Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

tmp

Patrik Beka

Modelovanie ľudskej vizuálnej pozornosti metódami počítačového videnia a umelej inteligencie

Diplomová práca

Študijný program: Inteligentné softvérové systémy
Študijný odbor: Inteligentné softvérové systémy
Miesto vypracovania: Ústav počítačového inžinierstva a aplikovanej informatiky, FIIT STU, Bratislava
Supervisor: doc. Ing. Vanda Benešová, PhD.

Máj 2018

ČESTNÉ PREHLÁSENIE

Čestne vyhlasujem, že som bakalársku prácu vypracoval samostatne, na základe konzultácií a štúdia odbornej literatúry, ktorej zoznam som uviedol na príslušnom mieste.

.....

Patrik Beka

POĎAKOVANIE

Anotácia

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: Inteligentné softvérové systémy
Autor: Patrik Beka
Diplomová práca: Modelovanie ľudskej vizuálnej pozornosti metódami počítačového videnia a umelej inteligencie
Vedúci práce: doc. Ing. Vanda Benešová, PhD.
Máj 2018

Annotation

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Degree Course: Intelligent Software Systems
Author: Patrik Beka
Master thesis: The modeling of human visual attention using computer vision and artificial intelligence
Supervisor: doc. Ing. Vanda Benešová, PhD.
May 2018

Obsah

Zoznam obrázkov

1 Úvod

2 Vizuálna pozornosť

Termín vizuálna pozornosť možno definovať ako súbor všetkých faktorov, ktoré ovplyvňujú naše mechanizmy výberu podstatných častí v scéne a jej spracovanie, nezáležiac od toho, aké tieto mechanizmy sú (či už riadené stimulmi, očakávaniami, pamaťou, atď.) [3].

Tento pojem je často zamieňaný s vizuálnou pútavosťou(výraznosťou?), avšak tieto dva termíny nevyjadrujú úplne to isté. Presnejšou definíciou vizuálnej pútavosti (z angl. visual saliency) je, že sa jedná o značne subjektívnu perceptuálnu vlastnosť, vďaka ktorej niektoré veci vo svete (v scéne) vyčnievajú v porovnaní so svojimi susedmi kvôli ich vlastnostiam ako farba, jas, kontrast či orientácia [10]. Upútanie pozornosti ovplyvňujú mechanizmy spracovania scény, ktoré možno rozdeliť do na dve skupiny:

- spracovanie „zdola nahor“ (z angl. bottom-up)
- spracovanie „zhora nadol“ (z angl. top-down)

2.1 Bottom-up spracovanie

Vizuálne stimuly, ktoré upútajú pozornosť automaticky, mimovoľne, sa nazývajú bottom-up stimuly (alebo kontextovo riadené). Práve tieto riadia našu pozornosť a v podstate sú akousi známkou (ukazovateľom), že táto lokácia (alebo objekt v nej) je značne odlišná od svojho okolia a presne preto stojí za pozornosť. Ich príkladom môžu byť značky pri pozemných komunikáciách, bezpečnostné prvky vo vozidlách, ale aj správne umiestnené titulky v novinách, blogoch, či dizajnérmi nesprávne umiestnená reklama na webových stránkach zbytočne odtrhujúca našu pozornosť od podstatných vecí.

Hlavnou charakteristikou tohto spracovania je nevedomosť (obvykle bez predošlých informácií o pozorovanej scéne) a rýchlosť - priemerné spracovanie jedného objektu v scéne je na úrovni od 20 do 50 milisekúnd[11].

2.2 Top-down spracovanie

Tento typ spracovania vizuálnych signálov sa oproti vyššie uvedenému líši viacerými vecami. Tou prvou je, že sa riadi tzv. predvídateľnými mechanizmami a prináša so sebou bližšie nešpecifikovanú vedomosť o pozorovanej scéne - pozorovateľ má isté informácie ako predošlé skúsenosti, spomienky, alebo napríklad hľadá v scéne nejaký konkrétny objekt. Z tohto dôvodu sa nazýva aj spracovaním založeným na vedomostiach (z angl. knowledge-based processing[6]) alebo údajoch (a angl. data-driven processing[8])

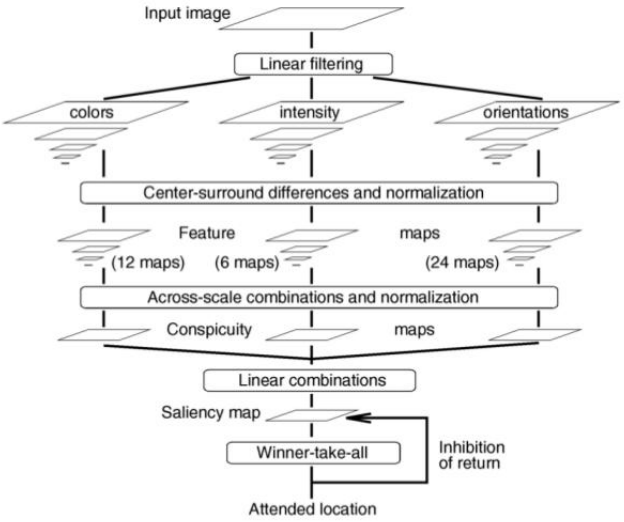
Druhým veľkým rozdielom oproti bottom-up spracovaniu je jeho rýchlosť, priemerný čas spracovania vizuálneho signálu sa pohybuje na úrovni 200 milisekúnd[11] a viac, čo je výrazne pomalšie.

2.3 Existujúce modely vizuálnej pozornosti

Existujúce modely vizuálnej pozornosti[16] možno rozdeliť nasledovne:

- hierarchické - využívajú hierarchické rozkladanie príznakov
- Bayesove - využívajú kombináciu výraznosti s predchádzajúcimi znalosťami
- rozhodovaco-teoretické - využívajú diskriminačnú teóriu výraznosti
- informaticko-teoretické - využívajú maximalizáciu informácie z daného prostredia
- grafické - predikcia výraznosti je založená na grafových algoritmoch
- vzorovo klasifikačné - využívajú strojové učenie zo vzorov s výraznými črtami

Jedným z najznámejších modelov vizuálnej pozornosti je Itti-ho hierarchický model[5]. Je to biologicky inšpirovaný bottom-up model, ktorý využíva hierarchické rozloženie vlastností a ich kombináciu do výslednej mapy výraznosti (z angl. saliency map). Ako je vidieť na obrázku nižšie, z obrázka sa vytvoria 3 typy máp a to podľa farby, intenzity a orientácie, ktorých kombináciou sa dosiahne už spomenutá mapa výraznosti.



Obr. 1: *Itti-ho hierarchický model vizuálnej pozornosti[5]*

3 Neurónové siete

Neurónová sieť je abstraktný výpočtovový model založený na princípe reálnych biologických neurosystémov. Základnou stavebnou jednotkou je tak rovnako ako u neurónových sietí živočíchov neurón, resp. model neurónu[1]. Ten spracováva rôzne množstvo vstupov (N) a výstupov (M). V minulosti sa zvykol vyjadrovať podľa nasledovnej matematickej špecifikácie:

$$o_i^{k+1} = f\left(\sum_{j+1}^{N} w_{ij}^k * o_j^k - \theta_i^{k+1}\right) \tag{1}$$

Pre vyššie uvedené platí:

0 <i $\leq$ M

0 <j $\leq$ N

o_i^{k+1} - výstupná hodnota i-teho neurónu patriaceho k+1 vrstve

k - číslo vrstvy

θ_{ij}^k - prah stimulácie i-teho neurónu k+1 vrstvy

w_{ij}^k - váha medzi i-tym neurónom vrstvy k+1 a j-tym neurónom vrstvy k
f() - funkcia

V súčasnosti sa však používa radšej matematické vyjadrenie zobrazené v rovnici 2. Vypustil sa z neho prah stimulácie neurónu, miesto ktorého sa používa tzv. predsudok (z angl. bias), čo je niečo ako predpokladaná hodnota (náš chýbajúci prah stimulácie) neurónu. Tá sa časom samozrejme mení.

Predpokladajme, že máme *m+1* vstupov so signálmi od x_0 po x_m a váhami od w_0 po w_m. Obvykle sa vstupu x_0 pridelí hodnota +1, čím sa stane predsudkom vstupu s $w_{k0} = b_k$. To necháva potom iba *m* vstupov do neurónu, od x_1 do x_m. Samotný výstup z k-teho neurónu je potom matematicky vyjadrený nasledujúcou rovnicou:

$$y_k = \phi(\sum_{j=0}^{m} w_{kj} * x_j) \tag{2}$$

Pre vyššie uvedené platí:
y_k - výstup k-teho neurónu
w_{kj} - váha j-teho neurónu spojeného s k-tym neurónom na ďalšej vrstve
x_j - j-ty neurón
ϕ - funkcia

Neurónová sieť sa sa môže skladať z viacerých vrstiev, na ktorých sú umiestnené neuróny. Prvá vrstva sa nazýva vstupná, posledná výstupná. Medzi nimi môže byť ľubovoľný počet skrytých vrstiev rôzneho typu. Každá vrstva (s výnimkou výstupnej) by mala ešte navyše obsahovať aktivačnú funkciu (kapitola 3.1). V našom prípade sa jedná o neurón umelý a funkciu, ktorá definuje výstup neurónu pre vstup alebo sadu vstupov.

3.1 Aktivačné funkcie

Aktivačná funkcia predstavuje matematické vyjadrenie použité k aproximácii vplyvu na neurón, zjednušene by bolo možné povedať, že pre sériu vstupov definuje výstupy. Aktivačných funkcií existuje niekoľko typov, každá vhodná na iný typ

úloh. Ako príklad je možné uviesť nasledujúce:

- **Softmax**

  Funkcia softmax[1] (inak aj normalizovaná exponenciálna funkcia) normalizuje daný *n* dimenzionálny vektor tak, že upraví jeho hodnoty do rozsahu (0,1), pričom ich súčet bude rovný 1. Jej matimatické vyjadrenie je nižšie.

  $$S_{vec_j} = \frac{e^{vec_j}}{\sum_{i=1}^{n} e^{vec_i}} \tag{3}$$

  Pre vyššie uvedené vyjadrenie platí:

  $\forall\, j \in 1..n$

  vec - konkrétny vector

  Keď si ako príklad vezmeme jednoduchý vektor [1, 2, 3], výsledok po aplikovaní softmaxu bude [0.09, 0.24, 0.67]. Ako môžeme vidieť, funkcia sa väčšinou používa na zvýraznenie väčších hodnôt a zárove potlačenie hodnôt, ktoré sú výrazne menšie ako maximálna hodnota.

- **ReLU**

  Upravená lineárna jednotka (z angl. rectified linear unit) je funkcia v tvare:

  $$f(x) = max(0, x) \tag{4}$$

  kde *x* je vstup do neurónu. Používa sa vďaka svojej jednoduchosti, keďže neobsahuje žiadne komplikované výpočty, čoho dôsledkom je aj jej značná rýchlosť. Jej využitie je možné pozorovať napríklad pri hlbokých neurónových sieťach.

- **Softplus**

  Je v podstate aproximáciou k predošlej ReLU s matematickým vyjadrením:

  $$f(x) = ln(1 + e^x) \tag{5}$$

[1] http://eli.thegreenplace.net/2016/the-softmax-function-and-its-derivative/

Rovnako ako pri ReLU je oborom hodnôt interval (0, ∞). Jej využitie je napríklad pri rozoznávaní reči.

- **Sigmoid**

  Táto funkcia sa používa hlavne keď je potrebné pracovať s pravdepodobnosťami, keďže jej výstup tvorí interval (0, 1). Jej matematické vyjadrenie je nasledovné:

$$S(t) = \frac{t}{1 - e^{-t}} \tag{6}$$

- **Tanh**

  Hyberbolický tangens. Často sa používa v rovnakých prípadoch ako Sigmoid, keďže matematicky sa dá vyjadriť aj za použitia Sigmoidu. Jeho vzorec je nasledovný:

$$tanh(x) = \frac{cosh(x)}{sinh(x)} = Sigmoid(2x) - Sigmoid(-2x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{7}$$

3.2 Učenie sa neurónovej siete

Základným prvkom toho, aby bola neurónová sieť schopná riešiť úlohy je učenie sa. Existujú viaceré typy učenia sa neurónovej siete, za zmienku stojí napríklad učenie sa s učiteľom (z angl. supervised learning) a učenie sa bez učiteľa (z angl. unsupervised learning[9]). Hlavným rozdielom medzi nimi je, že učenie s učiteľom musí prebiehať na predpripravenom datasete, ktorý musí obsahovať nejaké testovacie vstupné dáta (pre ktoré chceme vypočítať výstupnú funkciu) a takzvané štítky (z angl. labels), ktoré sú v podstate naše očakaváne výstupy. Učenie sa bez učeiteľa naproti tomu odvodzuje funkciu k popisu skrytej štruktúry z neoštítkovaných dát, teda bez štítkov, ktoré nám určujú očakaváne výstupy. Nie je tu teda žiadna chyba ani signál k ohodnoteniu potenciálneho riešenia.

Príkladom učenia s učiteľom môže byť napríklad jednoduchá neurónová sieť, ktorá má riešiť funkciu XOR[1], kedy potrebujeme reprezentovať vstupné dáta ako dvojicu núl a jednotiek. Štítkami sú v tomto prípade očakávané výstupy, takže napríklad pre vstup (dvojicu) [0,1] je štítkom 1. Takto pripravený dataset pre učenie sa by mal byť veľmi rozsiahly aby sa dosiahla maximálna presnosť.

Ďalej je potrebné použiť niektorý z algoritmov učenia. Široko používaným je na takýto typ úloh algoritmus učenia spätného šírenia (z angl. backpropagation). Tento algoritmus sa snaží minimalizovať chybu pri učení a to tak, že najprv sa vypočíta chyba na poslednej (výstupnej) vrstve. Tá sa potom šíri späť k vstupnej vrstve a aktualizujú sa váhy jednotlivých neurónov. V kombinácii s algoritmami učenia sa používajú optimizačné algoritmy, ktorých cieľom je nájdenie minima funkcie medzi váhami. Medzi základné optimizéry patria:

- **Gradient descent optimizer [18]:**

  Je to iteratívny algoritmus používaný k nájdeniu lokálneho minima funkcie, kedy podniká kroky k nájdeniu záporného gradientu[2] funkcie v aktuálnom bode. To je využívané pri určovaní rýchlosti učenia sa neurónovej siete.

  Existujú 3 hlavné varianty gradient descent optimizéru, ktoré počítajú sklon (gradient) funkcie. Delia sa hlavne podľa množstva dát určenému k spracovaniu, kedy sa robí kompromis medzi presnosťou aktualizácie parametra a časom, ktorý je potrebný na vykonanie tejto aktualizácie. Týmito typmi sú:

  - Dávkový gradient descent:

    Z angl. Batch gradient descent. Gradient sa počíta pre celý tréningový dataset, takže pre jednu aktualizáciu je potrebné ho prejsť celý a preto môže byť veľmi pomalý.

  - Stochaistický gradient descent:

    Tento typ je presným kontrastom voči dávkovému gradient descentu. Aktualizácia sa uskutočňuje pre každú vzorku z tréningového datasetu.

  - Mini-dávkový gradient descent:

    Je kompromisom medzi predošlými dvomi typmi. Aktualizácia prebieha pre malú dávku (batch) z datasetu o veľkosti n vzoriek.

- **Adam optimizer [13]:**

  V podstate vychádza priamo zo Stochaistického gradient descent optimizéru, resp. jeho modifikácie RMSProp algoritmu[21]. Rozdiel oproti Gradient

[2] zmena veličiny v závislosti od inej premennej

descent optimizéru je ale v tom, že je schopný variabilne určovať rýchlosť učenia neurónovej siete.

- **Ftrl optimizer:**

  Vychádza z algoritmu učenia FTRL-Proximal[15], celým názvom Nasleduj regularizovaného vodcu (z angl. Follow The (Proximal) Regularized Leader). Tento algoritmus je bez regularizácie v podstate identický s gradient descentom, avšak používa alternatívnu reprezentáciu koeficientov váh a tak môže byť regularizácia implementovaná efektívnejšie.

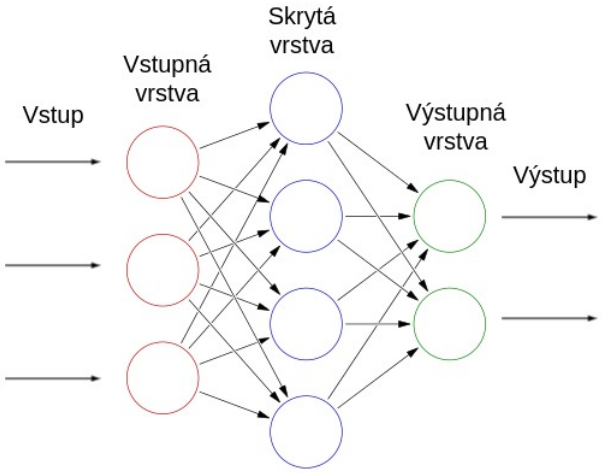
Po fáze učenia sa nasleduje validácia, pri ktorej sieť nemá prístup k štítkom. Na záver sa prejde k samotnému testovaniu neurónovej siete, kedy sa do nej posúvajú dáta rovnako bez toho, aby sieť mala prístup k štítkom. Na základe jej predikcie a štítkom k testovacím dátam sa určí jej presnosť. Na učenie, testovanie a validáciu by nemali byť použité tie isté dáta. Pomer dát k jednotlivým fázam by mal byť 80-10-10[17], čiže 80% dát je určených na učenie sa, 10% na validáciu a 10% na samotné otestovanie predikcií modelu siete.

Aj keď neurónové siete dokážu efektívne riešiť veľké množstvo úloh, problémom stále zostáva mať k dispozícii dostatok dát k učeniu neurónovej siete ešte pred riešením úloh. Taktiež je potrebné mať dostatok výpočtovej sily, aby sa problém neriešil pridlhý čas, a dostatok pamäte, keďže neurónové siete jej potrebujú značné množstvo.

3.3 Typy neurónových sietí

Neurónové siete majú niekoľko typov, ktoré sa rozlišujú hlavne podľa spôsobu prepojenia neurónov, ale aj podľa typu úloh, na ktoré sú určené, či podľa počtu vrstiev neurónov alebo štýlu učenia.

Najjednoduchší typ možno zobraziť ako jednu vstupnú vrstvu, jednu skrytú a jednu výstupnú, neuróny sú tu poprepájané z n-tej vrstvy do n+1 vrstvy, ako je možné vidieť na obrázku 2. Tento typ sa nazýva dopredná neurónová sieť (z angl. feedforward neural network) a môže mať aj viac ako len jednu skrytú vrstvu. Používa sa hlavne ak sa jedná o predikciu nelineárnej funkcie (napríklad carbon-13 NMR chemické posuny alkánov[20]).

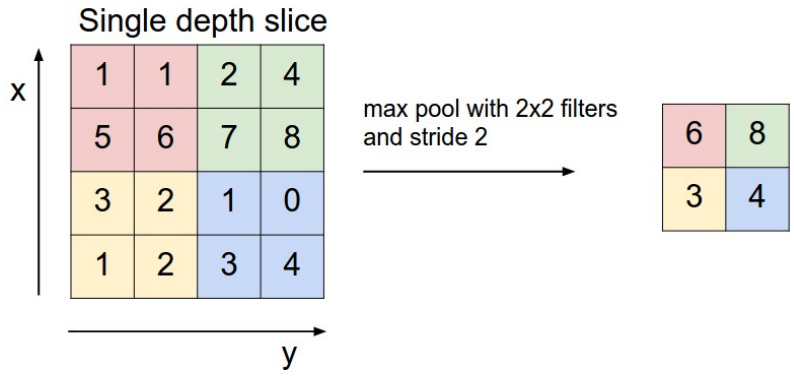


Obr. 2: *Príklad jednoduchej neurónovej siete*

Zložitejším typom sú rekurentné neurónové siete. Už z názvu vyplýva, že jednou z vecí, ktoré umožňujú, je rekurenciu. Vďaka nej prepojenia neurónov už nie sú jednosmerné len z jednej vrstvy na druhú, ale umožňuje prepojiť neuróny akokoľvek a tak vytvárať napríklad slučky či cykly. To dovoľuje zachytiť aj dynamické časovo obmedzené správanie a používať kontext z minulosti (avšak len niekoľko krokov dozadu), teda použiť niečo ako krátkodobú „pamäť“. Na rozdiel od doprednej neurónovej siete je možné spracovať aj ľubovoľnú sekvenciu vstupov. V praxi to znamená, že keď chceme napríklad predikovať ďalšie slovo vo vete, je dobré vedieť, ktoré slová boli pred ním. Tento typ sietí sa používa napríklad pri rozpoznávaní reči[19] alebo písma[7], či generovaní popisu k obrázkom[12], kedy však funguje v kombinácii s konvolučnou neurónovou sieťou (z angl. convolutional neural network). Tá je použitá na klasifikáciu obrázkov a rozoznávanie objektov, rekurentná sieť je použitá iba na výsledné generovanie jednoduchého popisu. Konvolučná neurónová sieť je ďalším typom neurónovej siete, ktorá sa používa pri práci s obrázkami (rozpoznávanie objektov, atď.). Podrobnejšie je tento typ popísaný nižšie, nakoľko je to typ, s ktorým budeme pracovať aj neskôr.

3.3.1 Konvolučné neurónové siete

Základ tejto siete tvorí vstupná konvolučná vrstva[3] s konvolučným filtrom, ten býva väčšinou malý (3x3, 5x5). Vstup tejto vrstvy musí byť v tvare *m* x *m* x *r*, kde *m* je šírka a výška obrázku, *r* je počet farebných kanálov. Napríklad pre RGB obrázok je r=3 (červená, zelená, modrá). Konvolučným filtrom sa prejde celý obrázok a výstupom z tejto vrstvy je niekoľko filtrov. Tie sa potom spracúvajú v ďalšie vrstve združovania (z angl. pooling layer[14]), ktorá tieto filtre rozvzorkuje. To prebieha nezávisle na každom získanom filtry z konvolučnej vrstvy. Rozvzorkovanie v podstate znamená, že sa zmení veľkosť filtrov použítim operácie MAX. Najbežnejšou formou spomínanej vrstvy je verzia s oknom (filtrom) o veľkosti 2x2 aplikovaným s krokom veľkosti 2. Toto sa dá jednoducho vysvetliť ako prejdenie každého výstupu z konvolučnej vrstvy oknom uvedenej veľkosti postupne po 2 políčkach na šírku aj výšku, pričom z každej štvorice v okne sa získa MAX operáciou maximum, s ktorým sa pracuje ďalej. Na obrázku[4] 3 je vidieť výsledok popisovaného postupu na jednoduchom príklade.

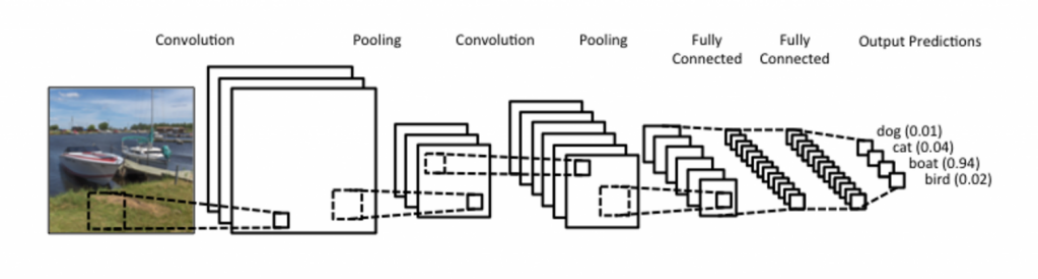


Obr. 3: *Príklad vrstvy združovania, pri ktorom sa rozvzorkuje výstup z konvolučnej vrstvy o veľkosti 4x4, filtrom 2x2, s krokom veľkosti 2 za použitia operácie MAX*

Takýchto konvolučných vrstiev s vrstvami združovania môže byť aj viac, nemusia ani nutne nasledovať po sebe. Po týchto vrstvách nasleduje plne prepojená vrstva alebo vrstvy (z angl. fully-connected layers), čo je vrstva, v ktorej majú

[3] http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/
[4] http://cs231n.github.io/convolutional-networks/

neuróny plné spojenie so všetkými aktiváciami v predošlej vrstve, rovnako ako pri bežných neurónových sieťach. Aktivačnou funkciou neurónov na tejto vrstve býva väčšinou ReLU. Po plne prepojenej vrstve (vrstvách) už nasleduje iba výstupná vrstva. Na obrázku[5] nižšie je jednoduchý náčrt vyššie popísané konvolučnej neurónovej siete.



Obr. 4: *Príklad konvolučnej neurónovej siete*

Využite tohto typu je v podstate všade, kde sa jedná o rozpoznávanie obrázkov. Či už ide o automatické vyznačenie tvárí pre označenie na facebooku, autonómne vozidlá, ktoré sa vedia riadiť sami (autopilot) alebo triedenie uhoriek na farmách v Japonsku[6]. Na tento konkrétny softvér bol použitý príklad kódu jednoduchej konvolučnej siete z tutoriálu[7] pre TensorFlow (knižnica pre prácu s neurónovými sieťami), s modifikáciou konvolučnej a združovacej vrstvy tak, aby bola sieť uspôsobená počtu tried uhoriek (10) a ich formátu obrázkov.

Z mnohých pokusov o autonómnu jazdu stoja za zmienku hlavne tie od Tesly a Google. Prototyp autonómneho systému vozidla od Google-u Dave-2[2] využíva model neurónovej siete s 9 vrstvami, jednu normalizačnú, 5 konvolučných a 3 plne prepojené vrstvy. Kamerami spracovaný obraz okolia s frekvenciou 10 snímkov za sekundu (tak nízky počet preto, aby sa predišlo veľkému množstvu príliš podobných obrázkov) je po jednom snímku rozdelený do YUV[8] úrovní a posunutý do neurónovej siete.

[5]https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

[6]https://cloud.google.com/blog/big-data/2016/08/how-a-japanese-cucumber-farmer-is-using-deep-learning-and-tensorflow

[7]https://www.tensorflow.org/versions/0.6.0/tutorials/mnist/pros/index.html

[8]farebný priestor používaný vo video aplikáciách

3.4 Framework-y pre prácu s neurónovými sieťami

V dnešnej dobe moderného internetu a dostupnosti technológií máme možnosť výberu z dostatočného množstva framework-ov pre potreby riešenia najrôznejších problémov neurónovými sieťami. Pri výbere môžeme brať do úvahy napríklad preferencie operačného systému (Windows, Linux, ...), programovacieho jazyka (Python, C++, Java, ...), ale aj benefity distribuovaného riešenia a omnoho viac. V nasledujúcich podkapitolách sú preto popísané niektoré z najpoužívanejších technológií pre implementovanie neurónových sietí.

3.4.1 TensorFlow

TensorFlow[9] je open-source softvérová knižnica, ktorá pre numerické výpočty používa graf dátového toku, kde uzly grafu reprezentujú matematické operácie a hrany multidimenzionálne dátové polia, tzv. tenzory. Graf je možné skonštruovať použitím jazykov s podporou frontend-u (C++, Python, ...).

Flexibilná architektúra umožňuje vykonávať výpočty na CPU alebo GPU (nepomerne rýchlejšie) na serveroch, desktopových počítačoch či dokonca aj mobilných zariadeniach. Pôvodne bol TensorFlow vyvinutý výzkumníkmi a inžiniermi v Google-i pre strojové učenie a hlboké učenie, avšak jeho využitie je oveľa širšie. Momentálne používa TensorFlow veľké množstvo programov, napríklad Google vyhľadávač, prekladač alebo YouTube.

Momentálne podporuje jazyky C++, Python, Java, Go, Swift.

Medzi hlavné výhody patrí:

- podpora pre jednoducho naučiteľné jazyky (Python)
- použtie výpočtovej grafovej abstrakcie
- vizualizácie pomocou TensorBoard-u[10] (interaktívne grafy pre priebeh učenia, pre model siete, ...)
- dostatočne nízko-úrovňový pre plnú kontrolu a implementáciu vlastnej (novej nie len preddefinovanej) funkcionality (v porovnaní napríklad s framework-om Keras(kapitola 3.4.4))

[9] https://www.tensorflow.org/

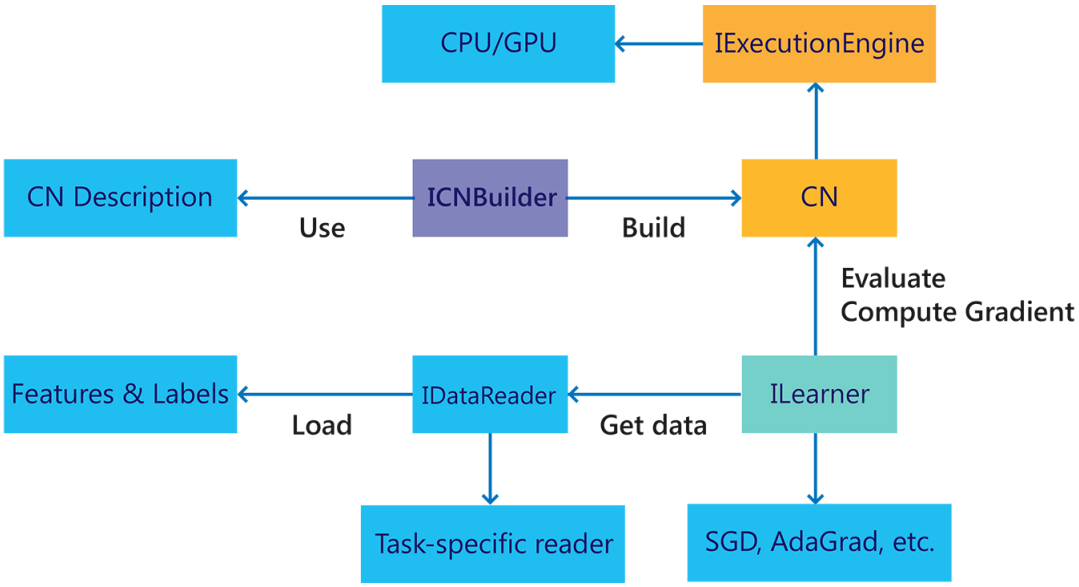
[10] https://www.tensorflow.org/programmers_guide/summaries_and_tensorboard

Ako nevýhody možno uviesť:

- nedostatok predtrénovaných modelov
- pri použití s určitými jazykmi (Python, Java, ...) je pomalý, nakoľko sa nejedná najrýchlejšie jazyky

3.4.2 Microsoft CNTK

Označuje knižnicu Microsoft Cognitive Toolkit[11], ktorá zlepšuje modularizáciu a údržbu separácie výpočtových sietí, zároveň postkytuje algoritmy učenia a popisy modelov. Má sa jednať o odpoveď na TensorFlow, poskytovaná funkcionalita je veľmi podobná, avšak je o niečo rýchlejší. Princíp a celá architektúra sú zachytené na diagrame na obrázku 5.



Obr. 5: *Diagram zobrazujúci architektúru fungovania Microsoft CNTK*[12]

Momentálne podporuje jazyky C++, C#, Python, Java.

Výhodami tohto framework-u sú:

[11] https://docs.microsoft.com/en-us/cognitive-toolkit/

[12] https://dzone.com/articles/progressive-tools10-best-frameworks-and-libraries

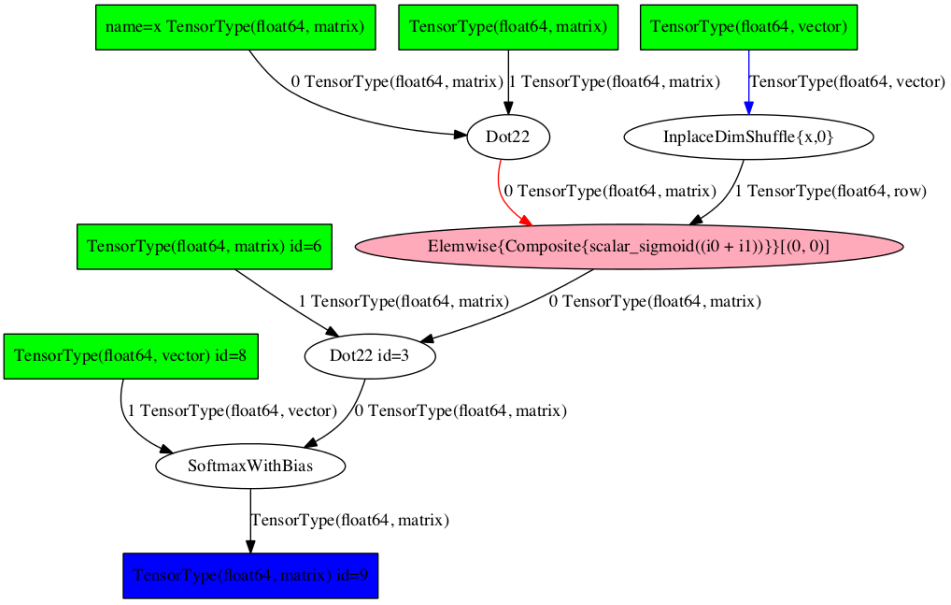
- flexibilita
- umožňuje distribuovaný tréning

Medzi nevýhody možno zaradiť:

- implementácia v novom jazyku, Network Description Language (NDL)
- nedostatok možností a nástrojov pre vizualizácie

3.4.3 Theano

Theano[13] je veľmi silná knižnica umožňujúca definovanie, optimalizáciu a evaluáciu numerických operácií nad multidimenzionálnymi poliami s obrovskou efektivitou. Podobne ako TensorFlow, k abstrakcii výpočtov používa grafy ako možno vidieť na príklade na obrázku 6.



Obr. 6: *Príklad grafovej abstrakcie výpočtov použitím framework-u Theano*[14]

[13] https://github.com/Theano/Theano

[14] http://www.wildml.com/2015/09/speeding-up-your-neural-network-with-theano-and-the-gpu/

Momentálne podporuje iba programovací jazyk Python a v poslednej dobe vývoj tohto framework-u dosť upadá.

Medzi hlavné výhody patrí:

- veľmi dobrá optimalizácia pre CPU a GPU (najmä vďaka použitiu nízkoúrovňovej funkcionality naprogramovanie v jazyku C)
- vysoko efektívna knižnica pre numerické úlohy

Za najväčšie nevýhody sú pokladané:

- Theano samo o sebe je v porovnaní s ostatnými knižnicami príliš nízkoúrovňové
- potreba použitia s inými knižnicami s vyšším stupňom abstrakcie (napríklad Keras)

3.4.4 Keras

Ďalší open-source framework pre prácu s neurónovými sieťami, avšak na rozdiel od predošlých troch nie je vypracovaný ako koncové riešenie pre strojové učenie. Namiesto toho slúži ako rozhranie a poskytuje vyššiu úroveň abstrakcie pre jednoduchšie používanie ostatných framework-ov, z ktorých momentálne pre použitie ako backend podporuje TensorFlow a Theano. Myšlienka stojaca za celým projektom je: *"Byť schopný pretaviť myšlienku na výsledok s čo najmenším zdržaním je kľúčom k dobrému výskum"*[15], čo bude aj jedným z dôvodov prečo je práca s ním jednoduchšia.

Keras je v súčasnosti možné používať iba v programovacom jazyku Python.

Jeho hlavnými výhodami sú:

- jednoduchosť naučenia, používateľsky veľmi prívetivé
- ľahká rozšíriteľnosť
- bezproblémový beh aj na CPU aj na GPU
- bezproblémové fungovanie aj s Theano-m a s TensorFlow-om

[15] https://keras.io/

Za jedinú nevýhodu môže byť považovaná nemožnosť použitia ako nezávislý framework - vždy je potrebný nejaký ďalší backend.

3.4.5 Spark MLlib

Škálovateľná knižnica pre strojové učenie, široko využívaná najmä v distribuovaných systémoch hlavne kvôli svojej efektivite. Veľmi jednoducho je ju môžné pripojiť do Hadoop workflow-u, poskytuje množstvo algoritmov pre strojové učenie optimalizovaných pre výpočty v už spomínaných distribuovaných systémoch na dátach vo veľkom meradle[16].

Momentálne poskytuje podporu pre jazyky Python, Java, Scala a R.

Najväčšími výhodami sú:

- vysoká rýchlosť na dátach vo veľkom meradle
- dostupnosť v jazykoch, ktoré podobnými framework-ami nie sú často podporované

Za nevýhody možno pokladať:

- strmá krivka učenia
- dostupnosť v jazykoch, ktoré podobnými framework-ami nie sú často podporované

Nevýhodou v porovnaní s vyššie uvedenými framework-ami môže byť nižšie množstvo implementovaných algoritmov, avšak vývoj rozhodne nezaháľa a pridávanej funcionality je stále viac a viac.

[16] https://spark.apache.org/mllib/

4 Existujúce riešenia

5 Metriky používané na ohodnotenie modelov vizuálnej pozornosti

Tradične sa tieto modely evaluujú vzhľadom na pohyb očí, resp. samotné fixácie. K tomu slúži značný počet rôzne fungujúcich metrík[4], najpoužívanejšie sú:

- NSS - Normalizovaná cesta pútavosti (z angl. Normalized Scanpath Saliency). Využíva priemer hodnôt pútavosti na *n* fixácií v normalizovanej mape podľa nasledovného vzorca:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{s(x_h^i, y_h^i) - \mu_s}{\sigma_s} \tag{8}$$

- AUC - Oblasť pod ROC krivkou (z angl. Area Under the ROC Curve). Ľudské fixácie sú považované za pozitívnu sadu a niektoré body na obrázku sú vybrané ako negatívna sada. K mape pútavosti je potom pristupované ako k binárnemu klasifikátoru na separáciu pozitívnych vzorkov od negatívnych. Presnosť podľa tejto metriky je daná nasledovne:
  - 0.90 - 1 = výborná
  - 0.80 - 0.90 = dobrá
  - 0.70 - 0.80 = priemerná
  - 0.60 - 0.70 = slabá
  - 0.50 - 0.60 = veľmi slabá

- sAUC - Zamiešaná oblasť pod ROC krivkou (z angl. shuffled Area Under the ROC Curve) je mierna modifikácia vyššie uvedenej metriky, kedy ako negatívna sada nie sú vybrané len niektoré body, ale všetky body, ktoré nie sú ľudskými fixáciami, sú považované za negatívne. Určenie presnosti na základe hodnôt je rovnaké ako pri AUC.

- CC - Korelačný koeficient, určuje prakticky podobnosť v tomto prípade dvoch máp výraznosti, kde jedna je výsledok modelu vizuálnej pozornosti a druhá je reálna mapa vypočítaná z fixácií.

$$CC(s,h) = \frac{cov(s,h)}{\sigma_s \sigma_h} \tag{9}$$

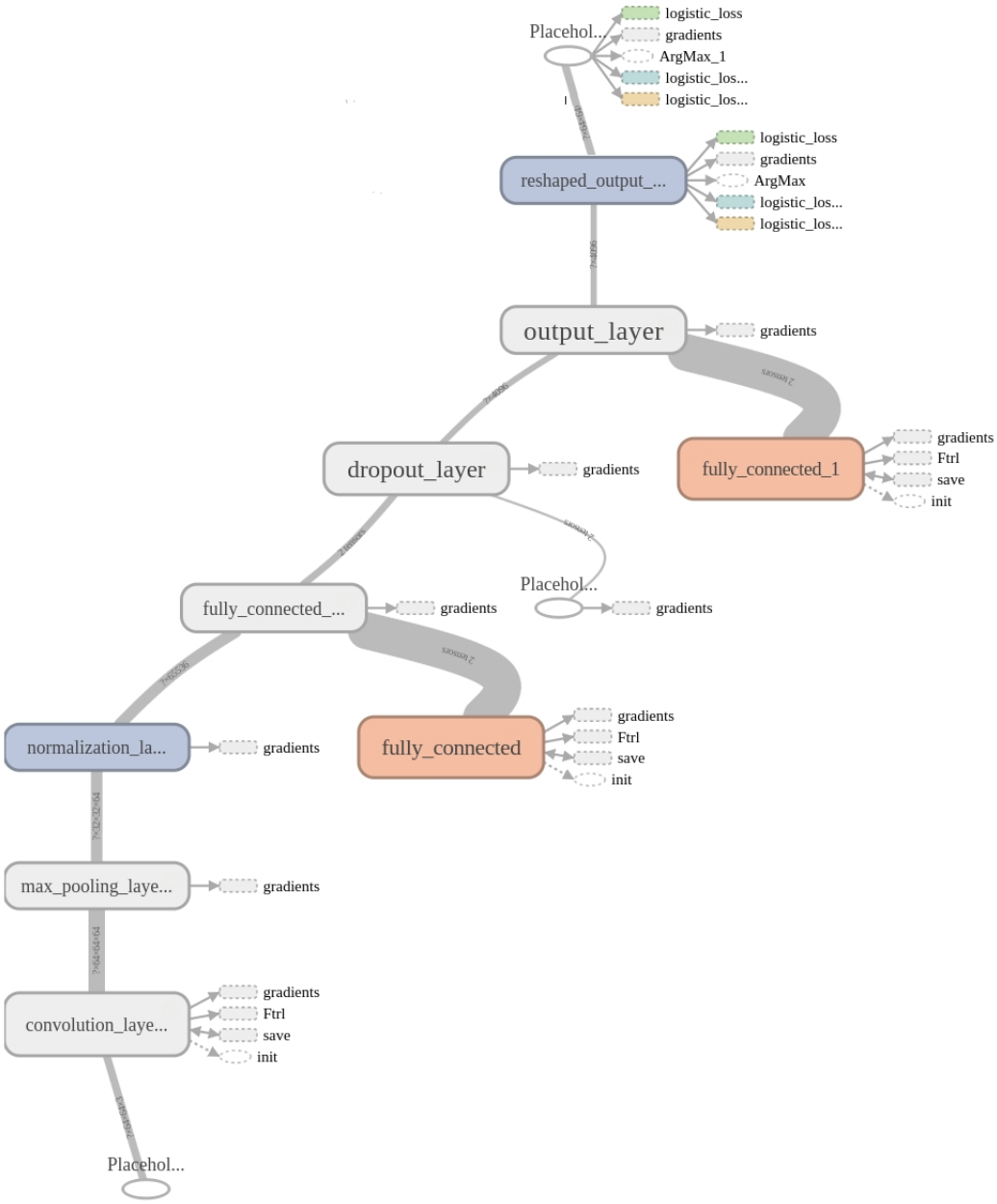
6 Návrh

6.1 Prvotné experimenty

6.2 Návrh neurónovej siete

Celá architektúra je načrtnutá na schéme na obrázku 7 vytvorenej pomocou nástroja TensorBoard[17].

[17] https://www.tensorflow.org/get_started/summaries_and_tensorboard/



Obr. 7: *Grafická schéma neurónovej siete, zdola vstup, vrstvy neurónovej siete, výstup (predikovaná mapa výraznosti)*

6.3 Dataset

7 Zhrnutie

Literatúra

[1] Kvasnička V. a kol. *Úvod do teórie neurónových sietí*. Iris, 1997.

[2] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[3] Ali Borji and Laurent Itti. State-of-the-art in visual attention modeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):185–207, 2013.

[4] Ali Borji, Hamed R Tavakoli, Dicky N Sihite, and Laurent Itti. Analysis of scores, datasets, and models in visual saliency prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 921–928, 2013.

[5] Renwu Gao, Faisal Shafait, Seiichi Uchida, and Yaokai Feng. A hierarchical visual saliency model for character detection in natural scenes. In *International Workshop on Camera-Based Document Analysis and Recognition*, pages 18–29. Springer, 2013.

[6] E Bruce Goldstein. *The Blackwell handbook of sensation and perception*. John Wiley & Sons, 2008.

[7] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868, 2009.

[8] Richard Langton Gregory. *Concepts and mechanisms of perception.* Charles Scribner's Sons, 1974.

[9] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer, 2009.

[10] Laurent Itti. Visual salience. *Scholarpedia*, 2(9):3327, 2007.

[11] Laurent Itti and Christof Koch. Computational modelling of visual attention. *Nature reviews neuroscience*, 2(3):194, 2001.

[12] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

[13] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Fei-Fei Li, Andrej Karpathy, and J Johnson. Cs231n: Convolutional neural networks for visual recognition, 2015.

[15] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230. ACM, 2013.

[16] Patrik Polatsek. Saliency maps. Prezentácia, 2015.

[17] David MW Powers. Roc-concert: Roc-based measurement of consistency and certainty. In *Engineering and Technology (S-CET), 2012 Spring Congress on*, pages 1–4. IEEE, 2012.

[18] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[19] Hasim Sak, Andrew W Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH*, pages 338–342, 2014.

[20] Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. Introduction to multilayer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39:43–62, 1997.

[21] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2), 2012.