Nové možnosti fyzickej interakcie hráča s počítačovou hrou.

Andrej Lúčny, FMFI UK Bratislava

Abstrakt

Pri hraní počítačovej hry nemusí byť hráč telesne pasívny, avšak do nedávna potreboval na fyzické zapojenie sa do hry špecifický hardvér, ktorý si musel kúpiť, zapojiť a nastaviť. Avšak vplyvom zdokonaľovania strojového učenia a cloudových služieb je čoraz viac je možné na tento účel použiť aj úplne bežne dostupné a všadeprítomné prostriedky. Konkrétne zmieňujeme rozpoznávanie tvárí a postáv z obyčajnej web-kamery, ovládanie hry hlasom a využitie novej generácie chatbotov.

Úvod

Fyzická interakcia hráča spočíva v ovládaní počítačovej hry pomocou pohybov tela hráča alebo jeho hlasom. Bolo vyvinutých viacero hardvérových prostriedkov na snímanie údajov potrebných k realizácii takejto interakcie. Okrem rôznych dotykových snímačov sú predovšetkým 3D kamery, medzi ktorými dominuje kamera Kinect. Tieto kamery nasvecujú scénu pre nás neviditeľným infračerveným žiarením, odrazy ktorého sníma špeciálna kamera. Z deformácie nasvecovaného obrazca je potom možné vypočítať hĺbku obrazu. Táto informácia potom výrazne uľahčuje detekciu postavy hráča, určenie kostry tela hráča a podobne.

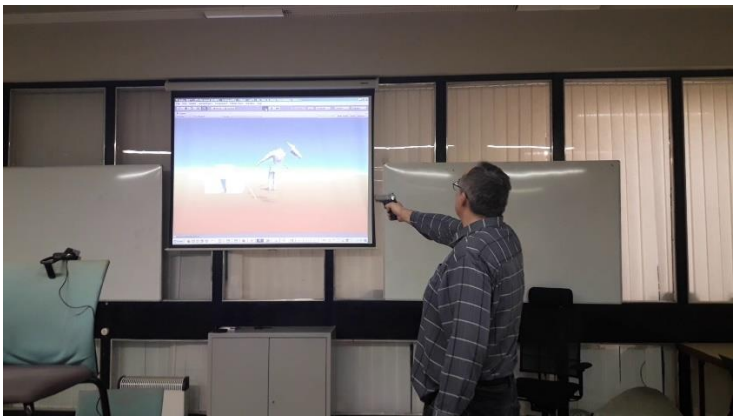
Potreba kúpy špecifického a nie príliš lacného hardvéru obmedzuje masovosť použitia fyzickej interakcie hráča s hrou. Pokiaľ by bola možná s prostriedkami, ktoré sú bežnou výbavou hráčskeho počítača alebo hráča, tento spôsob interakcie s hrou by sa stal dostupnejším a jej potenciálny trh oveľa väčším. Takými prostriedkami sú v súčasnosti nielen web kamera zabudovaná v hráčskych notebookoch, ale napríklad aj mobilný telefón so základnou výbavou (kamera, mikrofón, pripojenie do Internetu). Rovnakú službu spraví aj lacný hardvér, ktorú nie je primárne určený pre hry, napríklad laserové ukazovadlo.

Cieľom tohto príspevku je inšpirovať tvorcov počítačových hier k využívaniu fyzickej interakcie, najmä tej, ktorá môže mať masových charakter a doteraz nebola reálne dostupná. Fyzická interakcia je nielen zaujímavá, ale aj spoločensky veľmi prospešná, lebo odbúrava práve nepríjemnú stránku hier – nedostatok pohybu hráča.

Využitie lacného hardvéru

Určité možnosti fyzickej interakcie hráča pomocou dostupných a lacných prostriedkov sa ponúkali k dispozícii už dávnejšie. Príkladom môže byť použitie laserového ukazovadla či laserovej pištole z hračkárstva, ktorej laserový lúč monitoruje obyčajná kamera (Obr 1.). Na stenu premietneme dátovým projektorom určitú scénu a hráč v nej laserovým lúčom triafa objekty. Miesto dopadu laserového lúča – hoci ho niekedy ťažko vidno i očami – dokáže obyčajná kamera spoľahlivo zdetegovať, pokiaľ na ňu pridáme dostatočne tmavý filter. Nastaviteľnú filtráciu dosiahneme ak filter zrealizujeme dvomi prekrývajúcimi sa polarizačnými filtrami. Ich vzájomnou polohou môžeme potom určovať intenzitu filtra. Z kamery potom dostaneme čierny obrázok s prípadnou bielou bodkou v čase výstrelu. Ďalej je len vecou správnej kalibrácie odvodiť z polohy zásahu parametre lúča, ktorý v hernom engine odsimuluje výstrel. Pomocou collider-ov možno potom vyvolať v zasiahnutých objektoch explóziu, ktorá objekt zdeštruuje. Samozrejme informáciu o smerovaní výstrelu musíme do herného engine nejako dostať, nakoľko spracovať obraz z fyzickej kamery sa nám ľahšie urobí v externej aplikácii. Najprístupnejším riešením je použiť TCP protokol, pričom aplikácia bude serverom a hra klientom. Keďže aplikácia beží na

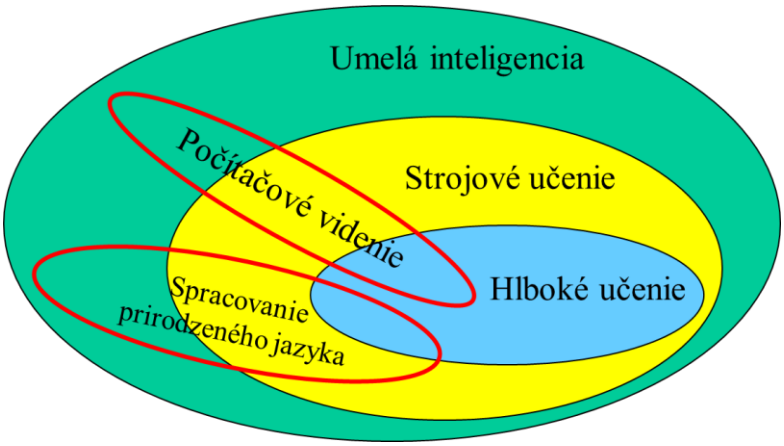
rovnakom počítači a TCP port na ktorom počúva, môžeme pevne zvoliť, pre hru to nebude predstavovať problémy s nastavením jej konfigurácie, pripája sa vždy na localhost.



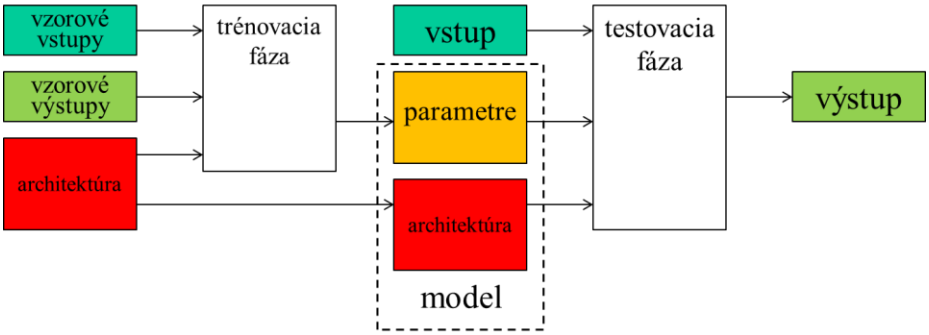
Obr. 1. Streľba na avatara s laserovej pištole. Herné prostredie je premietané na plátno, ktoré sníma kamera, ktorú na obrázku vidno vľavo dolu.

Využitie nových metód umelej inteligencie.

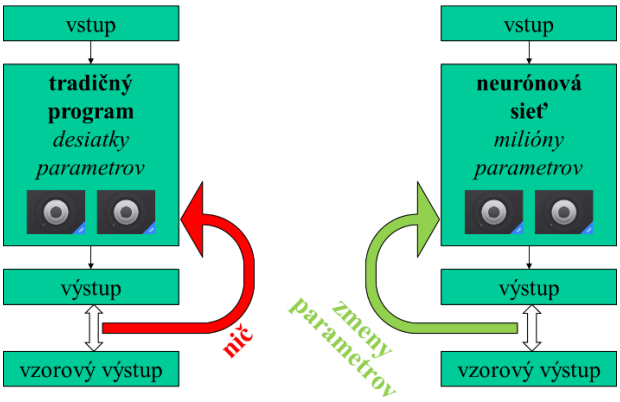
Prejdime však k tým prostriedkom, ktoré sa stali reálnymi len v nedávnej dobe a prináša ich pokrok v oblasti umelej inteligencie. Typickou úlohou na ktorej možno tento pokrok demonštrovať je už spomínaná detekcia postavy hráča. Kým klasické riešenie používa špeciálnu kameru, ktorá vie poskytnúť informáciu o hĺbke obrazu, pomocou umelej inteligencie je možné získať analogickú informáciu z obyčajnej kamery. To, že je to vôbec možné by nás nemalo prekvapiť, lebo človek takú schopnosť má a to aj keď sa pozerá jedným okom. Úžasný je však spôsob, akým sa podarilo napodobniť túto schopnosť v počítači. Vďaka tomu, že sme už vynašli spôsob ako z obrazu určiť postavu hráča so špeciálnou kamerou, môžeme rovnakú scénu snímať obyčajnou kamerou a získať tak nielen mnoho anotovaných vzoriek , t. j. sadu obrázkov z obyčajnej kamery, ku ktorým máme želaný výsledok v podobe detegovanej postavy hráča. S pomocou takého dataset-u a metód umelej inteligencie vynájdených pomerne nedávno a známych pod názvom hlboké učenie [9] – vieme nájsť riešenie tejto úlohy s použitím obyčajnej kamery.



Obr. 2. Vzťah umelej inteligencie ku strojovému a hlbokému učeniu. Červenou linkou sú vyznačené aplikačné oblasti.



Obr. 3. Základná schéma strojového učenia



Obr. 4. Porovnanie tradičného programu a neurónovej siete. Neurónová sieť je čokoľvek majúce parametre o čom vieme z rozdielu želaného a skutočného výstupu povedať, ako zmeniť parametre, aby sa tento rozdiel zmenšil.

Hlboké učenie (Obr. 2) je technika strojového učenia (Obr. 3) založená na špeciálnom type tzv. neurónových sietí. Pod neurónovou sieťou si je možné predstaviť čokoľvek, čo má nejaké parametre a čo vie spočítať z nejakého vstupu nejaký výstup, pričom z rozdielu tohto výstupu a želaného výstupu (počítame ho tzv. chybovou funkciou), vieme povedať ako zmeniť parametre, aby sa tento rozdiel zmenšil (Obr. 4). Každý algoritmus má nejaké parametre a pri reálnych úlohách je ich pomerne veľa. Vývojári softvéru ich nie celkom správne nazývajú bulharskými konštantami. Pokiaľ tieto parametre nastavuje človek, zvládne rozumne použiť maximálne desiatky parametrov. Neurónová sieť je špecifický algoritmus, ktorý skoro nič iné než bulharské konštanty neobsahuje a pri tzv. hlbokom učení má spravidla desiatky miliónov parametrov. Je nemysliteľné, aby ich nastavenie bolo hľadané ručne. Vďaka uvedenej základnej vlastnosti je ho však možné nájsť postupne tzv. trénovaním. Začneme s nejakým náhodným nastavením parametrov neurónovej siete. Zo dátovej kolekcie vyberieme nejakú časť vzoriek a spustíme na nej takto nastavenú neurónovú sieť. Z rozdielu výstupu a želaného výstupu odvodíme zmenu parametrov. Zmeníme ich a celý proces opakujeme. Opakujeme ho dovtedy kým odchýlka medzi výstupmi a želanými výstupmi nie je prijateľne malá alebo kým sa výkon siete nezlepšuje. Pokiaľ zmenu parametrov – tzv. gradient – odvodzujeme zo všetkých vzoriek, trénovací algoritmus radíme medzi tzv. Gradient Descent. Pokiaľ z jedinej vzorky, tak je to tzv. Stochastic Gradient Descent. Najobľúbenejšou metódou je však odvodiť gradient z náhodne vybranej skupiny vzoriek a vtedy hovoríme o Batch Gradient Descent. V určitých prípadoch, keď je vzoriek v skupine málo (napr. 15), ide o miniBatch Gradient Descent. Princíp trénovacieho algoritmu je jednoduchý, ale má svoje úskalia. Za prvé, zmena

parametra je daná len orientačne, t. j. vieme či parameter treba zvýšiť alebo znížiť, že ho treba zvýšiť viac ako iný parameter, ale nevieme presne o koľko. Finta, akou sa snaží algoritmus odhadnúť o koľko zmeniť parameter, je jadrom tohto algoritmu a od rôznych prístupov rozlišujeme viac konkrétnych optimalizátorov, najbežnejšie sú *rmsprop* a *ADAM*. Za druhé, aj keď sa nám podarí parametre nastaviť tak, že na vzorkách dosahujeme výbornú zhodu skutočného a želaného výstupu, ešte stále to nemusí byť nič platné. Potrebujeme totiž aby sieť fungovala aj pre vstupy, ktoré vo vzorke nemáme. Pomocou rozdelenia vzoriek na trénovaciu a testovaciu časť sa môžeme presvedčiť či nenastalo tzv. preučenie (overfitting), kedy sieť veľmi dobre funguje na trénovacej časti vzoriek a vôbec nefunguje na testovacej. Hovoríme vtedy o nízkej predikčnej schopnosti siete. Práve vyriešenie tohto problému pomocou techniky zvanej Dropout, ku ktorej sa neskôr pridali ďalšie ako je Batch normalization, naštartovalo boom hlbokého učenia. Zlomovým okamih sa odohral v roku 2012, kedy Alex Krizhevsky so sieťou AlexNet vyhral súťaž v určovaní objektov na obrázkoch (ImageNet, ILSVRC2012) s neuveriteľným náskokom vyše dvanásť percentuálnych bodov pred ostatnými riešeniami [13]. Keďže zlepšenia v tejto brandži sa bežne pohybujú medzi jedným až dvomi percentuálnymi bodmi, presvedčilo to všetkých, že táto technológia dozrela a začala prinášať ovocie. Odvtedy neuplynulo veľa času, ale v danej oblasti sa urobilo mnoho dobrej práce, ktorá ovplyvnila a z väčšej časti ešte len ovplyvní mnohé oblasti vedy, techniky a celkovo ľudskej spoločnosti.

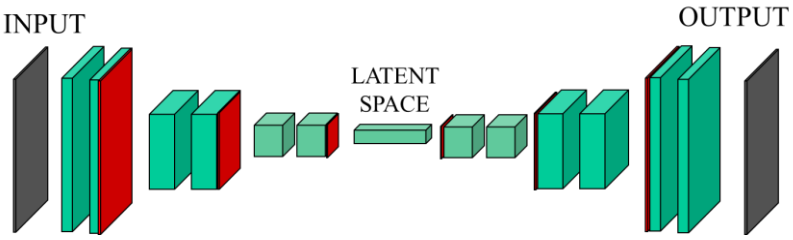
Každý výdobytok hlbokého učenia vznikne tak, že sa uhádne vhodná architektúra neurónovej siete a prácne zhromaždená vzorka dát (vstupov a želaných výstupov) sa trénovacím procesom premení na správne nastavenie parametrov siete. Používateľ potom dostane dve veci – architektúru siete a parametre siete (tzv. váhy), dohromady to nazývame modelom. V súčasnosti je dostupných mnoho hotových modelov, ktoré sú zhromažďované na webových úložiskách zvaných trochu expresívne zverince (model zoo). Jeden model deteguje ľudské tváre na obraze, iný vracia opornú kostru postavy na obraze, ďalší zmysluplne ofarbí čiernobiely obrázok. Všetky spomenuté modely možno dostať trénovaním pri ktorom sa gradient počíta priamo zo sady výstupov a k nim prislúchajúcich želaných výstupov na základe určitej chybovej funkcie, napríklad *mse* (priemer z druhých mocnín rozdielov).

Vo zverinci modelov nájdeme však aj také, ktoré vznikli použitím rafinovanejšej chybovej funkcie ako je metrická chybová funkcia. Príkladom takého modelu je vektorizácia tvárí [8], pri ktorej vytrénujeme neurónovú sieť tak, že pre každú tvár vráti nejaké číslo – tzv. deskriptor (napríklad v podobe vektora 1024 reálnych čísel od -1 po 1) a to tak, že tváram rovnakej osoby vracia podobné čísla a tváram rôznych osôb čísla rôzne. Vychádzame pritom z datasetu, ktorý obsahuje tváre anotované menom danej osoby, pričom z každej osoby máme viacero jej tvárí (z rôznych pohľadov, s rôznym výrazom, s rôznym účesom a podobne). Takýto dataset však neobsahuje vstup a želaný výstup (t. j. tvár a číslo) – nevieme aké presné číslo má sieť odpovedať, vieme len akú má mať toto číslo vlastnosť vo vzťahu k iným výstupom siete. Aj toto sa však dá pomocou vhodnej chybovej funkcie akou je metrická chybová funkcia zvládnuť. Úprava váh siete sa pri ich použití odvodí z niekoľko málo najhorších prípadov aktuálnych výstupov. Vezmeme vzorku tvárí, spustíme pre každú tvár sieť a dostaneme tak pre každú tvár číslo. Potom kontrolujeme po dvojiciach, ktoré dvojice sú zlé. Zlé sú tie, ktoré zodpovedajú tvári rovnakého človeka a majú príliš rôzne číslo ako aj tie, ktoré zodpovedajú tváram rôznych ľudí a majú podobné číslo. Pre dvojicu zlých čísel vieme povedať ako ich upraviť, aby sa k sebe priblížili alebo vzdiali. A z týchto zmien niekoľko málo najhorších prípadov odhadneme potrebnú úpravu váh. Je takmer neuveriteľné, že to funguje, ale funguje to. Po trénovaní dostaneme sieť, ktorú môžeme použiť na rozpoznávanie tvárí nasledovným spôsobom. Vezmeme sadu osôb a pre každú máme niekoľko fotiek ich tvárí. Zistíme

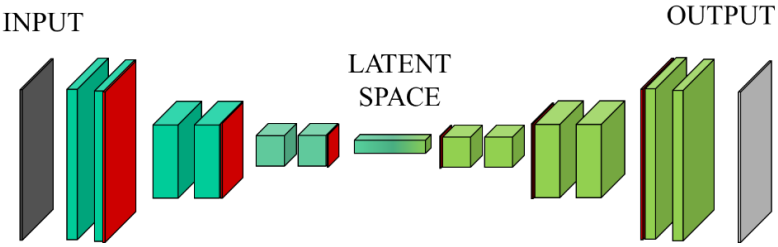
a zapamätáme si aké čísla pre ne sieť vráti. Potom vezmeme neznámu tvár, zistíme jej číslo a porovnáme ho so zapamätanými číslami. Ak medzi nimi nájdeme číslo podobné, určíme že ide o danú osobu.

A funguje to nielen pre tváre. Podobne možno spracovať všetky slová v jazyku tak, aby pre ne sieť vrátila číslo tak, že dvom slovám podobného významu vráti podobné čísla a dvom číslam rôzneho významu vráti rôzne čísla [4]. Následne možno dokonca urobiť sieť (tá už bude musieť mať aj tzv. rekurentné spojenia), ktorá pre dve výpovede z niekoľko málo viet vráti podobné čísla ak ich význam je rovnaký a rôzne čísla ak je ich význam iný. S takou sieťou potom ľahko skonštruujeme chatbot-a novej generácie. Kým klasický chatbot aplikuje na otázku rôzne textové transformácie a snaží sa tak dopátrať čo sa ho človek opýtal, chatbot využívajúci hlbokú neurónovú sieť vypočíta k otázke jej číslo a porovná ho z číslami ukážok akými vetami sa dá opýtať na určitý zámer. Tvorca chatbota, teda dopredu pripraví len zoznam zámerov a ku každému zámeru uvedie niekoľko príkladov, akým sa dá na zámer opýtať. Z týchto príkladov sa vypočítajú ich čísla a to, ktoré je najbližšie k číslu položenej otázky, určuje zámer.

Ako sme už spomenuli, tvorca modelu musí uhádnuť vhodnú architektúru neurónovej siete. Pri tomto sa trochu pristavíme, aby sme vysvetlili , prečo sa tomuto druhu neurónových sietí vlastne hovorí hlboké a aby sme trochu prenikli do podstaty toho, v čom sú inovatívne. Predchodcom všetkých hlbokých sietí je autoencoder [2], či presnejšie tzv. hlboký autoencoder (Obr. 5). Je to neurónová sieť pozostávajúca z viacerých blokov tzv. konvolučných vrstiev, ktoré sú prekladané vrstvami meniacimi dimenziu dát.



Obr. 5. Hlboký autoencoder – praotec všetkých hlbokých neurónových sietí



Obr. 6. Hlboká neurónová sieť typu encoder – decoder. Pozostáva akoby z dvoch polovíc rôznych autoencoderov. Jej najjednoduchšie použitie slúži na ofarbenie čiernobielych filmov.

Konvolučná vrstva pozostáva z neurónov, ktoré sú prepojené na neuróny vo vrstvách predchádzajúceho bloku a to len na tie, ktoré sa nachádzajú v blízkom okolí ich pozície: najčastejšie sa pritom používajú okolia 3x3 alebo 5x5, každý neurón má teda spravidla 9 či 25 spojení do každej vrstvy predchádzajúceho bloku. Neurón pracuje tak, že robí vážený priemer výstupných hodnôt pripojených neurónov z predchádzajúceho bloku, pripočíta k nemu tzv. bias a aplikuje na to tzv. aktivačnú funkciu. Typickou aktivačnou funkciou pre autoencoder je ReLu, ktorá zo záporného čísla urobí nulu a kladné ponechá. S každým spojením má teda neurón asociovanú určitú váhu – kladné alebo záporné reálne číslo, pričom všetky neuróny v jednej vrstve majú tieto váhy ako aj bias rovnaké. Čiže hoci neurónov sú vo vrstve

desaťtisíce, napríklad 300x300 a každý z nich má napríklad 3x3 spojení na každú vrstvu predchádzajúceho bloku plus bias, celá vrstva má púhych 3x3 x počet vrstiev v predchádzajúcom bloku plus 1 (za bias) parametrov. To je dosť málo a aby sieť nabrala spomínané desať milióny parametrov, musí byť takých vrstiev v sieti fakt veľa, inými slovami: musí byť hlboká. Treba pritom poznamenať, že takéto zdieľanie váh viacerými neurónmi odlišuje konvolučnú vrstvu od všetkých ostatných typov sietí, je to vlastnosť nevídaná, pretože neurónové siete viac či menej napodobňujú činnosť mozgu človeka a je nepredstaviteľné, že by táto vlastnosť bola biologicky relevantná. Technológia neurónových sietí sa tu teda vymaňuje z rámca, ktorý jej poskytuje príroda a vynachádza mechanizmus lepší, než je možné v prírode dosiahnuť. Zdieľanie váh má za následok dôležitú vec: pokiaľ sieť vystavíme určitému podnetu, jej reakcia je rovnaká bez ohľadu na jeho pozíciu. Takže ak je predkladáme obraz 300x300 pixelov a učíme sieť rozpoznávať na ňom loptičku, nezáleží na tom, na aké miesto na obraze ju umiestnime: pri trénovaní ju môžeme dávať vždy vľavo hore a pri testovaní ju spozná i v strede či vpravo dole, hoci v takej polohe ju nikdy predtým „nevidela“.

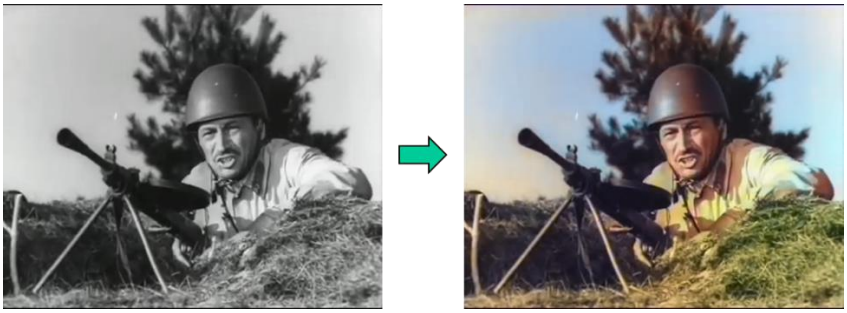
Ako sme už spomenuli konvolučné vrstvy sú v autoencoderi prekladané vrstvami, ktoré menia dimenziu dát. Konkrétne v prvej polovici dimenziu dát redukujú a v druhej ju expandujú. Redukcia spočíva spravidla vo výbere maxima s výstupov niekoľkých (napríklad 2x2) susedných neurónov, expanzia v rozkopírovaní jednej hodnoty na viaceré. Dimenzia dát sa teda postupne redukuje, napríklad zo vstupných 300x300 cez 150x150, 75x75 až na napríklad 1x1024 (tento blok siete nazývame latentný priestor) a potom zase postupne expanduje presne na pôvodnú veľkosť (prevod dvojrozmerných dát na jednorozmerné sa robí špeciálnou vrstvou zvanou Flatten).

Autoencoder trénujeme tak, že mu na vstup predkladáme určité dáta, napríklad obrázky snehuliakov a požadujeme, aby dával výstup čo najpresnejšie zodpovedajúci vstupu. To sa môže javiť ako načisto zbytočné, avšak pokiaľ sa nám podarí autoencoder natrénovať, môžeme druhú polovicu (za latentným priestorom) zahodiť a zvyšok nám poskytuje veľmi zaujímavý kódovací stroj. Keďže výstupy autoencodera sú až na nepresnosť trénovania totožné so vstupmi, každý jeden blok konvolučných vrstiev musí obsahovať informáciu, ktorá sa svojou obsažnosťou vyrovná vstupným dátam. A to sa týka aj latentného priestoru. Prvá polovica autoencodera teda zakóduje napríklad vstupný obrázok s 90000 pixelmi (0..255) do vektora 1024 reálnych čísel (-1..1). Na rozdiel od bežného kódovania, ktoré dáva premenlivú dĺžku a stabilnú kvalitu kompresie, autoencoder produkuje kód vždy rovnakej dĺžky a rôznej kvality kompresie (pričom dobrú kvalitu dosahuje na predložených vzoroch a im podobným dátam – napríklad ak trénujeme s obrázkami snehuliakov, tak snehuliakov kódujeme s vysokou kvalitou, zatiaľ čo obrázok stromu sa po zakódovaní a rozkódovaní, môže od pôvodného dosť líšiť).

Kód produkovaný autoencoderom pozostáva z čísel, ktoré vystihujú dáta s ktorými sieť pracuje oveľa úspornejšie – extrahuje to, čo je v nich podstatné. S týmto kódom môžeme pracovať rovnocenne k vstupným dátam. A to je veľmi cenná vlastnosť pre hľadanie transformácii dát. Už len z hľadiska veľkosti dát zaiste ľahšie nájdeme vhodnú transformáciu nad 1000 prvkami, než nad 90000. Navyše kód môže obsahovať prvky zodpovedajúce inherentným vlastnostiam dát a oddeľovať tak od seba nezávislé zložky popisujúce dáta. Že to tak naozaj je, dá sa dobre demonštrovať pri kódovaní ľudských tvárí autoencoderom: medzi prvkami latentného priestoru je možné identifikovať farbu kože, vzdialenosť očí alebo pohlavie [22].

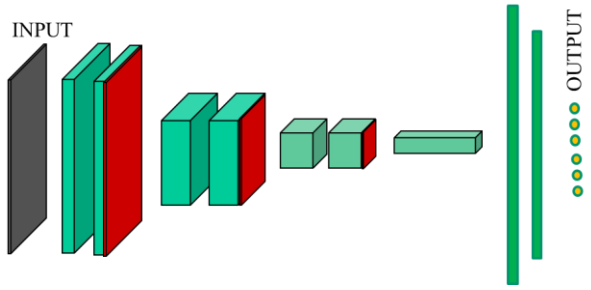
Hoci samotný autoencoder je prakticky nanič (dáva na výstupe kópiu vstupu), predstavuje základný návod ako má architektúra hlbokej neurónovej siete vyzerať. Najjednoduchšie transformácie môžeme

dokonca realizovať púhym prevzatím architektúry autoencodera. Samozrejme ho už nebudeme trénovať na výstupy totožné so vstupmi, takže naša architektúra pozostáva vlastne z polovíc dvoch rôznych autoencoderov. Nazývame ju preto architektúrou encoder – decoder (Obr. 6) a patrí medzi tzv. plne konvolučné siete [2]. Najjednoduchším príkladom použitia takejto siete je ofarbenie čiernobieleho filmu [23]. Vezmeme moderný ruský vojnový film, z neho ľahko vyrobíme čiernobiely. Potom budeme trénovať túto hlbokú neurónovú sieť tak, aby pre čiernobiely vstup dávala zodpovedajúci farebný výstup. Ak sa nám to podarí, úspešne takou sieťou ofarbíme čiernobiely sovietsky vojnový film (Obr. 7).

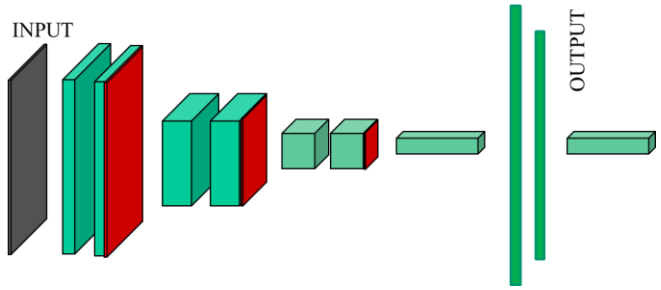


Obr. 7. Čiernobiely sovietsky film „Dolgie versty vojny“ ofarbený hlbokou neurónovou sieťou typy encoder – decoder. Na ofarbenie bol použitý caffe model *colorization v2*.

O niečo pokročilejšia úloha, ktorá sa rieši pomocou encodera – decodera je sémantická segmentácia obrazu [1]. Má význam napríklad pre autonómne vozidlá a obraz z kamery spracúva tak, že pre každý pixel sieť povie, či je súčasťou cesty, chodníka, budov, oblohy, iných áut alebo chodcov.



Obr. 8. Hlboká neurónová sieť typická pre klasifikátory ako je VGG. Pozostáva akoby z polovice autoencodera na ktorú je pripojený perceptron. Výstupom zo siete sú pravdepodobnosti kategórii.



Obr. 9. Hlboká neurónová sieť typická pre detektory ako je YOLO. Pozostáva akoby z polovice autoencodera na ktorú je pripojený perceptron, jeho výstupom však nie sú pravdepodobnosti ale výskyty detegovaných objektov.

Okrem transformácii jedných dát na druhé, vedia hlboké neurónové siete riešiť úlohy klasifikácie a detekcie. Pri návrhu architektúry pre ich riešenie spravidla kombinuje prvú polovicu autoencodera s klasickou neurónovou sieťou – tzv. perceptronom. Perceptron [19] má trochu iný charakter ako konvolučné siete: nie je hlboký a jeho neuróny nezdieľajú váhy. Napriek malému množstvu vrstiev (2 až 3) má teda veľa parametrov a je o ňom matematicky preukázané, že dokáže aproximovať každú rovnomerne spojitú funkciu [6]. Teoreticky je to veľmi silná mašinka, schopná sa naučiť skoro čokoľvek, avšak kŕmiť ju napríklad obrazom 300x300 pixelov, nevedie v praxi k dobrým výsledkom. Oveľa lepšie výsledky dostaneme ak do nej vstupuje vektor 1x1024. Takže prvá polovica autoencodera dokáže bez straty informácie premeniť dáta nevhodné pre perceptron na dáta vhodné. Architektúra potom pozostáva z prvej polovice autoencodera, perceptronu a výstupnej vrstvy. Toto spojenie dvoch šikovných mašiniek strojového učenia využíva väčšina hlbokých neurónových sietí slúžiacich pre klasifikáciu (kategorizáciu) alebo detekciu, ako sú VGG, ResNet a ďalšie [18]. V prípade klasifikácie je na výstupe vektor pravdepodobností príslušnosti vstupu k daným kategóriám (vrstva nazývajúca sa Softmax) (Obr. 8). V prípade detekcie je to blok vektorov spravidla dĺžky 7, ktoré popisujú umiestnenie detegovaného objektu vo vstupných dátach (napríklad obdĺžnik na obraze), vierohodnosť detekcie a typ detegovaného objektu (Obr. 9).

Siete ako ResNet [10] dokážeme natrénovať na klasifikáciu (napríklad urči, či je na obrázky pes, mačka, myš, kôň, ... hroch alebo niečo iné) a im podobné varianty ako je YOLO detektor [17] na detekciu (urči, kde na obraze je pes, mačka, myš, ...) Stačí mať k tomu dostatočne veľký dataset, t. j. súbor anotovaných (ručne) určených dát. Získať kvalitný dataset je pri väčšine aplikácii najnáročnejšia časť úlohy. Našťastie existujú viaceré iniciatívy, ktoré tieto datasety vytvárajú a poskytujú voľne k použitiu. Zbierku takých datasetov ponúkajú významné univerzity (Berkeley, Caltech, ...) ako aj na túto činnosť špecializované organizácie (NIST, Kaggle, ...). Najznámejšie zbierky datasetov možno nájsť pod názvami MNIST, CIFAR, ImageNet, Kaggle, UC Berkeley, Caltech, COCO a mnohými ďalšími [18].

Trénovanie nám navyše výrazne môže urýchliť tzv. transfer learning, učenie prenosom. Spočíva v tom, že zvolenú architektúru netrénujeme z náhodného nastavenia váh, ale ako východzí stav vezmeme váhy získané pri inej, podľa možnosti všeobecnejšej úlohe. Je totiž veľmi pravdepodobné, že ak budeme trénovať detektor snehuliakov, tak značná časť siete bude podobná ako v už hotovom detektore áut. Ide o podobné úlohy a obe musia pracovať s hranami, základnými tvarmi ako je oblúk alebo úsečka a podobne. Možno si taktiež pomôcť postupným trénovaním jednotlivých častí siete. Napríklad pre sieť, ktorá pozostáva z polovice autoencodera a perceptronu, vyjdeme z váh získaných pri trénovaní celého autoencodera, tieto váhy zafixujeme a doučíme váhy perceptronu. Potom to pretrénujeme ešte celé. Tieto triky výrazne pomáhajú nielen rýchlosti ale aj kvalite výsledku a preto s každým typom siete (VGG, ResNet, AlexNet, DarkNet, GoogleNet, SegNet, ...) je poskytovaný rad tzv. predtrénovaných (pretrained) modelov, ktoré sú vhodným východiskom trénovania.

Prácu s použitím hlbokého učenia si však môžeme ešte výraznejšie uľahčiť, pokiaľ sa nám podarí nájsť hotový model, ktorý vyhovuje našej aplikácii. Množstvo modelov je voľne dostupných v už spomenutých zverincoch modelov. Spravidla model tvoria dva súbory, z ktorých jeden popisuje architektúru a druhý váhy, získané jej trénovaním. Problémom je, že je viacero softvérov na trénovanie sietí (Pytorch, Caffe, TensorFlow, Teano, MxNet, Keras, DarkNet, Dlib, ...) a každý z nich má svoj formát modelov. Takže často narazíme na problém, že model by aj dostupný bol, ale nie je vo formáte, v ktorom by sme ho potrebovali. A to je problém často neriešiteľný, lebo prevody medzi jednotlivými modelmi sú obtiažne až nemožné. Preto vznikajú aj iniciatívy na zjednocovanie modelov a ich formátov, napríklad Gluon.

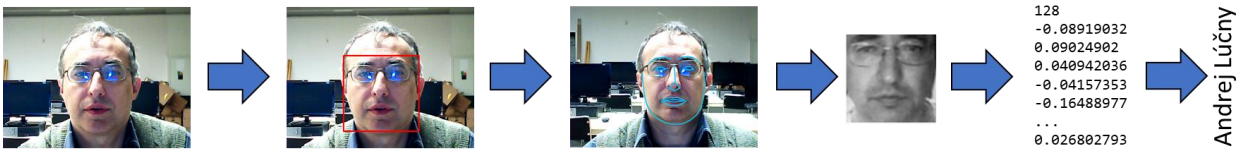
Pri použití modelu hlbokej siete v aplikácii zápasíme s výkonom počítača na ktorom model spúšťame. Vo všeobecnosti sú hlboké neurónové siete pomalšie než klasické metódy. Avšak, za prvé, poskytujú lepšiu kvalitu a za druhé, sú úlohy pri ktorých sú rýchlejšie. Typickou takou úlohou je detektor objektov. Klasické riešenie spočíva v aplikovaní klasifikácie na všetky možné miesta obrazu, berúc do úvahy aj rôzne veľkosti miesta – takto to robí napríklad HOG detektor [7]. A to je pomalšie než raz spustiť hlbokú neurónovú sieť s architektúrou YOLO (čo je skratka od veľavravného You Only Look Once), ktorá zvláda analogickú detekciu [17].

A to samozrejme platí len pokiaľ uvažujeme o behu hlbokých sietí na CPU (Central Processing Unit). Rýchlejšie časy sa dosahujú so správnou grafickou kartou (kompatibilnou s rozhraním CUDA, ktoré využíva jeho nadstavba cuDNN od NVIDIA), t.j. na hardvéri vybavenom s GPU (Graphics Processing Unit). Možné je taktiež pridať výpočtový výkon ako USB zariadenie – VPU (Visual Processing Unit) – ako sú Movidius Neural Computing Stick a Intel Neural Computing Stick 2. V prípade, že máme len CPU môžeme si aspoň trochu pomôcť cez softvérový akcelerátor ako je Inference Engine od Intelu šírená v knižnici OpenVINO: jej súčasťou je aj optimalizátor, ktorý k dodanému modelu vyrobí menšiu, rýchlejšiu ale z hľadiska kvality len o málo horšiu alternatívu.

Aplikácie herného priemyslu majú veľkú výhodu v tom, že ich odberatelia spravidla disponujú hráčskym PC alebo notebookom, ktorý je vybavený výkonnou grafickou kartou s GPU – boli to koniec koncov práve grafické karty pre počítačové hry, ktoré umožnili celému sektoru hlbokého učenia vzniknúť. Horšie je to s mobilnými aplikáciami, lebo hoci v mobiloch je úctyhodný výpočtový výkon, nevyrovná sa počítaču. Tam sa skúšajú rôzne triky ako modely čo najviac „odľahčiť“ a hľadá sa kompromis medzi výkonom a kvalitou.

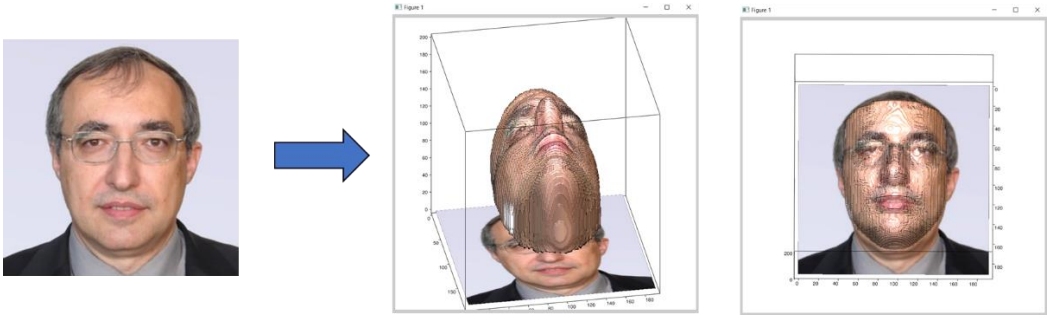
Čo sa z uvedeného dá konkrétne využiť? Je toho určite veľa, obmedzíme sa preto na zopár príkladov.

Už sme spomínali, že hlboké neurónové siete dokážu nielen detegovať tvár na obraze, ale aj tvári priradiť číselný deskriptor, pomocou ktoré možno odlíšiť tvár jedného človeka od druhého [8]. Je pritom použitých hneď viacero sietí aj iných metód (Obr. 10). YOLO detektor rozpozná, kde na obraze je tvár, potom sa rozpoznajú črty tváre a časť tváre potrebná pre rozpoznanie je na základe týchto čŕt prevedená do normalizovanej podoby. Tá vstupuje do ďalšej hlbokej siete, ktorá jej priradí deskriptor. Ten sa porovnáva s deskriptormi v databáze známych osôb a na základ toho je určená osoba. Pri počítačovej hre sa vďaka tomu hráč nemusí nijako identifikovať, pokiaľ ho už hra pozná.



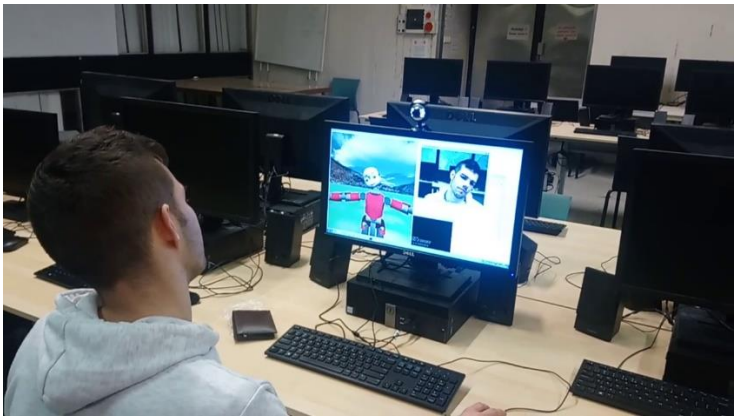
Obr. 10. Identifikácia hráča na základe spracovania obrazu z kamery. Tvár na obraze bola nájdená pomocou Caffe modelu res10_300x300_ssd_iter_140000 (architektúra ResNet). Črty tváre boli rozpoznané klasickejšou metódou a to kaskádnym regresorom z knižnice Dlib. Z tejto knižnice bol aj dlib_face_recognition_resnet_model_v1.dat použitý pre finálny prevod tváre na deskriptor (vektor 128 čísel od -1 po 1). Pozoruhodné je, že má na takmer identickú architektúru ako detektor, t.j. ResNet – líši sa len vo výstupnej vrstve – avšak trénovanú z úplne iných dát a inou chybovou funkciou.

Hráča však možno nielen identifikovať, ale ho aj preniesť do hry, t. j. vyrobiť k nemu zodpovedajúceho avatara, ktorý má jeho tvár. Na to by donedávna bolo potrebné hráčovu tvár nasnímať 3D scanerom, čo je veľmi drahá záležitosť. Avšak podarilo sa natrénovať model VRN [12], hlbokú neurónovú sieť, ktorá robí 3D rekonštrukciu tváre z obyčajnej 2D fotky (Obr. 11).

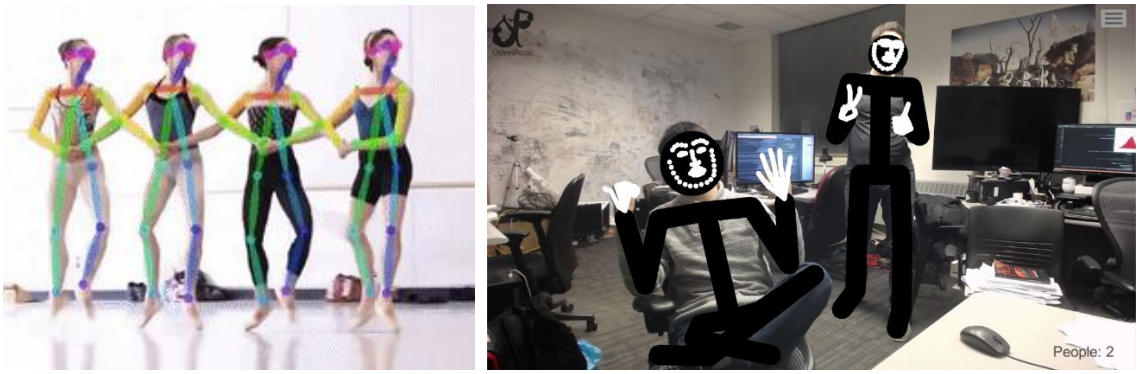


Obr. 11. 3D rekonštrukcia tvárovej časti hlavy z 2D fotky. Použitý bol tensorFlow model VRN.

Detekcia tváre sa dá úspešne použiť aj pre interakciu s hrou na základe pohybu hlavy (Obr. 12). Využívame pritom prvé dve transformácie z Obr. 10, a na základe čŕt tváre (nos, oči) zisťujeme náklon hlavy a premietame ho napríklad do náklonu hlavy postavy v hre.



Obr. 12. Ovládanie hry pohybom hlavy. Použitý je ResNet model pre rozpoznanie tváre (Caffe) a kaskádny regresor na detekciu čŕt tváre (Dlib).



Obr. 13. Detekcia pózy postavy. Prevzaté z [5].

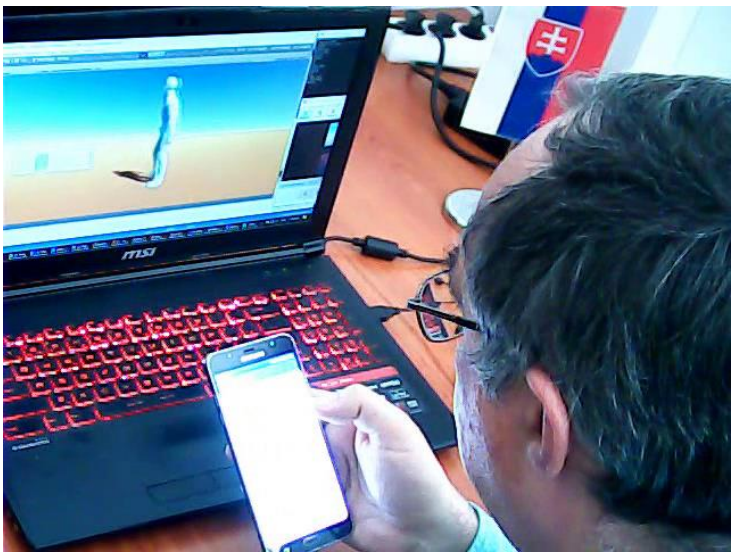
Zložitejšie pohyby tela je možné detegovať pomocou modelov OpenPose [5], ktoré sú dostupné aj ako plugin do Unity3D na https://github.com/CMU-Perceptual-Computing-Lab/openpose. Kvalita rozpoznania polohy celej postavy je však zatiaľ limitovaná (Obr. 13). Podobne to platí pre modely rozpoznávania gest ruky (OpenHand [16]). Usilovne sa pracuje na rozpoznávaní emócii tvárí [14] a ďalších aspektov človeka – a teda hráča.

Využitie kognitívnych cloudových služieb.

Funkcionalita viacerých modelov získaných hlbokým učením je dostupná vo forme cloudových služieb [11], t.j. tak, že vstup pre danú neurónovú sieť pošleme cez Internet do centra, ktoré je doslova na druhom konci sveta, tam sa mimoriadne výkonným hardvérom sieť na vstupe spustí a výsledok príde cez Internet naspäť. Často prv, než by ho mohol spočítať lokálny hardvér. Dostupnosť cloudových služieb z EÚ je cca 80ms, čo je prijateľná doba. Pre služby v rámci USA je to ešte menej, cca 40 ms. Väčšina cloudových centier sa fyzicky nachádza v Silicon Valley v Kalifornii. Najznámejšími cloudmi, ktoré poskytujú tieto služby (často sa im pridáva atribút „kognitívne“) sú IBM Watson, MicroSoft Azure a Google cloud.

V prípade využitia týchto služieb sme ušetrení všetkých problémov so zháňaním datasetu, trénovaním či zháňaním hotového modelu. Má to však aj nevýhody. V prvom rade musíme mať spojenie do Internetu, bez neho je služba nefunkčná. Čo je však horšie, avšak logické, je to, že každé jedno zavolanie služby je spoplatnené. A hoci poplatky nie sú nijako závratné, je to z hľadiska použitia veľmi limitujúci faktor.

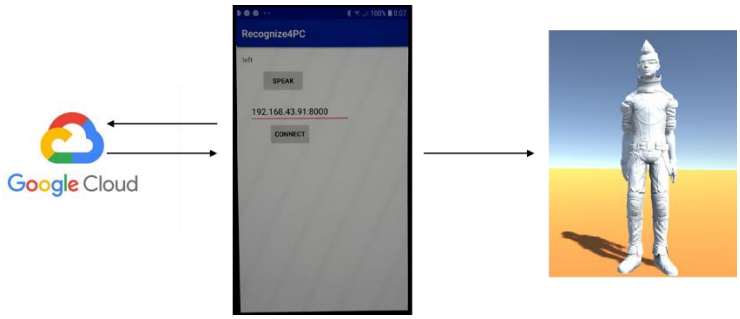
Existujú však aj vzácne výnimky, kedy môžeme cloud zavolať bez obmedzení, napríklad keď zavoláme Google cloud cez operačný systém smartfónov Android (od verzie 7). Môžeme takto ľahko zariadiť ovládanie hry hlasom, pričom používame smartfón ako mikrofón (Obr. 14). Potrebujeme na to len vhodnú aplikáciu pre Android, akou je naša Recognize4PC [15], voľne dostupná na adrese https://github.com/andylucny/Recognize4PC, inštaluje sa z .apk súboru.



Obr. 14. Ovládanie hry hlasom. Mobilný telefón tu slúži ako mikrofón. Prevod reči na text sa realizuje volaním cloudovej služby z aplikácie v smartfóne.

Táto aplikácia nahrá reč, odošle ju cez Internet do cloudu, kde sa premení na text a ten je odoslaný späť do aplikácie. Tá ho po lokálnej sieti pošle cez TCP do hry. V hre teda spracúvame textové pokyny

a musíme brať v úvahu len to, že jeden pokyn sa môže povedať viacerými rôznymi vetami. Kvalita rozpoznávania reči poskytovaná Google Cloudom je veľmi slušná a to dokonca aj v slovenčine, nemusíme dokonca ani zadať, aký jazyk používame. Do aplikácie v smartfóne však musíme zadať IP adresu počítača, na ktorom beží hra (Obr. 15).



Obr. 15. Volanie cloudovej služby cez Android vo smartfóne. Konkrétne ide o premenu reči na text, teda o ovládanie hry hlasom.

Samozrejme, pri spracovaní hlasového príkazu v hre je problém pokryť všetky možné varianty, akými sa dá daný povel povedať. Riešením tohto problému je využitie inej cloudovej služby a to služby pre rozpoznanie zámeru z vety v prirodzenom jazyku, t. j. základnej služby pre moderné chatboty [4]. Cloud túto službu dokáže vykonať vďaka už spomínanej vektorizácii slov a viet pomocou hlbokých neurónových sietí s pomerne komplikovanou architektúrou, ktorá obsahuje aj rekurentné prvky. Žiaľ pre slovenčinu takáto služba zatiaľ nie je dostupná v rozumnej kvalite.

Záver

Pokúsili sme predstaviť súčasné možnosti využitia moderných metód umelej inteligencie pre tvorbu hier, v ktorých hráč fyzicky interaguje s hrou – hlasom, pohybom tela a podobne – a pritom používajú len bežne dostupný hardvér. Sústredili sme sa na tie možnosti, ktoré sme sami vyskúšali. Keďže táto oblasť sa prudko vyvíja, už v tejto chvíli sú dostupné ďalšie podobné metódy a ďalšie možno očakávať v najbližšej budúcnosti. Sľubne sa zlepšuje kvalita napríklad detekcie emócii z tváre človeka alebo pózy tela a rúk z obyčajnej kamery. Čitateľovi sme sa snažili priblížiť túto problematiku a to nielen z hľadiska použitia, ale nechali sme ho nazrieť aj dovnútra vecí, tak aby mal aspoň určitú predstavu, vďaka čomu to celé funguje a aké varianty technického prevedenia sú možné. Dúfame, že tak podnietime čitateľa k tvorbe inovatívnych počítačových hier.

Použitá literatúra

[1] Badrinarayanan, V. - Kendall, A. - Cipolla, R. (2017) SegNet: A Deep Convolutional Encoder- Decoder Architecture for Image Segmentation. IEEE Trans. Pattern Anal. Mach. Intell, 39, pp. 2481–2495

[2] Ballard, D. (1987) Modular learning in neural networks. Proceedings AAAI.

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

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

[5] Cao, Z. - Hidalgo G. - Simon, T. - Wei, S. - Sheikh, Y. (2018) OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. preprint arXiv:1812.08008

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

[7] Dalal, N. - Triggs, B. (2005) Histograms of Oriented Gradients for Human Detection. CVPR '05 Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), Volume 1, pp. 886-893

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

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

[10] He, K. - Zhang, X. - Ren, S. - Sun, J. (2016) Deep Residual Learning for Image Recognition. Microsoft Research

[11] Hwang, K. (2017) Cloud Computing for Machine Learning and Cognitive Applications. The MIT Press, 2017

[12] Jackson, A. - Bulat, A. - Argyriou, V. - Tzimiropoulos, G. (2017) Large Pose 3D Face Reconstruction from a Single Image via Direct Volumetric CNN Regression. International Conference on Computer Vision

[13] Krizhevsky, A. - Sutskever, I. - Hinton, G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Advances in neural information processing systems 25(2)

[14] Loconsole, C. - Chiaradia, D. - Bevilacqua, V. - Frisoli, A. (2014) Real-Time Emotion Recognition: An Improved Hybrid Approach for Classification Performance. ICIC 2014: Intelligent Computing Theory, pp. 320-331

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

[16] Odhner, L. - Jentoft, L. - Claffee, M. - Corson, N. - Tenzer, Y. - Ma, R. - Buehler, M. - Kohout, R. - Howe, R. - Dollar, A. (2014) A Compliant, Underactuated Hand for Robust Manipulation. International Journal of Robotics Research, vol. 33(5), pp. 736-52, 2014

[17] Redmon, J. - Farhadi, A. (2018) YOLOv3: An Incremental Improvement. ArXiv 2018

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

[19] 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

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

[21] Sutskever, I. - Vinyals, O. - Le, Q. (2014) Sequence to Sequence Learning with Neural Networks. NIPS'14 Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, pp. 3104-3112

[22] Xu, H. (2018) Generate Faces Using Ladder Variational Autoencoder with Maximum Mean Discrepancy. Intelligent Information Management 10(04), pp. 108-113

[23] Zhang, R. - Isola, P. - Efros, A. (2016) Colorful Image Colorization. In ECCV, 2016