

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-182905-73665

Bc. Patrik Beka

**Modelovanie ľudskej vizuálnej
pozornosti metódami počítačového
videnia a umelej inteligencie**

Diplomová práca

Študijný program: Inteligentné softvérové systémy

Študijný odbor: Inteligentné softvérové systémy

Miesto vypracovania: Ústav počítačového inžinierstva a aplikovanej informatiky,
FIIT STU, Bratislava

Vedúci práce: doc. Ing. Vanda Benešová, PhD.

Apríl 2019

ČESTNÉ PREHLÁSENIE

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

.....

Bc. Patrik Beka

POĎAKOVANIE

Rád by som pod'akoval pani doc. Ing. Vande Benešovej, PhD. za rady a konzultácie počas vypracovávania diplomovej práce, spolu s možnosťou prezentácie práce na jej seminároch.

Anotácia

Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Študijný program: Inteligentné softvérové systémy

Autor: Bc. Patrik Beka

Diplomová práca: Modelovanie ľudskej vizuálnej pozornosti metódami počítačovo-viedenia a umelej inteligencie

Vedúci práce: doc. Ing. Vanda Benešová, PhD.

Apríl 2019

Vizuálna pozornosť zohráva veľmi dôležitú rolu v našom vnímaní sveta. Jej postupné modelovanie je kľúčovým procesom pri snahe výskumníkov umožniť počítačom vnímať a porozumieť pozorovanej scéne podobným spôsobom ako ľudia. Staršie postupy založené na matematických operáciách, s ktorých pomocou prebiehala extrakcia základných črt, postupne nahradzajú modernejšie metódy strojového učenia. Práve v tejto oblasti sa za posledných niekoľko rokov podarilo dosiahnuť obrovský pokrok, najmä vďaka neurónovým sietiam. Ich charakteristickým znakom je využitie veľkého množstva hierarchických vrstiev pre spracovanie nelineárnych informácií, čo prinieslo obrovské množstvo nových možností pre zachytenie pozornosti a detekciu salientných oblastí scény, berúc do úvahy aj jej sémantický kontext. Vďaka nielen týmto jedinečným vlastnostiam disponujú schopnosťou odvodiť si závislosti medzi pozorovaniami a objektami v scéne. Pri správnom trénovaní vedia práve tieto naučené závislosti aplikovať bez väčších problémov na nové dátá a tak sa čo najviac priblížiť svojimi predikciami reálnej pozornosti človeka.

Annotation

Slovak University of Technology Bratislava

FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

Degree Course: Intelligent Software Systems

Author: Bc. Patrik Beka

Master thesis: The modeling of human visual attention using computer vision and artificial intelligence

Supervisor: doc. Ing. Vanda Benešová, PhD.

May 2018

Visual attention plays very important role in our perception of world. Gradual modelling of visual attention is a key process in efforts of researchers to allow computers perceive and understand observed scene in a similar way as people do. Older methods based on the mathematical operations, which helped extract basic features, are gradually replaced by more sophisticated methods of machine learning. During last couple of years, a huge progress has been achieved, particularly in this area and mainly because of neural networks. Their characteristic feature is exploitation of a huge number of hierarchical layers for processing of nonlinear information. This brought on enormous possibilities for capturing attention and detection of salient areas in scene, taking into account its semantic context as well. Not only Because of these unique characteristics, neural networks have the ability to deduce dependencies between observations and objects in scene. Using the right training, they can apply these learned dependencies to the new data without any difficulties and so approximate their predictions to the real human attention as much as possible.

Obsah

1	Úvod	1
2	Vizuálna pozornosť	3
2.1	Spracovanie zdola nahor	3
2.2	Zhora nadol spracovanie	4
3	Neurónové siete	5
3.1	Aktivačné funkcie	6
3.2	Učenie sa neurónovej siete	8
3.2.1	Učenie sa s učiteľom	8
3.2.2	Učenie sa bez učiteľa	9
3.2.3	Učenie posilňovaním	10
3.2.4	Prenos učenia	10
3.2.5	Optimizačné algoritmy	11
3.3	Typy neurónových sietí	12
3.3.1	Rekurentné neurónové siete	13
3.3.2	Konvolučné neurónové siete	16
3.3.3	Autoenkóder	18
3.4	Framework-y pre prácu s neurónovými sietami	19
3.4.1	TensorFlow	20
3.4.2	Microsoft CNTK	21
3.4.3	Theano	22
3.4.4	Keras	23
3.4.5	Spark MLlib	23
4	Existujúce modely vizuálnej pozornosti	25
4.1	Modely k predikcii pozornosti zdola nahor	25
4.1.1	Itti-ho model	26
4.1.2	Metóda hierarchickej výraznosti	26
4.1.3	Model pre určovanie salientných oblastí webových stránok	27
4.2	Modely k predikcii pozornosti zhora nadol	27
4.2.1	Zameranie na emočný obsah	28

4.2.2	SAM	29
4.2.3	Pozornosť zhora nadol vedená titulkami	31
4.2.4	Redukcia sémantických medzier pri predikcii vizuálnej pozornosti	33
4.3	Detekcia objektov	34
4.3.1	Autoenkóder s VGG16 sietou	35
4.3.2	Vedená a upravená vizuálna pozornosť	36
5	Datasetsy vizuálnej pozornosti	39
6	Metriky pre modely vizuálnej pozornosti	41
7	Návrh	43
7.1	Dataset	43
7.2	Konvolučná neurónová siet'	44
7.3	Autoenkóder	46
7.4	Kombinácia VGG16 siete s autoenkóderom	48
8	Experiments	51
8.1	Implementačné prostredie	51
8.2	Prvotné experimenty	52
8.2.1	Úprava datasetu	52
8.2.2	Výsledky	53
8.3	Konvolučná neurónová siet'	54
8.4	Konvolučný autoenkóder	55
8.5	Autoenkóder s predtrénovaným modelom VGG16	58
8.6	Kombinácia autoenkóderu s VGG16 sietou	59
8.7	Vedená a upravená vizuálna pozornosť pri modeloch na detekciu objektov	64
8.8	Porovnanie výsledkov experimentov	68
9	Porovanie s existujúcimi riešeniami	69
10	Zhrnutie	71

Literatúra	73
A Plán práce a jeho plnenie	i
B Obsah priloženého elektronického nosiča	iii
C Technická dokumentácia	v
D Používateľská príručka	vii
E Podrobnejší diagram architektúry s kombináciou VGG16 a autoenkóderu	

Zoznam obrázkov

1	Jednoduchá neurónová siet'	13
2	Jednoduchá rekurentná neurónová siet'	14
3	Zobrazenie LSTM jednotky	15
4	Vrstva združovania - príklad vzorkovania	17
5	Konvolučná neurónová siet'	17
6	Príklad jednoduchého autoenkóderu	19
7	Architektúra fungovania Microsoft CNTK	21
8	Príklad grafovej abstrakcie výpočtov použitím framework-u Theano	22
9	Itti-ho hierarchický model vizuálnej pozornosti	26
10	Diagram modelu k predikcii máp výraznosti založených na emočnom obsahu	29
11	Diagram konvolučnej LSTM siete	30
12	Predikcia salientných oblastí na základe kl'účových slov vo vete .	32
13	Model neurónovej siete pre predikciu na základe kl'účových slov .	33
14	Model redukcie sémantických medzier pri predikcii pozornosti .	34
15	VGG16	35
16	Autoenkóder pre detekciu objektov	36
17	Návrh architektúry neurónovej siete	45
18	Diagram navrhnutého autoenkóderu	47
19	Náčrt architektúry VGG16 siete s autoenkóderom	49
20	Vizualizácia extrakcie regiónov záujmu	52
21	Porovnanie prvotných výsledkov	53
22	Vývoj chyby počas trénovania konvolučnej neurónovej siete . . .	54
23	Vzorka predikcie konvolučnej neurónovej siete	55
24	Vývoj chyby počas trénovania autoenkóderu	56
25	Porovnanie predikcií autoenkóderu voči reálnym mapám výraznosti	57
26	Vývoj chyby počas trénovanie siete s predtrénovaným modelom VGG16	59
27	Porovnanie vývoja chyby predikcie počas trénovania kombinácie autoenkóderu s VGG16 siet'ou	60

28	Porovnanie predikcií autoenkóderu s VGG16 siet'ou bez dodatočného trénovania voči reálnym mapám výraznosti	61
29	Porovnanie predikcií autoenkóderu s VGG16 siet'ou s dodatočným trénovaním voči reálnym mapám výraznosti	63
30	Porovnanie predikcií upravenej vizuálnej pozornosti voči reálnym mapám výraznosti	65
31	Porovnanie predikcií vedenej vizuálnej pozornosti voči reálnym mapám výraznosti	66
32	Podrobný diagram architektúry VGG16 s autoenkóderom	

1 Úvod

Vizuálna výraznosť zhora nadol (z angl. top-down) je veľmi dôležitou súčasťou vizuálnej pozornosti (popísaná v kapitole 2), nakoľko sa pri nej berie do úvahy sémantický kontext pozorovanej scény. To je niečo, čo zohráva klíčovú úlohu pri snahe výskumníkov umožniť počítačom vnímať a porozumieť obrázkom rovnakým spôsobom ako ľudia. V posledných rokoch sa ako prelomovým v tejto doméne javí nevídany pokrok v oblasti umelej inteligencie, konkrétnie neurónových sietí, ktorých popis spolu s niekoľkými typmi je spomenutý v kapitole 3. Ich charakteristickým znakom je využitie veľkého množstva hierarchických vrstiev pre spracovanie nelineárnych informácií. Vďaka tomu sa objavili nové možnosti detektie salientných častí skúmanej scény na základe jej sémantického kontextu.

Spomedzi riešení berúcich do úvahy spomínaný sémantický kontext a teda predikujúcich pozornosť zhora nadol (top-down), si v kapitole 4 predstavíme niekoľko zaujímavých príkladov spolu so zaužívanejšími modelmi k predikcii pozornosti zdola nahor (z angl. bottom-up). Pre ich natrénovanie je nutné mať čo najlepšie spracovaný dataset, najlepšie voľne dostupné sú popísané v kapitole 5. K ohodnoteniu kvality modelov po natrénovaní sa používa značné množstvo metrík, z ktorých napoužívanejšie sú vysvetlené v kapitole 6.

Na základe preštudovanej problémovej oblasti sme postupne navrhli niekoľko architektúr neurónových sietí (kapitola 7), od jednoduchších konvolučných až po zložitejšie kombinujúce niekoľko modelov (aj natrénovaných) určených pre riešenie rôznych úloh do jednej komplexnejšie siete. Na jednotlivé návrhy sme nadviazali experimentami v kapitole 8, kde sú postupne popísané priebehy a dosiahnuté výsledky počas hľadania najlepšieho modelu. Ten sme potom porovnávali s už existujúcimi riešeniami (kapitola 9) na viacerých datasetoch (kapitola 9) - náš model bol postupne dotrénovaný na jednotlivé vybrané datasety, aby sa výsledky čo najviac zlepšili. V závere práce sme už len zhrnuli dosiahnuté výsledky.

2 Vizuálna pozornosť

Termín vizuálna pozornosť možno definovať ako súbor všetkých faktorov, ktoré ovplyvňujú naše mechanizmy výberu podstatných častí v scéne a jej spracovanie, nezáležiac od toho, aké tieto mechanizmy sú (či už riadené stimulmi, očakávaniami, pamäťou, atď.). [4].

Tento pojem je často zamieňaný s vizuálnou výraznosťou, avšak tieto dva termíny nevyjadrujú úplne to isté. Presnejšou definíciou vizuálnej výraznosti (z angl. visual saliency) je, že sa jedná o značne subjektívnu perceptuálnu vlastnosť, vďaka ktorej niektoré veci vo svete (v scéne) vyčnievajú v porovnaní so svojimi susedmi kvôli ich vlastnostiam ako farba, jas, kontrast či orientácia [22]. Upútanie pozornosti ovplyvňujú mechanizmy spracovania scény, ktoré možno rozdeliť na dve skupiny:

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

2.1 Spracovanie zdola nahor

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

Hlavnou charakteristikou tohto spracovania je nevedomosť (obvykle bez predošlých informácií o pozorovanej scéne) a rýchlosť - priemerné spracovanie jedného objektu v scéne je na úrovni od 20 do 50 milisekúnd[23].

2.2 Zhora nadol spracovanie

Tento typ spracovania vizuálnych signálov sa oproti vyššie uvedenému líši viacerými vecami. Tou prvou je, že sa riadi tzv. predvídateľnými mechanizmami a prináša so sebou bližšie nešpecifikovanú vedomosť o pozorovanej scéne - pozorovateľ má isté informácie ako napríklad predošlé skúsenosti, spomienky, alebo hľadá v scéne nejaký konkrétny objekt. Z tohto dôvodu sa nazýva aj spracovaním založeným na vedomostach (z angl. knowledge-based processing[16]) alebo údajoch (z angl. data-driven processing[18])

Druhým veľkým rozdielom oproti spracovaniu zdola nahor je jeho rýchlosť, priemerný čas spracovania vizuálneho signálu sa pohybuje na úrovni 200 milisekúnd[23] a viac, čo je výrazne pomalšie.

3 Neurónové siete

Neurónová sieť - pojem reprezentujúci značne abstraktný výpočtový model, ktorého princíp fungovania bol postavený na reálnych biologických neurosystémoch. Ako už zo základnej školy vieme, základnou stavebnou jednotkou živých organizmov je bunka. Jej špecializovaným derivátom je nervová bunka - neurón, stavebný kameň nervových systémov. Pri neurónových sieťach je základnou jednotkou rovnako neurón (avšak nie živý ale umelý), respektívne jeho matematické vyjadrenie. To predpokladá, že umelý neurón je schopný spracovať N vstupov a k nim poskytnúť M výstupov. Jedna zo starších matematických špecifikácií ho popisuje nasledovne:

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

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

$f()$ - aktivačná funkcia

Modernejšie prístupy však odstránili prah stimulácie neurónu a miesto neho pridali tzv. predsudok (z angl. bias), ktorý reprezentuje predpokladanú (s časom sa meniacu) hodnotu stimulácie neurónu na ďalšej vrstve. Pre vysvetlenie hypoteticky predpokladajme, že máme umelý neurón a chceme, aby spracoval $m+1$ vstupov so signálmi od x_0 po x_m a váhami od w_0 po w_m . Pre x_0 pridelíme hodnotu +1, vďaka čomu sa stane predsudkom (biasom) k vstupu s $w_{k0} = b_k$. Týmto postupom nám teda zostane iba m vstupov so signálmi od x_1 po x_m . Popisovaný postup ako aj výstup z k-teho neurónu matematicky vyjadruje rovnica 2 nižšie:

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

Pre vyššie uvedené platí:

y_k - výstup k-teho neurónu

w_{kj} - váha j-teho neurónu spojeného s k-tym neurónom na d'alšej vrstve

x_j - j-ty neurón

ϕ - aktivačná funkcia

Takéto umelé neuróny sú potom umiestnené na vrstvách s prepojeniami napr. do d'alších vrstiev. Štandardne sa prvá vrstva nazýva vstupná a posledná výstupná. Tie medzi sebou d'alej môžu mať niekoľko d'alších vrstiev rôzneho typu. Tieto vrstvy sa nazývajú skryté a obvykle práve tieto bývajú kl'úcové pri učení sa a predikciach. Neuróny každej vrstvy navyše môžu obsahovať aj aktivačnú funkciu (klasicky býva pre všetky neuróny na vrstve rovnaká), ktorá definuje (upravuje) výstup z neurónu pre vstup alebo sériu vstupov.

3.1 Aktivačné funkcie

Aktivačná funkcia predstavuje matematické vyjadrenie použité k aproximácii vplyvu na neurón, zjednodušene by bolo možné povedať, že pre sériu vstupov definuje výstupy. Aktivačných funkcií existuje niekoľko typov, každá vhodná na iný typ úloh. Ako príklad je možné uviesť nasledujúce:

- **Softmax**

Funkcia 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

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

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 zároveň 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, ked'že neobsahuje žiadne komplikované výpočty, čoho dôsledkom je aj jej značná rýchlosť. Jej využitie je možné pozorovať napríklad pri hlbokých neurónovo-vých sietach.

- **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 ked' je potrebné pracovať s pravdepodobnosťami, ked'že 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, ked'že matematicky sa dá vyjadriť aj za použitia Sigmoidu. Jeho vzorec je

nasledovný:

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

3.2 Učenie sa neurónovej siete

Základným prvkom toho, aby bola neurónová siet' schopná riešiť úlohy je učenie sa. Existujú viaceré typy učenia sa neurónovej siete, základnými sú učenie sa s učiteľom (z angl. supervised learning), učenie sa bez učiteľa (z angl. unsupervised learning[19]) a učenie sa posilňovaním (z angl. reinforcement learning[34]).

3.2.1 Učenie sa s učiteľom

Učenie s učiteľom prebieha 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 očakávané výstupy. Najčastejším príkladom tohto typu učenia je algoritmus spätného šírenia chyby (z angl. backpropagation[35]).

Algoritmus spätného šírenia chyby

Hlavným princípom je snaha o minimalizovanie chyby predikcie pri učení a to tak, že po predikcii pri učení sa vypočíta hodnota chyby na poslednej výstupnej vrstve voči očakávanému výstupu (spomínaným štítkom). Tá sa potom tzv. "šíri" späť k vstupnej vrstve a vzhľadom na jej hodnotu sa postupne aktualizujú váhy jednotlivých neurónov.

Príklad použitia tohto algoritmu možno nájsť v jednoduchej neurónovej sieti určenej k riešeniu problému funkcie XOR[1]. Vstupné dátá je nutné reprezentovať ako dvojicu jednotiek a núl. Pre vstupnú dvojicu napríklad $[0, 1]$ je očakávaným výstupom číslo 1 , pre hodnotu $[0, 0]$ je to 0 . Tieto očakávané hodnoty zároveň označíme za spomínané štítky. Takto pripravíme celý dataset pre trénovanie, mal by byť dostatočne rozsiahly aby sa dosiahla maximálna presnosť (odhadnúť aké množstvo dát už možno považovať za dostatočné je značne netriviálny problém). Sieti sú potom počas učenia predkladané vstupy a štítky, na základe vypočítanej

chyby sú potom predikcie upravované až kým chyba nie je úplne minimalizovaná. Spomínaný problém veľkosti dostatočne rozsiahleho datasetu je považovaný za jeden z hlavných minússov učenia s učiteľom.

Učenie sa viacerých inštancií

MIL[33] - skratka pre anglický výraz Multiple Instance Learning, čiže učenie sa viacerých inštancií, je variácia učenia s učiteľom. Miesto obdržania individuálne oštítkovaných inštancií (reprezentujú triedy, ktoré chceme predikovať), obdrží model set oštítkovaných vreciek (z angl. bags), z ktorého každé obsahuje niekol'ko inštancií. V prípade jednoduchej binárnej klasifikácie je celé vrecko označené ako negatívna vzorka, ak všetky inštancie vo vnútri sú negatívne. Pokial' ale obsahuje aspoň jednu pozitívnu, je označené ako pozitívna vzorka. Pri zložitejších triedach si ako príklad môžeme uviesť predikciu nejakej ciel'ovej triedy pre obrázok na základe vizuálneho obsahu. Ciel'ová trieda bude "*pláž*" pre obrázok, ktorý obsahuje "*piesok*" a "*vodu*". V terminológii MIL je teda obrázok popísaný ako vrecko, matematicky reprezentované nasledovne:

$$X = \{X_1, \dots, X_N\} \quad (8)$$

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

3.2.2 Učenie sa bez učiteľa

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

3.2.3 Učenie posilňovaním

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

3.2.4 Prenos učenia

Z angl. transfer learning [49], týmto pojmom sa označuje zaujímavý problém pri strojovom učení - ako uložiť vedomosti naučené pri riešení danej úlohy a následne ich uplatniť pri riešení ďalšej odlišnej, ale súvisiacej úlohy. Najľahšie sa dá vysvetliť na príklade rozpoznávania objektov, kedy máme natrénovaný model napríklad na detekciu osobných áut a chceme vytvoriť ďalší, ktorý by detekoval kamióny. Pri klasickom postupe by sme potrebovali dostatočné veľký dataset kamiónov, na ktorom by sme model trénovali. To je však značne náročné čo sa týka zdrojov a času, preto môžeme použiť predtrénovaný model pre osobné autá a dotrénovať ho pre detekciu kamiónov. Takto môžeme výrazne znížiť čas nutný na trénovanie a zvýšiť presnosť detekcie.

Prenos učenia má 2 základné prístupy [8]:

- použitie predtrénovaného modelu - jedná sa o prípad uvedený ako príklad vyššie. Zvolíme si natrénovaný model pre podobnú úlohu ako chceme riešiť a ten následne znovupoužijeme (môže sa jednať o celý model, ale aj jednotlivé časti) a dotrénujeme ho pre náš problém. Znižuje čas potrebný na učenie, nevyžaduje tak veľký dataset a môže výrazne zvýšiť presnosť.
- použitie základného modelu - v prípade, že nemáme k dispozícii predtrénovaný model, môžeme si zvolať ako zdrojovú úlohu pre základný model podobnú, ako chceme riešiť my, avšak s hojným počtom dát. Tento model

natreňujeme a podobne ako pri predošlom prístupe ho potom znovupoužijeme a dotreňujeme na riešenie inej úlohy. Dobrým zvykom býva porovnanie výsledkov základného modelu napr. s naivnými Bayes-ovskými modelmi, aby sa zistilo, či sa model naučil aspoň nejaké relevantné črty. Tento prístup môže byť užitočný v prípade, že pre špecifickú úlohu máme málo dát.

3.2.5 Optimizačné algoritmy

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

- **Gradient descent optimizér [38]:**

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.

- Stochastický gradient descent:

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

- Mini-dávkový gradient descent:

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

² zmena veličiny v závislosti od inej premennej

- **Adam optimizér [27]:**

V podstate vychádza priamo zo Stochastického gradient descent optimizéru, resp. jeho modifikácie RMSProp algoritmu[47]. Rozdiel oproti Gradient descent optimizéru je ale v tom, že je schopný variabilne určovať rýchlosť učenia neurónovej siete.

- **Adadelta optimizér [51]:**

Jedná sa o rozšírenie Adagrad optimizéru[12], ktoré sa snaží zredukovať jeho agresívnu, monotónne klesajúcu rýchlosť učenia. Je robustnejší, prispôsobovanie spomínanej rýchlosťi učenia je založené na pohyblivom okne aktualizácií gradientu namiesto akumulácie všetkých minulých gradientov, ako je to pri Adagrad optimizéri. Vďaka tomu je Adadelta schopný pokračovať v učení aj po veľkom množstve epoch a aktualizácií gradientu.³

- **Ftrl optimizér:**

Vychádza z algoritmu učenia FTRL-Proximal[32], 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.

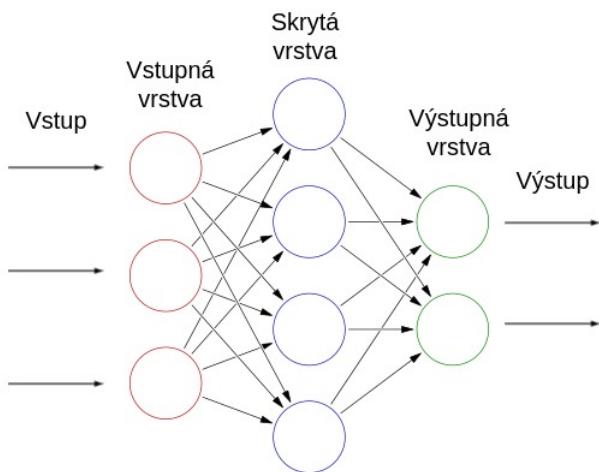
Aj keď neurónové siete dokážu efektívne riešiť veľké množstvo úloh, problémom stále zostáva mať k dispozícii dostatok dát k učeniu neurónovej siete ešte pred riešením úloh. Taktiež je potrebné mať dostatok výpočtovej sily, aby sa problém neriešil pridlhý čas, a dostatok pamäte, keďže neurónové siete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

3.3 Typy neurónových sietí

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

³<https://keras.io/optimizers/>

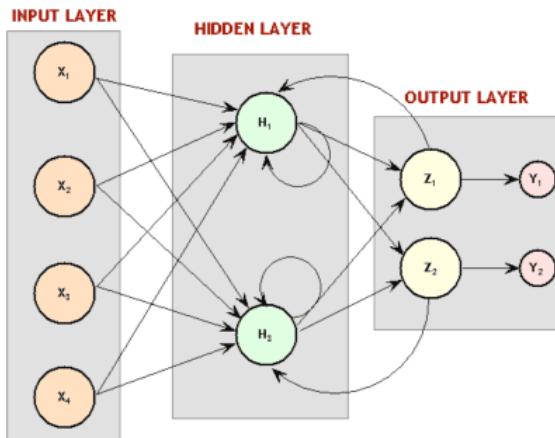
Najjednoduchší typ možno zobrazit ako jednu vstupnú vrstvu, jednu skrytú a jednu výstupnú, neuróny sú tu poprepájané z n -tej vrstvy do $n+1$ vrstvy, ako je možné vidieť na obrázku 1. Tento typ sa nazýva dopredná neurónová sieť (z angl. feedforward neural network) a môže mať aj viac ako len jednu skrytú vrstvu. Používa sa hlavne ak sa jedná o predikciu nelineárnej funkcie (napríklad carbon-13 NMR chemické posuny alkánov[45]).



Obr. 1: Príklad jednoduchej neurónovej siete

3.3.1 Rekurentné neurónové siete

Zložitejším typom neurónových sietí sú rekurentné neurónové siete. Už z názvu vyplýva, že jednou z vecí, ktoré umožňujú, je rekurenciu. Vďaka tejto prepojenia neurónov už nie sú jednosmerné len z jednej vrstvy na druhú, ale umožňuje prepojiť neuróny akokoľvek a tak vytvárať napríklad slučky či cykly (ukážka na obrázku 2).



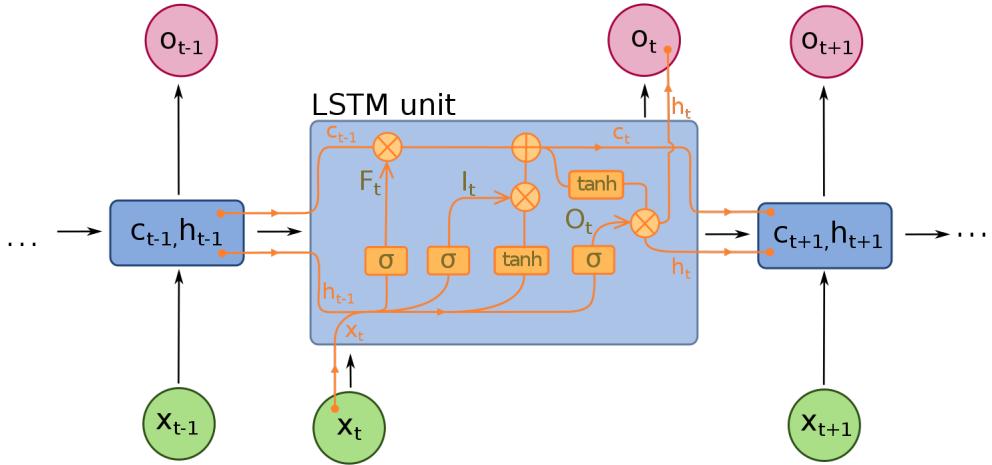
Obr. 2: Príklad jednoduchej rekurentnej neurónovej siete⁴

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

LSTM

LSTM[15] - skratka pre anglické pomenovanie Long short-term memory, čo v preklade znamená dlhá krátkodobá pamäť. Tento typ rekurentných sietí sa ukázal extrémne efektívny pri úlohách, kde je nutné pamätať si určitú informáciu (kratšieho charakteru) dlhý čas. Za to môžeme vďačiť LSTM jednotke, ktorej štandardná architektúra je zobrazená na obrázku 3.

⁴<http://www.mattmoocar.me/blog/RNNCountryLyrics/>

Obr. 3: Ukážka štruktúry LSTM jednotky⁵

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

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

Ked'že rekurentné siete na rozdiel od dopredných môžu na vstupe spracovať aj

⁵<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

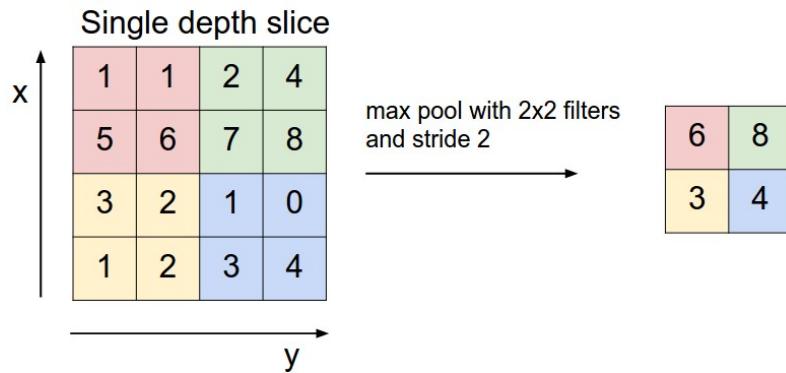
l'ubovoľnú sekvenciu (vektor) predstavujú s možnosťou "pamäte" značný posun. V praxi sa používajú napríklad pri úlohách so spracúvaním jazyka. Keby sme chceli napríklad predikovať nasledujúce slovo vo vete, je veľmi užitočné vedieť aké slová boli pred ním aby predikcia dávala zmysel. Práve v tejto oblasti sa LSTM siete ukázali veľmi efektívne. Využívajú sa d'alej napríklad aj pri rozpoznávaní reči[39] alebo písma[17], či generovaní popisu k obrázkom[26], kedy však fungujú v kombinácii s konvolučnou neurónovou sieťou (z angl. convolutional neural network). Tá je ďalším typom sietí a v tomto prípade bola použitá na klasifikáciu obrázkov a rozpoznávanie objektov, LSTM sieť bola použitá iba na generovanie výsledného jednoduchého popisu.

3.3.2 Konvolučné neurónové siete

Základ tohto typu siete tvorí vstupná konvolučná vrstva⁶ s konvolučným filtrom, ten býva väčšinou malý (3x3, 5x5). 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[29]), ktorá tieto filtre rozvzorkuje. To prebieha nezávisle na každom získanom filtrovi 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 2x2 aplikovaným s krokom veľkosti 2. Toto sa dá jednoducho vysvetliť ako prejdenie každého výstupu z konvolučnej vrstvy oknom uvedenej veľkosti postupne po 2 políčkach na šírku aj výšku, pričom z každej štvorce v okne sa získa MAX operáciou maximum, s ktorým sa pracuje ďalej. Na obrázku⁷ 4 je vidieť výsledok popisovaného postupu na jednoduchom príklade.

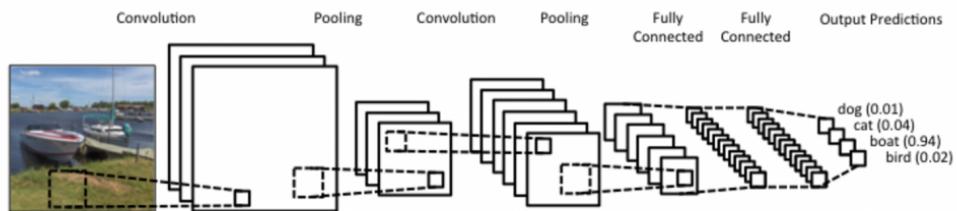
⁶<http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/>

⁷<http://cs231n.github.io/convolutional-networks/>



Obr. 4: 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 obrázku⁸ nižšie je jednoduchý náčrt vyššie popísanej konvolučnej neurónovej siete.



Obr. 5: Príklad konvolučnej neurónovej siete

Využíte tohto typu je v podstate všade, kde sa pracuje s obrázkami. Či už ide o automatické vyznačenie tvári pre označenie na facebook-u, autonómne vozidlá,

⁸<https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>

ktoré sa vedia riadiť sami (autopilot) alebo triedenie uhoriek na farmách v Japonsku⁹. Na tento konkrétny softvér bol použitý príklad kódu jednoduchej konvolučnej siete z tutoriálu¹⁰ pre TensorFlow (knižnica pre prácu s neurónovými sieťami, popísaná v kapitole 3.4.1), s modifikáciou konvolučnej a združovacej vrstvy tak, aby bola siet' uspôsobená počtu tried uhoriek (10) a ich formátu obrázkov.

Z mnohých pokusov o autonómnu jazdu stojia za zmienku hlavne tie od Tesly a Google. Prototyp autonómneho systému vozidla od Google-u Dave-2[3] 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 YUV¹¹ úrovní a posunutý do neurónovej siete.

3.3.3 Autoenkóder

Autoenkóder (z angl. autoencoder) je špeciálnym typom neurónovej siete, tradične je používaný ako algoritmus učenia bez učiteľa. K učenia však požíva štandardný algoritmus spätného šírenia chyby, kedy sú ale cielové hodnoty rovné tým vstupným, snaží sa teda naučiť funkciu:

$$y^{(i)} = x^{(i)} \quad (9)$$

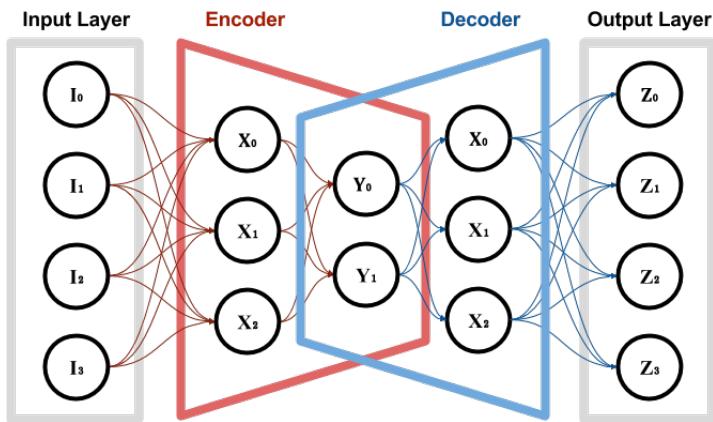
Jeho cielom je v podstate naučiť sa enkódovanú reprezentáciu dát, čo sa využíva napríklad pri komprimácii alebo k zníženiu dimenzionality dát[20]. Autoenkóder sa skladá z dvoch častí (ako je to vidieť na obrázku 6):

- enkóder - prvá časť, postupne sa od vstupnej vrstvy zmenšuje počet neurónov, čo v podstate komprimuje vstupnú informáciu do enkódovanie reprezentácie
- dekóder - druhá časť, jeho vstupom je práve enkódovaná reprezentácia dát a jeho cielom je z tejto reprezentácie zrekonštruovať pôvodné dátá

⁹<https://cloud.google.com/blog/big-data/2016/08/how-a-japanese-cucumber-farmer-is-using-deep-learning-and-tensorflow>

¹⁰<https://www.tensorflow.org/versions/r0.6.0/tutorials/mnist/pros/index.html>

¹¹farebný priestor používaný vo video aplikáciách



Obr. 6: Diagram reprezentujúci jednoduchý autoenkóder¹²

I keď' cieľom je na konci mať' na výstupnej vrstve dokonale zrekonštruovaný vstup, bohužiaľ to tak takmer nikdy nie je a pri autoenkóderoch sa tak jedná o stratovú kompresiu. Zatial' nie sú široko rozšírené a v praxi sa používajú najmä pri odstraňovaní šumu[48] z dát alebo pri znižovaní ich dimenziality. Uplatnenie tak nachádzajú hlavne pri spracovaní jazyka[30] a počítačovom videní - napríklad pri detekcii objektov[2].

3.4 Framework-y pre prácu s neurónovými siet'ami

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

¹²<https://www.alanzucconi.com/2018/03/14/an-introduction-to-autoencoders/>

3.4.1 TensorFlow

TensorFlow¹³ je open-source softvérová knižnica, ktorá pre numerické výpočty používa graf dátového toku, kde uzly grafu reprezentujú matematické operácie a hrany multidimenzionálne dátové polia, tzv. tenzory. Graf je možné skonštruovať použitím jazykov s podporou frontend-u (C++, Python, ...).

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

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

Medzi hlavné výhody patrí:

- podpora pre jednoducho naučiteľné jazyky (Python)
- použtie výpočtovej grafovej abstrakcie
- vizualizácie pomocou TensorBoard-u¹⁴ (interaktívne grafy pre priebeh učenia, pre model siete, ...)
- dostatočne nízko-úrovňový pre plnú kontrolu a implementáciu vlastnej (novej nie len preddefinovanej) funkcionality (v porovnaní napríklad s frameworkom Keras (kapitola 3.4.4))

Ako nevýhody možno uviesť:

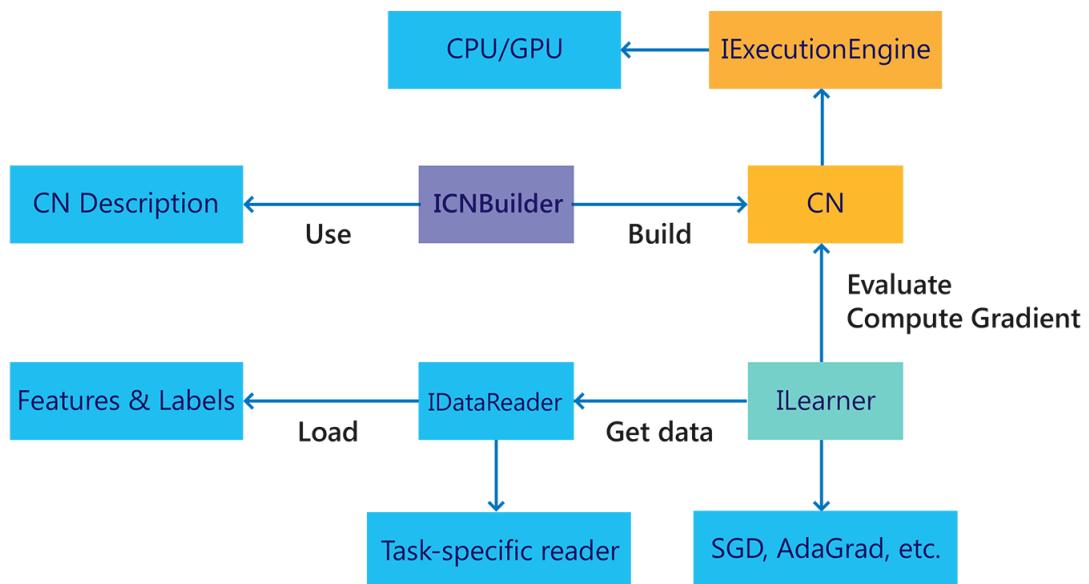
- nedostatok predtrénovaných modelov
- pri použití s určitými jazykmi (Python, Java, ...) je pomalý, nakoľko sa nejedná najrýchlejšie jazyky

¹³<https://www.tensorflow.org/>

¹⁴https://www.tensorflow.org/programmers_guide/summaries_and_tensorboard

3.4.2 Microsoft CNTK

Označuje knižnicu Microsoft Cognitive Toolkit¹⁵, ktorá zlepšuje modularizáciu a údržbu separácie výpočtových sietí, zároveň postkytuje algoritmy učenia a popisy modelov. Má sa jednať o odpoveď na TensorFlow, poskytovaná funkcia je veľmi podobná, avšak je o niečo rýchlejší. Princíp a celá architektúra sú zachytené na diagrame na obrázku 7.



Obr. 7: Diagram zobrazujúci architektúru fungovania Microsoft CNTK¹⁶

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

Výhodami tohto framework-u sú:

- flexibilita
- umožňuje distribuovaný tréning

Medzi nevýhody možno zaradit:

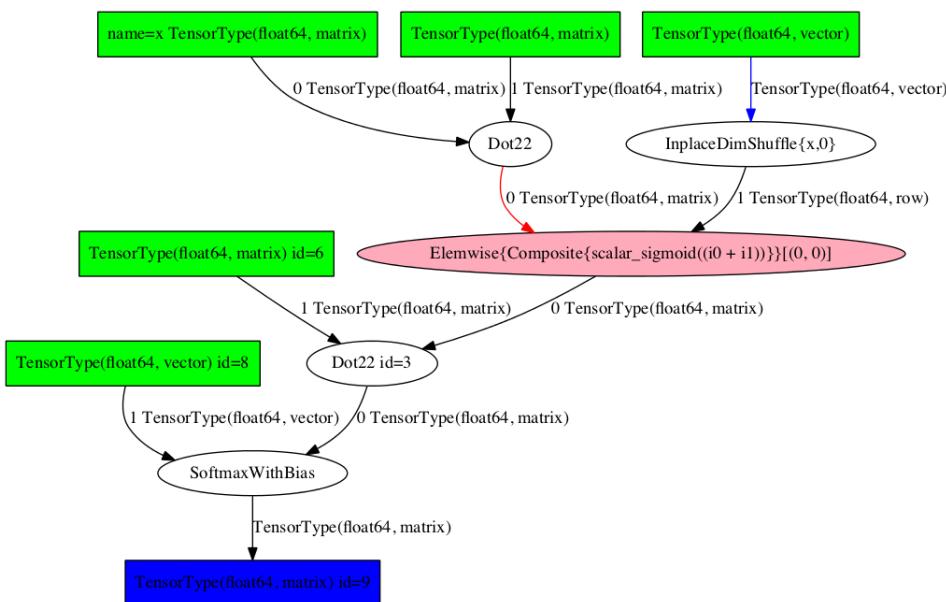
- implementácia v novom jazyku, Network Description Language (NDL)
- nedostatok možností a nástrojov pre vizualizácie

¹⁵<https://docs.microsoft.com/en-us/cognitive-toolkit/>

¹⁶<https://dzone.com/articles/progressive-tools10-best-frameworks-and-libraries>

3.4.3 Theano

Theano¹⁷ je veľmi silná knižnica umožňujúca definovanie, optimalizáciu a evaluáciu numerických operácií nad multidimenzionálnymi poliami s obrovskou efektivitou. Podobne ako TensorFlow, k abstrakcii výpočtov používa grafy, ako možno vidieť na príklade na obrázku 8.



Obr. 8: Príklad grafovej abstrakcie výpočtov použitím framework-u Theano¹⁸

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

Medzi hlavné výhody patrí:

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

Za najväčšie nevýhody sú pokladané:

¹⁷<https://github.com/Theano/Theano>

¹⁸<http://www.wildml.com/2015/09 speeding-up-your-neural-network-with-theano-and-the-gpu/>

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

3.4.4 Keras

Ďalší open-source framework pre prácu s neurónovými siet'ami, avšak na rozdiel od predošlých troch nie je vypracovaný ako koncové riešenie pre strojové učenie. Namiesto toho slúži ako rozhranie a poskytuje vyššiu úroveň abstrakcie pre jednoduchšie používanie ostatných framework-ov, z ktorých momentálne pre použitie ako backend podporuje TensorFlow a Theano. Myšlienka stojaca za celým projektom je: "*Byť schopný pretaviť myšlienku na výsledok s čo najmenším zdržaním je klúčom k dobrému výskumu*"¹⁹, čo bude aj jedným z dôvodov prečo je práca s ním jednoduchšia.

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

Jeho hlavnými výhodami sú:

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

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

3.4.5 Spark MLlib

Škálovateľná knižnica pre strojové učenie, široko využívaná najmä v distribuovaných systémoch hlavne kvôli svojej efektivite. Veľmi jednoducho je ju možné

¹⁹<https://keras.io/>

pripojiť do Hadoop workflow-u, poskytuje množstvo algoritmov pre strojové učenie optimalizovaných pre výpočty v už spomínaných distribuovaných systémoch na dátach vo veľkom meradle²⁰.

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

Najväčšími výhodami sú:

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

Za nevýhody možno považovať:

- strmá krivka učenia
- jednoduché používanie formou "pripoj a hraj" (z angl. plug-and-play) dostupné iba pre Hadoop

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

²⁰<https://spark.apache.org/mllib/>

4 Existujúce modely vizuálnej pozornosti

Existujúce modely vizuálnej pozornosti možno z pohľadu typu predikovaných máp výraznosti rozdeliť na modely k predikcii pozornosti zdola nahor (kapitola 4.1) a modely k predikcii pozornosti zhora nadol (kapitola 4.2). Ako prínosná sa ale javí aj detekcia objektov (kapitola 4.3) a modely postavené práve na nej. Ďalším zaujímavým rozdelením však je rozdelenie podľa použitých princípov a technológií [36], na modely:

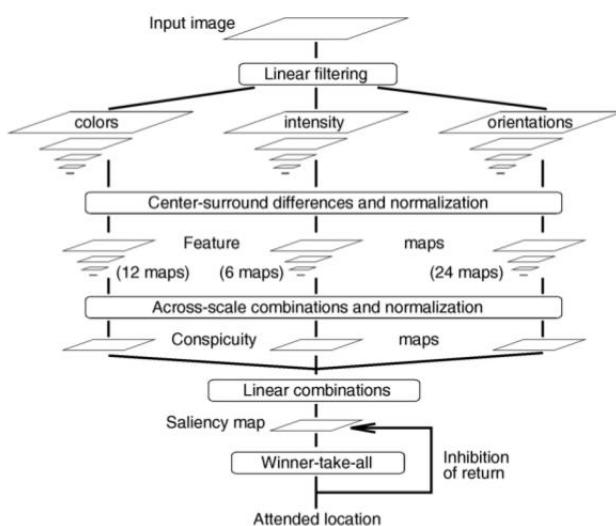
- hierarchické - využívajú hierarchické rozkladanie príznakov
- Bayesove - využívajú kombináciu výraznosti s predchádzajúcimi znalosťami
- rozhodovaco-teoretické - využívajú diskriminačnú teóriu výraznosti
- informaticko-teoretické - využívajú maximalizáciu informácie z daného prostredia
- grafické -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

4.1 Modely k predikcii pozornosti zdola nahor

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

4.1.1 Itti-ho model

Itti-ho model[13] je jedným z najznámejších modelov vizuálnej pozornosti, i keď patrí medzi tie staršie, dodnes je široko používaný a citovaný v mnohých článkoch. Je to biologicky inšpirovaný bottom-up model, ktorý využíva hierarchické rozloženie vlastnosí a ich kombináciu do výslednej mapy výraznosti (z angl. saliency map). Ako je vidieť na obrázku 9, zo vstupného obrázka sa vytvoria 3 typy máp a to podľa farby, intenzity a orientácie, ktorých kombináciou sa dosiahne už spomenutá mapa výraznosti.



Obr. 9: *Itti-ho hierarchický model vizuálnej pozornosti[13]*

4.1.2 Metóda hierarchickej výraznosti

Pri použíti metódy hierarchickej výraznosti (z angl. hierarchical-saliency method[14]) sa v podstate jedná o detekciu charakteristík v prirodzenej scéne. Tento postup je istým vylepšením predchádzajúceho Ittiho modelu a prakticky pozostáva z nasledujúcich dvoch krov:

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

1. výpočet mapy výraznosti S zo vstupného obrázka I
 2. evaluácia oblastí záujmu (z angl. regions of interest, ROIs), automatická klasifikácia všetkých pixelov do dvoch kategórií k získaniu masky M :
 - globálne výrazné oblasti (I)
 - zvyšok (O)
 3. vynásobenie masky M so vstupným obrázkom I k získaniu filtrovaného obrázka I'
- *druhý krok* - evaluácia lokálne výrazných častí vo vnútri globálne výrazných oblastí. Použije sa vyfiltrovaný obrázok I' z predchádzajúcich krokov k získaniu novej mapy výraznosti S' , ktorá je finálnou mapou.

4.1.3 Model pre určovanie salientných oblastí webových stránok

Veľmi zaujímavým riešením pre predikciu a určovanie salientných oblastí scény v doméne webových stránok je model od Chengyao Shen a Qi Zhao[40] využívajúci metódy strojového učenia. Model je postavený na použití metódy MKL (Multiple Kernel Learning - kombinuje viacero jadier SVM (Support Vector Machine) miesto jedného), ktorá bola trénovaná na princípe binárneho regresného problému. Rozdiel oproti predošlým popisovaným modelom je však v tom, že predikovaná nebola priamo výsledná mapa výraznosti ale iba vektor pohľadov (fixácií) na vstupné obrázky, na základe ktorého bola potom spomínaná mapa vypočítaná. Autori po vyhodnotení výsledkov zistili že ako najviac salientné oblasti sa v scéne javia v prvom rade ľudská tvár, oči a celkovo vrchná časť ľudského tela. Zaujímavým vylepšením ich modelu však bolo to, že tieto zistenia zapracovali formou predtrénovania MKL na tieto vysoko salientné oblasti.

4.2 Modely k predikcii pozornosti zhora nadol

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

4.2.1 Zameranie na emočný obsah

Niekol'ko riešení sa v posledných rokoch pokúsilo o predikovanie pozornosti zhora nadol (top-down) na základe emočného obsahu v pozorovanej scéne. Z nich stojí za zmienku napríklad model od Liu a spol. [31], ktorí sa zamerali na spomínaný emočný obsah v rôznych oblastiach (blokoch) obrázka, kde do úvahy brali:

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

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

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

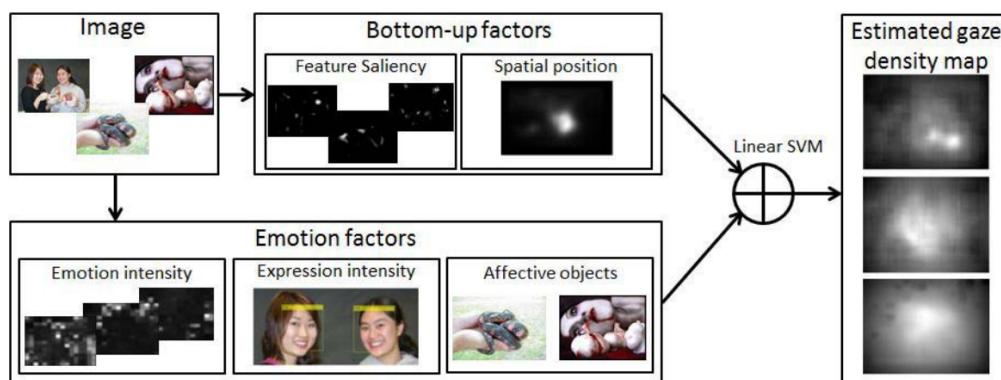
Ich narhované riešenie malo niekoľko častí. Prvou je extrakcia faktorov pozornosti zdola nahor (bottom-up) vo forme nízko-úrovňových máp výraznosti založených na črtách a mapy priestorovej polohy s dôrazom na centrum pre výpočet odhadu hustoty pohl'adu (z angl. gaze density estimation). K tomuto boli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

Druhou časťou je detekcia emócií, k čomu bol použitý emočný detektor založený na učení s použitím učenia viacerých inštancií (z angl. multiple instance learning, MIL) zo slabo oštítkovaných dát. Tými boli obrázky označené emočným typom²¹, ktorý bol určený po vypočítaní pravdepodobnosti vyskýtu jednotlivých typov pre každú časť obrázka. Touto matematickou operáciou sa autorom podarilo úspešne vyriešiť problém určenia dominantného typu emócie pre obrázok. Ďalej bol pre zlepšenie extrakcie emócií použitý mechanizmus založený na SVM, ktoré bolo použité na detekciu ľudských tvári a odhadnutie intenzity výrazu. Výstupom

²¹jedna z 8 skupín, do ktorých boli rozdelené emócie. Príkladom týchto skupín sú napr. hnev, láska, smútok, prekvapenie, atď.

z tejto časti je emočná mapa, ohodnocujúca intenzitu emócií, určená pre ďalšie spracovanie.

Treťou časťou bolo spracovanie výstupov z predošlých dvoch častí lineárnym SVM, ktoré vo výsledku z faktorov pozornosti zdola nahor (bottom-up) a emočných faktorov vygeneruje výslednú mapu hustoty pohľadu (z angl. gaze density map). Vyššie popísaný postup je zobrazený na diagrame na obrázku 10.



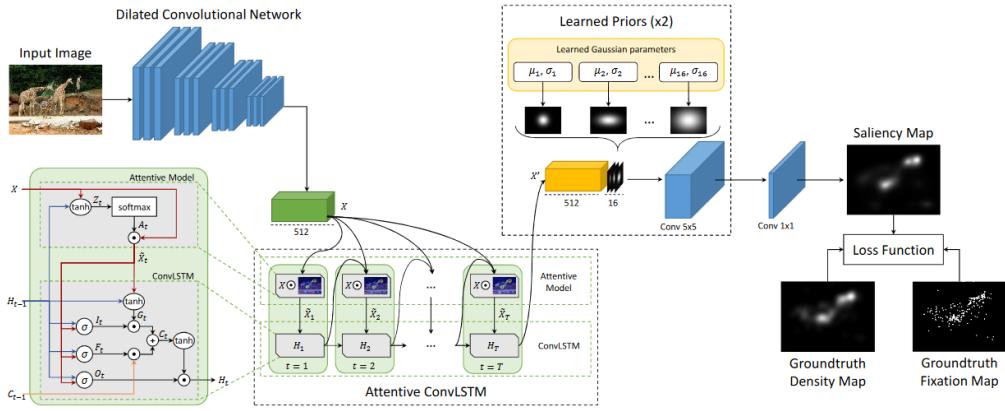
Obr. 10: Diagram modelu od Liu a spol. k predikcii máp výraznosti založených na emočnom obsahu[31]

Za veľkú výhodu tohto riešenia by sa dala pokladat' kombinácia viacerých faktorov (aj pozornosti zdola nahor (bottom-up) aj pozornosti zhora nadol (top-down emočné faktory) do výslednej mapy, vďaka čomu sa viac priblíži k reálnej pozornosti. Ako vieme tú neovplyvňujú čisto len jedny alebo druhé faktory, ale vždy je to ich kombinácia. Ako mierna nevýhoda by sa mohlo javiť rozdelenie emócií iba na 8 typov a určenie ako dominantný typ len jeden z nich, pričom ostatné sa ďalej neberú do úvahy.

4.2.2 SAM

SAM - skratka pre Saliency Attentive Model[11], ďalší z modelov schopných aj predikcie pozornosti zhora nadol (top-down). Založený je na architektúre nazvanej ML-Net (Multi-Level Network[10]), ktorá miesto klasických konvolučných vrstiev využíva dilatované konvolučné vrstvy, ktoré limitujú efekt zmenšovania. Táto architekúra použitá pre spracovanie vstupného obrázka je nasledovaná kon-

volučnými LSTM vrstvami, ich naučené výstupy potom spracováva už klasická konvolučná vrstva, výstupom ktorej je mapa výraznosti. Popisovaný postup je zobrazený na obrázku 11.



Obr. 11: Diagram modelu využívajúceho dilatované konvolučné vrstvy v kombinácii s LSTM vrstvami [11]

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

Spomínané LSTM vrstvy (podrobnejšie zobrazené aj na obrázku 11) využívajú kombináciu aktivačných funkcií *softmax*, *tanh* a *sigmoid*, kedy pri spracúvaní sekvenčne aktualizujú svoj interný stav. Ako vstup dostanú vektor extrahovaných číta z dilatovaných konvolučných vrstiev (X), ktoré spracúvajú vstupné obrázky, a generovanú mapu výraznosti z predošej LSTM vrstvy (s výnimkou prvej). Táto generovaná mapa je ešte upravená mechanizmami pozornosti, ktoré sa selektívne zameriavajú na rôzne časti obrázku. Vo výsledku je potom vektor extrahovaných číta (X) a upravená generovaná mapa (X_t) z predošej vrstvy ešte ďalej spracovávaná aktivačnými funkciami *tanh*.

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

Ďalšou zaujímavou vecou na tomto riešení je zvolenie funkcie chyby (z angl. loss function). Autori nepoužili žiadny z tradičnejších typov, ale použili lineárnu kombináciu troch rôznych evaluačných metrík (popísané v kapitole 6), v tvare:

$$L(y, y^{den}, y^{fix}) = \alpha L1(y, y^{fix}) + \beta L2(y, y^{den}) + \gamma L3(y, y^{den}) \quad (10)$$

Pre vyššie uvedené platí:

y - predikovaná mapa výraznosti

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

y^{fix} - binárna mapa fixácií

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

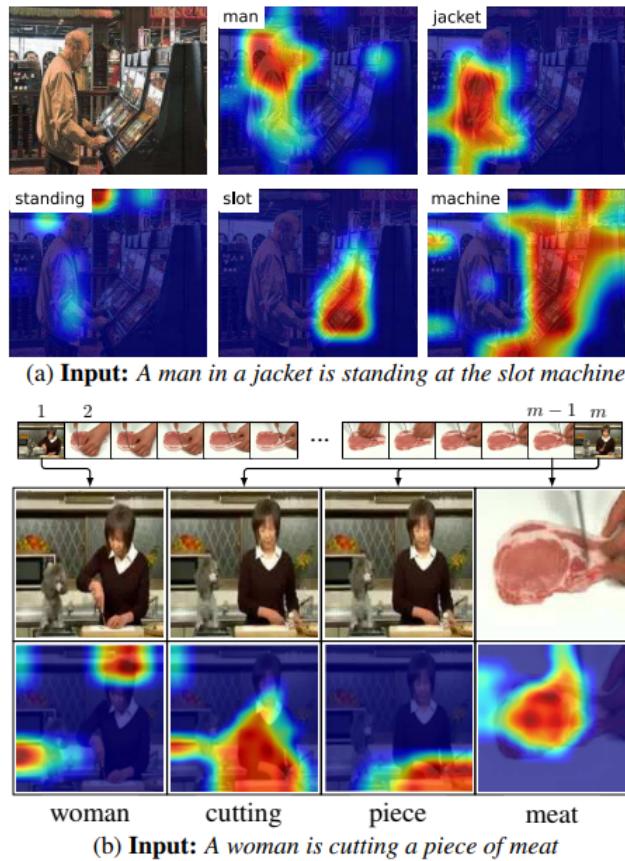
$L1, L2, L3$ - tri funkcie pre výpočet evaluačných metrík, za radom Normalizovaná cesta výraznosti (NSS, z angl. Normalized Scanpath Saliency), Lineárny korelačný koeficient (CC), Kullback-Leiblerova divergencia (KL-Div), všetky bližšie popísané v kapitole 6.

Za veľkú výhodu tohto riešenia pokladáme možnosť predikcie aj oboch typov pozornosti, v závislosti od datasetu, na ktorom je model natrénovaný. Taktiež možnosť nezávislej výmeny submodelov (dilatovaná konvolučná siet', konvolučná LSTM siet') sa javí ako veľmi výhodná, keďže ulahčuje prototypovanie bez nutnosti nového natrénovania celého modelu. Za miernu nevýhodu možno považovať značnú komplexitu modelu a s tým spojené nároky na hardvér, rovnako ako aj na čas nutný na natrénovanie od nuly, v prípade, že nechceme použiť predtrénované modely.

4.2.3 Pozornosť zhora nadol vedená titulkami

Jedným z ukážkových príkladov pozornosti zhora nadol (top-down) je úloha nájdenia nejakého konkrétneho objektu v scéne, čím sa inšpirovali výskumníci z Bostonskej univerzity. Predstavili model²² [37] schopný predikcie máp výraznosti na základe klúčových slov v zadanej vete a príslušnom obrázku alebo videu, príklad predikcií možno vidieť na obrázku 12.

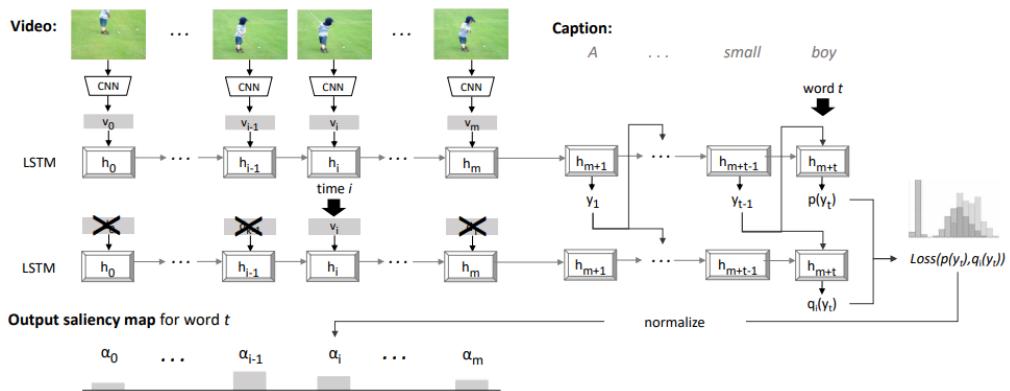
²²<http://ai.bu.edu/caption-guided-saliency/>



Obr. 12: Predikcia salientných oblastí na základe kl’účových slov vo vete, hore po a predikcia pre obrázok, dolu po b predikcia pre video[37]

Autori použili prístup nazvaný titulkami vedená vizuálna pozornosť’ (z angl. Caption-Guided Visual Saliency), ktorý produkuje mapy výraznosti pre nepohybливé obrázky alebo video. Ako základný model použili LSTM enkóder-dekóder (z angl. encoder-decoder), ktorý predikuje aj dočasné (z videa) aj priestorové (z obrázkov) salientné mapy na základe kl’účových slov vo vete (vstupné titulky, popis) a mapuje ich na vstupné obrázky (alebo video). Vstupné titulky sú spracovávané LSTM vrstvami. Na obrázku 13 je možné vidieť’ načrtnutý model riešenia. Ako prvá vstupná vrstva funguje konvolučná neurónová sieť’ reprezentujúca enkóder pre video, ktorá "zakóduje" reprezentáciu obrázkov aj s aktiváciami všetkých vizuálnych konceptov detekovaných vo videu. Tieto informácie posunie d’alej ako vektor pre LSTM vrstvy reprezentujúce dekóder - ten rozhoduje o tom, ktoré časti použije

pomocou LSTM výstupných brán k predikcii mapy výraznosti pre kl'účové slovo v čase t . Autori d'alej zvolili veľ'mi šikovné riešenie pre predikciu máp len pre obrázky - jednoducho výstup produkovaný poslednou konvolučnou vrstvou zmenili na tzv. "dočasnú" sekvenčiu (vektor): $V = (v_1, \dots, v_m)$. Tá je vytvorená sekvenčne scan-ovaním obrázka riadok po riadku z ľavého horného rohu do pravého dolného rohu. Prvá LSTM vrstva potom tento upravený vektor spracuje a posunie d'alej k dekódovaniu na kl'účové slová vo vete.

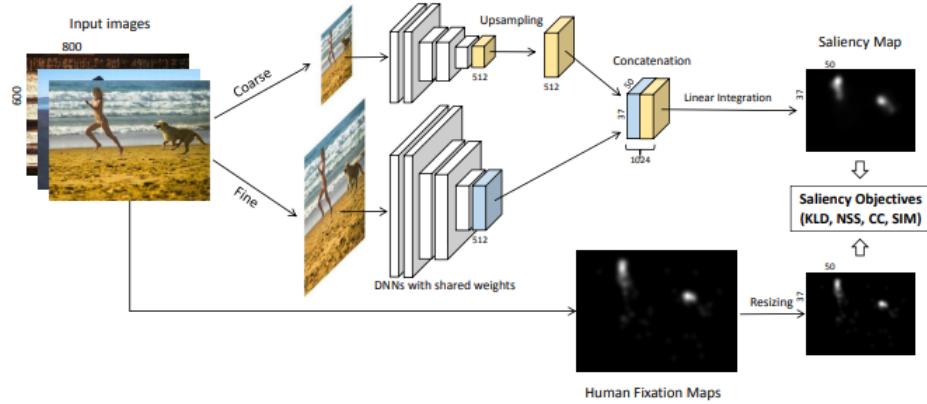


Obr. 13: Diagram modelu neurónovej siete[37], zhora vstup vľavo vo forme videa, vpravo titulky k nemu. V strede LSTM vrsty neurónovej siete, dolu výstupná mapa výraznosti pre kl'účové slová.

Popisané riešenie je ukážkovým modelom pre predikciu pozornosti počas hľadania objektov v scéne. Veľkou výhodou je použitie LSTM vrstiev pre zachytanie predošlého kontextu ako aj možnosť predikcie pre video aj obrázky iba s minimálnymi zmenami.

4.2.4 Redukcia sémantických medzier pri predikcii vizuálnej pozornosti

Riešenie od Huang a spol. [21] sa zaoberala zachytením sémantického kontextu scény ako prvku pozornosti zhora nadol, keďže práve to chýba v starších modeloch pre predikciu vizuálnej pozornosti. Pristúpili k tomu za použitia dvoch neurónových sietí (obrázok 14), ktorých predikcie na konci spojili pomocou spojovacej vrstvy do výslednej mapy vizuálnej pozornosti.



Obr. 14: Náčrt architektúri pri predikcii dvoch sietí so spojením výsledku do jednej mapy vizuálnej pozornosti

Riešenie sa prakticky skladá z 2 častí a to z:

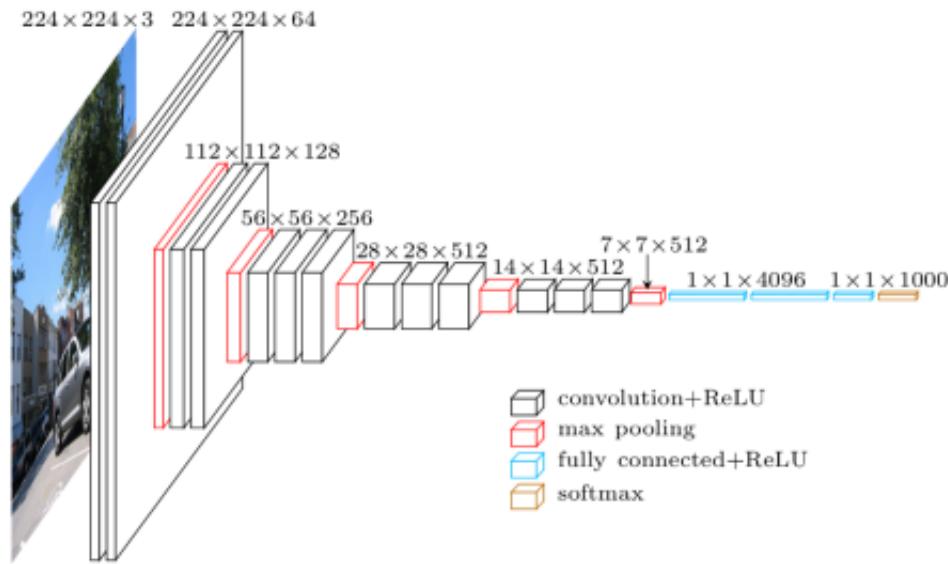
- tzv. "hrubozrnej"(z angl. coarse) - je zameraná na detekciu centier výrazných veľkých regiónov pozornosti, vstupom je menší obrázok
- tzv. "jemnozrnej"(z angl. fine) - je zameraná na detekciu salientných oblastí menšej veľkosti, vstupom je vačší obrázok

Autori experimentovali s architektúrami sietí pre detekciu objektov (s časti popisované v kapitole 4.3) ako napríklad AlexNet [28], VGG-16 [41] a GoogLeNet [46]). Najlepšie predikcie mala kombinácia VGG-16 siete, čím sme sa neskôr pri experimentovaní inšpirovali.

4.3 Detekcia objektov

Z pohľadu modelov vizuálnej pozornosti môžu za zmienku stáť aj modely pre detekciu objektov a ich klasifikáciu, ktorých je dostupné značné množstvo aj predtrénovaných. Ako príklad si môžeme uviesť konvolučnú sieť VGG16[42] určenú práve pre klasifikáciu objektov na obrázku. Skladá sa z 16 konvolučných vrstiev a veľkého množstva filtrov, ako možno vidieť na obrázku 15. Spracováva obrázky o veľkosti 224×224 , bola trénovaná na 4 GPU 2-3 týždne a skladá sa zhruba

zo 138 miliónov parametrov²³. Je častou voľbou pre ďalšie experimentovanie v oblasti pre svoju presnosť a dostupnosť už natrénovaného modelu.



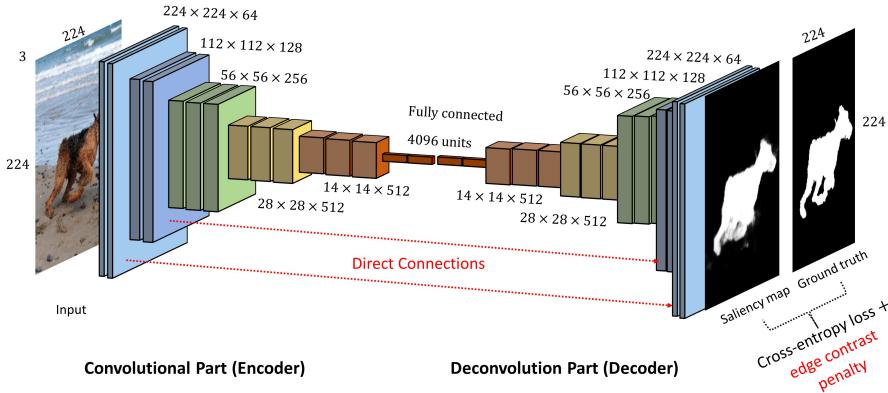
Obr. 15: Štruktúra neurónovej siete VGG-Net²⁴

4.3.1 Autoenkóder s VGG16 siet'ou

Ďalším zaujímavým riešením detekcie objektov v obrázku je autoenkóder od A. Meyer-a[2], ktorý využíva aj vyššie popísovanú siet' VGG16. Využil princíp konvolúcie-dekonvolúcie kedy práve VGG16 funguje ako enkóder a jej v podstate zrkadlová podoba s dekonvolučnými vrstvami funguje ako dekóder (zobrazené na obrázku 16)

²³<https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5>

²⁴<https://www.quora.com/What-is-the-VGG-neural-network>



Obr. 16: Štruktúra autoenkóderu pre detekciu objektov²⁵

Siet' je schopná aj priamych prepojení z konvolučných do dekonvolučných vrstiev a predkuje v podstate binárny obrázok, kde hodnota 1 reprezentuje pixel nájdeného objektu. Veľkou výhodou pri takejto sieti je možnosť použitia predtrénovaného modelu VGG16 pre enkóder.

4.3.2 Vedená a upravená vizuálna pozornosť

Pri natrénovaných modeloch k detekcii objektov vieme pári jednoduchými zmenami docieľiť, aby sme z nich dostali salientné mapy. Ako ukázali Springenberg a spol. [43], stačí zmeniť aktivačné funkcie a upraviť algoritmus spätného šírenia chyby. K tomu môžeme použiť dva spôsoby, vďaka ktorým dostaneme dva typy vizuálnej pozornosti:

- upravená alebo dekonvolučná (z angl. rectified or deconv saliency [52]) - v algoritme spätného šírenia chyby sa odsekávajú negatívne gradienty a propaguje sa len pozitívny gradient pri zvýšení výstupných hodnôt
- vedená vizuálna pozornosť (z angl. guided saliency) - podobne ako pri predošom type sa odsekávajú negatívne gradienty, propaguje sa však pozitívny gradient zodpovedný za pozitívne aktivácie

Po týchto zmenách vieme z vybraných konvolučných vrstiev dostať rôzne aktivácie a črty určitých filtrov, z ktorých potom po ďalších úpravách a spojeniach

²⁵https://github.com/arthurmeyer/Saliency_Detection_Convolutional_Autoencoder

získame salientné mapy. Hoci celkovo tieto mapy nemusia byť veľmi presné, toto riešenie je vhodné ako začiatok pri kritickom nedostatku dát, keďže využíva už natrénované modely detektie objektov. Tie sú voľne dostupné pre najrôznejšie experimentovanie.

5 Datasetsy vizuálnej pozornosti

Pre problém predikcie vizuálnej pozornosti existuje niekoľko voľne dostupných datasetov, ktoré sa líšia kvalitou, veľkosťou obrázkov, počtom fixácií či experimentami, pri ktorých boli zozbierané. Medzi najpoužívanejšie patria:

- CAT2000[5]:
 - obsahuje 4000 obrázkov z 20 rôznych kategórií
 - obsahuje pohľady 24 ľudí počas 5 sekúnd
 - obrázky sú naozaj rôznorodé, sú medzi nimi aj rôzne fraktály a obstraktné umenie
 - veľkosť obrázkov je *1920x1080*
- MIT1003[25]:
 - 1003 obrázkov prírodných a vonkajších scén (779 krajina, 228 portrét)
 - obsahuje pohľady 15 ľudí počas 3 sekúnd
 - veľkosť obrázkov je rôznorodá, od menších až po HD obrázky
- SALICON[24]
 - vizuálna pozornosť v kontexte (z angl. Saliency in Context)
 - obsahuje obrázky z datasetu MS COCO²⁶, tie obsahujú neikonické pohľady a objekty v kontexte
 - pozostáva z 10 000 trénovacích, 5 000 validačných a 5 000 testovacích obrázkov
 - veľkosť obrázkov je *1920x1080*
- DUT-OMRON[50]:
 - obsahuje viac než 5000 obrázkov
 - obsahuje pohľady 5 ľudí za prvé 2 sekundy

²⁶<http://cocodataset.org>

- má odstránené extrémy v dátach (z angl. outliers)
- viac ako 95% obrázkov má viac než 50 fixácií
- všetky obrázky nemajú iba jeden veľký objekt v strede - t.j. fixácie nie sú stále koncentrované v tej istej oblasti
- veľkosť obrázkov je $400x300$
- má dobre štruktúrovane spracované dáta

Z vyššie uvedených sa ako najlepšie javia posledné dva datasety, SALICON a DUT-OMRON, najmä z dôvodu spracovania a obsahu.

6 Metriky pre modely vizuálnej pozornosti

Obvykle sa modely k predikcii vizuálnej pozornosti evalujujú 2 spôsobmi, a to bud' vzhľadom na pohyb očí (resp. fixácie) alebo vzhľadom na originálnu mapu výraznosti. K tomu slúži veľké množstvo metrík [7][9], medzi tie najčastejšie používané patria:

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

$$\frac{1}{n} \sum_{i=1}^n \frac{s(x_h^i, y_h^i) - \mu_s}{\sigma_s} \quad (11)$$

- 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 výraznosti je potom pristupované ako k binárному klasifikátoru na separáciu pozitívnych vzoriek 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 oblast' 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.

$$CC(s, h) = \frac{cov(s, h)}{\sigma_s \sigma_h} \quad (12)$$

- KL-Div - Kullback-Leiblerova divergencia[9], všeobecné teoretické meranie rozdielu medzi dvoma pravdepodobnostými distribúciami. V oblasti predikcií vizuálnej pozornosti sa často nazýva aj AUC-Judd. Ako vstup berie mapu výraznosti P a mapu fixácií (z angl. ground truth fixation map) Q^D , evaluuje stratu informácie keď je P použité k aproximácii Q^D .

$$KL(P, Q^D) = \sum_i Q_i^D \log \left(\epsilon + \frac{Q_i^D}{\epsilon + P_i} \right) \quad (13)$$

- SIM - podobnosť (z angl. similarity), meria resp. vyjadruje podobnosť medzi dvoma distribúciami zobrazenými ako histogram. Počíta sa ako suma minimálnych hodnôt v každom pixeli po normalizácii.

$$SIM(P, Q^D) = \sum_i \min(P_i, Q_i^D) \quad (14)$$

$$kde \sum_i P_i = \sum_i Q_i^D = 1$$

7 Návrh

Na základe analýzy problémovej oblasti a existujúcich riešení sme sa rozhodli najprv použiť konvolučnú neurónovú sieť (jej popis a architektúra v kapitole 7.2), čo sa pretavilo aj do prvotných experimentov (kapitola 8.2).

Ďalej sme navrhli autoenkóder k predikcii vizuálnej pozornosti (kapitola 7.3) a pokúsili sa použiť už predtrénovaný model VGG16 siete s autoenkóderom od A. Meyer-a (oba popisované v 4.3) na našom predpripravenom datasete.

Neskôr sme v rámci pridávania dodatočných (top-down) informácií o scéne, akými sú napr. poloha konkrétnych objektov, vytvorili model, ktorý kombinuje VGG16 sieť s autoenkóderom. Jeho architektúra a princípy sú popísané v kapitole 7.4.

7.1 Dataset

Podarilo sa nám nájsť veľké množstvo datasetov, ktoré by sa potencionálne dali použiť pre našu neurónovú sieť, ako napr. CAT2000[6], NUSeF²⁷, či DUT-OMRON[50] (popísané aj v kapitole 5) Pôvodná myšlienka bola vytiahnutá z každého čo najlepšie vzorky (odstrániť rôzne abstraktné umenie, fraktály, atď.) a dať tak dokopy jeden veľký dataset, na ktorom by bolo možné experimentovať. To sme aj naozaj zrealizovali a takýto dataset použili pri prvotných experimentoch popísaných v kapitole 8.2.

Neskôr sa ale postupne ukázalo, že takto zostrojený dataset zložený z viacerých menších má značné nedostatky. Najväčšími problémami boli:

- rozdielna kvalita a veľkosť obrázkov
- rozdielna dĺžka pohľadov ľudí na obrázky (2, 5, 10 sekúnd)
- rozdielny počet fixácií pre obrázky
- rozdielnosť experimentov, pri ktorých sa dátá zbierali (voľné sledovanie, voľné sledovanie s detekciou anomálií, zapamätanie si scény, atď.).

²⁷<http://mmas.comp.nus.edu.sg/NUSeF.html>

- rozdielne zariadenia pre zachytenie fixácií s rôznou vzorkovacou frekvenciou

Z vyššie uvedených dôvodov sme sa preto rozhodli vybrať iba jeden dataset. Volba padla na SALICON, pretože je dosť rozsiahly, dobre spracovaný a hlavne obsahuje rôznorodé scény s kontextom. Z datasetu sme teda vybrali iba anotovanú vzorku dát (15 000 obrázkov) aby sme boli schopní vyhodnotiť predikcie metrikami vizuálnej pozornosti. Pre experimentovanie bol dataset rozdelený tradične v pomere 80 : 10 : 10 (trénovanie : validácia : testovanie).

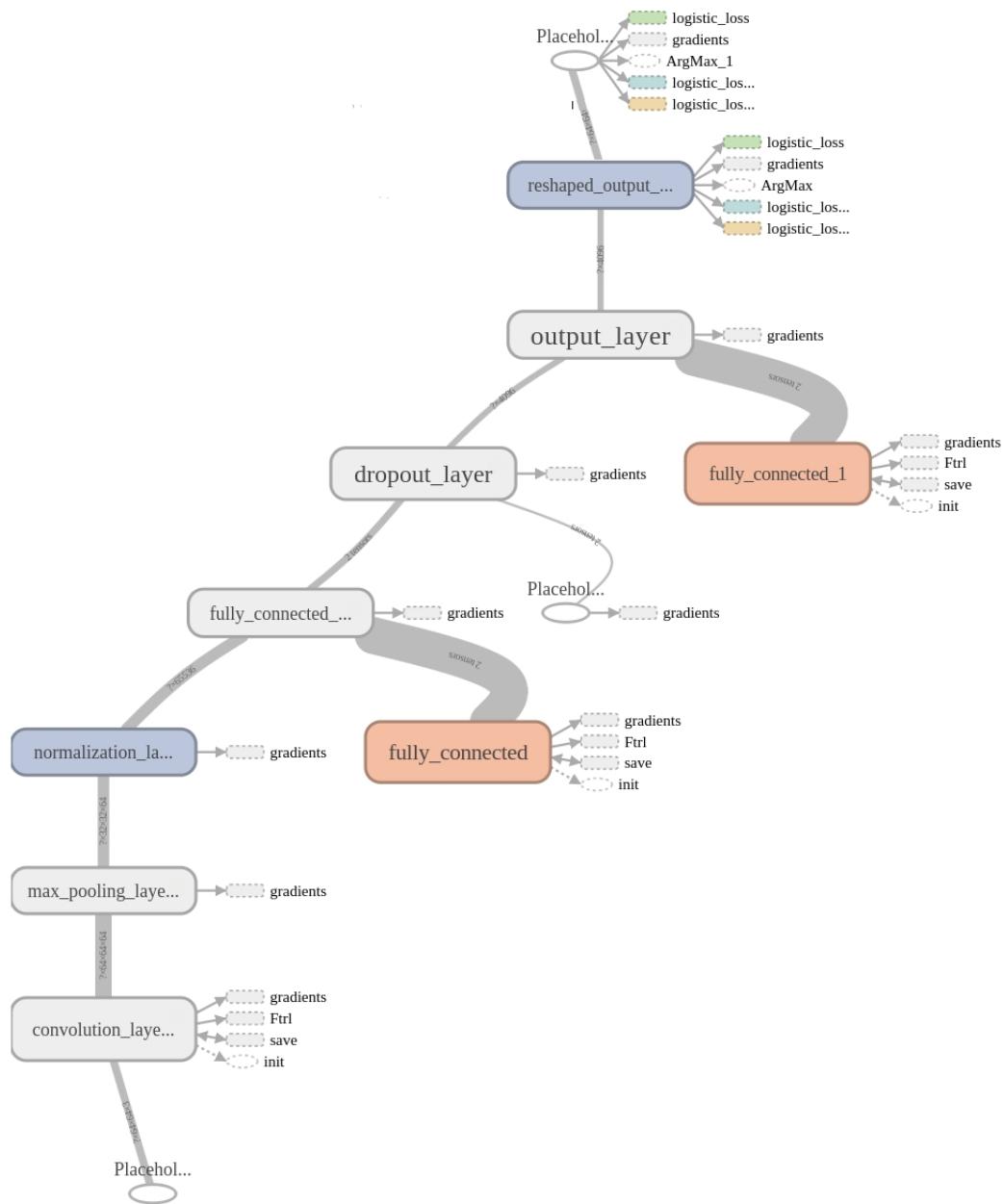
7.2 Konvolučná neurónová siet²⁸

Celá architektúra je načrtnutá na schéme na obrázku 17 vytvorenej pomocou nástroja TensorBoard²⁸.

Jedná sa o jednoduchú sieť so vstupnou konvolučnou vrstvou pre spracovanie obrázkov. Táto vrstva obsahuje konvolučný filter (veľkosť 5×5) s aktivačnou funkciou sigmoid. Výstup z nej ďalej pokračuje do vrstvy združovania, kde sa použije operácia MAX s filtrom o veľkosti 2×2 a krokom tiež s veľkosťou 2. Po nich nasleduje vrstva normalizácie, kde je celý výstup zlúčený do jednej širokej vrstvy. Za ňou sa nachádza plne prepojená vrstva (z angl. fully-connected layer) s aktivačnou funkciou sigmoid a vrstva výpadku (z angl. dropout layer[44]), ktorej hodnota (v rozmedzí od 0 do 1) určuje, aké percentuálne množstvo neurónov aj s prepojeniami bude dočasne skrytých. Táto možnosť umožňuje počas učenia sa predchádzat pretrénovaniu. Za vrstvou výpadku už nasleduje iba výstupná vrstva a jej transformácia na 2D maticu, obrázok predstavujúci mapu výraznosti, ktorú chceme dostať.

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

²⁸https://www.tensorflow.org/get_started/summaries_and_tensorboard/



Obr. 17: Diagram reprezentujúci architektúru neurónovej siete, zdola vstupná konvolučná vrstva nasledovaná ostatnými vrstvami siete až po výstupnú, spolu s transformáciou na 2D maticu reprezentujúcu predikovanú mapu výraznosti pre vstupný obrázok

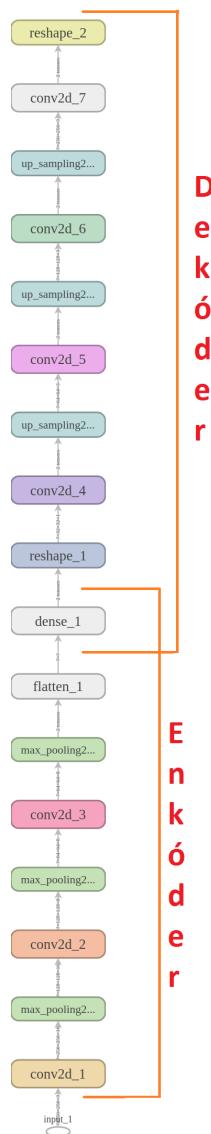
7.3 Autoenkóder

Ďalším z typov sietí, ktoré sme sa rozhodli otestovať je autoenkóder, konkrétnie jeho variácia s konvolučnými vrstvami pre spracovanie obrazu. Jeho úloha je však trochu iná oproti klasickému čo najlepšiemu zrekonštruovaniu vstupu na výstupnej vrstve, miesto toho bude z enkódovaných (komprimovaných dát) predikovať mapy vizuálnej pozornosti. To dosiahneme použitím algoritmu učenia spätného šírenia chyby, kedy ako vstupné dátá budú sieti poskytnuté obrázky a očakávanými výstupmi budú práve mapy vizuálnej pozornosti.

Autoenkóder sa klasicky skladá z dvoch častí, prehľadná štruktúra je zobrazená na obrázku 18. Prvou je enkóder, ktorý tvoria 3 konvolučné vrstvy, každá so svojou vlastnou vrstvou združovania, nasledované normalizačnou vrstvou a jednou plne prepojenou vrstvou (z angl. fully-connected layer). Konvolučné vrstvy majú každá filter o veľkosti 3×3 a aktivačnú funkciu ReLU (popísaná v kapitole 3.1), vrstvy združovania podobne ako pri predchádzajúcim type siete používajú operáciu MAX s filtrom o veľkosti 2×2 a krokom rovnako s veľkosťou 2. Cieľom týchto vrstiev je prakticky extrahovať relevantné črty a vstupný obrázok zmenšiť, resp. komprimovať, čo d'alej zabezpečuje spomínaná normalizačná a plne prepojená vrstva (bez aktivačných funkcií), ktorá už ale vstupný obrázok reprezentuje len ako komprimované jednorozmerné pole.

Druhou časťou spomínaného autoenkóderu je tzv. dekodér, ktorý sa snaží prakticky zrkadlovými operáciami získať informácie z komprimovaných dát, vďaka čomu je schopný predikovať mapy vizuálnej pozornosti. Jeho vstupnou vrstvou je v podstate výstupná vrstva z enkóderu, ktorej tvar ale musí byť najprv zmenený pre nasledujúce konvolučné vrstvy. Tie sú dokopy štyri, prvé tri konvolučné vrstvy obsahujú filter s veľkosťou 3×3 a aktivačnú funkciu ReLU. Každá z nich je ešte navyše nasledovaná vlastnou vrstvou prevzorkovania (z angl. upsampling layer), ich cieľom je postupná rekonštrukcia do tvaru vstupného obrázku. Obsahujú vzorkovací faktor 2, na rozdiel od vrstiev združovania v enkóderi však miesto operácie MAX (ktorá dáta, resp. obrázok zmenší) duplikujú riadky a stĺpce poskytnutých dát (obrázka), vďaka čomu sa zväčší o vzorkovací faktor. Posledná výstupná konvolučná vrstva obsahuje filter s veľkosťou 5×5 a aktivačnú funkciu sigmoid. Dáta na nej je ale ešte nutné pretransformovať na 2D maticu, reprezentujúcu nami

požadovanú mapu výraznosti. Spomínané trénovanie pomocou algoritmu spätného šírenia chyby využíva Adadelta optimizér (popísaný v časti 3.2.5).



Obr. 18: Diagram zobrazujúci architektúru autoenkóderu skladajúceho sa z dvoch častí - enkóderu a dekóderu. Postupne zdola prvá vstupná konvolučná vrstva nasledovaná ostatnými vyššie popísanými vrstvami enkóderu, potom dekóder so svojimi vrstvami s konvolúciou a prevzorkovaním až napokon úplne hore výstupná vrstva s predikciou mapy vizuálnej výraznosti.

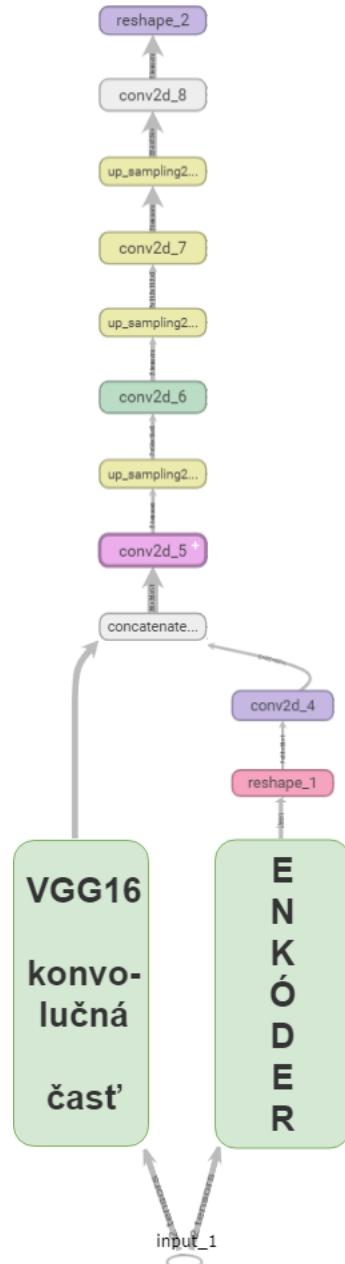
7.4 Kombinácia VGG16 siete s autoenkóderom

Pre experimentovanie so zapojením faktorov pozornosti zhora nadol sme návrhli siet', ktorá kombinuje predikcie vyššie navrhnutého autoenkóderu s detekciou objektov VGG16 siete. Náčrt architektúry je na obrázku 19, podrobnejší diagram sa nachádza v prílohe E.

Celá siet' je prakticky zložená z 3 častí:

- enkóder - časť siete zodpovedná za detekciu salientných častí, je totožná s enkóderom v 7.3 na obrázku 18.
- VGG16 konvolučná časť - obsahuje prvých 10 konvolučných vrstiev VGG16 siete (diagram na obrázku 15) spolu s príslušnými vrstvami združovania (z angl. max pooling). Je zodpovedná za detekciu objektov.
- dekóder - časť, v ktorej sa postupne spájajú a upravujú predikcie z predošlých dvoch častí. Rovnako ako v návrhu z kapitoly 7.3, obsahuje štyri konvolučné vrstvy, prvá však nie je nasledovaná vrstvou prevzorkovania, ale vrstvou spájania - tá zabezpečuje spojenie predikcií z enkóderu a VGG16 konvolučnej časti. Po nej sú ďalej upravované pomocou ďalších dvoch konvolučných vrstiev nasledovaných vrstvou prevzorkovania, rovnako ako pri spomínanom predošom návrhu. Tiež obsahujú aktivačnú funkciu ReLU a filter o veľkosti 3×3 . Posledná výstupná konvolučná vrstva využíva filter s veľkosťou 5×5 a aktivačnú funkciu sigmoid, nasledovaná je už len vrstvou, na ktorej sa dátá pretransformujú na 2D maticu reprezentujúcu predikovanú mapu výraznosti.

Pri trénovaní sme využili prenos učenia (popísaný v kapitole 3.2.4), kedy vrstvy VGG16 konvolučnej časti boli už predtrénované k detekcii objektov. Počas trénovania celej siete sme potom experimentovali s vypnutím učenia pre spomínanú časť siete, keďže sme chceli zachovať kontext detekcie objektov. Samotné trénovanie prebiehalo pomocou algoritmu spätného šírenia chyby s použitím Adadelta optimizéru.



Obr. 19: Náčrt architektúry VGG16 siete s autoenkóderom, VGG16 konvolučná časť obsahuje 10 konvolučných vrstiev pre detekciu objektov, enkóder je identický s enkóderom použitým v 7.3, podrobná architektúra je v prílohe E

8 Experimenty

V priebehu práce bolo nutné experimentovať v naozaj veľkom množstve s najrôznejším počtom vecí od typov neurónových sietí, datasetov až po rôzne prístupy k riešeniu zadaného problému.

Prvotné experimenty vykonávané v rámci predmetu Počítačové videnie²⁹ sú zachytené v kapitole 8.2. Ďalšie experimenty potom prebiehali postupne s konvolučnou neurónovou siet'ou (kapitola 8.3), konvolučným autoenkóderom (kapitola 8.4), autoenkóderom s predtrénovanou siet'ou VGG16 (kapitola 8.5) a autoenkóderom v kombinácii s VGG16 siet'ou (kapitola 8.6). Rozdiel pri modeloch s VGG16 siet'ou je ten, že prvý spomínaný využíva VGG16 ako enkóder, v druhom je zapojená nezávisle ako samostatná časť. Pri každom modeli sme odskúšali nespočetné množstvo rôznych konfigurácií, pre popis sme ale vybrali len tie s najlepšími výsledkami pre jednotlivé časti. V závere v kapitole 8.8 uvádzame porovnanie modelov použitých pri experimentoch pomocou metrík a výber toho najlepšieho.

8.1 Implementačné prostredie

Všetky nižšie popisované experimenty boli naimplementované v jazyku Python za použitia najrôznejších knižníc, hlavne pre prácu s obrázkami a neurónovými siet'ami (bližšie popísané v prílohe C). Najdôležitejšími knižnicami sú:

- TensorFlow³⁰ - open-source knižnica pre prácu s neurónovými siet'ami a strojovým učením
- Keras³¹ - vysoko úrovňová knižnica pre prácu s neurónovými siet'ami, beží na nižšie úrovňovými knižnicami ako Theano či TensorFlow
- Matplotlib³² - knižnica pre 2D vykresľovanie v Python-e

²⁹<http://vgg.fit.stuba.sk/teaching/computer-vision/>

³⁰<https://www.tensorflow.org/>

³¹<https://keras.io/>

³²<https://matplotlib.org/>

- OpenCV³³ - open-source knižnica pre počítačové videnie a strojové učenie
Trénovanie modelov v experimentoch prebiehalo na GPU.

8.2 Prvotné experimenty

Pre prvotné experimenty sme sa rozhodli zvoliť problém predikcie vizuálnej pozornosti v častiach obrázkov, konkrétnie v tzv. regiónoch záujmu (z angl. regions of interest, ROIs), ktoré sme zvolili v okolí fixácií na obrázky.

8.2.1 Úprava datasetu

V tejto fáze prvotných experimentov sme pracovali s datasetom zloženým z niekol'kých voľne dostupných datasetov vizuálnej pozornosti (CAT2000[6], NUSeF³⁴, ...). Keďže vo viacerých z nich chýbali úplné informácie k výpočtu máp výraznosti, rozhodli sme sa ich získať z dostupných obrázkov máp výraznosti, kedy sme ich načítali ako jednofarebný obrázok v odtieňoch sivej (z angl. grayscale) - hodnota pixelu vtedy prakticky reprezentuje intenzitu. Ďalej sme spolu pre vstupné obrázky a mapy výraznosti extrahovali regióny záujmu, ktoré sme zvolili v okolí fixácií - vizualizácia popisovanej extrakcie je zobrazená na obrázku 20. Nevyhovujúce časti datasetov (ako napr. abstraktné umenie, fraktály, cartoon obrázky, ...) boli odfiltrované.



Obr. 20: Vizualizácia extrakcie regiónov záujmu, vľavo obrázok, vpravo mapa výraznosti k nemu

Takto získané dátá boli ďalej pre neurónovú sieť normalizované, dokopy sme

³³<https://opencv.org/>, <https://pypi.org/project/opencv-python/>

³⁴<http://mmas.comp.nus.edu.sg/NUSEF.html>

si nakoniec zaistili zhruba 500 000 vzoriek.

8.2.2 Výsledky

Navrhovanú konvolučnú neurónovú sieť (kapitola 7.2) sme trénovali na priravenom datasete, ktorý bol rozdelený štandardne v pomere 80:10:10 (80 - trénovanie, 10 - validácia, 10 - testovanie). Validácia prebiehala po každej iterácii a trénovanie končilo v momente keď sa chyba na validačných dátach začala výrazne odlišovať oproti najnižšej dosiahnutej (pomaly dochádzalo k pretrénovaniu). V závere mala siet' chybu predikcie na testovacích dátach na úrovni 0.29, chyba bola počítaná ako priemer chýb v každom bode obrázka. Na obrázku 21 možno vidieť porovanie predikovaných máp výraznosti s originálnymi a so vstupnými obrázkami.



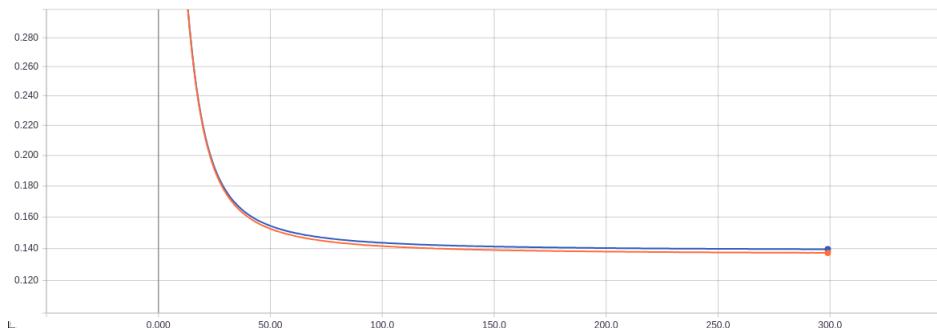
Obr. 21: Porovnanie predikcií (dolu) s originálnymi mapami výraznosti (v strede) voči vstupným obrázkom (hore)

Pri snahe vypočítať metriky pre predikcie sme narazili na problém, ktorý sme si na začiatku neuvedomili. Väčšina metrík evaluuje mapu výraznosti voči binárnej matici reprezentujúcej fixácie na obrázok. Vzhľadom na to, že našim vstupom boli regióny záujmy v okolí fixácií, vo väčšine prípadov tieto binárne matice obsahovali len jednu fixáciu. Vďaka tomu mali metriky (AUC, sAUC, NSS) nezmyselne vysoké hodnoty. Z tohto dôvodu ich teda považujeme za nerelevantné a jediná metrika, podľa ktorej sa môžeme riadiť, je korelačný koeficient, keďže

ten evaluuje predikovanú mapu voči tej pôvodnej. Jeho hodnoty na testovacích dátach sa v priemere pohybovali na hranici 0.563.

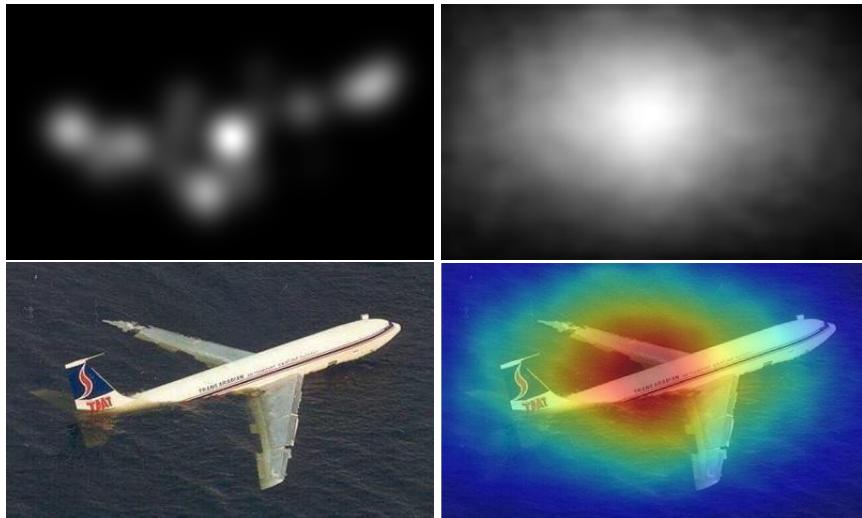
8.3 Konvolučná neurónová siet^{*}

Ďalším experimentom bolo použitie klasickej konvolučnej neurónovej siete rovnako ako v predchádzajúcim prípade, tentokrát ale na celých obrázkoch. Ako dataset bol použitý spracovaný DUT-OMRON dataset (popísaný v kapitole 5) rozdelený podobne v pomere 80:10:10 (80 - trénovanie, 10 - validácia, 10 - testovanie), validácia bola tiež po každej epoche, trénovanie ale končilo v momente keď validačná chyba začala stúpať[†]. Na obrázku nižšie možno vidieť vývoj jednotlivých chýb počas trénovania. Chyba bola počítaná ako priemier chýb predikcií v jednotlivých bodoch obrázka.



Obr. 22: Graf zobrazujúci vývoj chyby neurónovej siete po odstránení extrémov počas 300 epoch - modrá farba reprezentuje chybu na trénovacích dátach, oranžová chybu na validačných dátach

Z vyššie uvedených grafov vyplýva, že siet[‡] bola schopná najvýraznejšie znížiť chybu predikcií behom prvých 50 epoch. Z jej počiatočnej hodnoty približne 0.6709 klesla až na 0.1321 na trénovacích dátach, na validačných dosahovala hodnotu 0.1396. Pri finálnom testovaní bola priemerná chyba približne 0.1459. Na prvý pohľad sa to môže javiť ako solídny výsledok, po vizualizovaní predikcií sme ale zistili, že siet[‡] sa prakticky naučila predikovať ako najvýraznejšiu časť obrázka vždy len jeho stred, príklad predikcie je na obrázku 23.

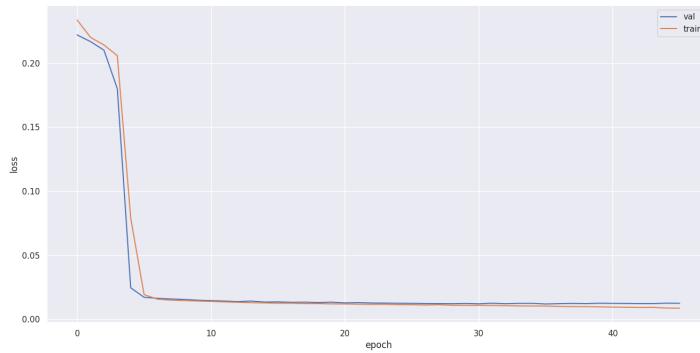


Obr. 23: Vľavo hore originálna mapa vizuálanej pozornosti, vpravo hore predikovaná mapa, vľavo dole obrázok pre prislúchajúce mapy výraznosti, vpravo dole zobrazenie predikcie na obrázku

Vyššie zobrazené správanie môže byť spôsobené napríklad relatívne malou veľkosťou vstupných obrázkov, kvôli veľkej pamäti ovej náročnosti siete.

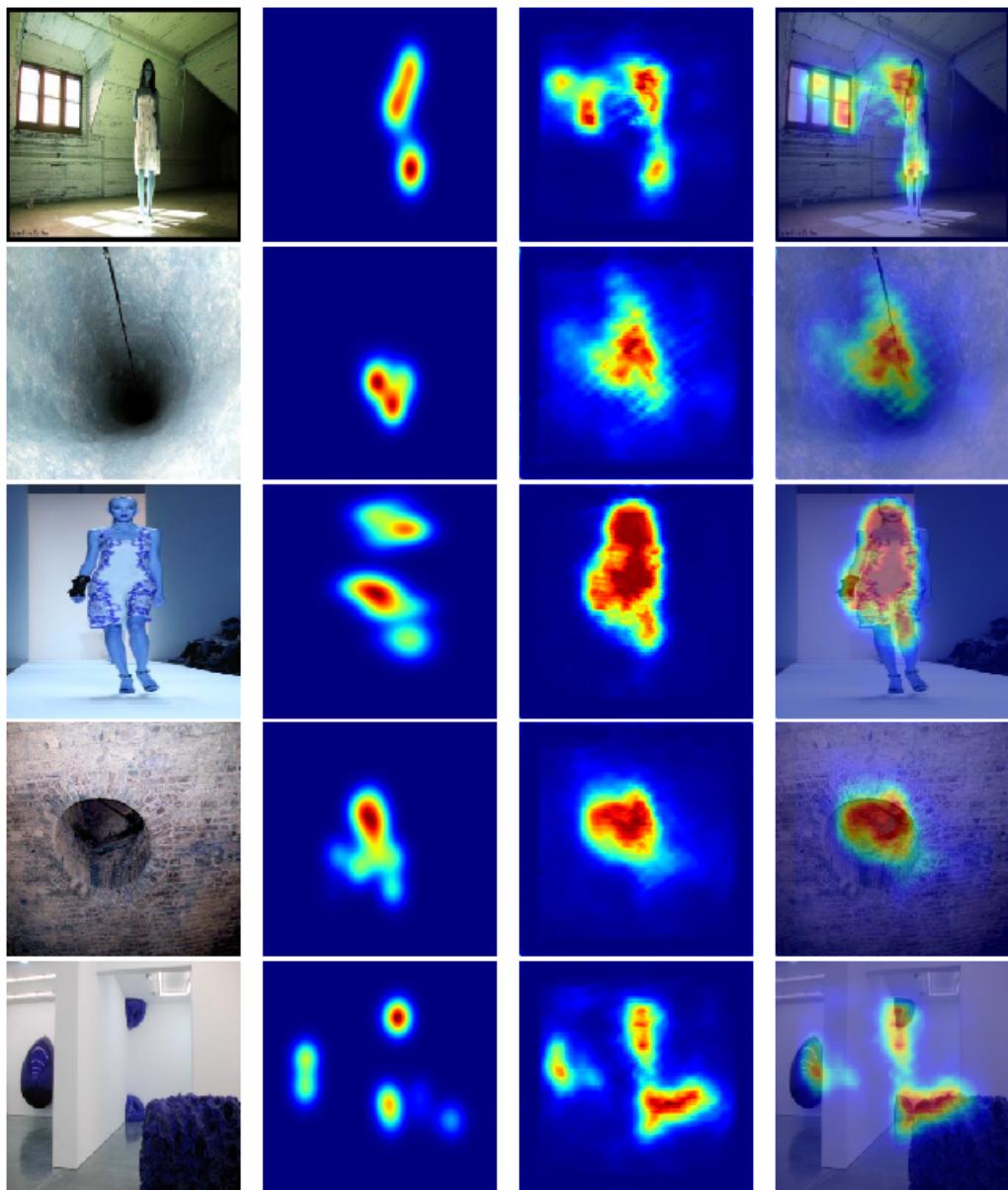
8.4 Konvolučný autoenkóder

Ďalšou architektúrou, ktorú sme sa rozhodli vyskúšať, bol konvolučný autoenkóder (navrhnutý v kapitole 7.3). Použitý dataset bol tento krát SALICON a jeho rozdelenie bolo rovnaké ako v predošlom experimente. Trénovanie ale končilo až keď validačná chyba za posledných n epoch neklesla. Ako funkcia chyby bola použitá priemerná absolútна chyba (z angl. mean absolute error, MAE), ktorá meria rozdiel medzi dvomi spojitémi premennými, v našom prípade medzi reálnou a predikovanou mapou výraznosti, resp. ich pixelmi. Priebeh učenia možno vidieť na obrázku 24.



Obr. 24: Vývoj chyby počas trénovania, oranžovaná reprezentuje trénovaciu, modrá validačnú

Z obrázku je vidieť, že učenie prebiehalo počas 45 epoch, chyba najvýraznejšie klesla počas prvých šiestich epoch. Z počiatočnej hodnoty 0.2336 (trénovacie dát) a 0.2219 (validačné dát) bola schopná klesnúť až na hodnotu 0.00874 (trénovacie dát) a 0.01245 (validačné dát). Priemerná hodnota chyby pri testovaní na dátach, ktoré siet' nikdy nevidela, bola na úrovni 0.01301. Vizualizácie porovnanie predikcií sú zobrazené na obrázku 25.



Obr. 25: Porovnanie predikcií autoenkóderu, zľava vstupný obrázok, potom originálna mapa výraznosti, predikovaná mapa výraznosti a nakoniec predikcia zobrazená na obrázku.

Z vyššie uvedených vizualizácií jasne vidieť, že siet' bola schopná naučiť sa isté výrazné súvislosti z dát, predikcie však nie sú úplne presné. Na rozdiel od siete z predošlého experimentu už nepredikuje všetko do stredu, ale je schopná označiť aj viacero výrazných objektov. To môže byť spôsobené napríklad väčšou

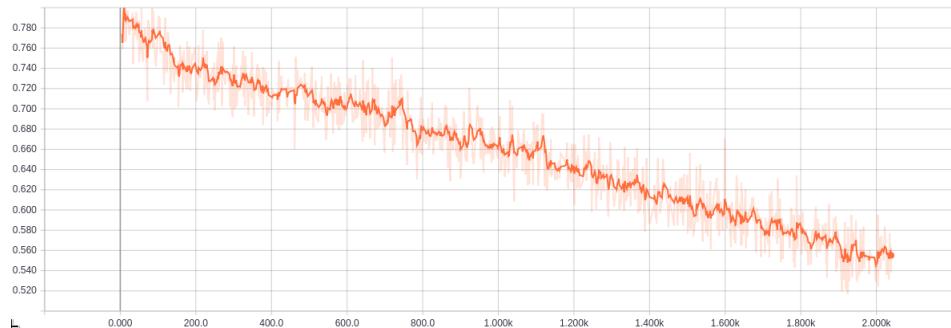
veľkosťou vstupných obrázkov. Nakol'ko autoenkóder si interne vstupné dátá zkomprimuje a zníži ich dimezionalitu, je v porovnaní zo spomínanou sieťou z predchádzajúceho experimentu pri rovnako veľkých vstupných obrázkoch menej pamäťovo náročný.

Zaujímavým javom, ktorý sme si všimli pri experimentovaní s týmto typom siete je, že hodnota chyby predikcie siete pri našom probléme nie je úplne smerodajná informácia. Pri použití napr. priemernej štvorcovej chyby (z angl. mean square error, MSE) bola výsledná chyba *0.0000431*, ale predikcie boli v podstate len prázdne mapy bez akýchkoľvek výrazných častí. Tiež pri použití ešte väčšieho množstva konvolučných vrstiev sa použitá priemerná absolútна chyba zmenšila počas trénovania viac, avšak siet' sa v tomto prípade naučila predikovať najviac výrazné časti obrázkov vždy len do stredu, podobne ako pri konvolučnej neurónovej sieti z predošlého experimentu. Práve preto sme kvalitu modelov a predikcií vyhodnocovali metrikami vizuálnej pozornosti (kapitola 6), ktorých prehľadné porovnanie aj so závermi je uvedené v 8.8.

8.5 Autoenkóder s predtrénovaným modelom VGG16

Pre tento experiment sme si vybrali siet' od A. Meyer-a³⁵ popisovanú v kapitole 4.3, ktorá je voľne dostupná. Jedná sa o variáciu autoenkóderu využívajúceho siet' VGG16 ako enkóder k predikcii binárnej mapy zobrazujúcej dominantné objekty v scéne. Pôvodná hypotéza bola, že vďaka použitému predtrénovanému modelu VGG16 pre detekciu objektov bude siet' lepšie rozumieť vstupným obrázkom a preto pri dodatočnom dotrénovaní k predikcii máp vizuálnej pozornosti bude dávať lepšie výsledky. Po menších úpravách siete sme pristúpili k trénovaniu na našom datasete, postupný vývoj chyby počas trénovania možno vidieť na obrázku 26. Validácia prebiehala každých 100 epoch, trénovanie končilo v momente keď minimálna hodnota chyby za posledných *n* epoch neklesla.

³⁵https://github.com/arthurmeyer/Saliency_Detection_Convolutional_Autoencoder



Obr. 26: Graf zobrazujúci vývoj chyby po odstránení extrémov pri použití predtrénovaného modelu VGG16

Na prvý pohľad sa môže zdať, že siet' sa bola schopná naučiť aspoň nejaké závislosti z dát. Po vizualizovaní dát sme ale zistili, že sme sa veľmi mylili a snaha o dotrénovanie nepriniesla žiadne ovocie. Vizualizované výsledné dáta, ktoré mali reprezentovať mapy výraznosti, rozhodne ako mapy výraznosti nevyzerali a siet' sa nenaučila nič užitočné. Výsledkom boli len úplne náhodné hodnoty, ktoré sa ani zd'aleko neblížili realite. Preto možno považovať tento experiment za nepríjemnú slepú uličku.

8.6 Kombinácia autoenkóderu s VGG16 siet'ou

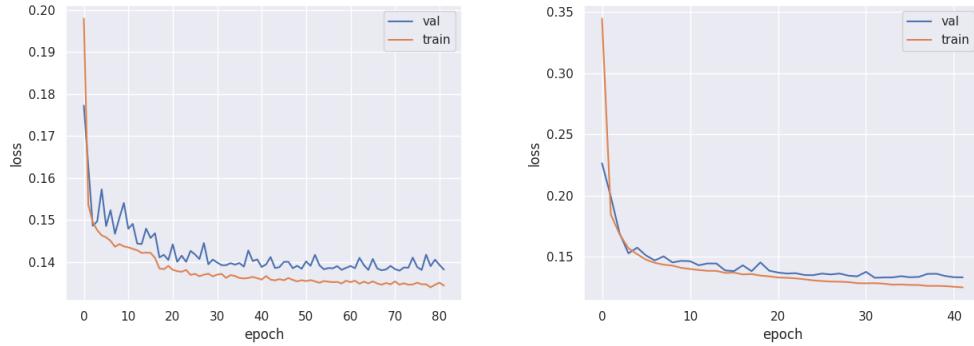
Hoci experiment s VGG16 siet'ou použitou ako enkóder v autoenkóderi nevyšiel, stále sa nám zdalo ako dobrý nápad zapojiť do predikcií detekciu objektov. Preto sme sa pri ďalšej architektúre (kapitola 7.4) inšpirovali riešením riešením zachytávajúcim sémantickú medzeru pri predikcii vizuálnej pozornosti (kapitola 4.2.4). Podľa tohto návrhu sme zostrojili neurónovú siet' kombinujúcu detekciu objektov pomocou konvolučných vrstiev predtrénovanej siete VGG16 a predikciu vizuálnej pozornosti pomocou autoenkóderu.

Siet' sme potom postupne trénovali v 2 dvoch konfiguráciach:

- s vypnutím učenia pre vrstvy VGG16 siete - išlo nám o snahu zachovať kontext detekcie objektov.
- so zapnutím učenia pre vrstvy VGG16 siete - po odskúšaní vyššie uvedenej konfigurácie sme zistili, že mapy vizuálnej pozornosti nemajú až tak vysokú

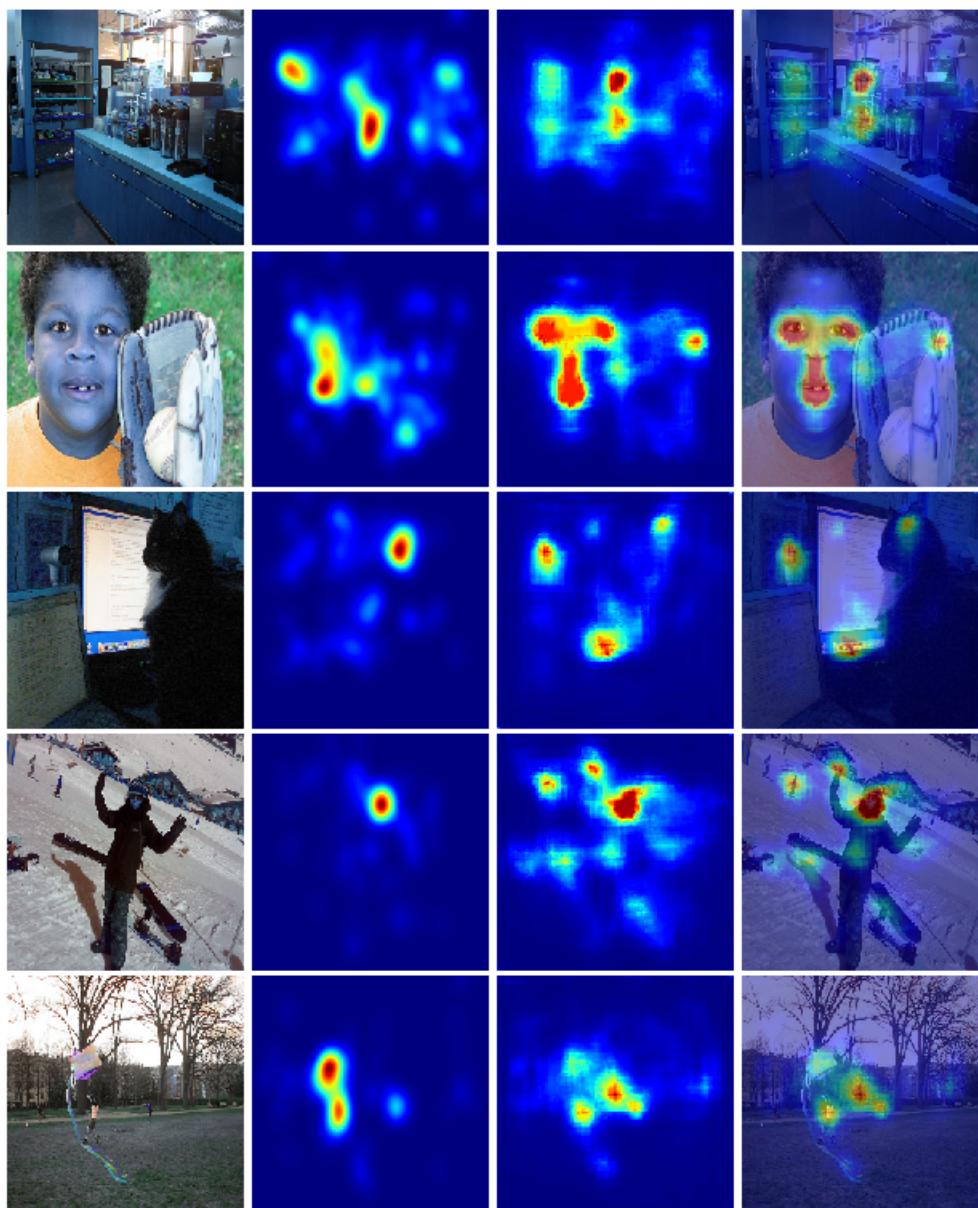
mieru granularity a chceli sme to zmeniť dotrénovaním vrstiev pre detekciu objektov.

Na obrázku 27 je zobrazený vývoj trénovacej a validačnej chyby počas učenia. Ako chyba bola zvolená binárna krížová entropia (z angl. binary cross entropy) v kombinácii s adadelta optimizérom a algoritmom spätného šírenia chyby. Učenie končilo v momente, keď sa za posledných n epoch nezlepšila validačná chyba.



Obr. 27: Porovnanie vývoja chyby počas trénoania, oranžová reprezentuje chybu na trénovacích dátach, modrá na validačných. Vľavo graf pre vypnuté učenie na vrstvách detekcie objektov, vpravo pre zapnuté.

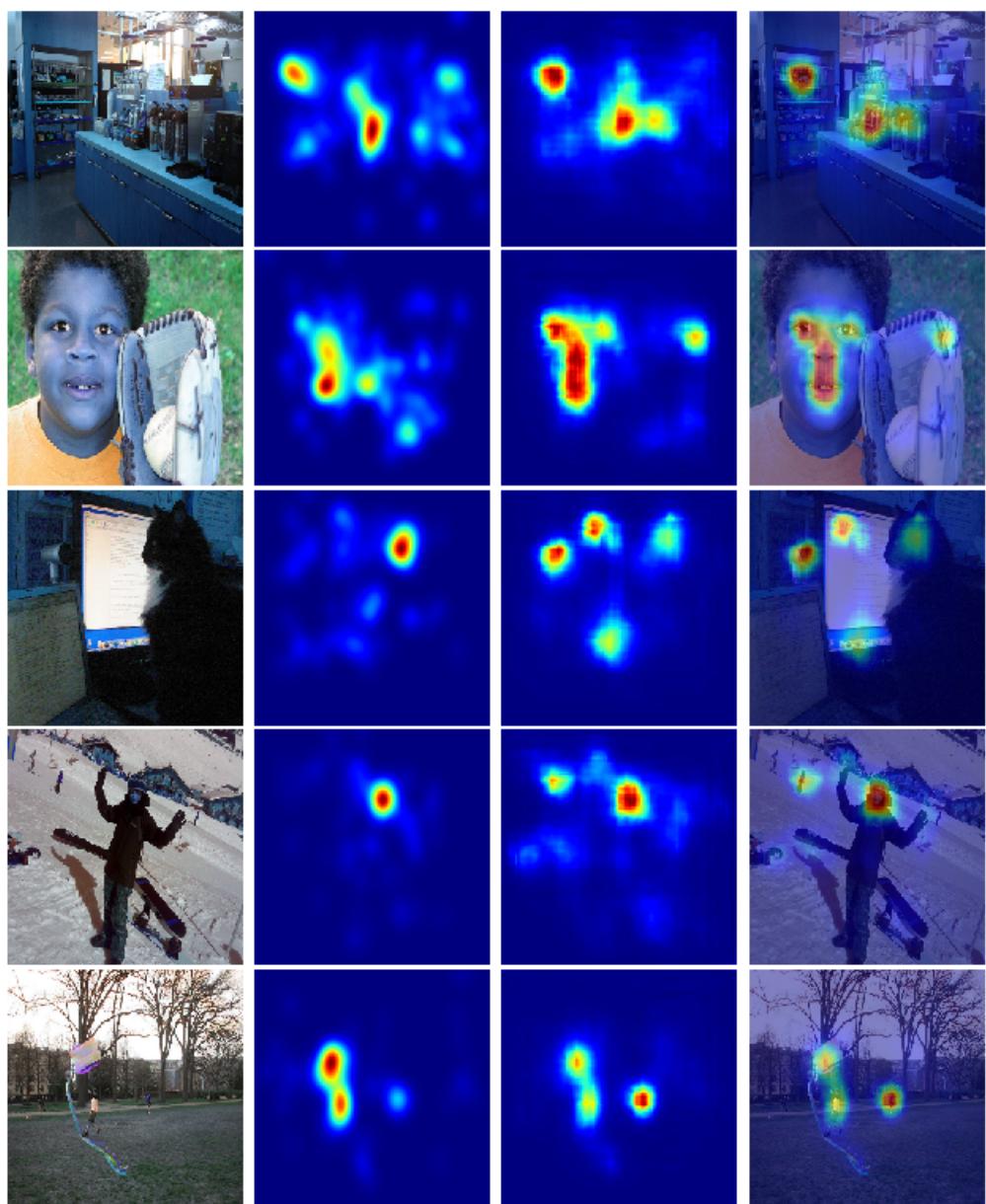
Ako je vidieť, pri zapnutom učení aj na vrstvách pre detekciu objektov (na obrázku 27 vpravo) bol priebeh učenia stabilnejší, kratší a aj samotná chyba bola nižšia. Jej hodnoty boli na trénovacích dátach 0.125 a na validačných 0.1332, pre vypnuté učenie na vrstvách detekcie objektov (obrázok 27 vľavo) bola trénovacia chyba na úrovni 0.1344 a validačná 0.1385. Podobne na tom bola aj chyba na testovacích dátach, a to 0.1391 oproti 0.1442 (so zapnutým učením menšia), čo je vzhľadom aj na takmer o polovicu menší počet epoch lepší výsledok. Na obrázkoch nižšie sú zobrazené predikcie pre jednotlivé konfigurácia spolu s porovnaním voči reálnym mapám vizuálnej pozornosti.



Obr. 28: Porovnanie predikcií autoenkóderu s VGG16 siet'ou bez zapnutého učenia, zl'ava vstupný obrázok, potom originálna mapa výraznosti, predikovaná mapa výraznosti a nakoniec predikcia zobrazená na obrázku.

Ako je vidieť na obrázku 28, v predikovaných mapách s vypnutým učením na vrstvách VGG16 siete cítiť vplyv detektie objektov. Siet' vedela relatívne dobre odhadnúť salientné miesta v prípade, že na obrázku nebolo príliš veľa

objektov. Problém je ale v intenzite týchto označených miest (tá býva vyššia ako v originálnych mapách) a v situáciách, kedy je na obrázku väčšie množstvo objektov. Vtedy sú v predikovanej mape vizuálnej pozornosti označené za salientné aj objekty, ktoré reálne nie sú (napríklad osoby v pozadí, atď.).



Obr. 29: Porovnanie predikcií autoenkóderu s VGG16 siet'ou so zapnutým učením, zľava vstupný obrázok, potom originálna mapa výraznosti, predikovaná mapa výraznosti a nakoniec predikcia zobrazená na obrázku.

Pri zapnutí učenia na vrstvách pre detekciu objektov sa predikcie o niečo zlepšili, sieť pri väčšom počte objektov v scéne už neoznačila tak veľké množstvo za salientné, pri čom aj predikcia intenzity v mapách je miernejšia a viac sa blíži

reálnym mapám vizuálnej pozornosti. Zlepšenie koniec koncov potvrdzujú aj metriky vizuálnej pozornosti (popísané v kapitole 6), ktorými sme ohodnotili obe konfigurácie modelu. Výsledky sú v tabuľke 1.

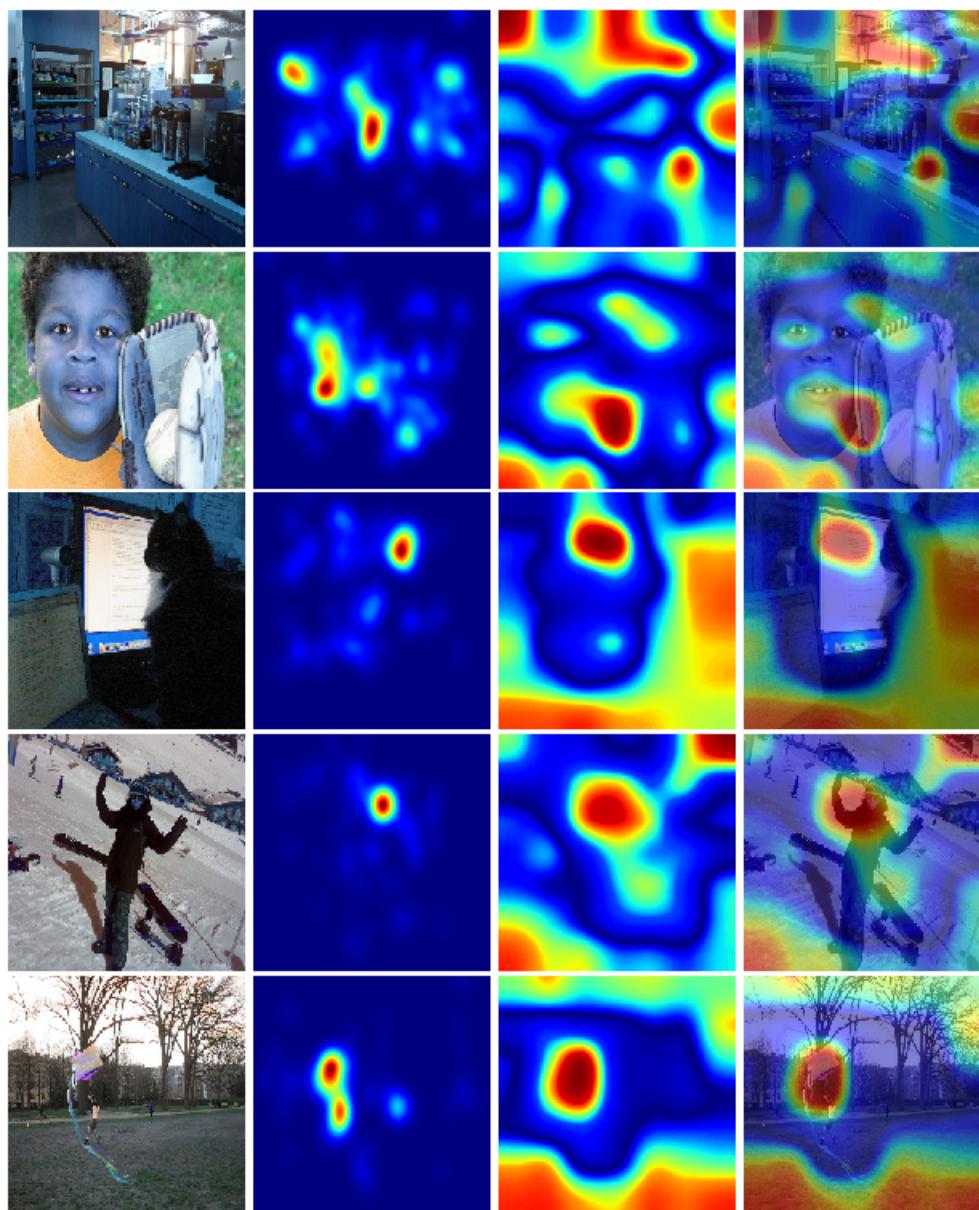
Tabuľka 1: Porovnanie hodnôt metrík pre obe testované konfigurácie

	Model s vypnutým učením pre VGG16 siet'	Model so zapnutým učením pre VGG16 siet'
CC	0.6863	0.8065
SIM	0.6043	0.6958
AUC	0.7678	0.7832
sAUC	0.7154	0.7227
NSS	1.1630	1.2381

8.7 Vedená a upravená vizuálna pozornosť pri modeloch na detekciu objektov

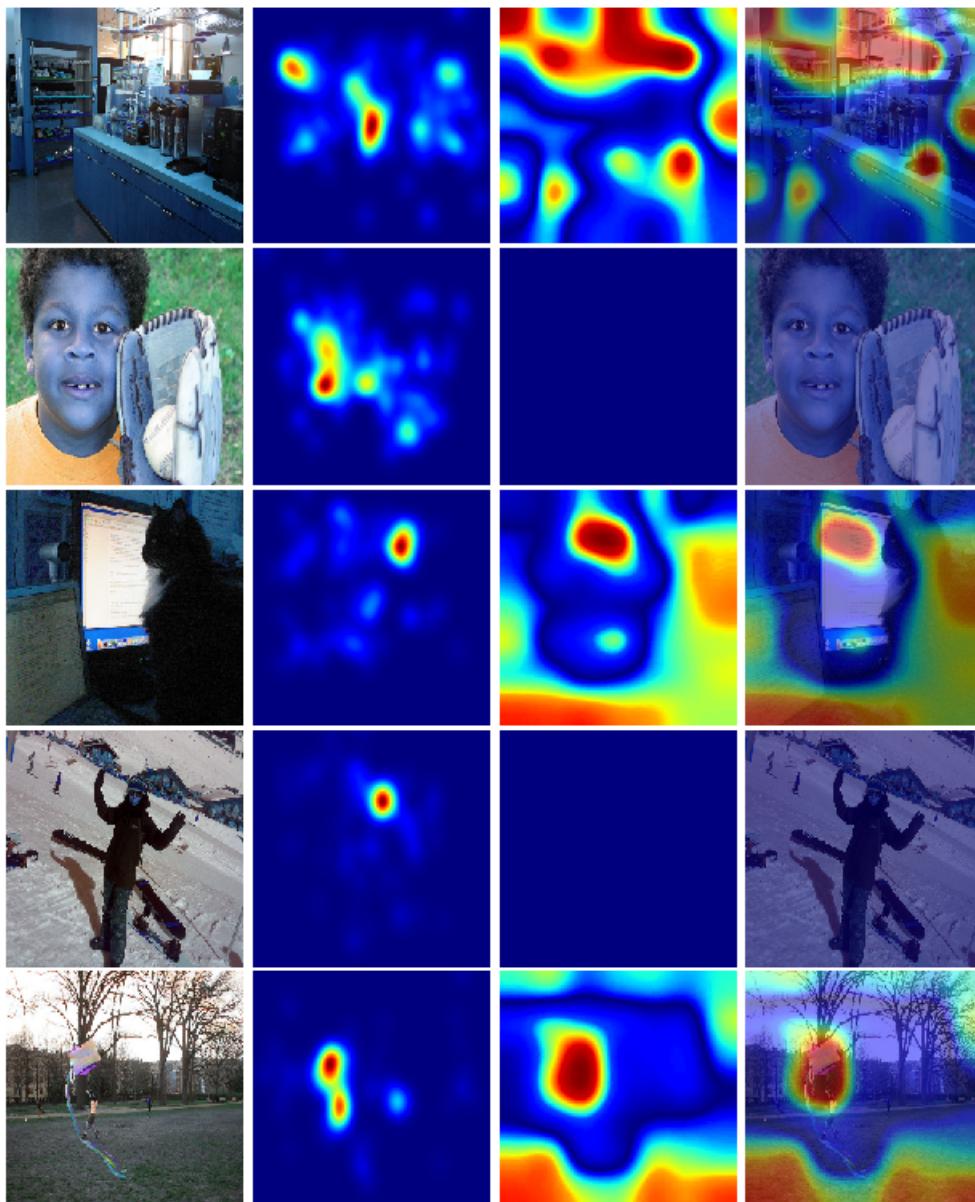
Ako sme popisovali v kapitole 4.3.2, pri natrénovaných modeloch k detekcii objektov vieme pári jednoduchými zmenami docieľiť, aby sme z nich dostali salientné mapy. V rámci jedného experimentu sme teda zobraťi natrénovanú VGG16 siet', zmenili sme *softmax* aktivačnú funkciu na lineárnu a postupne použili obe popísané úpravy algoritmu spätného šírenia chyby. Vizualizácie predikcií sú zobrazené na obrázkoch 30 (pre upravenú vizuálnu pozornosť) a 31 (pre vedenú vizuálnu pozornosť).

8.7 Vedená a upravená vizuálna pozornosť pri modeloch na detekciu objektov 65



Obr. 30: Predikcie upravenej vizuálnej pozornosti, zľava vstupný obrázok, potom originálna mapa výraznosti, predikovaná mapa výraznosti a nakoniec predikcia zobrazená na obrázku.

Vyššie zobrazené vizualizácie ukazujú, že v niektorých prípadoch sa predikcia dokázala zhruba trafiť do salientných miest, tie sú však oveľa rozsiahlejšie ako v originálnych mapách a celkovo majú predikované mapy dosť nízku granularitu.



Obr. 31: Predikcie vedenej vizuálnej pozornosti, zľava vstupný obrázok, potom originálna mapa výraznosti, predikovaná mapa výraznosti a nakoniec predikcia zobrazená na obrázku.

Pri vedenej vizuálnej pozornosti v zásade pretrvávajú podobné problémy, návyše sa však dosť často stávalo, že sme z modelu neboli schopní dostať žiadne mapy vizuálnej pozornosti. To môže súvisieť s tým, že po úprave algoritmu spätného šírenia chyby sa propaguje iba pozitívny gradient zodpovedný za pozitívne

8.7 Vedená a upravená vizuálna pozornosť pri modeloch na detekciu objektov 67

aktivácie. Celkovo predikcie nie sú veľmi presné, presnejšie ale boli mapy práve prvého typu, čo dokázali aj hodnoty metrík vizuálnej pozornosti (kapitola 6). Ako je vidieť v tabuľke 2, hodnoty metrík nedosahujú veľké čísla, rozdiely medzi dvomi typmi predikcií nie sú až tak veľké.

Tabuľka 2: Porovnanie hodnôt metrík pre predikcie vedenej a upravnej vizuálnej pozornosti

	Predikcie vede-nej vizuálnej pozornosti (z angl. guided saliency)	Predikcie upra-venej vizuálnej pozornosti (z angl. rectified saliency)
CC	0.0259	0.0707
SIM	0.2805	0.3040
AUC	0.4973	0.5183
sAUC	0.5018	0.5155
NSS	0.0204	0.0763

8.8 Porovnanie výsledkov experimentov

Do výsledného porovnania sme nakoniec zobrali iba modely s najlepšími výsledkami, pri porovnaní sme sa riadili najmä metrikami vizuálnej pozornosti, prehľadné zhrnutie ich hodnôt spolu s chybou na testovacích dátach a veľkosťou vstupných obrázkov je v tabuľke 3

Tabuľka 3: *Porovnanie konvolučnej neurónovej siete a autoenkóderu spolu s hodnotami metrik vizuálnej pozornosti*

	Konvolučná neurónová siet'	Autoenkóder	Kombinácia autoenkóderu s VGG16 siet'ou	Upravená vizuálna pozornosť
Veľkosť vstupných obrázkov	64x64	256x256	224x224	224x224
Chyba na testovacích dátach	0.1459	0.01301	0.1391	-
Metriky				
CC	0.3891	0.4634	0.8065	0.0707
SIM	0.3058	0.3989	0.6958	0.3040
AUC	0.6892	0.7767	0.7832	0.5183
sAUC	0.6631	0.7245	0.7227	0.5155
NSS	0.7334	1.1288	1.2381	0.0763

Z vyššie uvedenej tabuľky jasne vyplýva, že najlepšie výsledky má autoenkóder, resp. jeho kombinácia s VGG16 siet'ou. I keď hodnoty AUC majú dosť podobné, korelačný koeficient (CC) a podobnosť (SIM), ktoré validujú predikované mapy voči reálnym, nie voči fixáciám, sú výrazne vyššie. Preto sme sa ho rozhodli ako najlepší model, ktorý sa nám podarilo vytvoriť, d'alej porovnať s už existujúcimi modelmi na viacerých datasetoch.

9 Porovanie s existujúcimi riešeniami

Náš najlepší model (autoenkóder kombinovaný s VGG16 sietou pre detekciu objektov) sme porovnali s existujúcimi riešeniami - s modelom pre redukciu sémantických medzier (kapitola 4.2.4) a s modelom SAM (Saliency Attentive Model, kapitola 4.2.2). Porovanie prebiehalo pomocou metrík vizuálnej pozornosti na viacerých datasetoch (popísané v kapitole 5). Ich výsledné hodnoty sú v tabuľke 4.

Tabuľka 4: Porovnanie nášho najlepšieho modelu s existujúcimi riešeniami na rôznych datasetoch pomocou metrík

	Kombinácia autoen- kóderu VGG16 sietou	Model pre redukciu sémantických medzier	SAM (Sa- liency Atten- tive Model)
Salicon			
CC	0.8065	-	0.811
SIM	0.6958	-	-
AUC	0.7832	-	0.878
sAUC	0.7227	-	0.782
NSS	1.2381	-	3.15
MIT1003			
CC	0.6161	0.74	0.738
SIM	0.5075	0.60	-
AUC	0.8193	0.85	0.908
sAUC	0.7369	0.74	0.613
NSS	1.7388	2.12	2.231
CAT2000			
CC	0.7157	-	0.845
SIM	0.6169	-	-
AUC	0.8768	-	0.874
sAUC	0.8237	-	0.539
NSS	2.1150	-	2.233

Pre výpočet metrík nášho riešenia sme ako základný model použili ten natrénovaný na SALICON datasete, pre ďalšie datasety (MIT1003, CAT2000) sme ho už

len postupne dotrénovali. Počas trénovalia aj dotrénovávania boli dátá pre každý dataset rozdelené v pomere 80 : 10 : 10 (trénovalie : validácia : test). Výsledné hodnoty metrík boli vypočítané na testovacích dátach pre jednotlivé datasety.

Ako je vidieť v predošlej tabuľke, hodnoty nášho modelu sa vo väčšine prípadov blížia hodnotám uvedených riešení, len zriedkavo sú lepšie. Naše predikcie sú teda konkurencie schopné, nie sú ale ani zd'aleko najlepšie možné.

10 Zhrnutie

Vypracovaný diplomový projekt k problematike predikcie vizuálnej pozornosti sa venuje najmä jej časti berúcej do úvahy sémantický kontext scény s názvom pozornosť zhora nadol (z angl. top-down). Po preštudovaní uvedenej oblasti, dostupných datasetov a existujúcich riešení, s dôrazom na najnovšie prístupy využívajúce strojové učenie a neurónové siete, sme navrhli sériu modelov, s ktorými sme neskôr experimentovali.

V prvotných experimentoch sme sa snažili využiť rôzne kombinácie datasetov, nakoniec nám ale ako najlepšie vyšlo použiť iba jeden, z dôvodu prílišnej rôznorodosti dát v nich (rôzne veľkosti obrázkov, zozbierané počas rôznych experimentov, atď.). Naša voľba pôvodne padla na DUT-OMRON, neskôr sme ale skončili pri datasete SALICON - ten je rozsiahlejší a o niečo lepšie spracovaný. Na ňom sme trénovali navrhnuté modely, najlepšie z nich vyšiel koncept kombinácie autoenkóderu s konvolučnými vrstvami VGG16 siete pre detekciu objektov, kedy práve časť VGG16 siete fungovala ako samostaný model, ktorého predikcie sa v rámci dekóderu spájali s tými z enkóderu. Týmto spôsobom sa nám podarilo dostať do siete informáciu o pozícii objektov, čo sa prejavilo aj na zlepšení samotných predikcií máp vizuálnej pozornosti. Tie sme nakoniec porovnávali s predikciami dvoch existujúcich modelov vizuálnej pozornosti.

Porovnávanie prebiehalo s modelom pre redukciu sémantických medzier (popísaný v kapitole 4.2.4) s modelom SAM (popísaný v kapitole 4.2.2). Porovnávané boli hodnoty metrík na troch datasetoch, SALICON, MIT1003 a CAT2000. Ako základný model pre naše riešenie sme zvolili ten natrénovaný práve na datasete SALICON, pri čom sme potom postupne dotrénovali siet na zvyšné dva. Dosiahnuté hodnoty sa vo väčšine prípadov blížili hodnotám spomínaných riešení, ale len zriedkavo boli vyššie. To dokazuje, že zvolený prístup bol správny a predikcie sú konkurencie schopné, stále nie sú však najlepšie možné a je čo vylepšovať. Určite by sa dalo ďalej experimentovať práve s časťou pre detekciu objektov, v prípade možnosti trénovania na väčších grafických kartách by napríklad bolo možné zapojiť VGG16 časť do dekóderu skôr a tým získať väčšie množstvo trénovateľných parametrov, ktoré by potencionálne mohli zachytíť viac závislostí sémantického kontextu. Rovnako by dostať mohol pomôcť vačší dataset, ideálne čo

najrozmanitejší (aj vnútorné aj vonkajšie exteriéry, príroda, mesto, ľudia, atď.) - tu by sa dalo uvažovať o vlastných experimentoch, tie sú ale dosť časovo náročné a vyžadujú značný počet participantov.

Literatúra

- [1] Kvasnička V. a kol. *Úvod do teórie neurónových sietí*. Iris, 1997.
- [2] Meyer Arthur. Convolutional autoencoder for saliency detection: enforcing sharp boundary using edge contrast penalty. 2017.
- [3] 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.
- [4] Ali Borji and Laurent Itti. State-of-the-art in visual attention modeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):185–207, 2013.
- [5] Ali Borji and Laurent Itti. Cat2000: A large scale fixation dataset for boosting saliency research. *arXiv preprint arXiv:1505.03581*, 2015.
- [6] Ali Borji and Laurent Itti. Cat2000: A large scale fixation dataset for boosting saliency research. *arXiv preprint arXiv:1505.03581*, 2015.
- [7] 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.
- [8] J Brownlee. A gentle introduction to transfer learning for deep learning.
- [9] Zoya Bylinskii, Tilke Judd, Aude Oliva, Antonio Torralba, and Frédo Durand. What do different evaluation metrics tell us about saliency models? *arXiv preprint arXiv:1604.03605*, 2016.
- [10] Marcella Cornia, Lorenzo Baraldi, Giuseppe Serra, and Rita Cucchiara. A deep multi-level network for saliency prediction. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 3488–3493. IEEE, 2016.

- [11] Marcella Cornia, Lorenzo Baraldi, Giuseppe Serra, and Rita Cucchiara. Predicting human eye fixations via an lstm-based saliency attentive model. *arXiv preprint arXiv:1611.09571*, 2016.
- [12] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.
- [13] 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.
- [14] 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.
- [15] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.
- [16] E Bruce Goldstein. *The Blackwell handbook of sensation and perception*. John Wiley & Sons, 2008.
- [17] 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.
- [18] Richard Langton Gregory. *Concepts and mechanisms of perception*. Charles Scribner’s Sons, 1974.
- [19] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer, 2009.
- [20] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

- [21] Xun Huang, Chengyao Shen, Xavier Boix, and Qi Zhao. Salicon: Reducing the semantic gap in saliency prediction by adapting deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 262–270, 2015.
- [22] Laurent Itti. Visual salience. *Scholarpedia*, 2(9):3327, 2007.
- [23] Laurent Itti and Christof Koch. Computational modelling of visual attention. *Nature reviews neuroscience*, 2(3):194, 2001.
- [24] Ming Jiang, Shengsheng Huang, Juanyong Duan, and Qi Zhao. Salicon: Saliency in context. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.
- [25] Tilke Judd, Krista Ehinger, Frédo Durand, and Antonio Torralba. Learning to predict where humans look. In *2009 IEEE 12th international conference on computer vision*, pages 2106–2113. IEEE, 2009.
- [26] 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.
- [27] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [29] Fei-Fei Li, Andrej Karpathy, and J Johnson. Cs231n: Convolutional neural networks for visual recognition, 2015.
- [30] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*, 2015.

- [31] Huiying Liu, Min Xu, Jinqiao Wang, Tianrong Rao, and Ian Burnett. Improving visual saliency computing with emotion intensity. *IEEE transactions on neural networks and learning systems*, 27(6):1201–1213, 2016.
- [32] 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.
- [33] Fayyaz ul Amir Afsar Minhas and Asa Ben-Hur. Multiple instance learning of calmodulin binding sites. *Bioinformatics*, 28(18):i416–i422, 2012.
- [34] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.
- [35] Michael A Nielsen. *Neural networks and deep learning*. Determination Press, 2015.
- [36] Patrik Polatsek. Saliency maps. Prezentácia, 2015.
- [37] Vasili Ramanishka, Abir Das, Jianming Zhang, and Kate Saenko. Top-down visual saliency guided by captions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 7, 2017.
- [38] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.
- [39] 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.
- [40] Chengyao Shen and Qi Zhao. *Webpage Saliency*, pages 33–46. Springer International Publishing, Cham, 2014.
- [41] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

- [42] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [43] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.
- [44] 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.
- [45] Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39:43–62, 1997.
- [46] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
- [47] 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.
- [48] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec):3371–3408, 2010.
- [49] Jeremy West, Dan Ventura, and Sean Warnick. Spring research presentation: A theoretical foundation for inductive transfer. *Brigham Young University, College of Physical and Mathematical Sciences*, 1:32, 2007.
- [50] Chuan Yang, Lihe Zhang, Ruan Xiang Lu, Huchuan, and Ming-Hsuan Yang. Saliency detection via graph-based manifold ranking. In *Computer Vision and*

Pattern Recognition (CVPR), 2013 IEEE Conference on, pages 3166–3173. IEEE, 2013.

- [51] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.
- [52] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

A Plán práce a jeho plnenie

Plán práce:

- DP1:
 - analýza problematiky
 - prvotné experimenty
- DP2:
 - vybranie vhodného datasetu
 - experimentovanie s modelmi pozornosti zdola nahor
 - nájdenie najlepšieho modelu pre ďalšie experimentovanie
- DP3:
 - experimentovanie so sémantickým kontextom a faktormi pozornosti zhora nadol
 - výber najlepšieho modelu
 - testovanie a porovnanie s existujúcimi riešeniami

Moje vyjadrenie k plánu práce považujem za irelevantné, rovnako ako samotný plán, nakol'ko sa neustále menil a nemalo zmysel ho v jednotlivých semestroch určovať. K riešeniu projektu sme pristúpili agilne, pružne sme reagovali na zmeny a výsledky z jednotlivých experimentov, podľa ktorých sme počas celej práce upravovali jej smerovanie.

B Obsah priloženého elektronického nosiča

Štruktúra dát na elektronickom nosiči:

- **datasets** - datasety použité na trénovanie a testovanie, vzorka upraveného pripraveného datasetu pre neurónovú siet'
- **source_codes_with_models** - zdrojové kódy, aj s príslušnými uloženými modelmi
- **source_codes_with_models/autoencoder** - skripty pre trénovanie a predikcie autoenkóderu, uložený jeho natrénovaný model
- **source_codes_with_models/CNN** - skripty pre trénovanie a predikcie konvolučnej neurónovej siete
- **source_codes_with_models/vgg16_with_autoencoder** - skripty pre trénovanie a predikcie autoenkóderu s VGG16 siet'ou, uložený natrénovaný najlepší model
- **source_codes_with_models/nn_utils** - zdrojové kódy pomocných skriptov (napr. pre načítanie dát, výpočet máp fixácií, výpočet metrík atď.).)
- **source_codes_with_models/salicon_api** - aktualizované skripty pre prácu so salicon datasetom
- **document** - pdf verzia odovzdávanej práce

C Technická dokumentácia

Celé popisované riešenie bolo vyvinuté a otestované na Linux-ových operačných systémoch Ubuntu 18.04.2 LTS a Fedora 28 - na nich vieme zaručiť správne korektné fungovanie priložených skriptov. Tie sú napísané v jazyku Python vo verzii 3.5, za použitia nasledovných knižníc:

- TensorFlow - vo verzii 1.13.1, knižnica pre prácu s neurónovými sietami
- Keras - vo verzii 2.2.2, vysoko úrovňová knižnica pre prácu s neurónovými sietami, v našom riešení využíva nižšie úrovňový TensorFlow (za použitia napríklad knižnice Theano skripty nebudú fungovať)
- Matplotlib - verzia 3.0.0, knižnica pre 2D vykreslovanie a prácu z obrázkami, použitá pri vizualizáciách predikcií
- OpenCV - verzia 3.4.4.19, knižnica pre počítačové videnie a strojové učenie, použitá pre prácu s obrázkami
- SciPy³⁶ - verzia 1.1.0, knižnica pre prácu s dátami
- NumPy³⁷ - verzia 1.15.2, knižnica pre prácu s dátami
- Image Processing SciKit (scikit-image)³⁸ - verzia 0.14.1, knižnica pre jednoduché spracovanie obrázkov

Verzie knižníc uvádzame z dôvodu kompatibility riešenia, napríklad je možné, že skripty budú fungovať aj na iných verziach knižníc pre prácu s neurónovými sietami (Keras, TensorFlow), ale v iných verziach nemusia byť schopné načítať uložené modely.

Čo sa týka hardvérového vybavenia, celé riešenie bolo natrénované na grafickej karte Nvidia GeForce GTX 1080 Ti s veľkosťou 12gb a RAM s veľkosťou 32gb. Trénovanie aj predikcie je možné uskutočňovať aj na CPU, bude to však mať značné dopady na výkon. Trénovanie je možné aj na menšej grafickej karte, je však nutné zmeniť aj veľkosť dávky (z angl. batch size) pre jedno trénovanie.

³⁶<https://docs.scipy.org/doc/scipy/reference/index.html>

³⁷<https://www.numpy.org/>

³⁸<https://scikit-image.org/>

D Používateľská príručka

V zásade každý z modelov uvedených v návrhu má jeden priečinok, v ktorom sú jeho príslušné zdrojové kódy a do ktorého sa ukladajú natrénované modely a predikcie. Každý z nich má 2 základné skripty pre trénovanie a predikovanie, ktoré majú niekoľko konfiguračných možností:

- konvolučná neurónová siet':
 - **train.py** - trénovací skript, ako argumenty berie priečinok s obrázkami, mapami, ich veľkosť', počet epoch, veľkosť dávky a model, kde sa má natrénovaná siet' uložiť'. Príklad volania: *python3 train.py --images dataset/images-train --maps dataset/maps-train --n 64 --batch 100 --epochs 500 --model model.save*
 - **predict.py** - skript pre predikcie a testovanie, ako argumenty berie vstupné obrázky, natrénovaný model, priečinok kam má uložiť prvých 50 predikcií, priečinok s binárnymi mapami a s originálnymi mapami. Príklad volania: *python3 predict.py --images dataset/images-test --maps dataset/maps-test --model model.save --binary_maps dataset/binary-maps --save_to predicted_maps*
- autoenkóder:
 - **autoencoder.py** - trénovací skript, v zásade má ako povinné parametre len priečinok s obrázkami a mapami vizuálnej pozornosti. Nepovinných parametrov je d'aleko viac, od počtu epoch až po počty konvolučných vrstiev a filtrov, všetky ale majú predvolené hodnoty (pre konkrétné čísla treba pozrieť priamo do kódu). Príklad volania: *python3 autoencoder.py --maps /home/pbeka/dataset/salicon/maps_older/train+val/ --images /home/pbeka/dataset/salicon/images/train+val/ --batch 100 --epoch 500 --n 224 --conv_layers 2*
 - **autoencoder_predict.py** - skript pre predikcie a testovanie, povinné argumenty sú vstupné obrázky, natrénovaný model, priečinok s binárnymi mapami a s originálnymi mapami. Pre nepovinné parametre treba

pozrieť priamo do kódu. Príklad volania: *autoencoder_predict.py --images DUT-OMRON-image-test/ --maps DUT-OMRON-maps-test/ --binary_maps DUT-OMRON-fixations-test/ --model model.save*

- autoenkóder s VGG16 siet'ou:
 - **train_vgg16_with_autoencoder.py** - trénovací skript, povinné parametre sú rovnaké ako pri autoenkóderi. Príklad volania: *python3 train_vgg16_with_autoencoder.py --images salicon/images/train+val/ --maps salicon/maps_older/train+val/ --loss binary_crossentropy --optimizer adadelta --samples 13500 --batch 10*
 - **predict_vgg16_with_autoencoder.py** - skript pre predikcie a testovanie, povinné argumenty sú vstupné obrázky, priečinok s binárnymi mapami a s originálnymi mapami, funkcia chyby a optimizér (podľa nich sa načíta model, pri skripte pre trénovaní je názvom uloženého modelu práve ich kombinácia). Pre nepovinné parametre treba pozrieť priamo do kódu. Príklad volania: *python3 predict_vgg16_with_autoencoder.py --images salicon/images/test/ --maps salicon/maps_older/test/ --binary_maps salicon/binary_maps_2014/ --binary_format jpg --loss binary_crossentropy --optimizer adadelta*
 - **train_another_dataset.py** - skript pre dotrénovanie modelu na inom datasete. Ako argumenty berie priečinok s obrázkami, mapami a na-trénovaný model. Príklad volania: *python3 train_another_dataset.py --images cat2000/images-train+val --maps cat2000/maps-train+val --model models_all_train/adadelta_binary_crossentropy.model*
- upravený backpropagation:
 - **backprop_change_predict.py** - skript pre predikcie z VGG16 siete po zmene algoritmu spätného šírenia chyby. Argumentami sú priečinok s obrázkami, mapami vizuálnej pozornosti a mapami fixácií. Príklad volania: *python3 backprop_change_predict.py --images salicon/images/test/ --maps salicon/maps_older/test/ --binary_maps salicon/binary_maps_2014/*

E Podrobný diagram architektúry s kombináciou VGG16 a autoenkóderu

