezov známych už mnoho rokov i úplne nových objavov. Neurónové siete sú predmetom výskumu už dlhú dobu a za ten čas bolo vymyslených mnoho ich typov. Od bežných algoritmov sa líšia dvomi vlastnosťami. Za prvé majú oveľa viac parametrov. Aj bežný algoritmus môže mať nejaké konštanty, ktorými je možné ladiť jeho činnosť, ale je ich maximálne niekoľko desiatok. Hlboká neurónová sieť ich má spravidla desiatky miliónov. Stanovíme jej nejakú architektúru a predpokladáme, že pri správnom nastavení týchto parametrov robí to, čo potrebujeme. Pre veľký počet parametrov je však nemysliteľné, aby sme toto nastavenie hľadali ručne. Tým sa dostávame k druhému rozdielu. Keď spustíme bežný algoritmus a stane sa, že sa jeho výstup odlišuje od výstupu želaného, môžeme sa pokúsiť nejako modifikovať jeho parametre, aby sa to zlepšilo. Avšak netušíme ako ich máme modifikovať. Základnou vlastnosťou neurónovej siete je, že dokáže v takejto situácii povedať, ktorým smerom treba zmeniť každý z tých desať miliónov parametrov, aby sa rozdiel výstupu a želaného výstupu (tzv. hodnota chybovej funkcie) zlepšil. Toto zlepšenie možno potom cyklicky opakovať, až po kým nie je výstup siete uspokojivý, alebo už k ďalšiemu zlepšeniu nedochádza. To nazývame procesom trénovania siete.

Hoci sa k hlbokému učeniu nedospelo priamočiarou cestou, základnú myšlienku relevantnú pre väčšinu hlbokých neurónových sietí možno vysvetliť ako snahu o kombináciu dvoch neurónových sietí, ktorých činnosti ľahko porozumieme: (hlbokého) autoencoder-u a perceptron-u. Perceptron (Rosenblatt 1958) je neurónová sieť, v ktorej sú neuróny v aspoň dvoch vrstvách prepojené každý s každým a každé toto prepojenie má svoju špecifickú váhu. Z teoretického hľadiska je (pri vhodnom type aktivačnej funkcie neurónov) univerzálnym aproximátorom (Cybenko 1989), t.j. naučí sa s určitou presnosťou akúkoľvek rovnomerne spojitú funkciu. V princípe by táto mašinka mala sama o sebe stačiť na ľubovoľnú úlohu strojového učenia, prakticky ju však možno použiť na máločo. Relatívne dobré výsledky dáva na dátach, kde sú vstupom vektory reálnych čísel s nie príliš vysokou dimenziou. Naopak, zlé výsledky dáva na dátach ako je obraz z kamery. Je preto logické sa snažiť premeniť jedny dáta na druhé, napríklad obraz 640x480 pixelov na vektor 1024 reálnych čísel. A práve takúto premenu vie urobiť (hlboký) autoencoder (Rumelhart, Hinton, Williams 1986). Tento pozostáva z tzv. konvolučných vrstiev, v ktorých je každý neurón prepojený na málo susedných neurónov v predchádzajúcej vrstve, pričom váhy týchto spojení sú zdieľané so všetkými ostatnými neurónmi v danej vrstve (obr. 1). (Tým je zabezpečené, že sieť reaguje rovnako na určitý podnet nezávisle od jeho posunutia v rámci vstupných dát.) Konvolučné

vrstvy, ktoré dáta transformujú, sú v autoencoderi prekladané vrstvami, ktoré menia ich dimenziu: v prvej polovici autoencoderu ju redukujú, v druhej expandujú. Takže kým na vstupe i výstupe máme napríklad obraz 640x480, v strede autoencodera majú dáta podobu tých želaných 1024 reálnych čísel. Tento „stred“ autoencoderu nazývame latentným priestorom. Na čo je to dobré? Autoencoder môžeme trénovať, aby na výstupe dával presne to, čo mu dáme na vstup. Pokiaľ sa mu to na určitej sade dát podarí naučiť, môžeme jeho druhú polovicu zahodiť a máme z toho veľmi špecifický kompresný algoritmus. Kým bežný kompresný algoritmus dáva pre jednotlivé obrázky rôznu dĺžku kódu, ale udržuje rovnakú kvalitu kódovania, tu je to naopak: dĺžka kódu je rovnaká a kvalita kódovania sa mení. Do procesu kódovania autoencoderom je priamo premietnutá povaha dát, ktoré slúžili ako vzorky pre jeho trénovanie. Aby sme dostali z kompresnej polovice autoencodera hlbokú neurónovú sieť realizujúcu klasifikátor (určuje kategóriu) alebo regresor (aproximuje hodnotu funkcie), stačí teraz málo: pripojiť naň perceptron. Takáto sieť má väčšiu šancu na úspech, lebo najprv zakóduje vstupné dáta do vhodnej podoby a až potom ich preženie perceptronom. Hlboká neurónová sieť má teda pomerne hlbokú postupnosť konvolučných vrstiev prekladaných vrstvami redukujúcimi dimenziu dát a v závere aspoň dve plne prepojené vrstvy (obr. 2).



Obr. 1: Autoencoder



Obr. 2: Jedna z možností hlbokej neurónovej siete

Hlbokými sieťami nazývame však aj ďalšie siete, napríklad také, ktoré „perceptrónovú“ časť vôbec nemajú a sú len polotovarom, ktorý vstupným dátam priraďuje vhodný kód. Tieto siete sú trénované len nedávno zavedenými chybovými funkciami ako je napríklad metrická chybová funkcia. Umožňuje nový spôsob trénovania sietí, ktoré napríklad fotkám toho istého človeka priradia podobné vektor čísel, zatiaľ čo fotkám rôznych ľudí rôzne vektory.

Podobne za hlboké považujeme aj neurónové siete, ktoré majú architektúru zhodnú s autoencoderom, ale trénované sú z dát, kde výstup nie je totožný so vstupom – napríklad vstupom je čiernobiela podoba farebného výstupu. Takáto sieť sa potom naučí ofarbiť čiernobiely obrázok.

Ale to všetko by samo o sebe nestačilo, pretože čím viac vrstiev neurónová sieť má, tým ľahšie sa môže stať, že v procese trénovania sa už výkon nezlepšuje, avšak ešte stále je nedostatočný. Navyše sa môže stať, že proces trénovania prebehne uspokojivo, ale schopnosť siete fungovať na iných než trénovacích dátach je veľmi obmedzená (tzv. preučenie). Za úspechom hlbokého učenia stojí zo všetkého najviac práve riešenie týchto problémov. Aby bola zodpovednosť za fungovanie siete lepšie rozložená medzi všetky neuróny (aby schopnosti vrstvy nemohli zdegenerovať na schopnosti jedného neurónu), vkladajú sa do siete tzv. drop-out vrstvy. Tie v procese trénovania náhodne odstraňujú a vracajú jednotlivé neuróny do siete. Miesto toho je taktiež možné vkladať do siete normalizačné vrstvy, ktoré normalizujú signál, ktorý nimi prechádza. Dôležité je taktiež iniciálne nastavenie váh siete. Kým kedysi sa generovalo náhodne, dnes sa veľmi dôrazne hľadí na distribúciu iniciálnych hodnôt (optimálnym je tu tzv. Xavierove rozloženie).

Samozrejme hlboké učenie by nebolo prínosom bez výpočtového výkonu, ktorý je na natrénovanie hlbokých neurónových sietí potrebný. Kľúčovým výpočtovým prostriedkom pre tvorbu modelov hlbokého učenia je GPU – počítanie na množstve procesorov na grafických kartách pôvodne určených pre potreby herného priemyslu. Nemenej dôležité sú rozsiahle dátové úložiská, lebo na hlboké učenie potrebujeme veľké množstvo dát. Aj hlboké neurónové siete totiž fungujú len odtiaľ – potiaľ. Medzi to odtiaľ – potiaľ sa však dnes už môže zmestiť celý svet. Napríklad sa scanujú všetky ulice na svete a z týchto dát sa učia modely určené pre autonómne vozidlá. Veľkú úlohu v budúcnosti môžu zohrať taktiež kvantovo-mechanické počítače, ktoré potenciálne vedia uskutočniť proces trénovania siete v okamihu, avšak zatiaľ na to nestačí veľkosť ich pamäte (momentálne majú stovky, maximálne tisíce qubitov, zatiaľ čo ideálne by bolo mať ich miliardy).

Hlboké učenie prinieslo pokrok v mnohých oblastiach. V prvom rade v spracovaní obrazu: v detekcii, klasifikácii a rozpoznávaní objektov. Ale aj v iných oblastiach, napríklad rozpoznávaní reči alebo spracovaní jazykového textu. Pomocou hlbokej neurónovej siete je možné napríklad premeniť slovo na vektor čísel tak, aby podobné vektory zodpovedali podobným významom slov. Následne možno aj vety a celé výpovede premeniť na vektor čísel (pevnej dĺžky), kde podobné čísla zodpovedajú podobným výpovediam. To vedie k novej generácii chatbot-ov

(Brownlee 2018), ktoré už nefungujú na báze rôznych syntaktických transformácii jazykovej výpovede. Miesto toho sa vopred pripraví zoznam zámerov a ku každému sa uvedie niekoľko príkladov, akým sa dá na ne opýtať. Potom možno k akejkoľvek otázke zistiť, ku ktorému z uvedených zámerov má najbližšie a to tak, že hlboká neurónová sieť jej priradí vektor a porovnávame ho s vektormi, ktoré priradila spomínaným príkladom. Tvorba odpovede na rozpoznaný zámer môže byť pritom pomerne jednoduchá, napríklad vždy rovnakou, vopred pripravenou vetou.

Pokiaľ sa snažíme použiť model vytvorený hlbokým učením na mobilnom robotovi, môžeme naraziť na to, že nedisponujeme dostatočným výpočtovým výkonom. Ten síce nemusí byť tak obludný ako pri trénovaní modelu, ale aj tak značný, ak má byť model použitý v reálnom čase. Ako náhrada tu môže poslúžiť pripojenie na Internet, pretože mnohé modely sú dispozícii ako cloudové služby. Tie vykoná dostatočne výkonný hardware v dátovom centre prevádzkovateľa, robot len pošle vstup a prijme výstup. Dostupnosť cloudových služieb je celkom dobrá: v Európe jedno zavolanie trvá cca 80 ms.

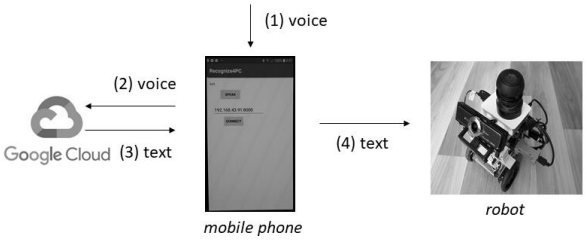
V dôsledku uvedeného pokroku sa samozrejme zvýšili nároky na to, čo by mal dokázať mobilný robot behajúci po stole. Čo nerozpráva, nepočuje a nevidí, už nie je trendy. Poďme sa teda v druhej časti nášho príspevku pozrieť na to, ako možno takúto funkcionalitu do mobilného robota z technického hľadiska dostať.

2 Integrácia metód UI do jednoduchého mobilného robota

Možnosti použitia spomínaných moderných metód uvedieme na príklade mobilného robota s podvozkom Boebot riadeného cez USB z Raspberry Pi3, ku ktorému je pripojená cez USB wide kamera a cez hlasový výstup reproduktor pre mobilné telefóny. Napriek malým rozmerom je Raspberry Pi3 vcelku normálny počítač s operačným systémom na báze Linuxu, ktorého výpočtový výkon posilníme pripojením Movidius Neural Stick do USB. Napájanie robotovi zabezpečí powerbanka pre mobilné telefóny.

Softwarovú platformu sme postavili na knižniciach OpenCV, (Bradski 2000) – v distibúcii OpenVINO, aby vedela využívať Movidius – a dlib, (Davis 2009). Tieto prostriedky nainštalujeme pomerne ľahko, len to dosť dlho trvá (hodiny). Ďalej nainštalujeme espeak, ktorý nám zabezpečí rečovú syntézu aj v slovenčine. Potom nastavíme sieť tak, aby sa automaticky pripájala do vhodnej WIFI siete, napríklad do access pointu smartfónu, ktorý bude zároveň cez rozhranie „mobilné dáta“ pripojený do Internetu. Z mobilu si môžeme potom zistiť IP adresu robota avyužiť jeho rozhrania (ssh a VNC) na vzdialený prístup.

Obr. 3: Robot Pingpong V.



Obr. 4: Použitie cloudovej služby (mobilný telefón tu slúži jednak ako access point, jednak ako mikrofón)

Implementáciu aplikačného softvéru je vhodné začať s diaľkovým ovládaním. Na robotovi vyvinieme TCP server, ku ktorému sa môže pripojiť akýkoľvek klient, následkom čoho sú jeho pokyny pretlmočené do príkazov pre podvozok. Už v tejto fáze je dobre ako komunikačný protokol použiť prirodzený jazyk, t. j. požívať pokyn „doprava“ a nie nejaký kód ako trebárs „R“. Následne môžeme telnet klienta nahradiť iným, ktorý tieto pokyny posiela na základe rozpoznávania reči. Raspberry Pi3 nemá vstup pre mikrofón a navyše mikrofón na robotovi je problematický z hľadiska množstva šumov, ktoré zachytí. Pohodlne však môžeme ako mikrofón použiť smartfón. To je výhodné aj preto, že operačný systém Android, ktorým je vybavený, má nelimitovaný prístup do Google cloudu, ktorý nahovorenú reč prevedie do textu, pričom má aj podporu slovenčiny. Potrebujeme akurát vhodnú apku do mobilu, ktorá nám umožní proces nahrávania, pošle nahranú reč do cloudu, prijme odtiaľ text a pošle ho cez TCP do robota. Túto aplikáciu sme vytvorili v Android Studiu a je k dispozícii na Github-e[1] (Lúčny 2019). Komunikačný protokol s robotom môžeme potom obohatiť o ďalšie vlastnosti, napríklad aby na pokyn „Povedz *X*“ cez espeak povedal *X*, alebo aby pri prijatí neznámeho povelu povedal „Nerozumiem“.

Bohatšie pochopenie príkazov, môžeme dosiahnuť, keď rozšírime syntaktické manipulácie s textom prijatého pokynu o volanie chatbota. Tu žiaľ

[1] https://github.com/andylucny/Recognize4PC

nemáme zatiaľ dobré prostriedky – MicroSoft Azure, ktorý má slovenčinu pomerne dobre zvládnutú je predsa len vhodnejší na windowsoch a otvorené riešenia ako wit.ai majú slovenčinu v plienkach.

Obr. 5: Rozpoznávanie tváre.

Ďalej doplníme spracovanie obrazu o detekciu tváre človeka, detekciu jej tvárových čŕt a rozpoznávanie o koho ide. Na túto úlohu treba mať v prvom rade vhodnú kameru. Jej rozlíšenie nesmie byť príliš veľké, lebo potom je príjem obrazu na Raspberry pomalý. Na druhej strane ani moc malé, lebo inak tvár na obraze nebude mať dostatok pixelov. Nesmie mať taktiež príliš malý uhol záberu (FOV).

Z hľadiska spracovania obrazu musíme najprv nájsť tvár, pričom je vhodný detektor, ktorý nie je orientovaný len na pohľad spredu, napr. res10_300x300_ssd_iter_140000.caffemodel z openCV (na to, aby tento model hlbokého učenia bežal v akceptovateľnom čase, využívame Movidius stick). Ďalej aplikujeme detektor tvárových čŕt (ten je postavený na kaskárdnom regresore, takže hoci je tiež výsledkom metódy počítačového učenia, je rýchly) – tu sme použili shape_predictor_68_face_landmarks.dat z knižnice dlib. Získanie čŕt tváre nám umožní tvár z obrazu dostať do normalizovanej podoby, ktorá je menej závislá na natočení hlavy. Hlavne ju potrebujeme vhodne otočiť a potom presne orezať. Pomocou ďalšieho modelu dlib_face_recognition _resnet_model_v1.dat z dlib túto normalizovanú podobu tváre premeníme na vektor a ten dokážeme porovnávať z databázou analogických vektorov z predošlých záberov (obr. 5). Pokiaľ je dostatočne podobný vektoru, ktorý už v databáze máme, ide o tú istú osobu a nemusíme si ho zapamätať. Na druhej strane, ak je odlišný, potrebujeme vedieť o koho ide. Na tento účel sme rozšírili hlasové ovládanie robota o pokyny „Toto je *X*“ a „Ja som *X*“, ktoré platia, pokým robot kontinuálne detekuje aktuálnu tvár. Keď robot s takýmto pokynom zistí, že vektor reprezentujúci tvár je od súčasných vzoriek vzdialený, zaradí ho pod udané meno do databázy. Pokiaľ pokyn uvádzajúci meno robot nedostane, môže na základe detekcie neznámej tváre položiť otázku „Ako sa voláte?“ Naopak, keď narazí na známeho človeka, môže povedať „Vy ste *X*“.

(Pozoruhodné je, že architektúra oboch použitých modelov hlbokého učenia je totožná, hoci ich funkcia je značne odlišná: jeden dáva tie obdĺžniky na obraze, kde je tvár a druhý dáva vektory, ktoré sú podobné ak ide tváre toho istého človeka a rozdielne ak ide o tváre dvoch rôznych ľudí. Konkrétne tu ide v oboch prípadoch o architektúru ResNet, avšak pri jej trénovaní sú použité iné dátové vzorky a hlavne iná chybová funkcia: v prvom prípade je to mean square error, v druhom už spomínaná metric loss function. Rosebrock, 2018)

Vo chvíli, kedy potrebujeme dostať z procesu prijímajúceho pokyny hlasového ovládania údaj o mene uvidenej tváre do procesu rozpoznávania tvárí, narazíme na potrebu výmeny dát medzi rôznymi procesmi a potrebu ich synchronizácie. Na tento účel je momentálne najpoužívanejším prostriedkom Robotic Operation System (ROS). Je to však dosť komplikované riešenie, preto my sme použili vlastnú architektúru vyvinutú v minulosti a to Agent-Space (www.agentspace.org, Lúčny 2004).

Pri prevádzke tohto robota sa ukazuje, že hoci je schopný udiviť svoji schopnosťami laikov, strašne moc

mu k dokonalosti chýba. Chýba mu nielen vyšší výpočtový výkon, ale aj základné percepčné schopnosti. Veľmi citeľným nedostatkom je napríklad to, že nevie rozlíšiť z ktorého smeru na neho hovoríme. Mimoriadnu potrebu tejto schopnosti, odhalíme po pár minútach interakcie s robotom, pričom inak by nám možno táto potreba ani neprišla na myseľ.

3 Záver

Prežívame obdobie búrlivého rozvoja umelej inteligencie a to najmä v oblasti percepcie. Je veľmi náročné, čo len stíhať študovať nové objavy, ktoré sa na nás valia každý deň. Podobne je to s technickými riešeniami, ktoré premietajú spomínané objavy do praktických aplikácii. Integrácia viacerých schopností, ktoré sa stali technicky dostupnými do jedného systému zaiste vyvolá potrebu po výkonnejšom hardvéri ale aj po z architektonického hľadiska solídnejších softvérových riešeniach (Kelemen 2001).

Literatúra

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Brownlee, J. (2018). Deep Learning for Natural Language Processing. *Jason Brownlee*.

Cybenko, G. (1989) Approximations by superpositions of sigmoidal functions. *Mathematics of Control, Signals, and Systems,* 2(4), 303–314.

Davis E. King (2009) Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research 10*, str. 1755-1758.

Goodfellow, I. - Bengio, Y. - Courville, A. (2016). Deep Learning. MIT Press, 2016.

Kelemen, J. (2001). From statistics to emergence – exercises in systems modularity In: *Multi-Agent Systems and Applications,* (Luck, M., Mařík, V., Štepánková, O. Trappl, R.), Springer, Berlin, str. 281-300.

Lúčny, A. (2004). Building complex systems with Agent-Space architecture. *Computing and Informatics,* Vol. 23, str. 1001-1036.

Lúčny, A. (2019). Easy Controlling a Robot using Voice for Hobbiests. *Reserchgate.net*, DOI: 10.13140/RG.2.2.25215.25761.

Rosebrock, A. (2018). Deep Learning for Computer Vision with Python. 2nd edition. *PyImageSearch.com.*

Rosenblatt, F. (1958), The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, *Cornell Aeronautical Laboratory, Psychological Review*, v65, No. 6, pp. 386–408.

Rumelhart, D. – Hinton G. – Williams, R. (1986). Learning internal representations by error propagation. *Parallel Distributed Processing*. Vol 1: Foundations. MIT Press, Cambridge, MA, 1986.