

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-5212-73665

Patrik Beka

Určovanie pútavých častí grafických rozhraní

Bakalárska práca

Študijný program: Informatika

Študijný odbor: 9.2.1 Informatika

Miesto vypracovania: Ústav informatiky, informačných systémov a softvérového
inžinierstva, FIIT STU, Bratislava

Supervisor: Ing. Mário Šajgalík, PhD.

Máj 2017

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

V prvom rade chcem podľakovať svojmu vedúcemu Ing. Máriusovi Šajgalíkovi, PhD. za odborné rady a čas strávený pri konzultovaní tejto práce v prvom semestri a počas skúškového obdobia.

Ďalej by som rád podľakoval pani doc. Ing. Vande Benešovej, PhD. za rady a konzultácie v druhom semestri, spolu s možnosťou prezentácie práce na jej seminároch.

Moje veľké podľakovanie patrí aj Jakubovi Kazimírovi, ktorý mi bol ochotný vždy pri akomkoľvek probléme pomôcť, bez ohľadu na množstvo a intenzitu mojich nepríjemných sarkastických poznámok.

Ďalej ďakujem Tomášovi Lachovi za pevné nervy počas množstva prebdených nocí strávených programovaním a písaním našich bakalárskych projektov. Za tolerovanie môjho neustáleho „sťažovania sa“ a za zavedenie spoločného pracovného režimu založeného na motivačných odmenách po splnení určenej časti práce.

Spoločnosti ST. NICOLAUS, a. s. patrí nasledujúce podľakovanie za stabilnú kvalitu ich produktov s o niečo vyšším percentuálnym obsahom antistresových látok ako je obvykle znesiteľné. Práve tieto v kombinácii s kávou významne dopomohli dokončeniu tejto práce formou motivačných prestávok, vďaka ktorým zostal autor bez akejkoľvek väčšej fyzickej, či psychickej ujmy.

Za technickú podporu vždy keď som mal nejaký problém s fakultným serverom, či už z dôvodu nedostatku práv pri jeho používaní alebo jednoducho len mojej nevedomosti, by som rád podľakoval Bc. Lukášovi Martákovi, ktorý mi vždy, bez ohľadu na pokročenosť hodiny či stupiditu mojej otázky, dokázal ochotne pomôcť.

Za pomoc pri úprave šablóny a celej práce v LaTeX-u ďakujem Ondrejovi Kaščákovi.

Za úpravu angličtiny jednak v článku na IIT. SRC, ale aj v anotácii ďakujem Lucii Martančíkovej.

Veľké ďakujem patrí aj všetkým 20 participantom nášho experimentu na získanie dát, bez vás by pravdepodobne výsledky tejto práce boli mnohonásobne horšie.

Anotácia

Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Študijný program: Informatika

Autor: Patrik Beka

Bakalárská práca: Určovanie pútavých častí grafických rozhraní

Vedúci práce: Ing. Mária Šajgalík, PhD.

Máj 2017

Pútavý a graficky príťažlivý dizajn je mimoriadne dôležitou súčasťou každej webovej stránky, hoci sa to tak na prvý pohľad nemusí javiť. Niekoľko môže namietať, že oveľa dôležitejší je samozrejme jej obsah, avšak bez dobrého dizajnu bude návštevník stránky len znechutený a návrat na stránku si v budúcnosti radšej dvakrát rozmyslí. Z tohto pohľadu je dobrý dizajn dôležitý nielen pre používateľov stránok, ale rovnako aj majiteľov, pretože tí by mali byť schopní umiestňovať obsah na základe jeho dôležitosti. Nepodstatné reklamy tak, aby nepôsobili rušivo a relevantné informácie na miesta, kde ich používateľ ihneď zaregistruje.

Táto práca sa zaobráva práve analýzou určovania pútavých častí, spolu s popisom existujúcich riešení ich určenia, a popisuje nami zvolenú metódu k ich predikcii založenú na strojovom učení, konkrétnie neurónových sieťach, ktoré sú v posledných rokoch čoraz viac rozšírené. Sú schopné naučiť sa určité zákonitosti priamo z dát a následne ich aplikovať na doteraz nevidené nové dáta. Nakoľko sme chceli aby boli predikcie možné jednoducho z obrázka webovej stránky, použili sme konvolučné neurónové siete, ktoré sú schopné s dostatočnou presnosťou predikovať pútavé časti na nových nevidených stránkach, čo koniec koncov dokázalo aj porovnanie s inými riešeniami, v ktorom naša metóda dopadla veľmi dobre.

Annotation

Slovak University of Technology Bratislava

FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

Degree Course: Informatics

Author: Patrik Beka

Bachelor thesis: Determination of the eye-catching parts in graphical interfaces

Supervisor: Ing. Mária Šajgalík, PhD.

May 2017

Eye-catching and graphically attractive design is an extremely important part of every website, although it might not appear to you at first sight. Someone could argue that the content is of course much more important, but without a good design, future comeback of website visitor is unlikely. From this point of view, good design is important not only for the users of websites, but also for the owners, because they should be able to place the contain according to its importance. The insubstantial ads on spot, which will not appeal bothering and the relevant informations on places, where the user notices it immediately.

This thesis describes the analysis of determining eye-catching parts (alongside with existing solutions) and also describes our method of determination based on machine learning, specifically neural networks. These networks are able to learn certain regularities directly from the data and subsequently apply them on previously unseen data. We wanted to predict eye-catching parts just from the screenshot of website, so we used convolutional neural network, which is able to predict interesting parts on new unseen websites with the sufficient accuracy. Comparing with other solutions, our method proved to be very good.

Obsah

1	Úvod	1
2	Analýza	3
2.1	Vizuálna výraznosť	3
2.2	Existujúce modely vizuálnej pozornosti	3
2.3	Neurónové siete	4
2.3.1	Typy neurónových sietí	10
2.3.2	Konvolučné neurónové siete	11
2.4	Určenie pútavých častí webových stránok pomocou strojového učenia	13
2.4.1	Riešenia na báze segmentácie stránok	13
2.4.2	Riešenia vychádzajúce z obrázku stránky	15
3	Návrh	19
3.1	Návrh neurónovej siete	19
3.2	Dataset	21
4	Implementácia	25
4.1	Spracovanie datasetu pre neurónovú sieť	25
4.2	Trénovanie neurónovej siete	27
4.3	Využívanie natrénovaného modelu	28
5	Experimenty	29
5.1	Model s validáciou vs. model bez validácie	29
5.2	Model s vrstvou výpadku vs. model bez vrstvy výpadku	30
6	Výsledky	31
6.1	Evaluácia modelu neurónovej siete	31
6.2	Metriky používané na ohodnotenie modelov vizuálnej pozornosti .	31
6.3	Porovnanie s inými riešeniami	32
7	Zhrnutie	35
	Literatúra	39

A Technická dokumentácia	i
A.1 Hardvér	i
A.2 Použité knižnice	i
A.2.1 TensorFlow	i
A.2.2 Matplotlib	ii
A.3 Zdrojové kódy najdôležitejších funkcií	ii
A.3.1 Funkcia load_fixations	ii
A.3.2 Funkcia pre výpočet máp výraznosti	iii
A.4 Funkcia na vytvorenie modelu neurónovej siete	iv
A.5 Funkcia pre trénovanie modelu neurónovej siete	v
A.6 Funkcia slúžiaca k predikciám na natrénovanom modeli	viii
B Používateľská príručka	xi
C Plán práce na riešení projektu	xiii
D Príspevok na konferenciu IIT. SRC 2017	xv
E Protokol z experimentu s cieľom získania dát pre neurónovú sieť	xxi
F Obsah priloženého elektronického nosiča	xxv

Zoznam obrázkov

1	Itti-ho hierarchický model vizuálnej pozornosti	4
2	Jednoduchá neurónová sieť	10
3	Vrstva združovania - príklad vzorkovania	12
4	Konvolučná neurónová sieť	12
5	Vizualizácia mapovania dôležitosti blokov webových stránok	15
6	Vizualizácia pohľadu formou teplotných máp	17
7	Návrh neurónovej siete	20
8	Vizualizácia prvého datasetu	22
9	Vizualizácia druhého datasetu	22
10	Ukážka výpočtu heatmapy	26
11	Ukážka náhodného zamiešania datasetu pre trénovanie	27
12	Trénovanie s validáciou vs. trénovanie bez validácie	30
13	Model s vrstvou výpadku vs. bez nej	30
14	Náš model vs. Itti-ho model	33

1 Úvod

Výsledkom bakalárskej práce je prototyp schopný predikovania pútavých častí grafických rozhraní webových stránok len z ich obrázka. Tieto časti určuje naša vizuálna pozornosť, ktorá je popísaná v časti 2.1.

Predikcie pútavosti (inak aj výraznosti) môžu slúžiť ako veľká pomoc pre dizajnérov stránok, ktorým môžu významne pomôcť pri určení častí, ktoré zaujmú ako prvé a najviac. Práve k tomu v súčasnosti neslúži až také veľké množstvo nástrojov, ako by sa dalo predpokladať. V zásade sú najpoužívanejšie 2 typy modelov vizuálnej pozornosti, a to buď staršie využívajúce rôzne rozkladanie príznakov, výraznosť, či grafové algoritmy (časť 2.2) alebo novšie založené na strojovom učení (časť 2.4). Práve v strojom učení sa v posledných rokoch do popredia dostávajú neurónové siete, popísané v časti 2.3, ktoré sme si na riešenie problému určovania pútavých častí zvolili aj my.

Konkrétnie sme vybrali konvolučnú neurónovú sieť, určenú pre spracovanie obrazu, ktorá bude predikovať pútavé oblasti webových stránok vo forme máp výraznosti, resp. teplotných máp. Jej návrh aj s grafickou schémou je v časti 3.1. Dataset potrebný na natrénovanie takejto siete by mal byť čo najväčší, ten náš tvoria obrázky webových stránok a pohľady na ne. Jeho získanie aj s bližším popisom sa nachádza v časti 3.2.

Celý model vybratej siete je naimplementovaný v jazyku Python, s použitým open-source knižníc preň. Tie sú spolu so spracovaním datasetu, trénovaním a využívaním natrénovaného modelu popísané v kapitole 4.

Rôzne experimenty vedúce až k finálnemu prototypu modelu sú opísané v kapitole 5. Za ňou nasleduje jedna z najpodstatnejších kapitol venujúca sa dosiahnutým výsledkom. Na tie sa treba pozerať z dvoch strán. Najprv z pohľadu neurónovej siete a toho, ako dobre sa neurónová sieť dokázala naučiť predikovať na predložených dátach (časť 6.1) a potom z pohľadu kvality predikcií, ktoré je nutné porovnať metrikami vizuálnej pozornosti, popísanými v časti 6.2, s inými už existujúcimi modelmi (časť 6.3).

V záverečnom zhrnutí je už len v krátkosti zosumarizovaný celý postup a riešenie problému, spolu s dosiahnutými výsledkami a presnosťou.

2 Analýza

V tejto časti je postupne popísaná vizuálna výraznosť v časti 2.1, časť 2.2 sa venuje modelom vizuálnej pozornosti k jej určeniu. Ďalej sú v časti 2.3 popísané neurónové siete, ich aktivačné funkcie, spôsoby učenia a ich typy, z ktorých bližšie sú popísané konvolučné neurónové siete. Záverečná časť 2.4 sa venuje práve riešeniam problému určenia pútavých častí grafických rozhraní web stránok pomocou strojového učenia.

2.1 Vizuálna výraznosť

Vizuálna výraznosť sa dá definovať ako značne subjektívna perceptuálna vlastnosť, vďaka ktorej niektoré veci vo svete vyčnievajú v porovnaní so svojimi susedmi a tak okamžite upútajú pozornosť.[12] Tá sa spolu s pútavosťou (výraznosťou) delí na dva typy a to:

- pozornosť „zdola nahor“ (z angl. bottom-up) - riadená čisto iba pútavými stimulmi, ktoré sú v podstate známkou toho, že „táto lokácia je značne odlišná od svojho okolia a stojí za pozornosť.“ Je rýchla a nevedomá.
- pozornosť „zhora nadol“ (z angl. top-down) - riadená tzv. predvídateľnými mechanizmami, ako napríklad hľadaním niečoho konkrétneho na webovej stránke. Je pomalšia, vedomá a ovplyvnená našou pamäťou.

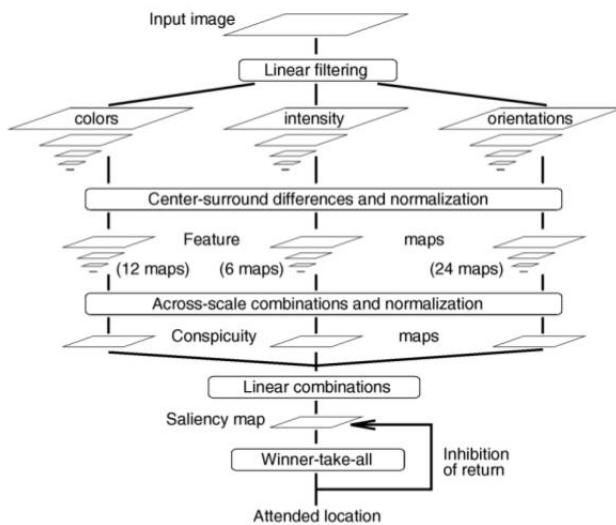
2.2 Existujúce modely vizuálnej pozornosti

Existujúce modely vizuálnej pozornosti[18] 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 algoritnoch
- vzorovo klasifikačné - využívajú strojové učenie zo vzorov s výraznými črtami

Jedným z najznámejších modelov vizuálnej pozornosti je Itti-ho hierarchický model[9]. Je to biologicky inšpirovaný bottom-up model, ktorý využíva hierarchické rozloženie vlastnosí 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[9]*

2.3 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 zvyklo vyjadrovať

podľa nasledovnej matematickej špecifikácie:

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

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 (nás 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:

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

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 - matematické vyjadrenie použité k aproximácii vplyvu na neurón. V našom prípade sa jedná o neurón umelý a funkciu, ktorá definuje výstup neurónu pre vstup alebo sadu vstupov. Aktivačných funkcií existuje niekoľko typov, každá vhodná na iný typ úloh. Niektoré z nich sú popísané nižšie.

Aktivačné funkcie

- **Softmax**

Funkcia softmax¹ (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 matematické vyjadrenie je nižšie.

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

Pre vyššie uvedené vyjadrenie platí:

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

vec - konkrétny vector

Ked' 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:

$$f(x) = \max(0, x) \quad (4)$$

kde x je vstup do neurónu. Používa sa vďaka svojej jednoduchosti, keďže neobsahuje žiadne komplikované výpočty, čoho dôsledkom je aj jej značná

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

rýchlosť. Jej využitie je možné pozorovať napríklad pri hlbokých neurónových sieťach.

- **Softplus**

Je v podstate aproximáciou k predošej ReLU s matematickým vyjadrením:

$$f(x) = \ln(1 + e^x) \quad (5)$$

Rovnako ako pri ReLU je oborom hodnôt interval $(0, \infty)$. 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:

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

- **Tanh**

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

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

Učenie sa

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[11]). Hlavným rozdielom medzi nimi je, že učenie s učiteľom musí prebiehať na predpripravenom datasete, ktorý musí obsahovať nejaké testovacie vstupné dátá (pre ktoré chceme vypočítať výstupnú funkciu) a takzvané štítky (z angl. labels), ktoré sú v podstate naše očakávane výstupy. Učenie sa bez učiteľa naproti tomu odvodzuje funkciu k popisu skrytej štruktúry

z neoštítkovaných dát, teda bez štítkov, ktoré nám určujú očakávané 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 dátá 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 ako Gradient descent optimizer[20] alebo Adam optimizer[14], popísané nižšie. Tieto algoritmy sú určené k nájdeniu minima funkcie medzi váhami.

Gradient descent optimizer

Je to iteratívny algoritmus používaný k nájdeniu lokálneho minima funkcie, kedy podniká kroky k nájdeniu záporného gradientu² funkcie v aktuálnom bode. To je využívané pri určovaní rýchlosťi 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:

² zmena veličiny v závislosti od inej premennej

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

V podstate vychádza priamo zo Stochastického gradient descent optimizéru, resp. jeho modifikácie RMSProp algoritmu[27]. 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[16], 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.

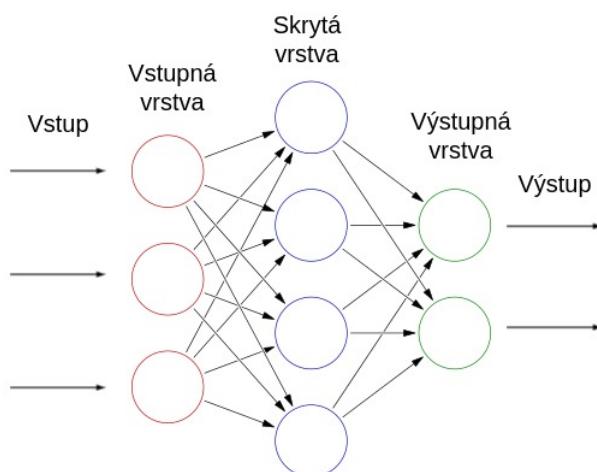
Po fáze učenia sa nasleduje validácia, pri ktorej sieť nemá prístup k štítkom. Na záver sa prejde k samotnému testovaniu neurónovej siete, kedy sa do nej posúvajú dátá rovnako bez toho, aby sieť mala prístup k štítkom. Na základe jej predikcie a štítkom k testovacím dátam sa určí jej presnosť. Na učenie, testovanie a validáciu by nemali byť použité tie isté dátá. Pomer dát k jednotlivým fázam by mal byť 80-10-10[19], 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.

2.3.1 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[26]).



Obr. 2: Príklad jednoduchej neurónovej siete

Zložitejším typom sú rekurentné neurónové siete. Už z názvu vyplýva, že jednu 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

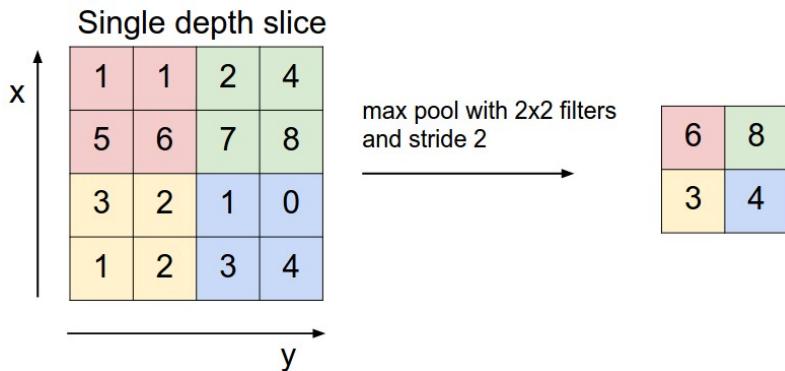
pri rozpoznávaní reči[22] alebo písma[10], či generovaní popisu k obrázkom[13], kedy však funguje v kombinácii s konvolučnou neurónovou sieťou (z angl. convolutional neural network). Tá je použitá na klasifikáciu obrázkov a rozoznávanie objektov, rekurentná sieť je použitá iba na výsledné generovanie 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.

2.3.2 Konvolučné neurónové siete

Základ tejto siete tvorí vstupná konvolučná vrstva³ s konvolučným filtrom, ten býva väčšinou malý (3×3 , 5×5). Vstup tejto vrstvy musí byť v tvare $m \times m \times r$, 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[15]), ktorá tieto filtre rozvzorkuje. To prebieha nezávisle na každom získanom filtrov z konvolučnej vrstvy. Rozvzorkovanie v podstate znamená, že sa zmení veľkosť filtrov použitím operácie MAX. Najbežnejšou formou spomínanej vrstvy je verzia s oknom (filtrom) o veľkosti 2×2 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 obrázku⁴ 3 je vidieť výsledok popisovaného postupu na jednoduchom príklade.

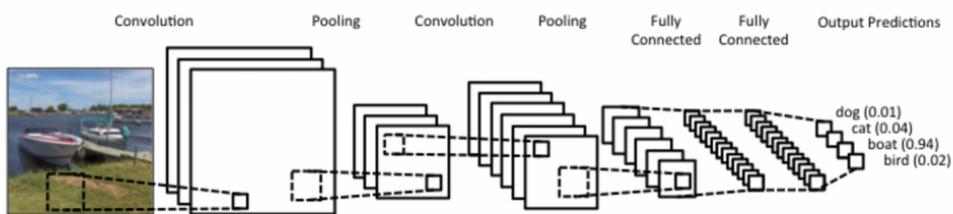
³<http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/>

⁴<http://cs231n.github.io/convolutional-networks/>



Obr. 3: Príklad vrstvy združovania, pri ktorom sa rozvzorkuje výstup z konvolučnej vrstvy o veľkosti 4×4 , filtrom 2×2 , 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 predošej 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 obrázku⁵ nižšie je jednoduchý náčrt vyššie popísanej konvolučnej neurónovej siete.



Obr. 4: Príklad konvolučnej neurónovej siete

Využíte tohto typu je v podstate všade, kde sa jedná o rozpoznávanie obrázkov. Či už ide o automatické vyznačenie tvári pre označenie na facebooku, autonómne

⁵<https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>

vozidlá, ktoré sa vedia riadiť sami (autopilot) alebo triedenie uhoriek na farmách v Japonsku⁶. Na tento konkrétny softvér bol použitý príklad kódu jednoduchej konvolučnej siete z tutoriálu⁷ 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 YUV⁸ úrovní a posunutý do neurónovej siete.

2.4 Určenie pútavých častí webových stránok pomocou strojového učenia

V nasledujúcej časti sú opísané niektoré z existujúcich riešení problému určenia pútavých častí grafického rozhrania webovej stránky za použitia strojového učenia. V podstate existujú 2 prístupy, z ktorých jeden vychádza z HTML kódu stránku a druhý len z jej obrázku, screenshot-u.

2.4.1 Riešenia na báze segmentácie stránok

Tento spôsob na začiatku vyhádza z HTML kódu stránky, ktorú na jeho základe rozdelí tzv. segmentovaním na hlavné časti (segmenty, bloky) stránky. Segmentácia používa buď segmentačné algoritmy na rozdelenie stránky do blokov podľa rôznych vlastností, alebo používa dátové štruktúry na reprezentáciu komponentov a elementov HTML dokumentu. Najpoužívanejšie prístupy k segmentácii sú:

- Segmentácia založená na DOM[7] (dokumentový objektový model, z angl. document object model):

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/r0.6.0/tutorials/mnist/pros/index.html>

⁸farebný priestor používaný vo video aplikáciách

HTML dokument je reprezentovaný ako DOM strom. Tagy môžu reprezentovať jeden blok stránky, napr. P-paragraf, TABLE-tabulka, atď. I keď tento strom dokáže veľmi presne reprezentovať štruktúru HTML dokumentu (stránky), často nie je dostatočne presný pri rozdelení sémanticky (vizuálne) rôznych blokov.

- Segmentácia založená na lokácii[21] (z angl. location-based segmentation):

Stránka sa rozdelí na 5 hlavných častí: stred, vľavo, vpravo, hore, dolu. Problémom je však, že nie vždy sa toto rozdelenie hodí na každú stránku a v prípade, že je stránka príliš dlhá (scroll-ovateľná) sa časti, ktoré boli na začiatku napr. dolu, posunú smerom hore a rozdelenie stránky sa tým zmení.

- VIPS[6] - segmentácia stránky založená na pohľade (z angl. vision-based page segmentation):

VIPS je v podstate kombinácia predchádzajúcich dvoch algoritmov. Delí stránku aj na základe farby, veľkosti blokov atď. V prvom rade vyberie vhodné uzly z DOM stromu a nájde medzi nimi separátory, ktoré naznačujú horizontálne a vertikálne línie webovej stránky. Na základe DOM stromu a separátorov sa vytvorí sémantický strom stránky, v ktorom je každý segment stránky reprezentovaný ako samostatný uzol v strome, koreňom stromu je samotná stránka. Spojitosť segmentácie stránky je kontrolovaná preddefinovaným stupňom koherencie (z angl. Predefined Degree of Coherence (PDoC[17])) – každému uzlu je daný určitý stupeň súvislosti, čo zabezpečuje, že VIPS dokáže efektívne držať obsah pokope, zatiaľ čo sémanticky odlišné bloky od seba.

Ako príklad riešenia používajúceho spomínanú segmentáciu je možné uviesť prácu výskumníkov z Microsoftu [21]. Tí použili VIPS algoritmus na segmentovanie a 5 ľudí, ktorí manuálne oštítkovali každý blok webovej stránky. Bloky sú očíslované od 1 po 4 podľa dôležitosti (1 – nepodstatné informácie ako reklamy, 4 – najdôležitejšia časť stránky, hlavný obsah). Z nich sa potom zostrojí model dôležitosti blokov, čo je vlastne funkcia mapovania každého bloku a jeho dôležitosti zobrazená ako:

$$\{ \text{block features} \} \rightarrow \text{block importance}$$

Vizualizácia mapovania dôležitosti blokov je zobrazená na časti stránky na obrázku 5.



Obr. 5: Vizualizácia mapovania dôležitosti blokov webových stránok[21]. 1 - najmenej dôležité informácie, 4 - najpodstatnejšia časť stránky

Na odhad dôležitosti blokov sa potom nahliada ako na regresný problém a na jeho riešenie je použitá neurónová sieť, v ktorej sú bloky reprezentované ako dvojica (x, y) , kde x je reprezentácia bloku a y je dôležitosť (berie sa ako reálne číslo). Typ siete je RBF (radial basis function) a vyhľadávacou technikou je štandardný gradient descent. Sieť zahrňa 3 vrstvy, každú s inou funkciou. Prvú (vstupnú) vrstvu tvoria zdrojové uzly, druhá je jediná skrytá vrstva a zabezpečuje nonlineárnu transformáciu zo vstupného priestoru do skrytého. Obsahuje RBF neuróny, ktoré kalkulujú vstup skrytej vrstvy kombináciou vážených vstupov a odhadov. Tretia (výstupná) vrstva dodáva dôležitosť blokov do reprezentácie blokov aplikovanej v prvej (vstupnej) vrstve.

Výhodou tohto riešenia je rozdelenie stránky podľa logických blokov, nevýhodu vidíme v začiatocnom ohodnotení blokov ľuďmi podľa dôležitosti, pretože mi to príde príliš individuálne. Tiež rozdelenie stránky do stromu s jednotlivými blokmi a časťami je v závislosti od stránky dosť pamäťovo náročná operácia.

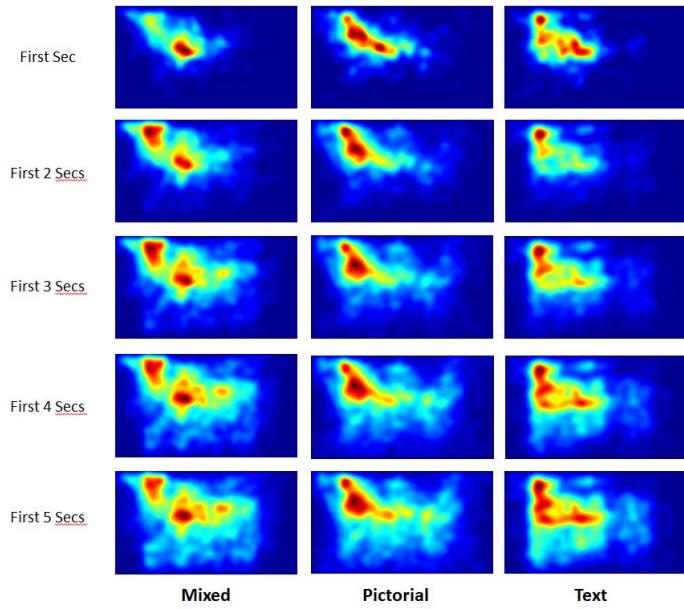
2.4.2 Riešenia vychádzajúce z obrázku stránky

Tento typ riešenia používa k určeniu dôležitých (pútavých) častí stránky jej screenshot. Autormi modelu sú Chengyao Shen a Qi Zhao[23]. Tí na začiatku

rozdelili ich dataset do niekoľkých kategórií podľa obsahu stránky, t. j. ilustrované (prevládajú obrázky), textové (prevažne blogy, články, atď.) a mix predoších dvoch kategórií. V každej z nich bolo približne 50 obrázkov s rozlíšením 1360x768. Na obrázky web stránok potom pozeralo 11 subjektov a ich sekvencie pohľadov boli zaznamenané pomocou MATLABU s Psychtoolbox[4] a systémom Eyelink 1000.

Na predikciu pohľadov[24] na web stránky bola použitá metóda MKL (Multiple Kernel Learning - kombinuje viacero jadier SVM⁹ miesto jedného). MKL bolo trénované ako binárny regresný problém, z datasetu bolo použitých 119 obrázkov na trénovanie, zvyšných 30 bolo použitých na finálne testovanie. Predikovaný bol vektor pohľadov (fixácií) na obrázok. Vyhodnotením datasetu bolo zistené, že človeka mimo iné upúta pri pohľade na web stránku v prvom rade ľudská tvár, oči a celkovo horná časť tela, či neprimerane veľké logo. Taktiež sa dá predpovedať, že človek sa začne pozerať v prvom rade najprv do strednej oblasti a oblasti viac vľavo hore od stredu. Tieto zistenia dokázali veľmi dobre zpracovať aj do ich modelu. Na obrázku nižšie je možné vidieť vizualizáciu pohľadu na rôzne typy stránok v prvých 5 sekundách formou teplotných máp.

⁹Support Vector Machine - modely učenia s príslušnými algoritmami učenia, ktoré analyzujú dátá použité pre klasifikáciu a regresnú analýzu[5]



Obr. 6: Vizualizácia pohľadov v prvých 5 sekundách. Vľavo mix stránok, v strede prevažne ilustrované stránky a vpravo textové.[23]

Za výhodu tohto riešenia oproti predošlému pokladám hlavne vygenerovanie heat mapy, ktorá vypovie o dôležitých častiach stránky ďaleko viac ako očíslované HTML bloky. Taktiež zistenia, že človek si ako prvé všíma napríklad ľudské tváre, sú pre dizajnéra web stránky určite velké plus oproti dôležitosti informácií na základe ich umiestnenia.

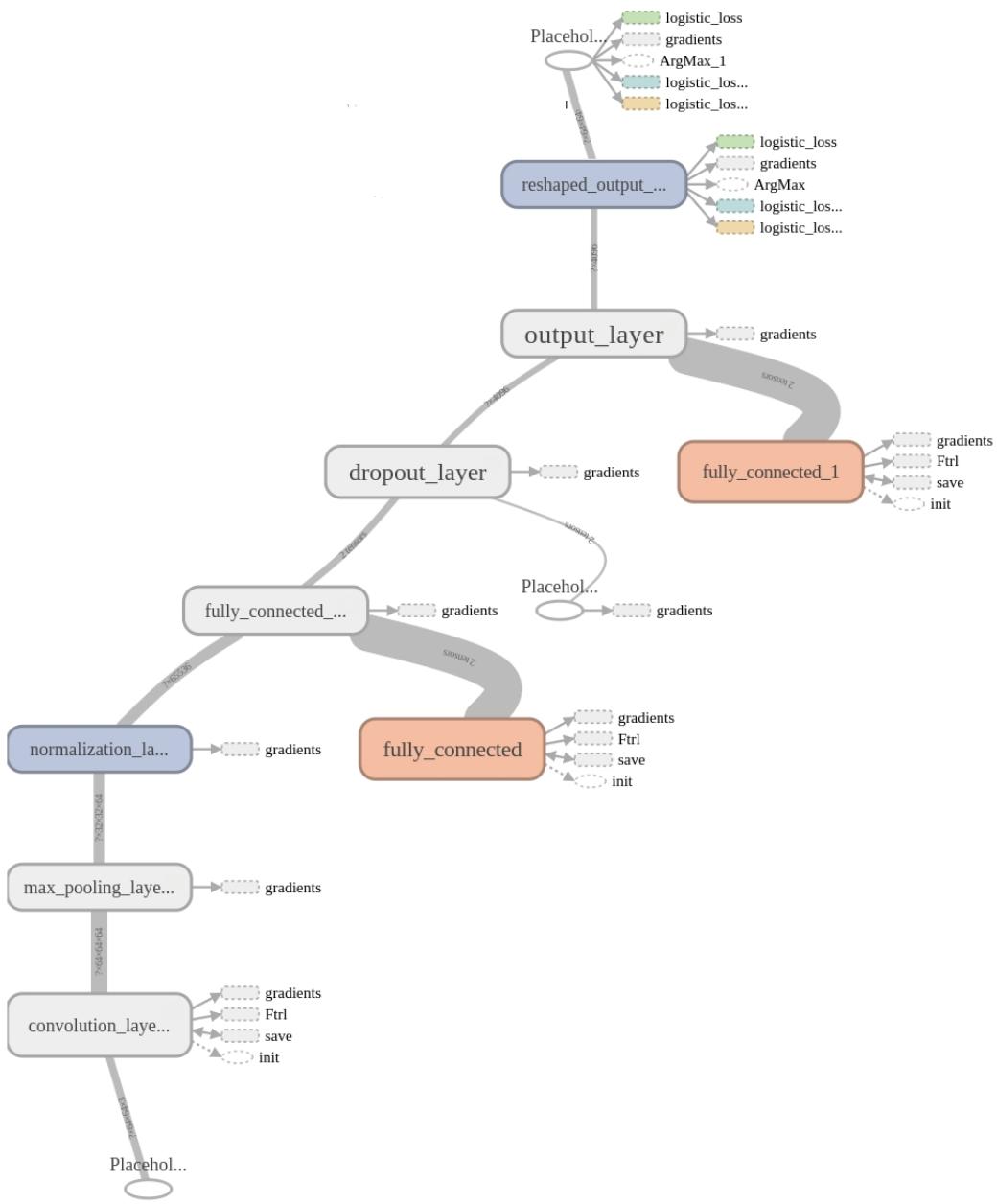
3 Návrh

V nasledujúcich riadkoch tejto kapitoly je rozobratý postupný návrh neurónovej siete určenej k riešeniu zadaného problému, spolu s dvomi typmi datasetov, s ktorými sme pracovali, v časti 3.2.

3.1 Návrh neurónovej siete

K predikcii pohľadov na webové stránky je vhodné použiť konvolučnú neurónovú sieť, keďže webstránky sú vo forme obrázkov. Neurónová sieť bude predikovať priamo výslednú teplotnú mapu pohľadov (mapu výraznosti). Sieť by mala pozostávať z konvolučnej a združovacej vrstvy (vrstiev) pre spracovanie obrázku, nasledovaných normalizačnou vrstvou, plne prepojenou vrstvou a vrstvou výpadku (z angl. dropout layer). Za nimi nasleduje už len výstupná vrstva. Ako aktivačnú funkciu sme vybrali sigmoid, nakoľko sa bude predikovať mapa výraznosti, t. j. v podstate pravdepodobnosť pohľadu. Celá architektúra je načrtnutá na schéme na obrázku 7 vytvorenéj pomocou nástroja TensorBoard¹⁰.

¹⁰https://www.tensorflow.org/get_started/summaries_and_tensorboard/



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

Na konvolučnej vrstve sa použije konvolučný filter o veľkosti 5×5 , ktorým sa prejde vstupný obrázok. Výstupom z nej budú mapy (obrázky) po aplikovaní filtra. Tieto dátá budú spracovávané vo vrstve združovania s použitím operácie

MAX, okna (filtra) s veľkosťou 2×2 a posunom s veľkosťou 2. Výstup všetkých filtrov je zlúčený do jednej širokej vrstvy a to normalizačnej. Z nej nasledujú prepojenia do plne prepojenej vrstvy, za ktorou nasleduje vrstva výpadku[25]. Tá prakticky obsahuje hodnotu (od 0 do 1) určujúcu kočko neurónov bude dočasne „odstránených“ z modelu spolu so všetkými spojeniami, ako vstupnými tak aj výstupnými. To by malo zabrániť pretrénovaniu siete. Za vrstvou výpadku nasleduje už len výstupná vrstva, ktorá je nakoniec ešte aj transformovaná do 2D matice reprezentujúcej predikovanú mapu výraznosti.

Aktivačná funkcia sigmoid bude použitá iba na prvej konvolučnej vrstve a na plne prepojenej vrstve. K trénovaniu sa použije FTRL optimizér spolu s náhodným generátorom dát, ktorý bude z datasetu určenému pre fázu trénovania náhodne vyberať dátá na učenie. Tým sa zabezpečí simulácia náhodnosti vstupných dát a teda sa sieť bude schopná lepšie učiť.

3.2 Dataset

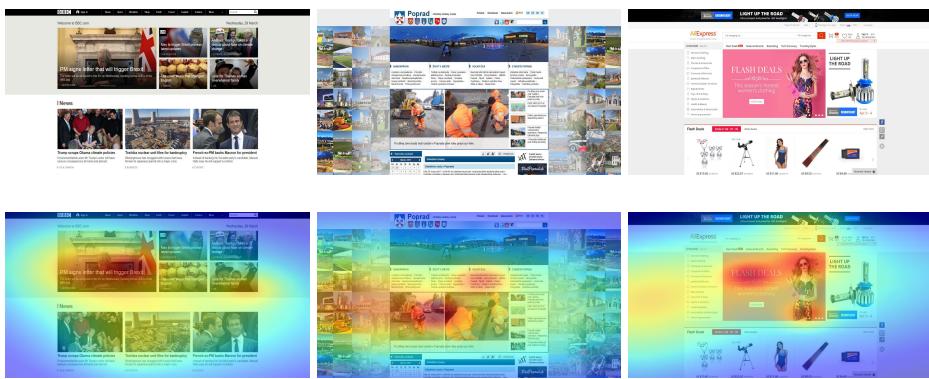
Počas práce na modeli neurónovej siete sme mali možnosť pracovať s dvomi typmi datasetov, obsahujú prakticky sériu obrázkov, fixácie pohľadov ľudí na ne a ich dĺžky:

- prvý dataset - bol nazbieraný ako časť experimentu bakalárskej práce[8] študentky Márie Dragúňovej. Bol zameraný na pozornosť „zhora nadol“ (top-down), keďže pri experimente, v ktorom bol zaznamenaný pohľad, boli fixácie na obrázku silno ovplyvnené tým, že participanti boli inštruovaní aby našli na obrázku web stránky niečo konkrétnie (napr. pri obrázku nemocnice mali nájsť kde sa dajú zobraziť informácie o CT a MRI). Práve to spôsobilo, že neurónová sieť v podstate nebola schopná naučiť sa nič potrebné z týchto dát. Jeho vizualizácia nasleduje na obrázku 9.



Obr. 8: Grafická vizualizácia prvého datasetu, hore obrázky webových stránok, dolu teplotné mapy k prislúchajúcim obrázkom zobrazené na nich)

- druhý dataset - vzhľadom na nevhodný predošlý dataset sme vykonali vlastný experiment s cieľom zozbierania dát pohľadov. Ten však bol podľa potreby zameraný na pozornosť „zdola nahor“. Jeho vizuálizácia aj s popísaným experimentom je nižšie.



Obr. 9: Grafická vizualizácia druhého datasetu, hore obrázky webových stránok, dolu teplotné mapy k prislúchajúcim obrázkom zobrazené na nich)

Experiment na získanie datasetu pre neurónovú sieť

Pripravený experiment pozostával zo screenshot-ov 50 rôznych webových stránok a jednej abstraktnej snímky v odtieňoch sivej. Participantom boli postupne všetky ukázané na 5 sekúnd úlohou bolo zapamatievanie si obsahu a rozloženia web stránky, nakoľko na konci experimentu im bola položená kontrolná otázka, či nasledujúci obrázok webovej stránky bol alebo nebol medzi prešlými 50 stránkami. Otázka na konci bola v podstate z nášho pohľadu irelevantná, jej cieľom však bola motivácia participantov k dôkladnému zapamätaniu si obsahu stránky, keďže za správnu odpoveď mohol participant získať ľubovoľnú odmenu. Medzi dvomi obrázkami bola na sekundu zobrazená už spomínaná abstraktná snímka v odtieňoch sivej na rozhodenie pozornosti.

Obrázky boli zobrazované na monitore s rozlíšením *1920x1200*, pohľad bol zaznamenávaný zariadením Tobi TX-300 s frekvenciou záznamu pohľadu 300Hz, čo zabezpečilo dostatok zaznamenaných fixácií za 5 sekúnd pohľadu na obrázok. Experimentu sa zúčastnilo 20 participantov, protokol z jeho vykonania je priložený v prílohe E.

4 Implementácia

Zdrojový kód môjho riešenia je napísaný v jazyku Python s použitím nasledujúcich hlavných knižníc:

- TensorFlow¹¹ - open-source knižnica pre strojové učenie. Použitá na vytvorenie modelu neurónovej siete a prácu s ňou.
- Matplotlib¹² - knižnica pre 2D vykreslovanie. V mojom riešení je použitá hlavne na výsledné zobrazenie (resp. uloženie) predikovaných máp výraznosti (teplotných máp) na konkrétnom obrázku webovej stránky.
- PIL (Python Image Library¹³) - knižnica pre prácu s obrázkami. Použitá je hlavne na načítanie obrázkov webových stránok a zmenu ich rozmerov podľa potreby.

Celá implementácia aj spolu s ukážkami zdrojových kódov funkcií je ešte podrobnejšie popísaná v prílohe A.

4.1 Spracovanie datasetu pre neurónovú sieť

Získaný dataset pozostával z 50 obrázkov a pohľadov 20 ľudí na ne. Najprv ho bolo nutné upraviť k trénovaniu neurónovej siete. Úprava pozostávala zo zmenšenia a zmeny rozlíšenia obrázkov web stránok pre neurónovú sieť na štvorec o veľkosti $a \times a$. To zabezpečuje funkcia ***resize_image (image, size)***, ktorá ako parametre dostane obrázok a veľkosť ($size = a$), na ktorú má byť obrázok prevedený.

Ďalej bola nutná normalizácia a vyfiltrovanie fixácií, k čomu slúži funkcia ***load_fixations (path)***. Ako argument dostane CSV súbor exportovaných dát z experimentu, ktorý má za úlohu spracovať. Súbor je spracovávaný sekvenčne po riadkoch, pri čom sú ignorované pre nás nepodstatné záznamy, ako napr. pohľady na medzisnímku pre rozhodenie pozornosti. Fixácie mimo obrázka sú rovnako preskočené a každá fixácia je znormálizovaná. Dáta (meno obrázka, fixácie a ich dĺžky) sú potom uložené pre participantov jednotlivovo.

¹¹<https://www.tensorflow.org/>

¹²<https://matplotlib.org/>

¹³<http://www.pythontware.com/products/pil/>

Pre neurónovú sieť je ešte nutné vypočítať mapy výraznosti (teplotné mapy) z vyfiltrovaných fixácií použitím Gaussovo rozdelenia, ktorého rovnica je ukázaná vo vzorci 9. Najprv však bolo nutné prepočítať fixáciu na konkrétny pixel na obrázku, pomocou nasledujúceho vzorca:

$$[x, y] = [fixation_x * a, fixation_y * a] \quad (8)$$

$[x, y]$ - bod v obrázku

a - veľkosť strany obrázka (keďže je štvorec)

$fixation_x$ - x-ová normalizovaná súradnica fixácie

$fixation_y$ - y-ová normalizovaná súradnica fixácie

Toto zabezpečilo rovnakú veľkosť obrázkov a prislúchajúcich máp výraznosti k nim. Hodnota Gaussovo rozdelenia je vypočítaná prakticky pre všetky body z fixácií.

$$f(x | \mu, \sigma^2) = \frac{1}{\sqrt{2\sigma^2\pi}} * e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (9)$$

x - aktuálny bod, pre ktorý je počítané Gaussovo rozdelenie

$(x - \mu)^2$ - vzdialenosť medzi bodom a fixáciou

σ^2 - dĺžka fixácie

Obe vyššie uvedené matematické operácie zobrazuje časť kódu na obrázku 10.

```
point_x = int(round(data[j][0] * 63)) # X coordinate of fixation
point_y = int(round(data[j][1] * 63)) # Y coordinate of fixation
variance = int(data[j][2])

if point_y > 63 or point_y < 0 \
    or point_x > 63 or point_x < 0: # view outside of image
    j += 1
    continue
sqrt_2_pi = 1 / math.sqrt(2 * math.pi * variance)
for y in range(0, 64):
    for x in range(0, 64):
        ni = abs((point_x - x) ** 2) + abs((point_y - y) ** 2) # distance between points
        heatmap[y][x] += sqrt_2_pi * math.e ** (-ni / (2 * variance)) # part of the counting
```

Obr. 10: Ukážka výpočtu heatmapy, kde premenná data obsahuje fixácie a ich dĺžky pre všetky obrázky, heatmap je už konkrétna mapa výraznosti pre obrázok web stránky

Kód je súčasťou funkcie *load_coordinates_for_image (file_with_coordinates, path)*, ktorej prvý parametrom je súbor s fixáciami participanta a druhým je adresár (cesta) kde sa majú uložiť vypočítané heatmapy. Heatmapy sú ukladané v tvare *heatmap_i_image_j*, kde *i* reprezentuje číslo participanta (začína od nula) a *j* reprezentuje číslo obrázka (od 0 do 49).

Takto pripravený dataset máp výraznosti pre neurónovú sieť je neskôr rozdelený v pomere 80-10-10 (trénovanie-validácia-testovanie).

4.2 Trénovanie neurónovej siete

Trénovanie neurónovej siete prebieha na už spomínaných 80% dát z datasetu, 10% je venovaných validácií. Najprv je pomocou funkcie *neural_network_model (data, keep_prob)* vytvorený model siete, ktorého schéma aj s popisom je v Návrhu v časti 3.1. Parameter *data* predstavuje vstup pre neurónovú sieť, v našom prípade obrázok, resp. obrázky. Parameter *keep_prob* určuje hodnotu na vrstve výpadku (z angl. dropout layer), tá je pri trénovaní 0.5, pri validácii a testovaní 1.0.

Po rozdelení datasetu, nastavení krížových entropií (z angl. cross entropy) a optimizéru je neurónová sieť trénovaná v 15 iteráciách po 30 epoch. V každej iterácii sa trénuje a validuje na celom dataseite k tomu určenom, ten je však vždy na začiatku náhodne zamiešaný (ako je vidieť v ukážke kódu na obrázku 11), neurónová sieť sa teda učí na náhodných dátach a nie vždy na úplne tých istých.

```
indexes = np.arange(len(images))
np.random.shuffle(indexes)

for index in indexes: # randomly mixes dataset
    randomize_images[i] = images[index]
    randomize_heatmaps[i] = heatmaps[index]
    i += 1
```

Obr. 11: Ukážka náhodného zamiešania datasetu pre trénovanie

Vstupom do neurónovej siete je teda séria obrázkov, konkrétnie o veľkosti 64×64 . Výstupom sú heatmapy (mapy výraznosti) rovnakej veľkosti.

Trénovanie končí buď po uplynutí všetkých epoch alebo v momente, kedy chyba predikcie na validačnom dataseite prestane klesať a začne rásť. To značí, že sieť sa už neučí, ale začína sa pretrénovať.

Po tomto bude nasleduje otestovanie neurónovej siete na zvyšných 10% datasetu a uloženie celého natrénovaného modelu.

4.3 Využívanie natrénovaného modelu

Predikovať heatmapy k novým obrázkom pomocou natrénovaného modelu umožňuje funkcia `predict(model, directory_with_images, directory_for_saved_predictions, heatmap_type, metrics_set=None`, ktorá ako parametre dostane spomínaný model, adresár s obrázkami (na ich veľkosťi nezáleží), adresár kam sa majú uložiť výsledné predikcie a typ heatmapy, ktorá sa má zobraziť. Ako voliteľný parameter je zvolenie/nezvolenie výpočtu metrík ohodnocujúcich kvalitu predikcií vizuálnej pozornosti. Po predikcii máp výraznosti sa tak v závislosti od voliteľného parametra môžu volať funkcie na výpočet už spomínaných metrík, ktoré sú prevzaté z open-source-ového repozitára¹⁴. K zobrazeniu/uloženiu predikcií slúži funkcia `visualize_heatmap (heatmap, orig, path_for_saving, heatmap_type)`, tá má ako parametre heatmapu (teda našu predikciu), originálny obrázok, pre ktorý sa robila predikcia, cestu kam sa má uložiť výsledok a nakoniec typ zobrazovanej heatmapy.

¹⁴<https://github.com/herrlich10/saliency>

5 Experimenty

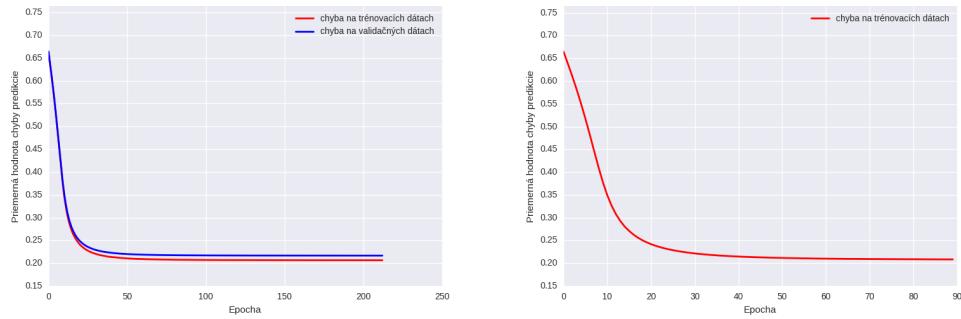
Počas zstrojovania neurónovej siete sme vykonali niekoľko experimentov, týkali sa hlavne konfigurácie siete, rôznych aktivačných funkcií, optimizérov a veľkosti datasetu pri trénovaní. Väčšie experimenty sú popísané v častiach 5.1 a 5.2, zistenia z tých menších sú zhrnuté nasledovne:

- najlepšie sa sieť učí, keď dostane naraz celý trénovací dataset, nie len jeho časti
- pridanie ďalších plne prepojených vrstiev predikcie zhoršilo
- pridanie viacerých konvolučných vrstiev nemalo prakticky žiadny vplyv na predikcie
- použitím štandardnej ReLU aktivačnej funkcie dosahovala chyba predikcie enormné hodnoty, bez ohľadu na rýchlosť učenia a použitý optimizér (Adam, Gradient descent, ...)
- aktivačná funkcia Sigmoid v kombinácii s Ftrl optimizérom, ktorého rýchlosť učenia bola 0.2, dávala najnižšiu chybu predikcií

Na základe vyššie uvedeného sme teda do ďalších experimentov pokračovali s modelom neurónovej siete, kde bola aktivačná funkcia Sigmoid, Ftrl optimizér, rýchlosť učenia 0.2 a rovnaký počet konvolučných a plne prepojených vrstiev ako v zobrazenom modeli v časti 3.1.

5.1 Model s validáciou vs. model bez validácie

Nakoľko máme dosť malý dataset, experimentovali sme s modelom, kde nebola použitá validácia a dátá pre ňu určené boli pridelené k trénovacím dátam. Výsledok však dosiahol iba zhoršenie priemernej chyby predikcie o 0.001. Tréning bez validácie končil v momente, keď chyba na tréningových dátach začala rásť, s validáciou končil až keď začala rásť na validačných dátach. To vyústilo vo viac ako 2x dlhší tréning, ako môžete vidieť na obrázku 12.

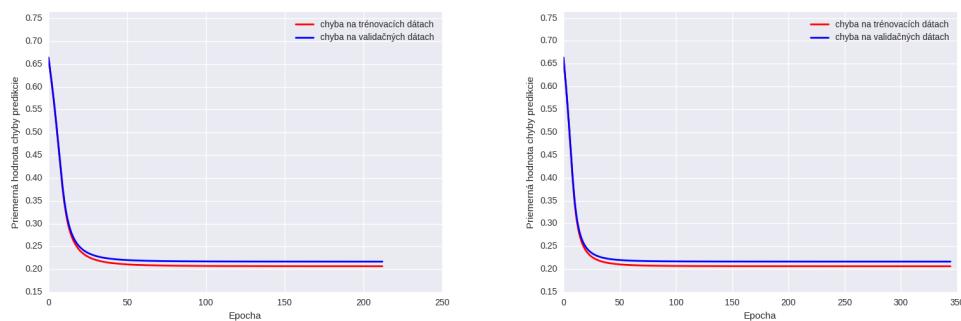


Obr. 12: Grafy poklesu chyby pri jednotlivých trénovaníach, vľavo trénovanie s validáciou, vpravo bez nej

Chyba na natrénovanom modeli bez validácia bola na testovacích dátach 0.2088, na trénovacích 0.2061. Na natrénovanom modeli s validáciou bola chyba na testovacích dátach 0.2078, na validačných 0.2166 a na trénovacích 0.2066. Takže zlepšenie bolo síce minimálne, ale stále badateľné. Hlavný dôvod, prečo validácia nemala až taký účinok, je malé množstvo dát.

5.2 Model s vrstvou výpadku vs. model bez vrstvy výpadku

Ďalšiu konfiguračnú zmenu, s ktorej implementáciou sme experimentovali, bolo pridanie vrstvy výpadku. Výsledok bol však len ten, že sieť sa dokázala naučiť to isté za rýchlejší čas, predikcie sa ale nezlepšili, ako je vidieť na nasledujúcich grafoch na obrázku 13.



Obr. 13: Grafy poklesu chyby pri jednotlivých trénovaníach, vľavo model s vrstvou výpadku, vpravo bez nej

6 Výsledky

V nasledujúcej kapitole sú rozobraté dosiahnuté výsledky. Na tie je nutné pozrieť sa z dvoch strán, najprv z pohľadu neurónovej siete a toho, ako dobre sa neurónová sieť dokázala naučiť predikovať na predložených dátach. Potom z pohľadu kvality predikcií a ich porovnania s inými riešeniami použitím metrík na evaluáciu modelov vizuálnej pozornosti.

6.1 Evaluácia modelu neurónovej siete

Na evaluáciu toho, ako dobre bola sieť schopná naučiť sa predikovať sme použili jednoduchý priemer chýb predikcií v jednotlivých bodoch heatmapy. Ten potom podľa očakávania klesal počas tréningu aj validácie štandardne, ako bolo ukázané na grafoch v časti 5.2. Jeho hodnota bola pri testovaní na testovacom datasete 0.2066. Vzhľadom na to, že pri trénovaní na 3/4 datasetu bola chyba 0.2511, predpokladáme, že ďalšie zlepšenie predikcií a zníženie chyby by bolo možné dosiahnuť väčším množstvom dát.

6.2 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[3], najpoužívanejšie sú:

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

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

- AUC - Oblast 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 binárному 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.

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

Z vyššie popísaných metrík sme k finálnej evaluácii a porovnaniu s inými riešeniami použili tri, a to CC, sAUC a NSS.

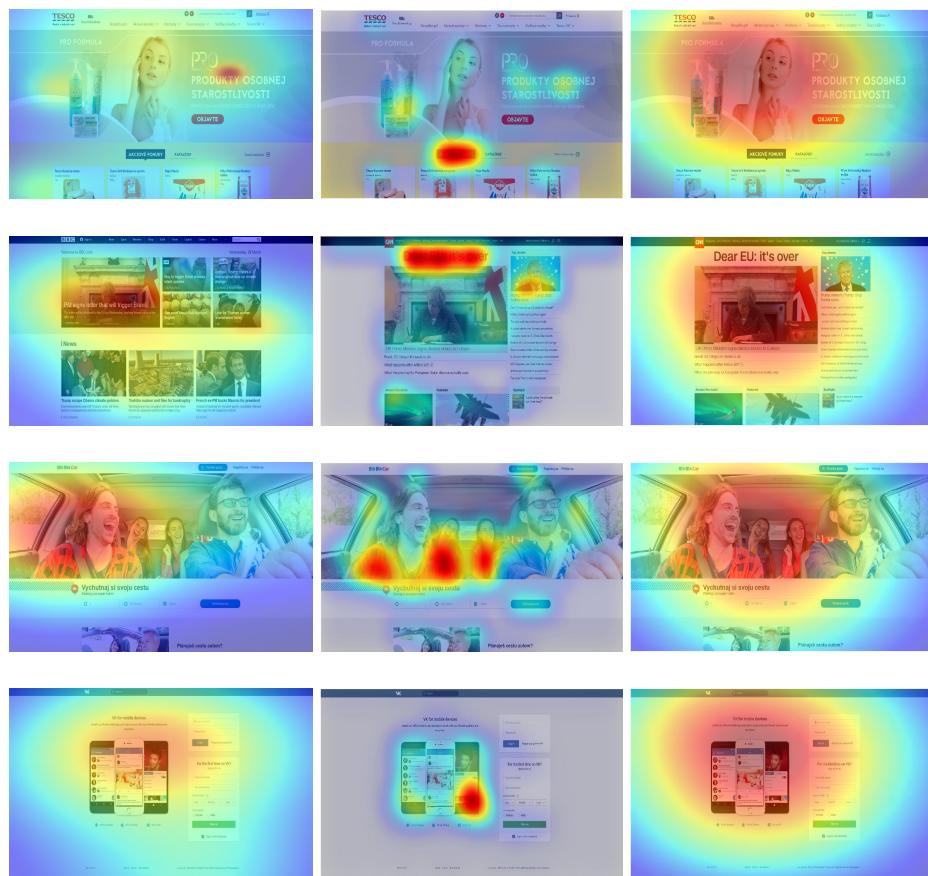
6.3 Porovnanie s inými riešeniami

Výsledné predikcie nášho riešenia sme porovnali s Itti-ho modelom vizuálnej pozornosti (popísaný v 2.2) pomocou vyššie uvedených metrík. Metriky boli počítané na nových testovacích dátach, ktoré naša neurónová sieť doteraz nevidela. Je však nutné spomenúť, že Itti-ho model nie je primárne zameraný na určovanie vizuálne pútavých častí grafických rozhraní web stránok, jedná sa skôr o všeobecný model vizuálnej pozornosti. Porovnanie a hodnoty metrík je možné nášť v tabuľke 1.

Tabuľka 1: Porovnanie hodnôt metrík pre Itti-ho a náš model

	Itti-ho model	Náš model
CC	0.4535	0.6620
sAUC	0.5785	0.7163
NSS	0.3309	0.6998

Z tabuľky je vidieť, že náš model bol podľa metrík schopný predikovať o niečo presnejšie mapy výraznosti ako Itti-ho model. Na obrázku 14 je zobrazené porovnanie predikcií formou vizualizácie.



Obr. 14: Grafická vizualizácia porovnania predikcií Itti-ho modelu s naším. Vľavo mapy výraznosti vypočítané z reálnych fixácií ľudí, v strede predikcia pomocou Itti-ho modelu, vpravo predikcia nášho modelu.

Na obrázku je možné pozorovať, že v niektorých situáciách je náš model

presnejší, v iných nie. Naša neurónová sieť sa naučila predikovať primárne do strednej hornej oblasti, čo však na väčšine typov stránok je aj reálne prvá oblasť, ktorú vnímame.

Najideálnejšie by bolo náš model porovnať s modelom od Chengyao Shen a Qi Zhao[23], popísaným v časti 2.4.2, ale ich dataset ani riešenie nie sú verejne dostupné. Hodnoty metrík preto nie sú vypočítané na tých istých dátach a porovnanie môže byť trochu zavádzajúce, prakticky ale náš model dosiahol hodnoty CC zhruba o 0.06 vyššie, sAUC je na tej istej úrovni a NSS má náš model o polovicu nižší, je to jediná metrika kde sme výrazne horší ako ich model.

7 Zhrnutie

Cieľom tejto práce bolo vytvorenie modelu schopného určovania pútavých častí grafických rozhraní webových stránok. Po preštudovaní zadania, témy a problémnej oblasti sme nadobudli značné znalosti a zistili sme nasledovné:

- vizuálna výraznosť (ktorá je len iným názvom pútavosti) sa delí v podstate podľa pozornosti na tzv. „zdola nahor“ (z angl. bottom-up) a „zhora nadol“ (z angl. top-down)
- existuje naozaj veľké množstvo už existujúcich modelov určovania vizuálnej pozornosti, od starších hierarchických modelov až po novšej vzorovo klasifikačné, využívajúce strojové učenie
- v súčasnosti je len veľmi málo modelov (takmer žiadne), ktoré využívajú k určeniu pútavosti grafických rozhraní neurónové siete

Po týchto zisteniach sme sa rozhodli použiť práve neurónovú sieť, ktorá by bola schopná predikovať pútavé časti iba z obrázka webovej stránky. K tomu sa výborne hodí práve konvolučná neurónová sieť, určená práve pre spracovanie obrazu. Po získaní datasetu zozbieraného vrámci experimentu k bakalárskej práci jednej študentky, sme začali s konštruovaním vhodnej architektúry neurónovej siete pre riešenie nášho problému. Dataset pozostával z obrázkov web stránok a pohľadov na ne. Rozhodli sme sa teda, že neurónová sieť bude predikovať priamo mapy výraznosti, na ktorých je jasne vidieť, čo je na stránke viac a čo menej pútavé. Problém bol však v tom, že dataset bol zameraný na pozornosť „zhora nadol“ (z angl. top-down), čo je pozornosť veľmi silno ovplyvnená nejakým cieľom, v našom prípade úlohou nájsť niečo na stránke. Neurónová sieť nebola schopná učiť sa na takýchto dátach, takže bolo nutné zohnať iný, lepší dataset, tento krát zameraný na pozornosť „zdola nahor“ (z angl. bottom-up), ktorú charakterizuje rýchlosť, nevedomosť a to, že je riadená čisto len pútavými stimulmi.

Na získanie takéhoto datasetu sme zostrojili vlastný experiment, ktorý obsahoval 50 obrázkov webových stránok rôzneho druhu. Každý bol zobrazený participantom experimentu na 5 sekúnd a ich úlohou bolo zapamätať si obsah a rozloženie

stránky. Úloha bola takto formulovaná z dôvodu, že keď si má človek stránku zapamätať tak ju len celú narýchlo nezoskenuje, ale sústredí sa práve na veci, ktoré ho zaujmú. Navyše vrámci motivácie bola na konci jednoduchá otázka, či nasledujúci obrázok bol alebo neboli medzi predošlými, za správnu odpoveď mohol participant získať ľubovoľnú odmenu. Medzi obrázkami web stránok bola ešte na 1 sekundu zobrazená abstraktná medzisnímka v odnieňoch šedej, na rozhodenie pozornosti. Experimentu sa zúčastnilo 20 participantov, čo nám v konečnom dôsledku dalo dokonca o niečo viac dát ako bolo v prvom datasete.

Po úprave týchto nových dát sme na nich následne učili neurónovú sieť. Experimentovali sme s rôznymi konfiguráciami, aktivačnými funkciemi či optimizérmi, ale ako najlepší sa ukázal model s jednou konvolučnou vrstvou, s jednou vrstvou združovania nasledovanou normalizačnou, plne prepojenou, výpadkovou a výstupnou vrstvou. Spomedzi aktivačných funkcií dávala najlepšie výsledky funkcia sigmoid na konvolučnej a plne prepojenej vrstve v spolupráci s Ftrl optimizérom ktorý mal rýchlosť učenia nastavenú na 0.2. Sieť v takejto konfigurácii s datasetom rozdeleným v pomere 80-10-10 (trénovanie-validácia-testovanie) bola schopná znížiť priemernú chybu predikcie na testovacích (doteraz nevidených) dátach na úroveň 0.2078.

Sieť sa bola schopná naučiť predikovať najpútavejšiu časť stránky vždy do stredu, čo sa môže na prvý pohľad javiť ako hlúposť. Treba si však uvedomiť, že stránky v dnešnej dobe sú ozaj stavané presne tak, že práve v strede sú najzaujímavejšie časti ako titulky článkov, či produkty za akciové ceny. To je naozaj presne to miesto, kam sa na stránke, bez toho aby sme na nej niečo hľadali, pozerá väčšina z nás. To koniec koncov z časti potvrdili aj porovnania vizualizácií predikovaných máp výraznosti oproti reálnym mapám.

Evaluácia takto natrénovanej siete prebehla metrikami vizuálnej pozornosti CC, sAUC a NSS, ktorých hodnoty sme porovnali s hodnotami metrík Itti-ho modelu na tých istých obrázkoch webových stránok. V porovnaní náš model dosiahol vyššie a lepšie hodnoty, ale je nutné upozorniť na to, že Itti-ho model nie je primárne stavaný na určovanie pútavých častí webových stránok. Model určený presne k tomu a trénovaný aj na prislúchajúcim datasete od Chengyao Shen a Qi Zhao[23], s ktorým by bolo najlepšie sa porovnať, bohužiaľ nie je verejne dostupný a ani nemá verejný dataset. Mohli sme tak porovnať jedine hodnoty metrík na roz-

ličných datasetoch, kde CC nám vyšlo o niečo vyššie, sAUC zhruba na rovnakej úrovni, iba NSS máme výrazne (takmer o polovicu) nižšie.

Celý vyššie popísaný model je naimplementový v jazyku Python za použitia knižnice TensorFlow pre strojové učenie, v našom prípade sa jedná o hlbokú neurónovú sieť. Vizualizácie máp výraznosti sú realizované pomocou knižnice Matplotlib, vďaka ktorej je umožnené vizualizovať aj heatmapy rozličných typov.

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, 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.
- [4] David H Brainard. The psychophysics toolbox. *Spatial vision*, 10:433–436, 1997.
- [5] Cortes C. and Vapnik V. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
- [6] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Vips: A vision-based page segmentation algorithm. 2003.
- [7] Soumen Chakrabarti. Integrating the document object model with hyperlinks for enhanced topic distillation and information extraction. In *Proceedings of the 10th international conference on World Wide Web*, pages 211–220. ACM, 2001.
- [8] Mária Dragúňová. Vyhodnocovanie používateľského zážitku analýzou pochádzu a emocií. Bachelor thesis, Bratislava: FIIT STU, 2016.
- [9] 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.

- [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] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer, 2009.
- [12] Laurent Itti. Visual salience. *Scholarpedia*, 2(9):3327, 2007.
- [13] 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.
- [14] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [15] Fei-Fei Li, Andrej Karpathy, and J Johnson. Cs231n: Convolutional neural networks for visual recognition, 2015.
- [16] 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.
- [17] Rupesh R Mehta, Pabitra Mitra, and Harish Karnick. Extracting semantic structure of web documents using content and visual information. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 928–929. ACM, 2005.
- [18] Patrik Polatsek. Saliency maps. Prezentácia, 2015.
- [19] 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.

- [20] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.
- [21] Ji-Rong Wen Ruihua Song, Haifeng Liu and Wei-Ying Ma. Learning block importance models for web pages. In *Proceedings of the 13th international conference on World Wide Web (WWW '04)*, pages 203–211, New York, USA, 2004.
- [22] 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.
- [23] Chengyao Shen, Xun Huang, and Qi Zhao. Predicting eye fixations on webpage with an ensemble of early features and high-level representations from deep network. *IEEE Transactions on Multimedia*, 17(11):2084–2093, 2015.
- [24] Chengyao Shen and Qi Zhao. *Webpage Saliency*, pages 33–46. Springer International Publishing, Cham, 2014.
- [25] 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.
- [26] Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39:43–62, 1997.
- [27] 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.

A Technická dokumentácia

V nasledujúcej časti sa nachádza technická dokumentácia, spolu s doteraz použitými technológiami. Jednotlivo sú popísané najdôležitejšie použité knižnice a funkciuálnita rozdelená v procedúrach.

A.1 Hardvér

Trénovanie modelu neurónovej siete prebiehalo na fakultnom serveri s grafickej kartou Nvidia GeForce GTX 980 Ti 6gb a RAM pamäťou o veľkosti 24gb, avšak samotné riešenie nie je viazané na určitý typ hardvéru, stačí len mať dostatok pamäte na vytvorenie modelu a spracovanie datasetu.

A.2 Použité knižnice

A.2.1 TensorFlow

TensorFlow¹⁵ 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++ a Python).

Flexibilná architektúra umožňuje vykonávať výpočty na CPU alebo GPU (ne-pomerne 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.

Centrálnou jednotkou v TensorFlow-e je tenzor. Ten pozostáva z hodnôt sformovaných v pole akejkoľvek veľkosti či dimenzie. Jadro TensorFlow-u tvorí výpočtový graf, čo je vlastne séria TensorFlow operácií, ktoré sú do grafu pridané ako uzly.

Klientske programy interagujú s TensorFlow-om vytvorením tzv. *Session*. Jej primárne podporovanou operáciou je *Run*, ktorá v podstate funguje ako *init*, keďže

¹⁵<https://www.tensorflow.org/>

do jej zavolania sú hodnoty všetkých premenných TensorFlow-u neinicializované.

Pre potreby strojového učenia je dostupné značné množstvo funkcií od nastavenia optimizérov, cez funkcie počítajúce chybu či presnosť až po tie zabezpečujúce samotné učenie.

A.2.2 Matplotlib

Ďalšia open-source knižnica¹⁶, pre zmenu určená pre 2D vykreslovanie nelen rôznych grafov. Poskytuje obrovské množstvo možností vizualizácií dát, ktoré ocení nielen hŕstka vedcov. Jeho základnou súčasťou je modul pyplot, ktorý prakticky implementuje vykresľovaciú funkcionality Matlabu.

A.3 Zdrojové kódy najdôležitejších funkcií

A.3.1 Funkcia load_fixations

Funkcia dostane ako parameter súbor, ktorý je exportom dát z experimentu. Tento roztriedi podľa participantov a vyberie z neho fixácie a ich dĺžky pre jednotlivé obrázky.

```
def load_fixations(path): # loads all recordings of fixations from one
    # export of experiment
    file = open(path, 'r')
    name = file.readlines()[1].rsplit(';', 13)[0] # gets name of first
    file = open(path, 'r') # recording
    fixations_y = [] # data neccesary for us - fixations and durations
    fixations_x = []
    durations = []
    images_name = []
    for line in file.readlines()[1:]:
        line_splited = line.rsplit(';', 13)
        if line_splited[3] == "grayscale.jpg" or \
            line_splited[3] == "FINAL.JPG":
            continue # skips grayscale and last image of experiment -
                      # it's not part of dataset for NN

        if name != line_splited[0]: # new recording, saving fixations
            save_sorted_by_participants(name, np.array(images_name),
                                         np.array(fixations_x, dtype=float),
                                         np.array(fixations_y, dtype=float),
                                         np.array(durations, dtype=float))
```

¹⁶<https://matplotlib.org/>

```

name = line_splited[0]
fixations_y = [] # empties variables
fixations_x = []
durations = []
images_name = []

Y = int(line_splited[12].rsplit('\n', 1)[0]) # fixation outside
# of image
if Y < 45 or Y > 1130:
    continue

fixations_y.append(Y / 1200.0) # normalization of fixation
fixations_x.append(int(line_splited[11]) / 1920.0)
durations.append(line_splited[10])
images_name.append(line_splited[3])
# saving last recording
save_sorted_by_participants(name, np.array(images_name),
                            np.array(fixations_x, dtype=float),
                            np.array(fixations_y, dtype=float),
                            np.array(durations, dtype=float))

```

A.3.2 Funkcia pre výpočet máp výraznosti

Táto funkcia dostane súbor fixácií jedného participanta a cestu, kam má uložiť mapy výraznosti - heatmapy. Tie postupne vypočítá jednotlivo pre každý obrázok.

```

def load_coordinates_for_image(file_with_coordinates , path):
    participant = file_with_coordinates[::-1]
    participant = int(participant[4:6][::-1]) # gets participant number
    image_names = np.loadtxt(file_with_coordinates ,
                            usecols=(0,) ,
                            delimiter=',',
                            dtype=str) # gets names of images
    # gets fixations and durations for images
    data = np.loadtxt(file_with_coordinates , usecols=(1, 2, 3),
                      delimiter=',')
    current_image = image_names[0]
    i = 0
    j = 0
    heatmap = np.zeros([64, 64])
    for name in image_names:
        # print name
        if current_image != name: # new image, heatmap for old will be saved
            print ("saving: " + str(current_image))
            save_heatmaps(participant - 2, heatmap, i, path + "/heatmaps/")
            current_image = name
        i += 1
        heatmap = np.zeros([64, 64])

```

```

point_x = int(round(data[j][0] * 63)) # X coordinate of fixation
point_y = int(round(data[j][1] * 63)) # Y coordinate of fixation
variance = int(data[j][2])

if point_y > 63 or point_y < 0 \
    or point_x > 63 or point_x < 0: # view outside of image
    j += 1
    continue
sqrt_2_pi = 1 / math.sqrt(2 * math.pi * variance)
# computing Gaussian distribution
for y in range(0, 64):
    for x in range(0, 64):
        # distance between points
        ni = abs((point_x - x) ** 2) + abs((point_y - y) ** 2)
        # part of the counting
        heatmap[y][x] += sqrt_2_pi * math.e ** (-ni / (2 * variance))

j += 1

print ("saving: " + str(current_image))
save_heatmaps(participant - 2, heatmap, i, path + "/heatmaps/")

```

A.4 Funkcia na vytvorenie modelu neurónovej siete

Jej účelom je vytvorenie modelu so špecifikovanými vrstvami. Prvý parameter určuje vstup do siete (séria obrázkov), druhý hodnotu na vrstve výpadku. Na konvolučnej vrstve je 64 filtrov, okno o veľkosti 5x5 s krokom 1. Vrstva združovania obsahuje okno 2x2, v normalizačnej vrstve sú všetky filtre po aplikovaní maxpoolingu spojené do jednej vrstvy. Plne prepojená vrstva spolu s konvolučnou majú aktivačnú funkciu Sigmoid. Výstupná vrstva je nakonci zmenená na 2D maticu reprezentujúcu heatmapu.

```

def neural_network_model(data, keep_prob):
    with tf.name_scope('convolution_layer_1'):
        w_conv = tf.Variable(tf.truncated_normal([5, 5, 3, 64], stddev=0.1))
        b_conv = tf.Variable(tf.zeros(shape=[64]))

        x_image = tf.reshape(data, [-1, 64, 64, 3])
        conv = tf.nn.sigmoid(tf.nn.conv2d(x_image, w_conv, strides=[1, 1, 1, 1],
                                         padding='SAME') + b_conv)
    print_shape(conv)

    with tf.name_scope('max_pooling_layer_1'):
        pool = tf.nn.max_pool(conv, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1],

```

```

        padding='SAME')
print_shape(pool)

with tf.name_scope('normalization_layer'):
    #norm = tf.reshape(pool2, [-1, 16 * 16 * 128])
    norm = tf.reshape(pool, [-1, 32 * 32 * 64])
print_shape(norm)

with tf.name_scope('fully_connected_layer_1'):
    l1 = fully_connected(norm, 64 * 64, activation_fn=tf.nn.sigmoid)
print_shape(l1)

#with tf.name_scope('dropout_layer'):
#    drop = tf.nn.dropout(l1, keep_prob)

with tf.name_scope('output_layer'):
    output = fully_connected(l1, 64 * 64, activation_fn=None)

with tf.name_scope('reshaped_output_layer'):
    output = tf.reshape(output, [-1, 64, 64])
print_shape(output)

return output

```

A.5 Funkcia pre trénovanie modelu neurónovej siete

Účelom tejto funkcie je natrénovanie modelu, jeho otestovanie a uloženie. Ako argumenty dostane cestu k datasetu a názov, pod ktorým má byť model uložený.

```

def train_neural_network(path, model):
    # loading dataset
    images = data_part.load_images(path + "/rsz_images_64x64/", 50, 64)
    images = np.array(images)
    images = clone_images(images, 20) # clones images for heatmaps
    heatmaps = load_all_heatmaps(20, path)

    print ("loaded images for neural network: " + str(images.shape))
    print ("loaded heatmaps for neural network: " + str(heatmaps.shape))

    # normalization of heatmaps
    if max(heatmaps.flatten()) > 1:
        heatmaps /= max(heatmaps.flatten())

    # splitting dataset

    # test data
    test_images = images[900:1000]

```

```

test_heatmaps = heatmaps[900:1000]
# validation data
valid_images = images[800:900]
valid_heatmaps = heatmaps[800:900]
# training data
images = images[0:800]
heatmaps = heatmaps[0:800]

graph = tf.Graph()

with graph.as_default():

    # input to NN - logits
    x = tf.placeholder('float', shape=[None, 64, 64, 3])
    # expected output - labels
    y = tf.placeholder('float', shape=[None, 64, 64])
    # value on dropout layer
    keep_prob = tf.placeholder(tf.float32)

    # creates model of neural network
    prediction = neural_network.model(x, keep_prob)

    # setting up parameters of model
    cross_entropy = tf.reduce_mean(
        tf.nn.sigmoid_cross_entropy_with_logits(logits=prediction, labels=y)
    )
    train_step = tf.train.FtrlOptimizer(0.2).minimize(cross_entropy)

    validation_cross_entropy = tf.reduce_mean(
        tf.nn.sigmoid_cross_entropy_with_logits(logits=prediction, labels=y)
    )
    test_cross_entropy = tf.reduce_mean(
        tf.nn.sigmoid_cross_entropy_with_logits(logits=prediction, labels=y)
    )

    ...
    config = tf.ConfigProto(
        device_count = {'GPU': 0}
    )
    ...

    # add variables to log file
    tf.summary.scalar("cross_entropy", cross_entropy)
    tf.summary.scalar("validation_cross_entropy", cross_entropy)
    summary = tf.summary.merge_all()

    saver = tf.train.Saver()

    with tf.Session(graph=graph) as sess:

```

```

sess.run(tf.global_variables_initializer())

# set up directories for log files
summary_writer = tf.summary.FileWriter(path + "logs/", sess.graph)
validation_writer = tf.summary.FileWriter(path + "logs/val/",
                                         sess.graph)

end = False

iteration_count = 0
for i in range(0, 15): # 15 iterations of training on whole
    # dataset
    iterations = 30 # 30 epoch of each iteration

    # randomly shuffles given dataset
    training_dataset = make_dataset_for_epoch(images, heatmaps)
    validation_dataset = make_dataset_for_epoch(valid_images,
                                                valid_heatmaps)

    # sets up variables for feed_dict
    train_data = {
        x: training_dataset[0],
        y: training_dataset[1],
        keep_prob: 0.5
    }
    val_data = {
        x: validation_dataset[0],
        y: validation_dataset[1],
        keep_prob: 1.0
    }

    print ("start training on series: " + str(i))
    iteration_loss = 100000

    for count in range(0, iterations): # start training
        _, c = sess.run([train_step, cross_entropy],
                       feed_dict=train_data)

        if math.isnan(c) == True: # too big loss
            print ("died at: " + str(count))
            return 0
            break

        print ("current loss: ", c)
        iteration_count += 1
        # writing trainig loss to log file
        summary_str = sess.run(summary, feed_dict=train_data)
        summary_writer.add_summary(summary_str, iteration_count)
        summary_writer.flush()

```

```

# validate data
val = sess.run(validation_cross_entropy, feed_dict=val_data)
print ("validation loss: ", val)
# writing validation loss to log file
summary_str = sess.run(summary, feed_dict=val_data)
validation_writer.add_summary(summary_str, iteration_count)
validation_writer.flush()

# if validation loss starts rising, stop training
if iteration_loss < val:
    print ("end of trainig, loss starts rising")

    end = True
    break
iteration_loss = val

if end:
    break

# test model on test data
test_data = {x: test_images, y: test_heatmaps, keep_prob: 1.0}
test_loss = sess.run(test_cross_entropy, feed_dict=test_data)
print('test loss: ' + str(test_loss))

# saving model
save_path = saver.save(sess, model)
print ("saved in: %s" % save_path)

```

A.6 Funkcia slúžiaca k predikciám na natrénovanom modeli

Jej prvým parametrom je natrénovaný model, druhým adresár s obrázkami určenými pre neurónovú sieť, tretím adresár, kam sa majú predikcie uložiť. Štvrtý parameter *heatmap_type* určuje výsledný typ zobrazenej heatmapy, ide v podstate o rôzne farebné modely dostupné v knižnici Matplotlib¹⁷. Posledný parameter *metrics* nie je povinný, je voliteľný. V prípade, že je zadaný (číslo 1) sú vypočítané metriky vykonaných predikcií.

```

def predict(model, directory_with_images,
           directory_for_saved_predictions, heatmap_type,
           metrics_set=None):
    graph = tf.Graph()
    with graph.as_default():

```

¹⁷https://matplotlib.org/examples/color/colormaps_reference.html

```

x = tf.placeholder('float', shape=[None, 64, 64, 3])
y = tf.placeholder('float', shape=[None, 64, 64])
keep_prob = tf.placeholder(tf.float32)

# creates model of neural network
prediction = neural_network_model(x, keep_prob)
cross_entropy = tf.reduce_mean(
    tf.nn.sigmoid_cross_entropy_with_logits(logits=prediction, labels=y))
)

"""

config = tf.ConfigProto(
    device_count = {'GPU': 0}
)
"""

# load names of all images in directory
files = os.listdir(directory_with_images)
images_for_prediction = []
images_original = []

for filename in files: # loads images
    images_original.append(directory_with_images + filename)
    images_for_prediction.append(data_part.load_images_for_prediction(
        directory_with_images + filename)[0])

saver = tf.train.Saver()

with tf.Session(graph=graph) as sess:
    saver.restore(sess, model) # restores saved model

predicted_heatmaps = make_prediction(prediction,
                                       {x: images_for_prediction,
                                        keep_prob: 1.0})

if metrics_set == 1:
    directory = input(
        "enter directory with original heatmaps and fixations:\n"
    )
    # computes metrics
    count_metrics(predicted_heatmaps, len(images_original),
                  directory)

for map, img in zip(predicted_heatmaps, images_original):
    print ("working on: " + str(img.rsplit('/', 1)[1]))
    # saving predicted heatmaps on image
    data_part.vizualize_heatmap(map, img,
                                directory_for_saved_predictions,
                                heatmap_type)

```


B Používateľská príručka

K používaniu programu, resp. modelu popísanému v bakalárskej práci, je nutné mať nainštalovaný jazyk Python a knižnice TensorFlow, Matplotlib, PIL, Skimage, SciPy, NumPy a používať operačný systém Linux, nakoľko väčšina ciest k súborom je viac či menej napevno zapísaná v zdrojovom kóde. Ten je rozdelený do viacerých súborov, každý je určený na niečo iné. Pre spustenie rôznej funkcionality je nutné spustiť rôzne súbory (všetky sa spúšťajú ako štandardné skripty jazyka Python) s prislúchajúcimi argumentami:

- **data_preprocessing.py** - zabezpečuje roztriedenie participantov z experimentov a filtráciu fixácií s ich dĺžkami. Spúšťa sa s jedným argumentom, ktorým je súbor exportu dát z experimentov vo formáte CSV, napr.: *python data_preprocessing.py /home/beky/dataset/BekaBP_Data_Export_1.csv*
- **data_to_heatmaps.py** - vytvorí z fixácií k obrázkom ich heatmapy a uloží ich. Spúšťa sa s 3 argumentmi, prvým je cesta k roztriedeným súborom partipantov s ich fixáciami. Tie musia byť uložené v tvare *Rec X.txt*, kde *X* je číslo súboru. Ďalšími dvomi argumentmi sú čísla *X*, najprv číslo prvého súboru, potom posledného. Ukážka: *python data_to_heatmaps.py /home/beky/dataset/processed_participants/ 5 13*
- **model_train_64x64.py** - súbor s modelom neurónovej siete a funkcionalitou k jej trénovaniu. Prvým argumentom je cesta k priečinku s datasetom, ktorý musí obsahovať priečinok *heatmaps/* s vypočítanými heatmapami k obrázkom a prečinok *rsz_images_64x64/* so samotnými obrázkami. Obrázky musia byť o veľkosti *64x64*, heatmapy rovnako. Tiež musia byť uložené v tvare *X.jpg*, kde *X* je číslo obrázka (od 0), podobne ako heatmapy, ktoré musia byť v tvare *heatmap_i_image_j.txt*, kde *i* reprezentuje číslo participanta (začína od nula) a *j* reprezentuje číslo obrázka. Druhým argumentom je názov, pod ktorým bude uložený model neurónovej siete. Príklad: *python model_train_64x64.py /home/beky/dataset/dataset_for_nn/ model_nn_v2.0*
- **model_predictions_64x64.py** - súbor so zdrojovým kódom umožňujúcim využívanie natrénovaného modelu. Argumentov je nie hned' niekoľko. Pr-

vým je natrénovaný model, druhým je adresár s obrázkami určenými pre neurónovú sieť, tretím adresár, kam sa majú predikcie uložiť. Štvrtý parameter určuje výsledný typ zobrazenej heatmapy, ide v podstate o rôzne farebné modely dostupné v knižnici Matplotlib¹⁸. Posledný parameter nie je povinný, je voliteľný. Určuje či sa majú počítať metriky predikcií, ktoré sa počítajú ak je jeho hodnota 1. Príklad: *python model_predictions_64x64.py /home/beky/dataset/model/model_nn_v2.0 /home/beky/dataset/images/ /home/beky/dataset/predictions/ jet*

¹⁸https://matplotlib.org/examples/color/colormaps_reference.html

C Plán práce na riešení projektu

Táto príloha obsahuje plán práce na riešení projektu v jednotlivých semestroch a počas skúškového obdobia:

- **Prvý semester:**

- 5. týždeň - získanie dát z UX-labu
- 6. - 7. týždeň - spracovanie dát, naštudovanie existujúcich riešení
- 8. týždeň - návrh
- 9. týždeň - naštudovanie konvolučných neurónových sietí
- 10. týždeň - osnova bakalárky so zdrojmi
- 11. týždeň - analýza
- 12. týždeň - konzultovanie prvej časti BP

- **Skúškové obdobie:**

- refaktORIZÁCIA a „upratanie“ doterajšej práce (hlavne kódu)
- implementácia návrhu
- mať funkčný prototyp schopný predikcie
- otestovať rôzne možnosti učenia (meniť veľkosti batch-ov, rýchlosť,...)
- v januári prekonzulovať dosiahnuté výsledky spolu s ďalším postupom

- **Druhý semester:**

- 1. týždeň - článok na IIT. SRC
- refaktORIZÁCIA a optimalizácia riešenia
- postupné dopisovanie práce a dokumentácie
- úprava článku na IIT. SRC
- príprava a vykonanie experimentu na získanie nového datasetu
- zapracovanie nového datasetu

- príprava posteru na IIT. SRC
- 27.4.2017 - IIT. SRC
- experimentovanie
- finalizácia

Plán som naplnil, i keď jeho časové rozpoloženie bolo najmä v druhom semestri trochu chaotickejšieho charakteru. To bolo spôsobené hlavne zistením, že vtedajší dataset bol prakticky nepoužiteľný a bolo nutné pripraviť a vykonať experiment na získanie lepšieho.

D Príspevok na konferenciu IIT. SRC 2017

Táto príloha obsahuje príspevok publikovaný na konferencii IIT. SRC 2017 v časti Computer Science and Artificial Intelligence.

Determination of the eye-catching parts in graphical interfaces

in Proceedings of IIT.SRC 2017

Patrik BEKA*

*Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 2, 842 16 Bratislava, Slovakia
patrik.beka@gmail.com*

Abstract. Eye-catching and graphically attractive design is an extremely important part of every website, although it might not appear to you at first sight. Our main purpose is to develop a neural network, capable of learning from given data and eventually predicting eye-catching and important parts of the websites. We selected method based on the convolution neural network, which is able to make predictions from the given images of the web pages. Final prediction is shown in the form of the heatmap, which determines the most engaging parts of the given web pages.

1 Introduction

Design of the good graphical interface for web pages is not an easy thing to do at all. Nowadays, when people are looking for informations almost only via internet, the good design has even bigger importance. Whether the visitor of page is interested in the content or not, is not as important as his attraction to the design of web page and the fundamental availability of informations. Therefore, design of the web page is certainly one of the most important preconditions for the future comeback.

Our task is to develop the system, that is capable of determining the eye-catching parts of web pages just from the web pages image (screenshot). In this thesis, we try to describe our method of determining important parts of the web pages using neural network, specifically convolution network, along with other similar solutions.

2 Related work

Convolution neural networks [7] [2] are widely used almost in every domain, where is the need for image recognition. Whether it is automatic face detection on Facebook, autonomous cars capable of self driving using autopilot by Tesla, Google, or software for sorting and classification of cucumbers in the Japanese farm¹. Prototype of self driving car by Google called Dave-2 [1] has a model of convolution neural network, which processes frames from cameras placed on the car. These cameras are recording environment in 10 frames per second and after some modifications, these frames are passed to the neural network. This network consists of 9 layers, one normalization layer, 5 convolution and 4 fully-connected layers.

Related work about judging the importance of different parts in a web page can be classified into two main approaches.

First approach

First approach is using HTML code of the web page to divide the whole web page to the main parts with operation called segmentation. Segmentation uses either segmentation algorithms to divide page into the blocks by different features, or data structures to represent components and elements of HTML document. Most commonly used approaches to segmentation are:

- DOM-based segmentation:

HTML document is represented as a DOM [4] (Document Object Model) tree. Tags are representing block of pages, for example P-paragraph, TABLE-table. Very accurate representation of

* Bachelor study programme in field: Informatics

Supervisor: Ing. Mária Šajgalík, Institute of Informatics, Information Systems and Software Engineering, Faculty of Informatics and Information Technologies STU in Bratislava

¹ <https://cloud.google.com/blog/big-data/2016/08/how-a-japanese-cucumber-farmer-is-using-deep-learning-and-tensorflow>

the structure of HTML document, but not accurate enough for dividing of visual different blocks.

- Location-based segmentation [10]:
Page is divided to 5 parts: center, left, right, bottom, top. Problematic may be scrollable pages, when page has different layout of the objects after scrolling.
- VIPS (Vision-based Page Segmentation [3]) algorithm:
Combination of the previous two examples of segmentation. Dividing page by color, size of blocks, etc. At first, the appropriate nodes, which imply the horizontal and vertical lines of web page, are found in the DOM tree. Based on that, semantic tree is created. It is a tree, where every segment is a separate node. Continuity of the segmented page is controlled by permitted degree of coherence (pDoC [9]), which ensures keeping content together, while semantically different blocks apart.

After segmentation, the importance of blocks is determining. It could be done using heuristics [8], human assessors (who manually label the blocks), etc.

As an example of working solution can be mentioned the work of Microsoft researchers [10]. They used VIPS algorithm for page segmentation and 5 human assessors to label each blocks of web pages. Blocks were numbered by importance from 1 to 4, from insignificant informations (such as adds) to the most important part of web page (news, products, etc.). Authors of this solution presume that people have consistent opinions about the importance of the same block in a page. After every block was labeled, the model of importance was created as a mapping function of every block and its importance:

$$\{block\ features\} \rightarrow \{block\ importance\} \quad (1)$$

For estimation of the block's importance was used neural network, where blocks are represented as tuple $\{x, y\}$. Number of block is x and its importance is y (real number). The network type was RBF (Radial Basis Function) and the neural network used standard gradient descent.

Second approach

Second approach to determining important parts of the web pages is based on the image of web page and a neural network that predicts the most engaging parts of web page in the form of heatmap, saliency map, etc.

Solution using this approach was created by Shen Chengyao and Zhao Qi [11]. They created a dataset divided to the categories by the content of web page

(pictorial, text, mixed). Each category contains 50 images, which were shown to the 11 subjects. Their sequences of views were recorded using MATLAB with Psychtoolbox and Eyelink 1000. For views prediction was used MKL (multiple kernel learning), which was trained as a binary regression problem.

3 Method of predicting eye-catching parts of the websites

After closer research of the problem domain, we decided to use second approach of the determining important parts of the web pages. Our method is based on the neural network, particularly convolution neural network, which is able to predict the eye-catching parts just from the image of web page. After experimenting with various architectures of neural networks, we decided to use architecture shown in figure 1, because it has the best results. Shown architecture contains one convolution layer and one max pooling layer followed by normalization, fully-connected and output layer.

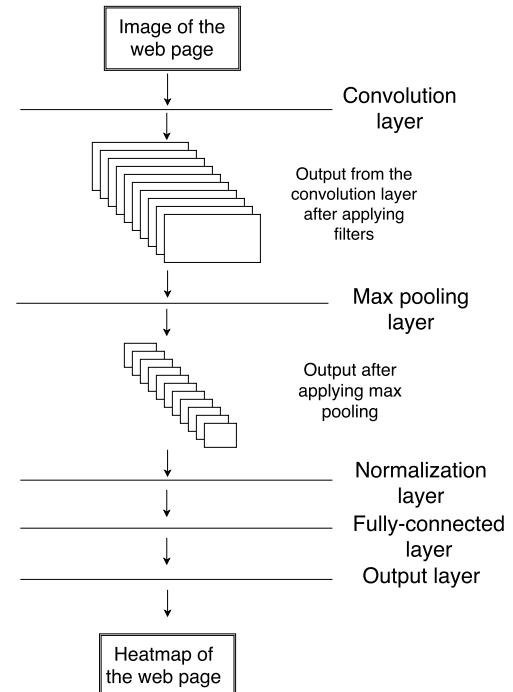


Figure 1. Diagram of the neural network's architecture

Convolution layer has convolution filter, whose size is 5x5. Activation function on this layer is standard ReLU:

$$f(x) = \max(0, x) \quad (2)$$

After processing data, data is passed to the max pooling layer, where the output from previous layer is processed using window, whose size is 2x2, and MAX operation. Output of all filters is merged into one wide flat layer, normalization layer. This layer is followed by fully-connected and output layer, where the whole prediction is made. The output layer contains final predicted heatmap for image, whose size is $m \times m$, where m is the size of both, image and heatmap.

4 Evaluation

Our dataset was collected as a part of experiments, that are described in bachelor thesis of Mária Dragúňová [5]. Now it contains 15 images of different web pages with 44 sequences of views (fixations) on these images in first 5 seconds.

Images size is 1920×1080 , what is too much. Therefore, the size is reduced to 256×256 , what is also better for neural network.

Fixations consist of X and Y coordinates for every view (range from 0 to 1) and duration of views. Number of fixations is variable for every image, the range is from 3 to 20. Heatmaps are calculated from these fixations using normal (Gaussian) distribution. First of all, every fixation is converted into the point on image of web page, as it is shown in equation 3. That ensures same size of heatmaps and images.

$$[x, y] = [fixation_x * 256, fixation_y * 256] \quad (3)$$

The probability density function of normal distribution is shown in equation 4, and it is calculated for every fixation from all other fixations.

$$f(x | \mu, \sigma^2) = \frac{1}{\sqrt{2\sigma^2\pi}} * e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (4)$$

x - current point for which the normal distribution is used

$(x-\mu)^2$ - distance between current point and other fixation

σ^2 - duration of fixation

For visualization of heatmaps on images, simple alpha blending is used, as you can see in figure 2.



Figure 2. From left: original image, heatmap, heatmap on image

Before passing data to the convolution neural network, we divide the data (images, heatmaps) to 16 smaller parts (buckets), so every image and heatmap is basically 4×4 grid (shown in figure 3), consisting of squares of size 64×64 . This division of data is done mostly because of the need for a larger training dataset.

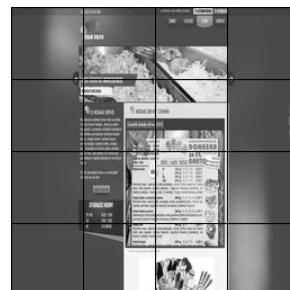
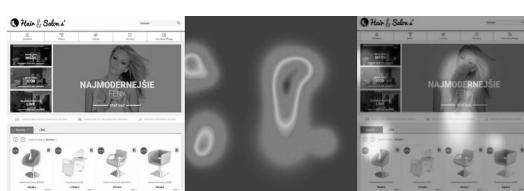


Figure 3. Visualization of the grid on image

The input data for neural network are parts of divided images, labels are equally divided heatmaps. Currently, accuracy of predictions is about 20%, what is not much.



5 Conclusion

This thesis describes a prototype of neural network, that is capable of predicting the eye-catching parts of the given web pages in form of the heatmap. It does not need any source code of the web sites, only screenshot is required. That may be considered as a solid advantage compared to some other solutions of similar problem.

Prototype is written in Python using Tensorflow² framework for neural networks. Results will be compared to the saliency model of Itti and Koch [6], but only after some modifications and more training. In future, we should consider optimization for network, maybe try to add more hidden layers. We may also perform another experiment with different web pages and let more people look at them, analyze it so we will have more data for training our convolution neural network and hopefully the dividing of data to 4x4 grid wouldn't be needed.

As a real-life application of our solution, I could imagine a web page, where people just upload screenshot of the web pages design and get a visualization of predicted engaging parts in form of the heatmap. That could provide solid informations for administrators of web sites, so they would be able to decide where to place important elements and where adds should be placed in order not to cause distraction.

Acknowledgement: This contribution was created with kind support of ČSOB Foundation and is partial result of the project University Science Park of STU Bratislava, ITMS 26240220084, co-funded by the ERDF.

References

- [1] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., et al.: End to End Learning for Self-Driving Cars. *arXiv preprint arXiv:1604.07316*, 2016.
- [2] Britz, D.: UNDERSTANDING CONVOLUTIONAL NEURAL NETWORKS FOR NLP. <http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/>.
- [3] Cai, D., Yu, S., Wen, J.R., Ma, W.Y.: VIPS: a Vision-based Page Segmentation Algorithm. Technical report, 2003.
- [4] Chakrabarti, S.: Integrating the document object model with hyperlinks for enhanced topic distillation and information extraction. In: *Proceedings of the 10th international conference on World Wide Web*, ACM, 2001, pp. 211–220.
- [5] Dragúňová, M.: Vyhodnocovanie používateľského zážitku analýzou pohľadu a emócií. Bachelor thesis, Bratislava: FIIT STU, 2016.
- [6] Itti, L., Koch, C.: A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, 2000, vol. 40, no. 10–12, pp. 1489 – 1506.
- [7] Li, F.F., Karpathy, A., Johnson, J.: CS231n: Convolutional neural networks for visual recognition, 2015.
- [8] Liu, Y., Wang, Q., Wang, Q. In: *A Heuristic Approach for Topical Information Extraction from News Pages*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 357–362.
- [9] Mehta, R.R., Mitra, P., Karnick, H.: Extracting semantic structure of web documents using content and visual information. In: *Special interest tracks and posters of the 14th international conference on World Wide Web*, ACM, 2005, pp. 928–929.
- [10] Ruihua Song, Haifeng Liu, J.R.W., Ma, W.Y.: Learning block importance models for web pages. In: *Proceedings of the 13th international conference on World Wide Web (WWW '04)*, New York, USA, 2004, pp. 203–211.
- [11] Shen, C., Zhao, Q. In: *Webpage Saliency*. Springer International Publishing, Cham, 2014, pp. 33–46.

² <https://www.tensorflow.org/>

E Protokol z experimentu s cieľom získania dát pre neurónovú sieť

Príloha obsahuje protokol z kvantitatívneho experimentu, s jeho popisom, cieľmi a priebehom sedení.

Protokol experimentu

Kontext experimentu

Meno a priezvisko Patrik Beka

Supervisor Ing. Mária Šajgalík

Názov projektu Určovanie pútavých častí grafických rozhraní webových stránok.

Názov projektu v AJ Determination of the eye-catching parts in graphical interfaces

Kľúčové slová konvolučné neurónové siete, predikcia pohľadov, heatmapy,

Stručný opis projektu

Cieľom projektu je zestrojiť neurónovú sieť schopnú predikcie pútavých častí grafických rozhraní webových stránok len zo screenshot-u danej stránky. K tomu je však potrebný dostaok relevantných dát, na ktorých by sa mohla neurónová sieť učiť. Tá je, resp. bude bližšie popísaná v mojej bakalárskej práci. Experiment je jednoduchý a krátky, participantom bude ukázaných 50 web stránok rôzneho druhu, každá na 5 sekúnd. Ako úlohu dostanú zapamätať si rozloženie a obsah stránky a to z toho dôvodu, že keď si budú mať niečo zapamätať, stránku len narýchlo neprejdú (nezoskenujú) ale budú sa sústrediť na jej črty. Vrámcí podnietenia motivácie bude na konci kontrolná otázka, či nasledujúci obrázok bol alebo nebol medzi predošlými 50. Cieľom je získať sekvencie pohľadov s ich dĺžkou, prípadne rovno vygenerované heatmapy.

Stručný opis projektu v AJ

The purpose of project is to design neural network, that is capable of predicting eye-catching parts of web pages, just from the screenshot. It is closer described (or it will be) in my bachelor thesis. Neural network requires lots of relevant data for training, the more the better. Proposed experiment is simply and not long. 50 web pages of different kind will be shown to the participants (each for 5 seconds). Their task will be to memorize layout and content, because people are used to just quickly “scan” the web page, but if they should memorize something, they tend to focus on multiple features. At the end will be shown control question: Was the followed picture in previous fifty? Question is there just to motivate participants to take the testing seriously. Our purpose in this experiment is to gain the sequence of views with durations, alternatively generated heatmaps.

Príprava experimentu

Cieľ experimentu

Nasnímanie pohľadov čo najviac participantov na web stránky.

Hypotézy

Kam sa ľudia pozerajú pri prvom stretnutí s web stránkou bez zadanej úlohy a čo si všímajú?
-dôležité informácie pri učení sa neurónovej siete, ktorá má predikovať pútavé časti grafických rozhraní webových stránok

Parametre experimentu

Kľudné prostredie, upozornenie participanta na kontrolnú otázku na konci a možnosť odmeny

podľa výberu v prípade správnej odpovede.

Použitý dataset

obsahuje 50 obrázkov rôznych webstránok

Účastníci

20 participantov (16 chlapov a 4 ženy vo veku 20 - 24 rokov)

Sledované metriky

fixácie pohľadov a ich dĺžky

Scenár experimentu

Participantom bude postupne ukázaných 50 screenshot-ov web stránok, každý na 5 sekúnd. Ich úlohou bude zapamätať si čo najlepšie obsah web stránky. Medzi každým obrázkov web stránky sa na sekundu zobrazí abstraktná grayscale snímka na rozhodenie pozornosti. Na konci je kontrolná otázka, či nasledujúci obrázok web stránky bol alebo neboli medzi predošlými 50. Participant je o kontrolnej otázke na konci informovaný ešte pred začiatkom experimentu.

Priebeh experimentu

Pilot

V podstate obsahoval len prvé testovanie ktorého sa zúčastnil len jeden participant. Testovanie odhalilo menšie chyby ako zlé časovanie na niektorých snímkach, či opakujúce sa snímky jednej web stránky. Inak bolo všetko v poriadku a pochopiteľné, chyby boli upravené pre ďalších priebeh experimentu.

Sedenie 1

Zúčastnilo sa ho 5 participantov, všetko prebiehalo podľa plánu a bez akýchkoľvek špeciálnych udalostí.

Celkovo sa uskutočnilo 6 sedení, na ktorých všetko prebiehalo tak ako malo, bez problémov. Na poslednom sedení bolo len nazbieraných od jedného participanta o niečo viac dát, nakoľko bolo pridaných ešte 5 ďalších stránok, ktoré poslúžili ako testovací dataset navyše pri výpočte metrik využívnej pozornosti.

Zhodnotenie experimentu

Výsledky experimentu

Experiment dopadol veľmi, podarilo sa zozbierať solídne množstvo dát pre neurónovú sieť.

Neočakávané udalosti

Pri prvom v podstate testovacom pilote zlé časovanie a opakujúce sa snímky, problém bol vyriešený a už sa neopakoval.

Čo sa podarilo

Ciel zozbieranie dát pre neurónovú sieť bol splnený.

F Obsah priloženého elektronického nosiča

K bakalárskej práci je priložený elektronický nosič (DVD), ktorého štruktúra a obsah sú nasledujúce:

- **dataset/** - priečinok obsahujúci kompletný dataset, vrátane obrázkov, surových a spracovaných dát. Jeho štruktúra je nasledujúca:
 - **data_exports/** - surové dáta exportované z experimentov
 - **data_for_nn/** - obsahuje spracovaný dataset pripravený k trénovaniu modelu neurónovej siete, rozdelený je nasledovne:
 - * **heatmaps/** - priečinok s vypočítanými heatmapami
 - * **rsz_images_64x64/** - priečinok s obrázkami s upravenou veľkosťou k trénovaniu neurónovej siete
 - **final_testing/** - priečinok obsahujúci fixácie, heatmapy a obrázky použité pri finálnom testovaní a výpočte metrík na evaluáciu modelu vizuálnej pozornosti, všetky rozdelené do prislúchajúcich priečinkov:
 - * **fixations/** - priečinok s fixáciami
 - * **heatmaps/** - priečinok s heatmapami
 - * **images/** - priečinok s obrázkami
 - **images_from_experiment/** - priečinok obsahujúci obrázky použité pri experimente
 - **processed_participants/** - priečinok obsahuje textové súbory reprezentujúce vyfiltrovaných participantov a ich fixácie na obrázky. Nachádza sa tu aj priečinok:
 - * **heatmaps/** - v ňom sú vypočítané heatmapy fixácií na každý obrázok
- **doc/** - priečinok obsahujúci dokument bakalárskej práce v pdf verzii
- **model/** - priečinok obsahuje natrénovaný model neurónovej siete
- **source_codes/** - priečinok obsahuje všetky zdrojové kódy k riešeniu bakalárskej práce