Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

tmp

Patrik Beka

Modelovanie l'udskej 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 výraznosťou, avšak tieto dva termíny nevyjadrujú úplne to isté. Presnejšou definíciou vizuálnej výraznosti (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 [13]. 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[14].

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[9]) alebo údajoch (a angl. data-driven processing[11])

Druhým veľkým rozdielom oproti bottom-up spracovaniu je jeho rýchlosť, priemerný čas spracovania vizuálneho signálu sa pohybuje na úrovni 200 milisekúnd[14] a viac, čo je výrazne pomalšie.

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:

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

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

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

- **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, základnými sú učenie sa s učiteľom (z angl. supervised learning), učenie sa bez učiteľa (z angl. unsupervised learning[12]) a učenie sa posilňovaním[21].

3.2.1 Učenie sa s učiteľom

Učenie s učiteľom prebieha 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. Najčastejším príkladom tohto typu učenia je algoritmus spätného šírenia chyby (z angl. backpropagation[22]).

Algoritmus spätného šírenia chyby

Hlavným princípom je snaha o minimalizovanie chyby predikcie pri učení a to tak, že po predikcii pri učení sa vypočíta hodnota chyby na poslednej výstupnej vrstve voči očakávanému výstupu (spomínaným štítkom). Tá sa potom tzv.

"šíri"späť k vstupnej vrstve a vzhľadom na jej hodnotu sa postupne aktualizujú váhy jednotlivých neurónov.

Príklad použitia tohto algoritmu možno nájsť v jednoduchej neurónovej sieti určenej k riešeniu problému funkcie XOR[1]. Vstupné dáta je nutné reprezentovať ako dvojicu jednotiek a núl. Pre vstupnú dvojicu napríklad *[0,1]* je očakávaným výstupom číslo *1*, pre hodnotu *[0,0]* je to *0*. Tieto očakávané hodnoty zároveň označíme za spomínané štítky. Takto pripravíme celý dataset pre trénovanie, mal by byť dostatočne rozsiahly aby sa dosiahla maximálna presnosť. Problém veľkosti dostatočne rozsiahleho datasetu je považovaný za jeden z hlavných mínusov učenia s učiteľom.

Učenie sa viacerých inštancií

MIL[20] - skratka pre anglický výraz Multiple Instance Learning, čiže učenie sa viacerých inštancií, je variácia učenia s učiteľom. Miesto obdržania individuálne oštítkovaných inštancií (reprezentujú triedy, ktoré chceme predikovať), obdrží model set oštítkovaných vreciek (z angl. bags), z ktorého každé obsahuje niekoľko inštancií. V prípade jednoduchej binárnej klasifikácie je celé vrecko označené ako negatívna vzorka, ak všetky inštancie vo vnútri sú negatívne. Pokiaľ ale obsahuje aspoň jednu pozitívnu, je označené ako pozitívna vzorka. Pri zložitejších triedach si ako príklad môžeme uviesť predikciu nejakej cieľovej triedy pre obrázok na základe vizuálneho obsahu. Cieľová trieda bude *"pláž"* pre obrázok, ktorý obsahuje *"piesok"* a *"vodu"*. V terminológii MIL je teda obrázok popísaný ako vrecko, matematicky reprezentované nasledovne:

$$X = \{X_1, ..., X_N\} \tag{8}$$

X_i reprezentuje vektor čŕt (inštanciu) extrahové z prislúchajúceho i-teho regiónu obrázku a *N* je celkový počet regiónov (inštancií) v obrázku. Vrecko je označené pozitívne (*"pláž"*) v prípade, že obsahuje oba regióny (inštancie) *"piesok"* a *"voda"*.

3.2.2 Učenie sa bez učiteľa

Základný popis o učení bez učiteľa hovorí, že sa jedná o odvodenie funkcie k popisu skrytej vnútornej štruktúry z neoštítkovaných dát. Dalo by sa teda tvrdiť, že tu nie sú žiadne signály, chyba či ukazovatele, ktoré by nám napovedali alebo ohodnocovali to, ako dobré je potenciálne riešenie (predikcia). Tento typ učenia je široko používaným napr. v oblasti dátových analýz.

3.2.3 Učenie posilňovaním

Tento typ učenia má podobne ako učenie s učiteľom k dispozícii istú spätnú väzbu o kvalite predikcií počas trénovania, nie je to však tak konkrétna informácia ako chyba predikcie voči správnym výstupom. Spätná väzba predstavuje ohodnotenie predikcie buď odmenou alebo trestom, v závislosti od toho aká dobrá bola. Hlavným cieľom je, ako by sa dalo očakávať, maximalizovanie získanej odmeny. Tú neurónová sieť získava metódou pokus- omyl pri upravovaní váh počas trénovania. Tento postup do značnej miery simuluje učenie sa v reálnom svete. Najčastejšie sa používa pri algoritmoch, ktoré sú stavané na hranie doskových alebo počítačových hier.

3.2.4 Optimizačné algoritmy

V kombinácii s algoritmami učenia sa používajú optimizačné algoritmy, ktorých cieľom je nájdenie minima funkcie medzi váhami. Medzi základné optimizéry patria:

- **Gradient descent optimizer [26]:**

  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:

[2] zmena veličiny v závislosti od inej premennej

- 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 [16]:**

  V podstate vychádza priamo zo Stochaistického gradient descent optimizéru, resp. jeho modifikácie RMSProp algoritmu[31]. Rozdiel oproti Gradient descent optimizéru je ale v tom, že je schopný variabilne určovať rýchlosť učenia neurónovej siete.

- **Ftrl optimizer:**

  Vychádza z algoritmu učenia FTRL-Proximal[19], 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[24], čiže 80% dát je určených na učenie sa, 10% na validáciu a 10% na samotné otestovanie predikcií modelu siete.

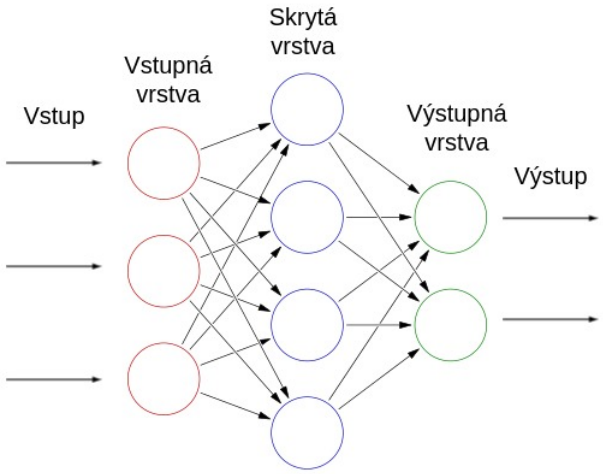
Aj keď neurónové siete dokážu efektívne riešiť veľké množstvo úloh, problémom stále zostáva mať k dispozícii dostatok dát k učeniu neurónovej siete ešte pred

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 1. 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[30]).

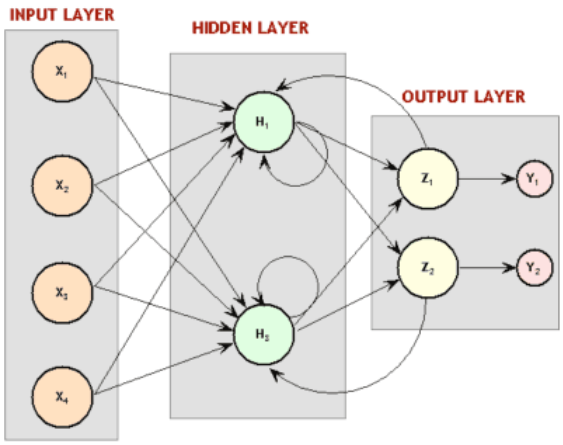


Obr. 1: *Príklad jednoduchej neurónovej siete*

3.3.1 Rekurentné neurónové siete

Zložitejším typom neurónových sietí 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 (ukážka na obrázku 2).



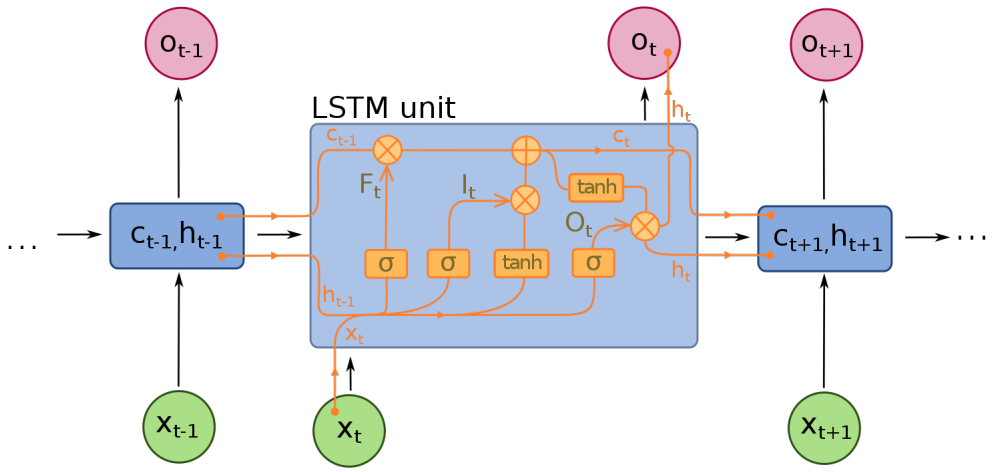
Obr. 2: *Príklad jednoduchej rekurentnej neurónovej siete*[3]

Vyššie uvedená funcionalita dovoľuje zachytiť aj dynamické časovo obmedzené správanie a používať kontext z minulosti, teda použiť niečo ako „pamäť“. Rekurentné siete majú veľké množstvo typov, od jednoduchších LSTM až po zložitejšie obojsmerné (z angl. bi-directional).

LSTM

LSTM - skratka pre anglické pomenovanie Long short-term memory, čo v preklade znamená dlhá krátkodobá pamäť. Tento typ rekurentných sietí sa ukázal extrémne efektívny pri úlohách, kde je nutné pamätať si určitú informáciu (kratšieho charakteru) dlhý čas. Za to môžeme vďačiť LSTM jednotke, ktorej štandardná architektúra je zobrazená na obrázku 3.

[3] http://www.mattmoocar.me/blog/RNNCountryLyrics/



Obr. 3: *Ukážka štruktúry LSTM jednotky*

Na rozdiel od klasickejších rekurentných sietí, kde jednotka obsahuje jednu vrstvu neurónov, LSTM jednotka (bunka) obsahuje až štyri, každá špecializovaná na niečo iné. Na obrázku vyššie reprezentuje prvá horizontálna čiara v jednotke jej stav. Informácia tadiaľto prakticky len "tečie", s malými lineárnymi zmenami ako pridanie alebo odobratie. Tieto akcie sú opatrne regulované tzv. bránami.

Spodná horizontálna čiara reprezentuje postupné vstupy do štyroch vrstviev zobrazených na obrázku ako štvorce s aktivačnými funkciami (σ - sigmoid, tanh). Vrstvy so sigmoid-om fungujú ako tzv. "strážcovia", vzhľadom na ich výstup v intervale <0,1> určujú aké veľké množstvo informácie má byť pustené ďalej (0 - nič, 1 - všetko). Týmto prakticky kontrolujú a udržujú stav celej bunky. Prvá sigmoid vrstva sa nazýva "zabúdacia brána"(z angl. forget gate layer) a určuje, aké množstvo informácií z aktuálneho stavu bunky bude zahodených. Druhá sigmoid vrstva sa nazýva "vstupná brána"(z angl. input gate layer) a určuje, ktoré hodnoty budú aktualizované. Tretia tanh vrstva vytvorí vektor hodnôt, ktoré je možné pridať k stavu bunky. Výstup z týchto dvoch vrstiev je skombinovaný k aktualizovaniu stavu bunky. Posledná sigmoid vrstva prakticky určuje aké hodnoty budú výstupom z celej jednotky. Výstup je založený na upravenom stave bunky, avšak vyfiltrovaný na základe hodnôt z poslednej sigmoid vrstvy.

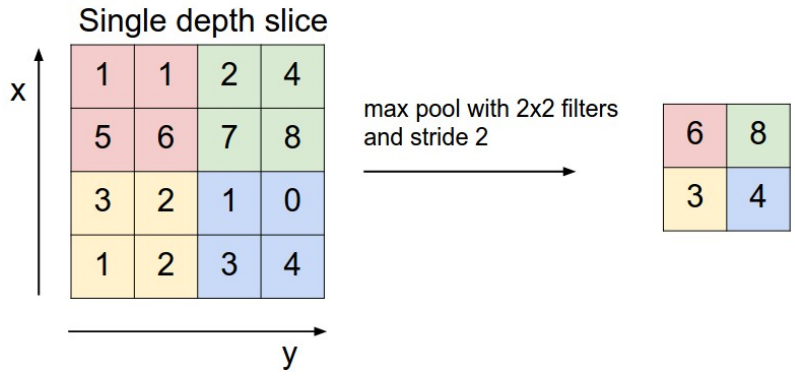
Keďže rekurentné siete na rozdiel od dopredných môžu na vstupe spracovať aj ľubovoľnú sekvenciu (vektor) predstavujú s možnosťou "pamäte"značný posun. V

praxi sa používajú napríklad pri úlohách so spracúvaním jazyka. Keby sme chceli napríklad predikovať nasledujúce slovo vo vete, je veľmi užitočné vedieť aké slová boli pred ním aby predikcia dávala zmysel. Práve v tejto oblasti sa LSTM siete ukázali veľmi efektívne. Využívajú sa ďalej napríklad aj pri rozpoznávaní reči[27] alebo písma[10], či generovaní popisu k obrázkom[15], kedy však funguje v kombinácii s konvolučnou neurónovou sieťou (z angl. convolutional neural network).Tá je ďalším typom sietí a v tomto prípade bola použitá na klasifikáciu obrázkov a rozpoznávanie objektov, LSTM sieť bola použitá iba na generovanie výsledného jednoduchého popisu.

3.3.2 Konvolučné neurónové siete

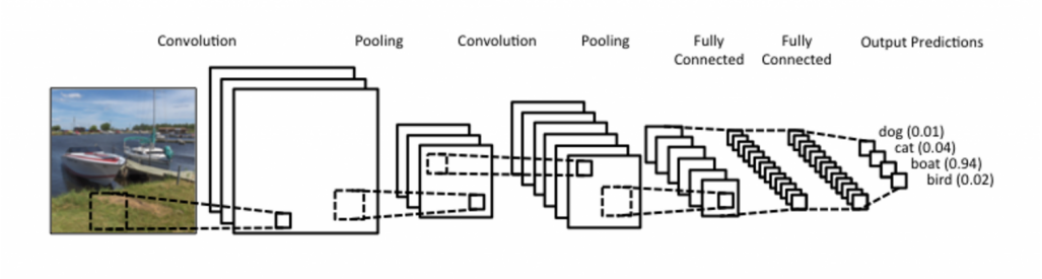
Základ tohto typu siete tvorí vstupná konvolučná vrstva[4] s konvolučným filtrom, ten býva väčšinou malý (3x3, 5x5). Vstup tejto vrstvy musí byť v tvare *m* x *m* x *r*, kde *m* je šírka a výška obrázku, *r* je počet farebných kanálov. Napríklad pre RGB obrázok je r=3 (červená, zelená, modrá). Konvolučným 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[17]), 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[5] 4 je vidieť výsledok popisovaného postupu na jednoduchom príklade.

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



Obr. 4: *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 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[6] nižšie je jednoduchý náčrt vyššie popísané konvolučnej neurónovej siete.



Obr. 5: *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

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

vozidlá, ktoré sa vedia riadiť sami (autopilot) alebo triedenie uhoriek na farmách v Japonsku[7]. Na tento konkrétny softvér bol použitý príklad kódu jednoduchej konvolučnej siete z tutoriálu[8] 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[9] úrovní a posunutý do neurónovej siete.

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[10] 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, ...).

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

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

[9] farebný priestor používaný vo video aplikáciách

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

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[11] (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 frameworkom Keras(kapitola 3.4.4))

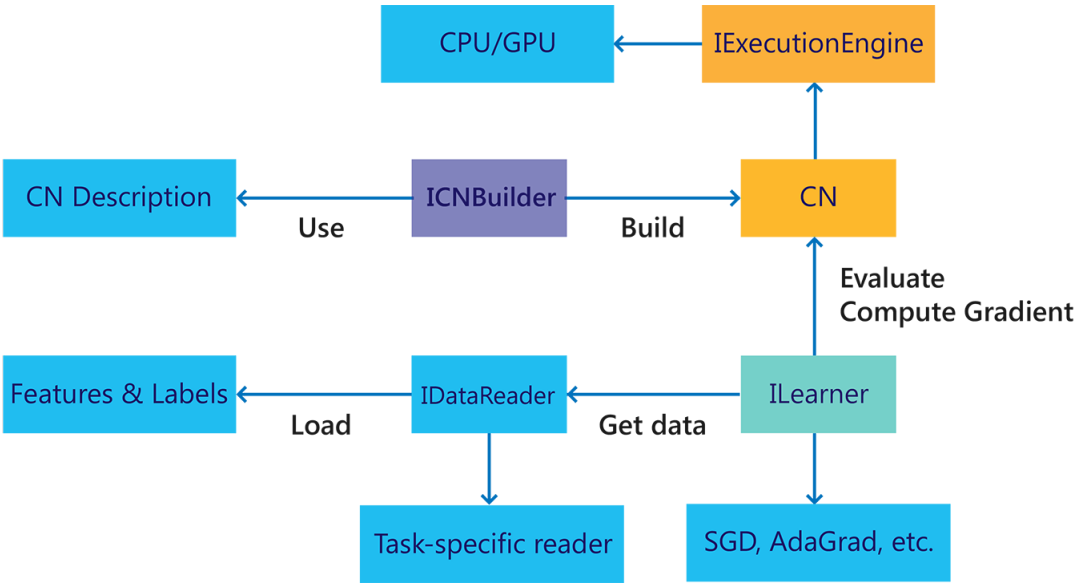
Ako nevýhody možno uviesť:

- 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[12], 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 6.

[11] https://www.tensorflow.org/programmers_guide/summaries_and_tensorboard
[12] https://docs.microsoft.com/en-us/cognitive-toolkit/



Obr. 6: *Diagram zobrazujúci architektúru fungovania Microsoft CNTK*[13]

Momentálne podporuje jazyky C++, C#, Python, Java.
Výhodami tohto framework-u sú:

- flexibilita
- umožňuje distribuovaný tréning

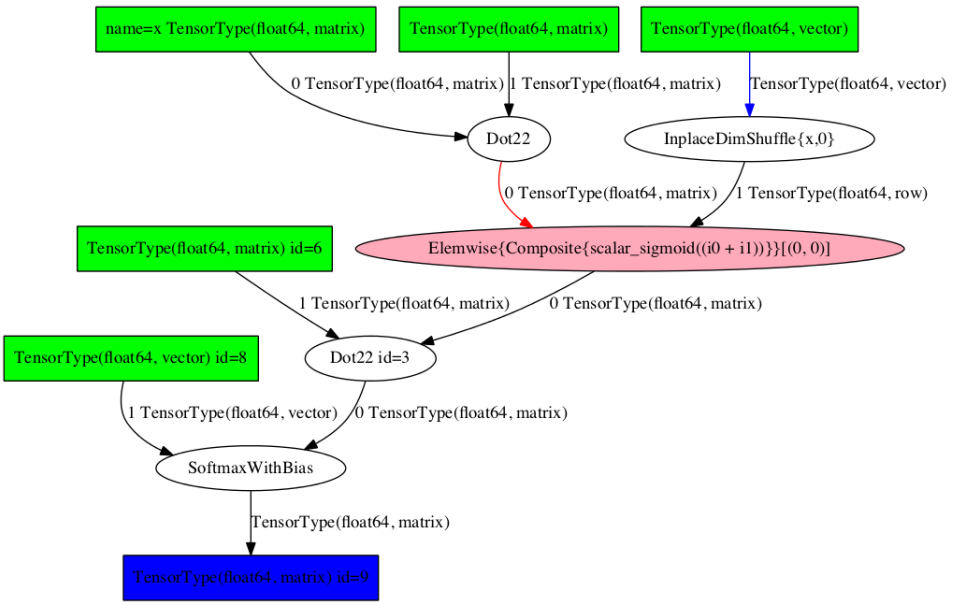
Medzi nevýhody možno zaradiť:

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

3.4.3 Theano

Theano[14] 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 7.

[13] https://dzone.com/articles/progressive-tools10-best-frameworks-and-libraries
[14] https://github.com/Theano/Theano



Obr. 7: *Príklad grafovej abstrakcie výpočtov použitím framework-u Theano*[15]

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)

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

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"*[16], 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
- rýchle prototypovanie

Za jedinú nevýhodu može 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[17].

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

Najväčšími výhodami sú:

[16] https://keras.io/

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

- 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.

4 Existujúce modely vizuálnej pozornosti

Existujúce modely vizuálnej pozornosti možno z pohľadu typu predikovaných máp výraznosti rozdeliť na bottom-up modely (kapitola 4.1) a top-down modely (kapitola 4.2). Ďalším zaujímavým rozdelením však je rozdelenie podľa použitých princípov a technológií [23], na modely:

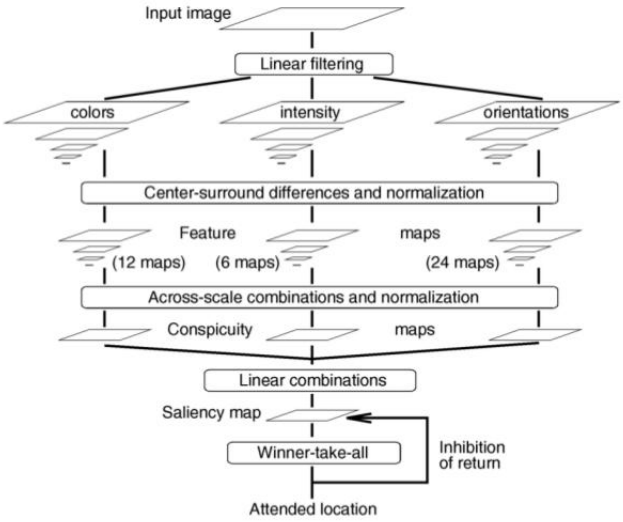
- hierarchické - využívajú hierarchické rozkladanie príznakov
- Bayesove - využívajú kombináciu výraznosti s predchádzajúcimi znalosťami
- rozhodovaco-teoretické - využívajú diskriminačnú teóriu výraznosti
- informaticko-teoretické - využívajú maximalizáciu informácie z daného prostredia
- grafické -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

4.1 Bottom-up modely

Nakoľko práve bottom-up pozornosť bola prvá skúmaná, existuje k jej predikcii obrovské množstvo najrozličnejších modelov. Niekoľko z nich je popísaných v nasledujúcich podkapitolách. Porovnaním týchto rôznych typov by sa dalo povedať, že ako najlepšie a najpresnejšie sa javia nové moderné postupy strojového učenia. Ich nevýhodou však je potreba dostatočne veľkého datasetu, na ktorom by mohli byť natrénované ako aj značne dlhé trénovanie a predikcie v porovnaní s tými, ktoré používajú rôzne matematické postupy alebo extrakcie čŕt a príznakov, kde fáza trénovania nie je nutná.

4.1.1 Itti-ho model

Itti-ho model[7] je jedným z najznámejších modelov vizuálnej pozornosti, i keď patrí medzi tie staršie, dodnes je široko používaný a citovaný v mnohých článkoch. 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 8, zo vstupného obrázka sa vytvoria 3 typy máp a to podľa farby, intenzity a orientácie, ktorých kombináciou sa dosiahne už spomenutá mapa výraznosti.



Obr. 8: *Itti-ho hierarchický model vizuálnej pozornosti[7]*

4.1.2 Metóda hierarchickej výraznosti

Pri použití metódy hierarchickej výraznosti (z angl. hierarchical-saliency method[8]) sa v podstate jedná o detekciu charakteristík v prirodzenej scéne. Tento postup je istým vylepšením predchádzajúceho Ittiho modelu a prakticky pozostáva z nasledujúcich dvoch krokov:

- *prvý krok* - extrakcia globálne výrazných oblastí (z angl. globally salient regions):

  1. výpočet mapy výraznosti *S* zo vstupného obrázka *I*
  2. evaluácia oblastí záujmu (z angl. regions of interest, ROIs), automatická klasifikácia všetkých pixelov do dvoch kategórií k získaniu masky M:
     - globálne výrazné oblasti (*1*)
     - zvyšok (*0*)
  3. vynásobenie masky *M* so vstupným obrázkom *I* k získaniu filtrovaného obrázka *I'*

- *druhý krok* - evaluácia lokálne výrazných častí vo vnútri globálne výrazných oblastí. Použije sa vyfiltrovaný obrázok *I'* z predchádzajúcich krokov k získaniu novej mapy výraznosti *S'*, ktorá je finálnou mapou.

4.1.3 Model pre určovanie salientných oblastí webových stránok

Veľmi zaujímavým riešením pre predikciu a určovanie salientných oblastí scény v doméne webových stránok je model od Chengyao Shen a Qi Zhao[28] využívajúci metódy strojového učenia. Model je postavený na použití metódy MKL (Multiple Kernel Learning - kombinuje viacero jadier SVM (Support Vector Machine) miesto jedného), ktorá bola trénovaná na princípe binárneho regresného problému. Rozdiel oproti predošlým popisovaným modelom je však v tom, že predikovaná nebola priamo výsledná mapa výraznosti ale iba vektor pohľadov (fixácií) na vstupné obrázky na základe ktorého bola potom spomínaná mapa vypočítaná. Autori po vyhodnotení výsledkov zistili že ako najviac salientné oblasti sa v scéne javia v prvom rade ľudská tvár, oči a celkovo vrchná časť ľudského tela. Zaujímavým vylepšením ich modelu však bolo to, že tieto zistenia zapracovali formou predtrénovania MKL na tieto vysoko salientné oblasti.

4.2 Top-down modely

Predikcia top-down pozornosti je značne zložitejšia, nakoľko sa do úvahy berie aj sématický kontext scény. Tým môže byť napríklad emočný obsah (kapitola 4.2.1), či hľadanie konkrétnych objektov v scéne (kapitola 4.2.3).

4.2.1 Zameranie na emočný obsah

Niekoľko riešení sa v posledných rokoch pokúsilo o predikovanie top-down pozornosti na základe emočného obsahu v pozorovanej scéne, z nich stojí za zmienku napríklad model od Liu a spol. [18]. Zamerali sa na spomínaný emočný obsah v rôznych oblastiach (blokoch) obrázka, kde do úvahy brali:

- emočné objekty, napr. hady a krv (evokujú silný strach)
- výrazy tváre

- všeobecný emočný obsah

Hlavné problémy, ktoré riešili, by sa dali zhrnúť do dvoch otázok:

- Ako získať a ohodnotiť intenzitu emócií pre obrázok?
- Aký typ emócie by mal byť dominantným pre jednotlivé obrázky?

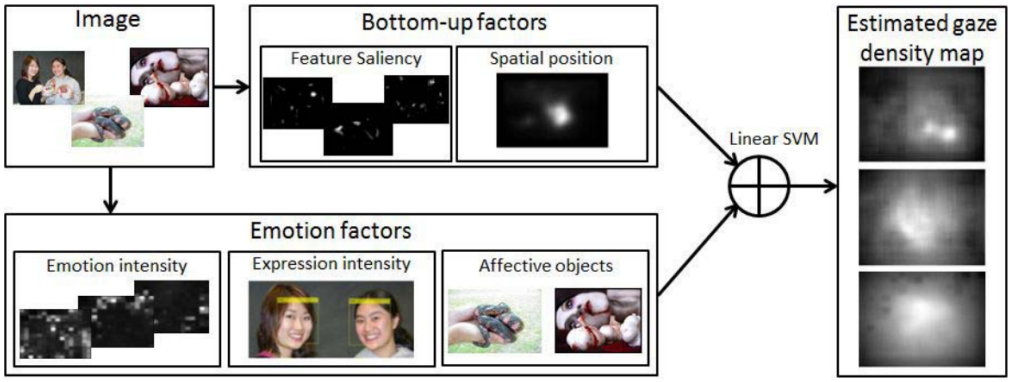
Ich narhované riešenie malo niekoľko častí. Prvou je extrakcia bottom-up faktorov vo forme nízko-úrovňových máp výraznosti založených na črtách a mapy priestorovej polohy s dôrazom na centrum pre výpočet odhadu hustoty pohľadu (z angl. gaze density estimation). K tomuto boli použité klasické hierarchické modely pre určovanie pozornosti (napr. Itti-ho model).

Druhou časťou je detekcia emócií, k čomu bol použitý emočný detektor založený na učení s použitím viacerých inštancií učenia (z angl. multiple instance learning (MIL [18])) zo slabo oštítkovaných dát. Tými boli obrázky označené emočným typom[19], ktorý bol určený po vypočítaní pravdepodobnosti vyskytu jednotlivých typov pre každú časť obrázka. Touto matematickou operáciou sa autorom podarilo úspešne vyriešiť problém určenia dominantného typu emócie pre obrázok. Ďalej bol pre zlepšenie extrakcie emócií použitý mechanizmus založený na SVM, ktoré bolo použité na detekciu ľudských tvárí a odhadnutie intenzity výrazu. Výstupom z tejto časti je emočná mapa ohodnocujúca intenzitu emócií určená pre ďalšie spracovanie.

Treťou časťou bolo spracovanie výstupom z predošlých dvoch lineárnym SVM, ktoré vo výsledku z bottom-up a emočných faktorov vygeneruje výslednú mapu hustoty pohľadu (z angl. gaze density map). Vyššie popísaný postup je zobrazený na diagrame na obrázku 9.

[18] variácia učenia s učiteľom. Miesto obdržania inštancií, ktoré sú individuálne oštítkované obdrží model set oštítkovaných vreciek (z angl. bag), z ktorého každé obsahuje niekoľko inštancií. V prípade jednoduchej binárnej klasifikácie je celé vrecko označené ako negatívna vzorka, ak všetky inštancie vo vnútri sú negatívne. Pokiaľ ale obsahuje aspoň jednu pozitívnu, je označené ako pozitívna vzorka.)

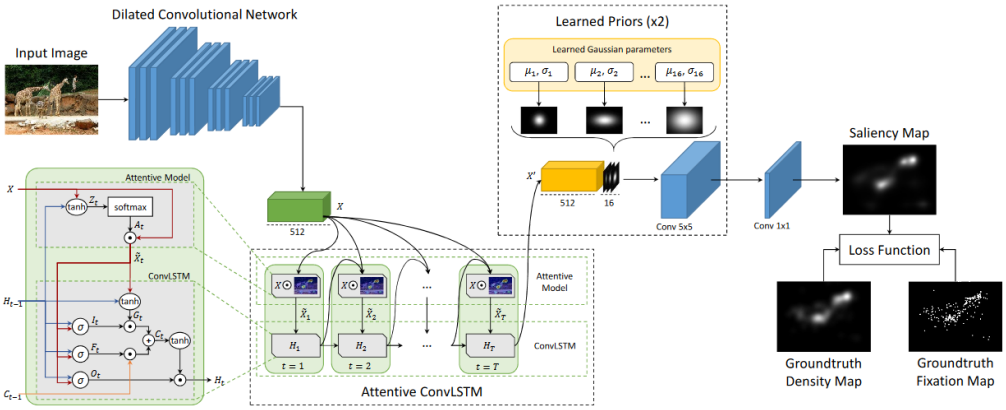
[19] jedna z 8 skupín, do ktorých boli rozdelené emócie. Príkladom týchto skúpín sú napr. hnev, láska, smútok, prekvapenie, atď.



Obr. 9: *Diagram modelu od Liu a spol. k predikcii máp výraznosti založených na emočnom obsahu[18]*

4.2.2 SAM

SAM - skratka pre Saliency Attentive Model[6], ďalší z modelov schopných aj predikcie top-down pozornosti. Založený je na architektúre nazvanej ML-Net (Multi-Level Network[5]), ktorá miesto klasických konvolučných vrstiev využíva dilatované konvolučné vrstvy, ktoré limitujú efekt zmenšovania. Táto architekúra použitá pre spracovanie vstupného obrázka je nasledovaná konvolučnými LSTM vrstvami, ich naučené výstupy potom spracováva už klasická konvolučná vrstva, výstupom ktorej je mapa výraznosti. Popisovaný postup je zobrazený na obrázku 10.



Obr. 10: *Diagram modelu využívajúceho dilatované konvolučné vrstvy v kombinácii s LSTM vrstvami [6]*

Vďaka architektúre zobrazenej na obrázku vyššie je možné nezávisle na sebe trénovať dilatovanú konvolučnú sieť (alebo použiť predtrénovanú, či o niečo inú) a konvolučnú LSTM sieť. Veľkou výhodou dilatovanej konvolučnej siete je stratégia zväčšenia výstupného rozlíšenia obrázka, zatiaľ čo sa zachováva mierka, na ktorej operujú konvolučné filtre a počet parametrov. To je rozdiel oproti klasickej konvolučnej sieti, kedy je vstupný obrázok po extrakcii čŕt obvykle zmenšený, čo môže značne zhoršiť presnosť predikcie máp výraznosti.

Spomínané LSTM vrstvy (podrobnejšie zobrazené aj na obrázku 10) využívajú kombináciu aktivačných funkcií *softmax*, *tanh* a *sigmoid*, kedy pri spracúvaní sekvenčne aktualizujú svoj interný stav. Ako vstup dostanú vektor extrahovaných čŕt z dilatovaných konvolučných vrstiev (X), ktoré spracúvajú vstupné obrázk,y a generovanú mapu výraznosti z predošlej LSTM vrstvy (s výnimkou prvej). Táto generovaná mapa je ešte upravená mechanizmami pozornosti, ktoré sa selektívne zameriavajú na rôzne časti obrázku. Vo výsledku je potom vektor extrahovaných čŕt (X) a upravená generovaná mapa (X_t) z predošlej vrstvy ešte ďalej spracovávaná aktovačnými funkciami *tanh*.

Výstup z týchto vrstiev je ďalej upravený štandardnými konvolučnými vrstvami do výsledných máp výraznosti.

Ďalšou zaujímavou vecou na tomto riešení je zvolenie funkcie chyby (z angl. loss function). Autori nepoužili žiadny z tradičnejších typov, ale použili lineárnu

kombináciu troch rôznych evaluačných metrík (popísané v kapitole 5), v tvare:

$$L(y, y^{den}, y^{fix}) = \alpha L1(y, y^{fix}) + \beta L2(y, y^{den}) + \gamma L3(y, y^{den}) \tag{9}$$

Pre vyššie uvedené platí:

y - predikovaná mapa výraznosti

y^{den} - pravdepodobnostná distribúcia (z angl. groundtruth density distribution)

y^{fix} - binárna mapa fixácií

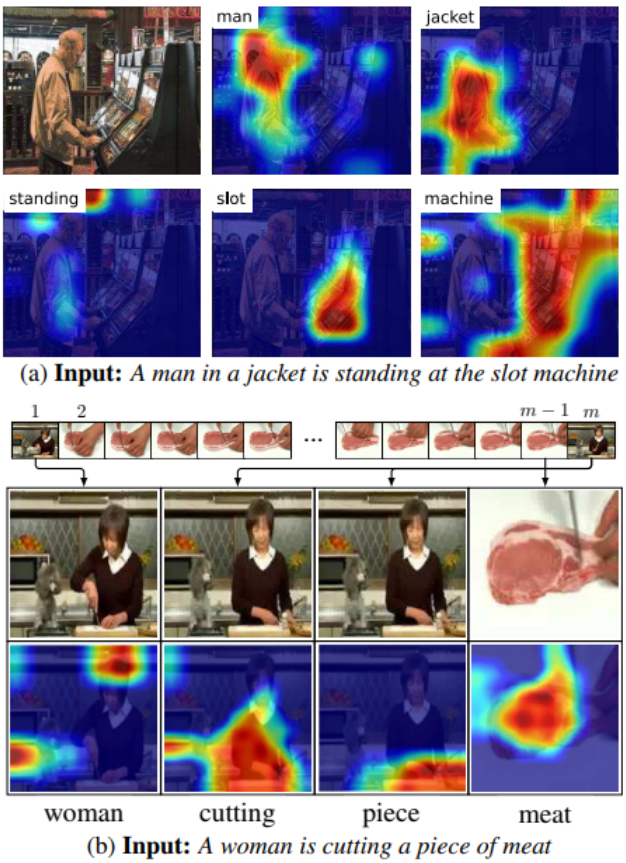
α, β, γ - tri skalárne hodnoty určené k vybalancovaniu troch funkcií chyby

$L1, L2, L3$ - tri funkcie pre výpočet evaluačných metrík, za radom Normalizovaná cesta výraznosti (NSS, z angl. Normalized Scanpath Saliency), Lineárny Korelačný Koeficient (CC), Kullback-Leiblerova divergencia (KL-Div), všetky bližšie popísané v kapitole 5

4.2.3 Top-down pozornosť vedená titulkami

Jedným z príkladov top-down pozornosti je úloha nájdenia nejakého konkrétneho objektu v scéne, čím sa inšpirovali výskumníci z Bostonskej univerzity. Predstavili model[20] [25] schopný predikcie máp výraznosti na základe kľúčových slov v zadanej vete a príslušnom obrázku alebo videu, príklad možno vidieť na obrázku 11.

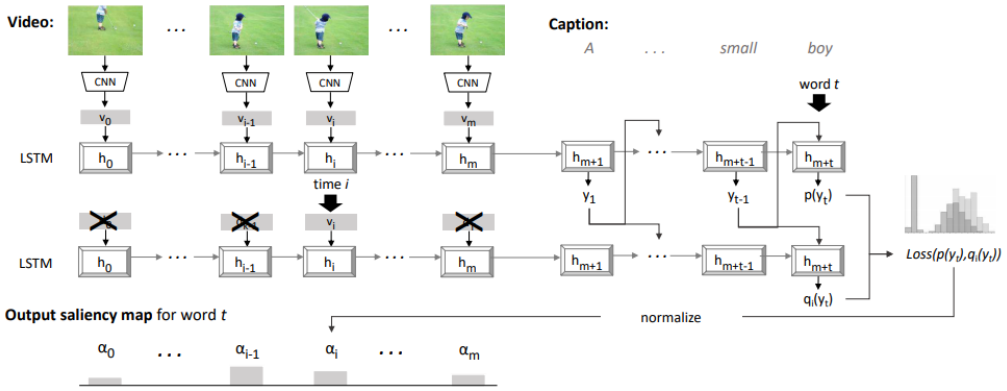
[20]http://ai.bu.edu/caption-guided-saliency/



Obr. 11: *Predikcia salientných oblastí na základe kľúčových slov vo vete, hore po* ***a*** *predikcia pre obrázok, dolu po* ***b*** *predikcia pre video[25]*

Autori použili prístup nazvaný titulkami vedená vizuálna pozornosť (z angl. Caption-Guided Visual Saliency), ktorý produkuje mapy výraznosti pre nepohyblivé obrázky alebo video. Ako základný model použili LSTM encoder-decoder, ktorý predikuje aj dočasné (z videa) aj priestorové (z obrázkov) salientné mapy na základe kľúčových slov vo vete (vstupné titulky, popis) a mapuje ich na vstupné obrázky (alebo video). Vstupné titulky sú spracovávané LSTM vrstvami. Na obrázku 12 je možné vidieť načrtnutý model riešenia. Ako prvá vstupná vrstva funguje konvolučná neurónová sieť reprezentujúca encoder pre video, ktorá "zakóduje" reprezentáciu obrázkov aj s aktiváciami všetkých vizuálnych konceptov detekovaných vo videu. Tieto informácie posunie ďalej ako vektor pre LSTM vrstvy reprezentujúce decoder - ten rozhoduje o tom, ktoré časti použije pomocou

LSTM výstupných brán k predikcii mapy výraznosti pre kľúčové slovo v čase *t*. Autori ďalej zvolili veľmi šikovné riešenie pre predikciu máp len pre obrázky - jednoducho výstup produkovaný poslednou konvolučnou vrstvou zmenili na tzv. "dočasnú"sekvenciu (vektor): $V = (v_1, ..., v_m)$. Tá je vytvorená sekvenčne scanovaním obrázka riadok po riadku z ľavého horného rohu do pravého dolného rohu. Prvá LSTM vrstva potom tento upravený vektor spracuje a posunie ďalej k dekódovaniu na kľúčové slová vo vete.



Obr. 12: *Diagram modelu neurónovej siete, zhora vstup vľavo vo forme videa, vpravo titulky k nemu. V strede LSTM vrtvy neurónovej siete, dolu výstupná mapa výraznosti pre kľúčové slová. [25]*

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 výraznosti (z angl. Normalized Scanpath Saliency). Využíva priemer hodnôt výraznosti na *n* fixácií v normalizovanej mape podľa nasledovného vzorca:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{s(x_h^i, y_h^i) - \mu_s}{\sigma_s} \tag{10}$$

- 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 výraznosti 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.

$$CC(s,h) = \frac{cov(s,h)}{\sigma_s \sigma_h} \tag{11}$$

- KL-Div - Kullback-Leiblerova divergencia, všeobecné teoretické merianie rozdielu medzi dvoma pravdepodobnostými distribúciami. V oblasti predikcií vizuálnej pozornosti sa často nazýva aj AUC-Judd. Ako vstup berie mapu výraznosti *P* a mapu fixácií (z angl. ground truth fixation map) Q^D, evaluuje stratu informáciie keď je *P* použité k aproximácii Q^D.

$$KL(P, Q^D) = \sum_i Q_i^D log\left(\epsilon + \frac{Q_i^D}{\epsilon + P_i}\right) \tag{12}$$

6 Návrh

Na základe analýzy problémovej oblasti a existujúcich riešení sme sa rozhodli najprv použiť konvolučnú neurónovú sieť (jej popis a architektúra v kapitole 6.1), čo sa pretavilo aj do prvotných experimentov (kapitola 6.2) vykonávaných v rámci predmetu Počítačové videnie[21].

6.1 Návrh neurónovej siete

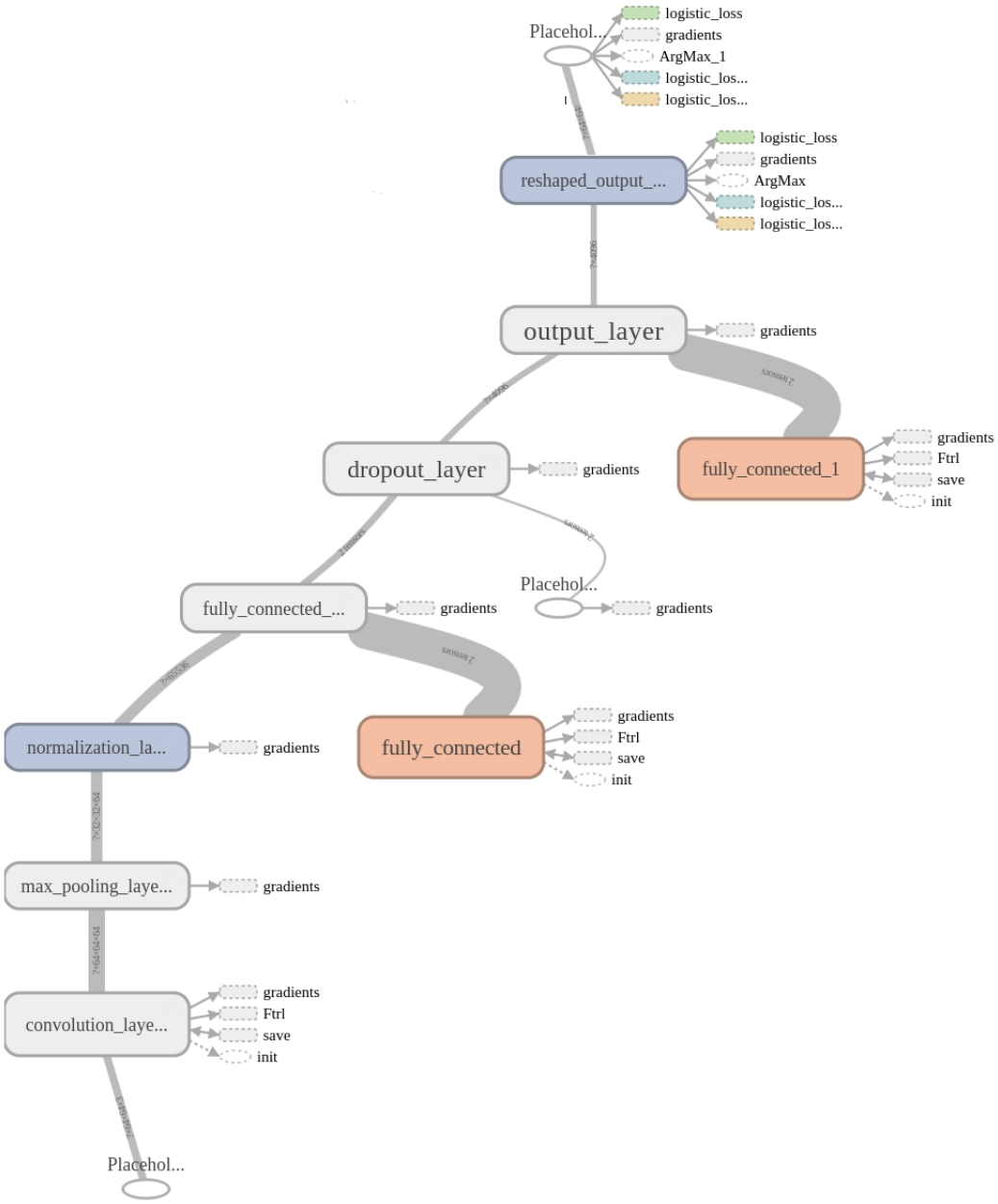
Celá architektúra je načrtnutá na schéme na obrázku 13 vytvorenej pomocou nástroja TensorBoard[22].

Jedná sa o jednoduchú sieť so vstupnou konvolučnou vrstvou pre spracovanie obrázkov. Táto vrstva obsahuje konvolučný filter (veľkosť *5x5*) s aktivačnou funkciou sigmoid. Výstup z nej ďalej pokračuje do vrstvy združovania, kde sa použije operácia MAX s filtrom o veľkosti *2x2* a krokom tiež s veľkosťou *2*. Po nich nasleduje vrstva normalizácie, kde je celý výstup zlúčený do jednej širokej vrstvy. Za ňou sa nachádza plne prepojená vrstva (z angl. fully-connected layer) s aktivačnou funkciou sigmoid a vrstva výpadku (z angl. dropout layer[29]), ktorej hodnota (v rozmedzí od 0 do 1) určuje, aké percentuálne množstvo neurónov aj s prepojeniami bude dočasne skrytých. Táto možnosť umožňuje počas učenia sa predchádzať pretrénovaniu. Za vrstvou výpadku už nasleduje iba výstupná vrstva a jej transformácia na 2D maticu, obrázok predstavujúci mapu výraznosti, ktorú chceme dostať.

Aktivačná funkcia sigmoid je použitá najmä preto, že mapa výraznosti je prakticky pravdepodobnostné rozdelenie, t. j. sieť sa snaží predikovať pravdepodobnosti výraznosti každého pixelu. Ako algoritmus učenia sme zvolili štandardný algoritmus spätného šírenia chyby (z angl. backpropagation) s trochu extravagatným FTRL optimizérom.

[21] http://vgg.fiit.stuba.sk/teaching/computer-vision/

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



Obr. 13: *Diagram reprezentujúci architektúru neurónovej siete, zdola vstupná konvolučná vrstva nasledovaná ostatnými vrstvami siete až po výstupnú spolu s transformáciou na 2D maticu reprezentujúcu predikovanú mapu výraznosti pre vstupný obrázok*

6.2 Prvotné experimenty

6.2.1 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] Marcella Cornia, Lorenzo Baraldi, Giuseppe Serra, and Rita Cucchiara. A deep multi-level network for saliency prediction. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 3488–3493. IEEE, 2016.

[6] Marcella Cornia, Lorenzo Baraldi, Giuseppe Serra, and Rita Cucchiara. Predicting human eye fixations via an lstm-based saliency attentive model. *arXiv preprint arXiv:1611.09571*, 2016.

[7] 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.

[8] 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.

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

[10] 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.

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

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

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

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

[15] 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.

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

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

[18] Huiying Liu, Min Xu, Jinqiao Wang, Tianrong Rao, and Ian Burnett. Improving visual saliency computing with emotion intensity. *IEEE transactions on neural networks and learning systems*, 27(6):1201–1213, 2016.

[19] 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.

[20] Fayyaz ul Amir Afsar Minhas and Asa Ben-Hur. Multiple instance learning of calmodulin binding sites. *Bioinformatics*, 28(18):i416–i422, 2012.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[22] Michael A Nielsen. *Neural networks and deep learning*. Determination Press, 2015.

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

[24] 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.

[25] Vasili Ramanishka, Abir Das, Jianming Zhang, and Kate Saenko. Top-down visual saliency guided by captions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 7, 2017.

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

[27] 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.

[28] Chengyao Shen and Qi Zhao. *Webpage Saliency*, pages 33–46. Springer International Publishing, Cham, 2014.

[29] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

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

[31] 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.