

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

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

Apríl 2019

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

.....

Bc. Patrik Beka

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	26
4.1	Modely k predikcii pozornosti zdola nahor	26
4.1.1	Itti-ho model	26
4.1.2	Metóda hierarchickej výraznosti	27
4.1.3	Model pre určovanie salientných oblastí webových stránok	28
4.2	Modely k predikcii pozornosti zhora nadol	28
4.2.1	Zameranie na emočný obsah	28

4.2.2	SAM	30
4.2.3	Pozornosť zhora nadol vedená titulkami	32
4.2.4	Redukcia sémantických medzier pri predikcii vizuálnej pozornosti	34
4.3	Detekcia objektov	35
5	Datasetsy vizuálnej pozornosti	39
6	Metriky pre modely vizuálnej pozornosti	39
7	Návrh	41
7.1	Dataset	41
7.2	Konvolučná neurónová sieť	42
7.3	Autoenkóder	45
7.4	Kombinácia VGG16 siete s autoenkóderom	48
8	Experimenty	50
8.1	Implementačné prostredie	50
8.2	Prvotné experimenty	51
8.2.1	Úprava datasetu	51
8.2.2	Výsledky	52
8.3	Konvolučná neurónová sieť	53
8.4	Konvolučný autoenkóder	54
8.5	Autoenkóder s predtrénovaným modelom VGG16	57
8.6	Kombinácia autoenkóderu s VGG16 sietou	58
8.7	Porovnanie výsledkov experimentov	58
9	Zhrnutie	60
Literatúra		62
A	Plán d' āszej práce	i
B	Obsah priloženého elektronického nosiča	i

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	27
10	Diagram modelu k predikcii máp výraznosti založených na emočnom obsahu	30
11	Diagram konvolučnej LSTM siete	31
12	Predikcia salientných oblastí na základe klúčových slov vo vete	33
13	Model neurónovej siete pre predikciu na základe klúčových slov	34
14	Model redukcie sémantických medzier pri predikcii pozornosti	35
15	VGG16	36
16	Autoenkóder pre detekciu objektov	37
17	Návrh architektúry neurónovej siete	44
18	Diagram navrhnutého autoenkóderu	47
19	Vizualizácia extrakcie regiónov záujmu	51
20	Porovnanie prvotných výsledkov	52
21	Vývoj chyby počas trénovania konvolučnej neurónovej siete	53
22	Vzorka predikcie konvolučnej neurónovej siete	54
23	Vývoj chyby počas trénovania autoenkóderu	55
24	Porovnanie predikcií autoenkóderu voči reálnym mapám výraznosti	56
25	Vývoj chyby počas trénovanie siete s predtrénovaným modelom VGG16	58

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 detekcie 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). K ich evaluácii sa používa značné množstvo metrík, z ktorých napoužívanejšie sú vysvetlené v kapitole 6.

Za cieľ na druhý semester sme si zvolili nájsť čo najlepší model neurónovej siete k predikcii pozornosti zdola nahor, ktorému by sme mohli neskôr postupne dodávať ďalšie informácie o pozornosti zhora nadol. Na to sme teda po rozanalizovaní problémovej oblasti nadviazali v kapitole 7, kde uvádzame navrhnuté 2 neurónové siete a to klasickú konvolučnú (kapitola 7.2) a autoenkóder (kapitola 7.3). Experimentovaniu nie len s nimi sa venuje kapitola 8, v ktorej závere sa nachádza zhrnutie a výsledky experimentov, rovnako ako výber najlepšieho modelu, s ktorým budeme pracovať aj ďalej.

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 [21]. 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[22].

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[15]) alebo údajoch (z angl. data-driven processing[17])

Druhým veľkým rozdielom oproti spracovaniu zdola nahor je jeho rýchlosť, priemerný čas spracovania vizuálneho signálu sa pohybuje na úrovni 200 milisekúnd[22] 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), respektíve 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_i^k - prah stimulácie i-teho neurónu k+1 vrstvy

w_{ij}^k - váha medzi i-tým neurónom vrstvy k+1 a j-tým 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 . Popisovany 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 niekol'ko d'alších vrstiev rôzneho typu. Tieto vrstvy sa nazývajú skryté a obvykle práve tieto bývajú kl'účové 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 niekol'ko 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 zárove potlačenie hodnôt, ktoré sú výrazne menšie ako maximálna hodnota.

- **ReLU**

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

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

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

- **Softplus**

Je v podstate aproximáciou k 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:

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

3.2 Učenie sa neurónovej siete

Základným prvkom toho, aby bola neurónová sieť schopná riešiť úlohy je učenie sa. Existujú viaceré typy učenia sa neurónovej siete, základnými sú učenie sa s učiteľom (z angl. supervised learning), učenie sa bez učiteľa (z angl. unsupervised learning[18]) a učenie sa posilňovaním (z angl. reinforcement learning[31]).

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 očakávane výstupy. Najčastejším príkladom tohto typu učenia je algoritmus spätného šírenia chyby (z angl. backpropagation[32]).

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 očakávaný 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 minusov učenia s učiteľom.

Učenie sa viacerých inštancií

MIL[30] - skratka pre anglický výraz Multiple Instance Learning, čiže učenie sa viacerých inštancií, je variácia učenia s učiteľom. Miesto obdržania individuálne oštitkovaných inštancií (reprezentujú triedy, ktoré chceme predikovať), obdrží model set oštitkovaných vreciek (z angl. bags), z ktorého každé obsahuje niekoľko inštancií. V prípade jednoduchej binárnej klasifikácie je celé vrecko označené ako negatívna vzorka, ak všetky inštancie vo vnútri sú negatívne. Pokiaľ ale obsahuje aspoň jednu pozitívnu, je označené ako pozitívna vzorka. Pri zložitejších triedach si ako príklad môžeme uviesť predikciu nejakej cielovej triedy pre obrázok na základe vizuálneho obsahu. Cielová 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 črt (inštanciu) extrahové z prislúchajúceho i-teho regiónu obrázku a N je celkový počet regiónov (inštancií) v obrázku. Vrecko je označené pozitívne ("*pláž*") v prípade, že obsahuje oba regióny (inštancie) "*piesok*" a "*voda*".

3.2.2 Učenie sa bez učiteľa

Základný popis o učení bez učiteľa hovorí, že sa jedná o odvodenie funkcie k popisu skrytej vnútornnej štruktúry z neoštitkovaných 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 [45], 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 kameóny. Pri klasickom postupe by sme potrebovali dostatočné veľký dataset kameónov, 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 dotrénuvať ho pre detekciu kameónov. Takto môžeme výrazne znížiť čas nutný na trénovanie a zvýšiť presnosť detekcie.

Prenos učenia má 2 základné prístupy [7]:

- 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 jednat 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

natrénujeme a podobne ako pri predošlom prístupe ho potom znovupoužijeme a dotrénujeme 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 optimizer [35]:**

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 optimizer [24]:**

V podstate vychádza priamo zo Stochastického gradient descent optimizéru, resp. jeho modifikácie RMSProp algoritmu[43]. Rozdiel oproti Gradient descent optimizéru je ale v tom, že je schopný variabilne určovať rýchlosť učenia neurónovej siete.

- **Adadelta optimizer [47]:**

Jedná sa o rozšírenie Adagrad optimizéru[11], 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 optimizer:**

Vychádza z algoritmu učenia FTRL-Proximal[29], 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.

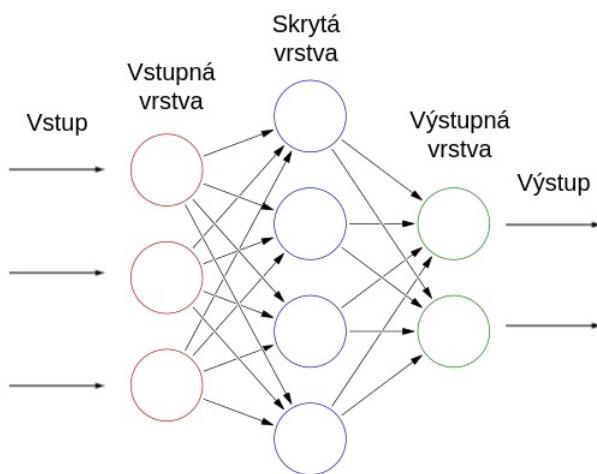
Aj keď neurónové siete dokážu efektívne riešiť veľké množstvo úloh, problémom stále zostáva mať k dispozícii dostatok dát k učeniu neurónovej siete ešte pred riešením úloh. Taktiež je potrebné mať dostatok výpočtovej sily, aby sa problém neriešil pridlhý čas, a dostatok pamäte, keďže neurónové siete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

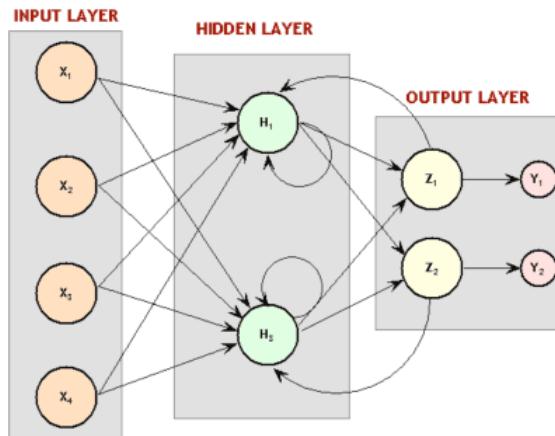
Najjednoduchší typ možno zobrazit' ako jednu vstupnú vrstvu, jednu skrytú a jednu výstupnú, neuróny sú tu poprepájané z n-tej vrstvy do n+1 vrstvy, ako je možné vidieť na obrázku 1. Tento typ sa nazýva dopredná neurónová sieť (z angl. feedforward neural network) a môže mať aj viac ako len jednu skrytú vrstvu. Používa sa hlavne ak sa jedná o predikciu nelineárnej funkcie (napríklad carbon-13 NMR chemické posuny alkánov[41]).



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).



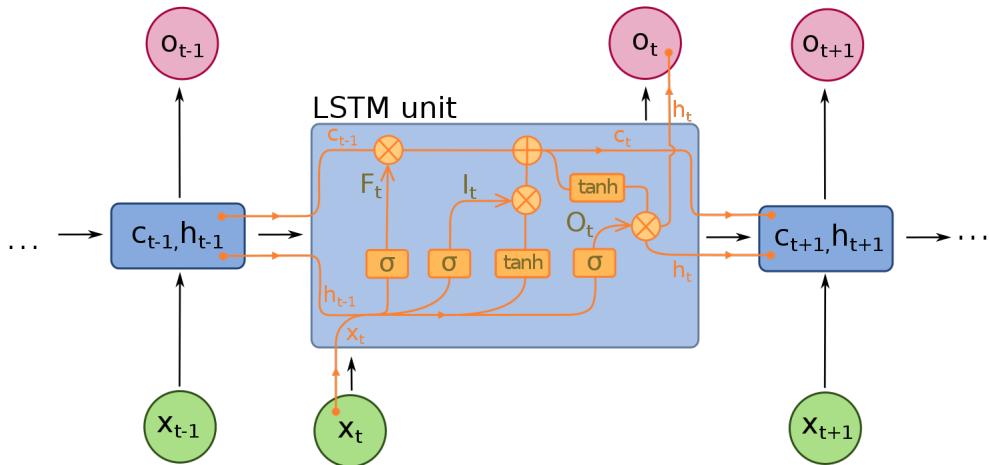
Obr. 2: Príklad jednoduchej rekurentnej neurónovej siete⁴

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[14] - 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/>

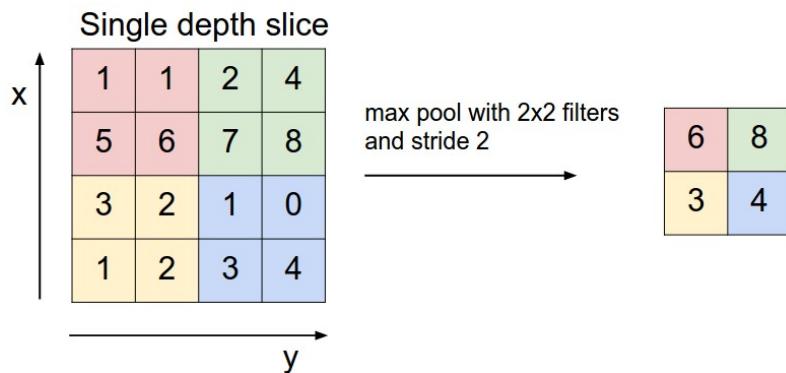
l'ubovoľnú sekvenciu (vektor) predstavujú s možnosťou "pamäte" značný posun. V praxi sa používajú napríklad pri úlohách so spracúvaním jazyka. Keby sme chceli napríklad predikovať nasledujúce slovo vo vete, je veľmi užitočné vedieť aké slová boli pred ním aby predikcia dávala zmysel. Práve v tejto oblasti sa LSTM siete ukázali veľmi efektívne. Využívajú sa d'alej napríklad aj pri rozpoznávaní reči[36] alebo písma[16], či generovaní popisu k obrázkom[23], kedy však fungujú v kombinácii s konvolučnou neurónovou sieťou (z angl. convolutional neural network). Tá je d'alším 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 (3×3 , 5×5). Vstup tejto vrstvy musí byť v tvaru $m \times m \times r$, 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 d'alšie vrstve združovania (z angl. pooling layer[26]), 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 2×2 aplikovaným s krokom veľkosti 2. Toto sa dá jednoducho vysvetliť ako prejdenie každého výstupu z konvolučnej vrstvy oknom uvedenej veľkosti postupne po 2 políčkach na šírku aj výšku, pričom z každej štvorice v okne sa získa MAX operáciou maximum, s ktorým sa pracuje d'alej. Na obrázku⁷ 4 je vidieť výsledok popisovaného postupu na jednoduchom príklade.

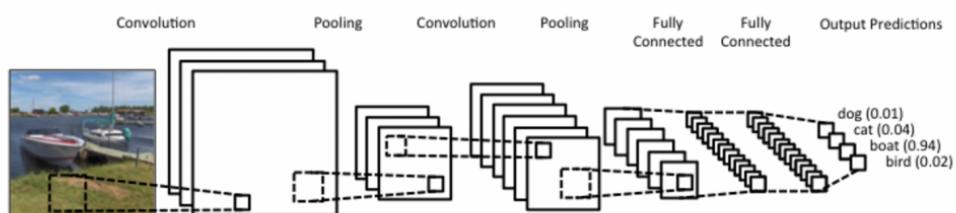
⁶<http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/>

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 sietach. 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 ciel'ové hodnoty rovné tým vstupným, snaží sa teda naučiť funkciu:

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

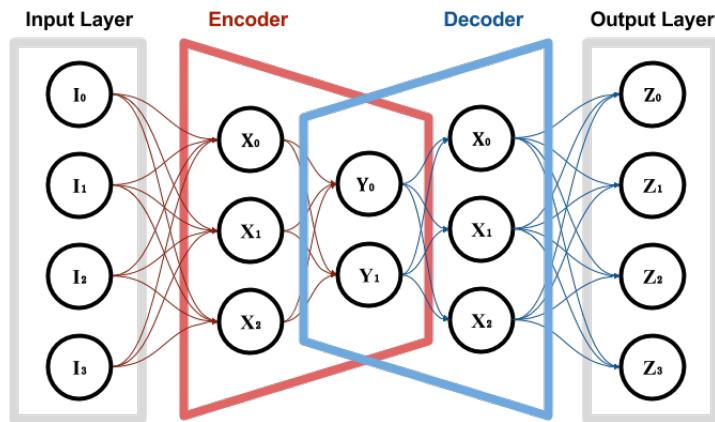
Jeho ciel'om je v podstate naučiť sa enkódovanú reprezentáciu dát, čo sa využíva napríklad pri komprimácii alebo k zníženiu dimenzionality dát[19]. Autoenkóder sa skladá z dvoch častí (ako je to vidieť na obrázku 6):

- 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 ciel'om 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 ked' ciel'om 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[44] z dát alebo pri znižovaní ich dimenzionality. Uplatnenie tak nachádzajú hlavne pri spracovaní jazyka[27] a počítačovom videní - napríklad pri detekcii objektov[2].

3.4 Framework-y pre prácu s neurónovými sieťami

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

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:

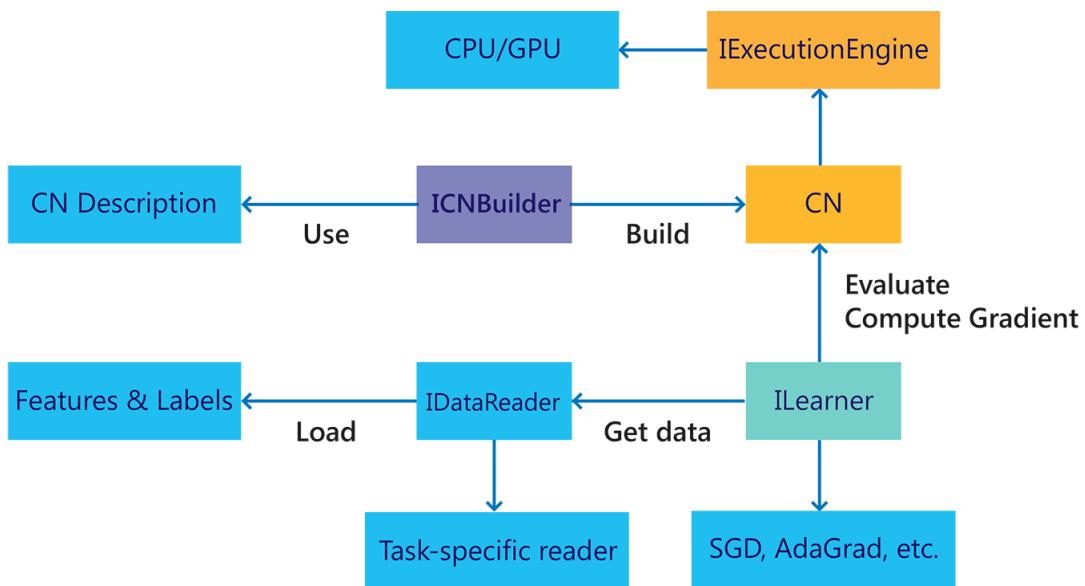
- nedostatok predtrénovaných modelov
- pri použití s určitými jazykmi (Python, Java, ...) je pomalý, nakol'ko sa nejedná najrýchlejšie jazyky

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’:

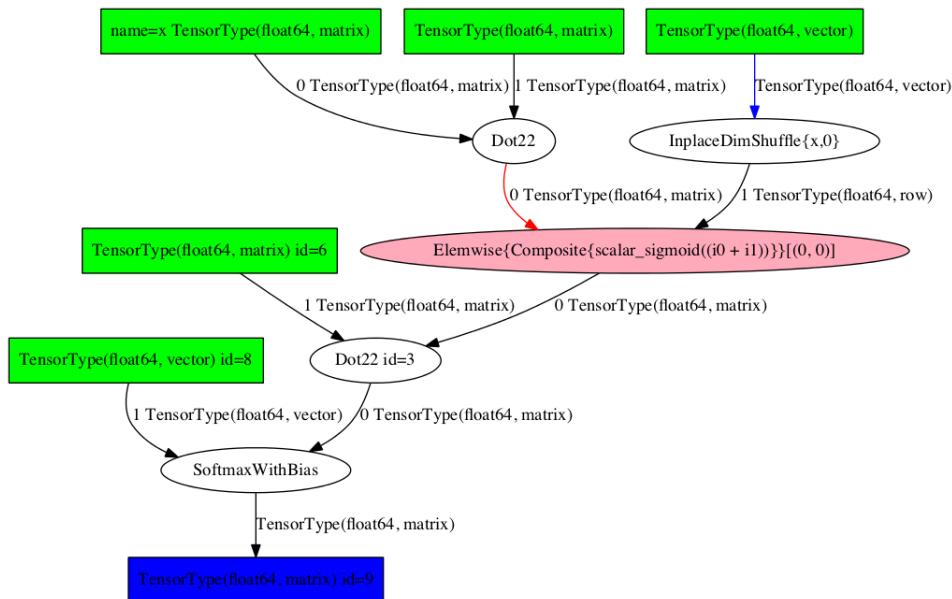
- implementácia v novom jazyku, Network Description Language (NDL)
- nedostatok možností a nástrojov pre vizualizácie

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 sietami, 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 rozšíritelnosť
- 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 môžné

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 porované

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 funkcionality 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). Ďalším zaujímavým rozdelením však je rozdelenie podľa použitých princípov a technológií [33], 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

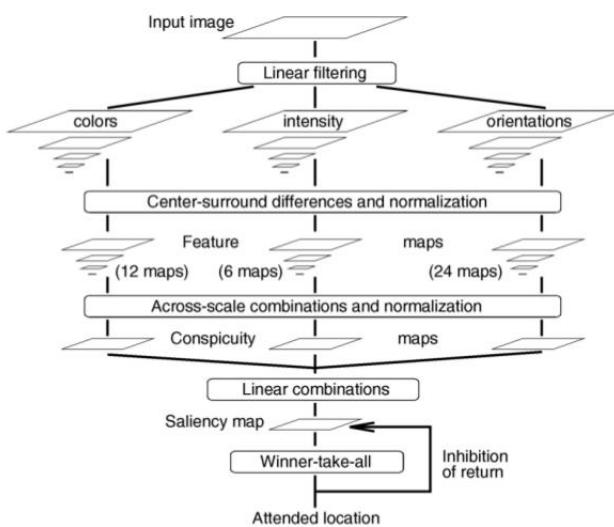
4.1 Modely k predikcii pozornosti zdola nahor

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[12] 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*[12]

4.1.2 Metóda hierarchickej výraznosti

Pri použíti metódy hierarchickej výraznosti (z angl. hierarchical-saliency method[13]) sa v podstate jedná o detekciu charakteristík v prirodzenej scéne. Tento postup je istým vylepšením predchádzajúceho Ittiho modelu a prakticky pozostáva z nasledujúcich dvoch krokov:

- *prvý krok* - extrakcia globálne výrazných oblastí (z angl. globally salient regions):
 1. výpočet mapy výraznosti S zo vstupného obrázka I
 2. evaluácia oblastí záujmu (z angl. regions of interest, ROIs), automatická klasifikácia všetkých pixelov do dvoch kategórií k získaniu masky M :

- globálne výrazné oblasti (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[37] 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, nakoľko sa do úvahy berie aj sématický kontext scény. Tým môže byť napríklad emočný obsah (kapitola 4.2.1), či hľadanie konkrétnych objektov v scéne (kapitola 4.2.3).

4.2.1 Zameranie na emočný obsah

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. [28], 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?

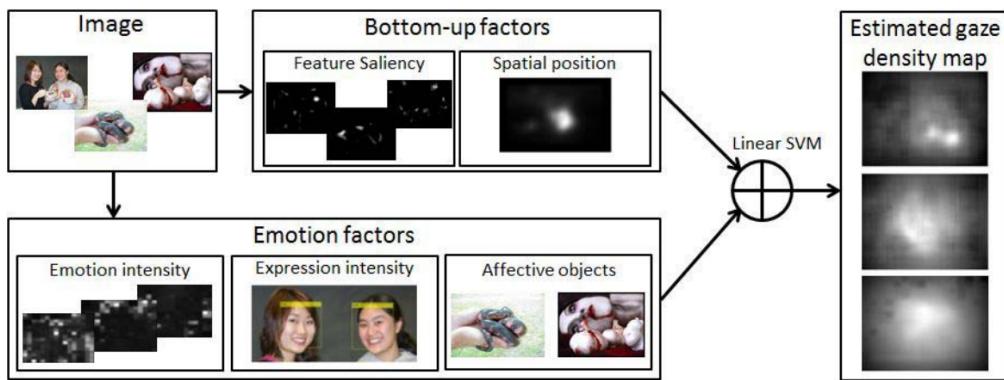
Ich narhované riešenie malo niekol'ko častí. Prvou je extrakcia faktorov pozornosti zdola nahor (bottom-up) vo forme nízko-úrovňových máp výraznosti založených na črtách a mapy priestorovej polohy s dôrazom na centrum pre výpočet odhadu hustoty pohľadu (z angl. gaze density estimation). K tomuto boli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

Druhou časťou je detekcia emócií, k čomu bol použitý emočný detektor založený na učení s použitím 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 vyskútu 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 z tejto časti je emočná mapa, ohodnocujúca intenzitu emócií, určená pre ďalšie spracovanie.

Tretiu časťou bolo spracovanie výstupov z predošlých dvoch častí lineárnym SVM, ktoré vo výsledku z faktorov pozornosti zdola nahor (bottom-up) a emočných

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

faktorov vygeneruje výslednú mapu hustoty pohľadu (z angl. gaze density map). Vyššie popísaný postup je zobrazený na diagrame na obrázku 10.

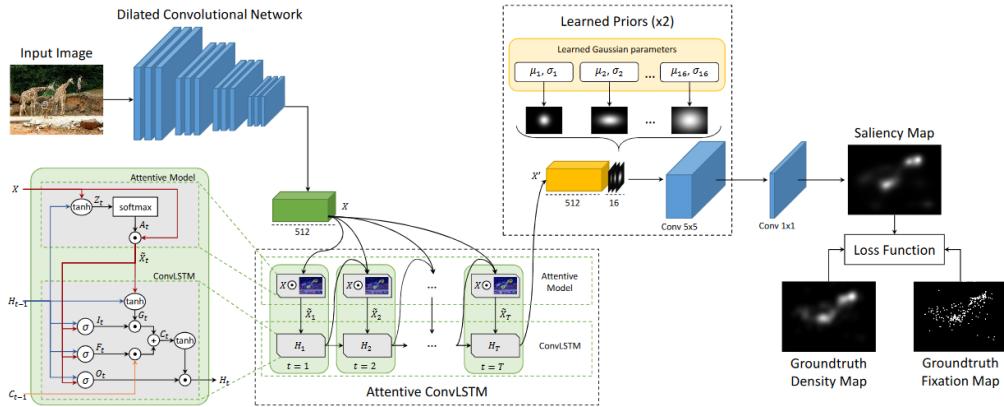


Obr. 10: Diagram modelu od Liu a spol. k predikcii máp výraznosti založených na emočnom obsahu[28]

Za veľkú výhodu tohto riešenia by sa dala pokladať 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[10], ďalší z modelov schopných aj predikcie pozornosti zhora nadol (top-down). Založený je na architektúre nazvanej ML-Net (Multi-Level Network[9]), ktorá miesto klasických konvolučných vrstiev využíva dilatované konvolučné vrstvy, ktoré limitujú efekt zmenšovania. Táto architektúra použitá pre spracovanie vstupného obrázka je nasledovaná konvolučnými LSTM vrstvami, ich naučené výstupy potom spracováva už klasická konvolučná vrstva, výstupom ktorej je mapa výraznosti. Popisovaný postup je zobrazený na obrázku 11.



Obr. 11: Diagram modelu využívajúceho dilatované konvolučné vrstvy v kombinácii s LSTM vrstvami [10]

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 čírt 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 čírt 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 čírt (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

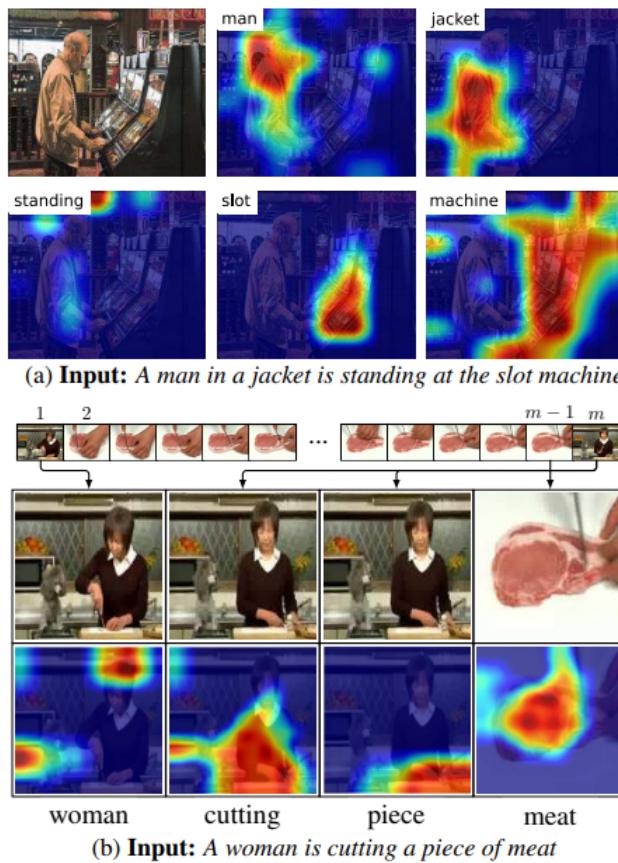
$L1, L2, L3$ - tri funkcie pre výpočet evaluačných metrík, za radom Normalizovaná cesta výraznosti (NSS, z angl. Normalized Scanpath Saliency), Lineárny korelačný koeficient (CC), Kullback-Leiblerova divergencia (KL-Div), všetky bližšie popísané v kapitole 6.

Za veľkú výhodu tohto riešenia pokladáme možnosť predikcie aj oboch typov pozornosti, v závislosti od datasetu, na ktorom je model natrenovaný. Taktiež možnosť nezávislej výmeny submodelov (dilatovaná konvolučná siet', konvolučná LSTM siet') sa javí ako veľmi výhodná, keďže uľahčuje prototypovanie bez nutnosti nového natrenovania celého modelu. Za miernu nevýhodu možno považovať značnú komplexitu modelu a s tým spojené nároky na hardvér, rovnako ako aj na čas nutný na natrenovanie 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 [34] schopný predikcie máp výraznosti na základe kľúčových slov v zadanej vete a príslušnom obrázku alebo videu, príklad predikcií možno vidieť na obrázku 12.

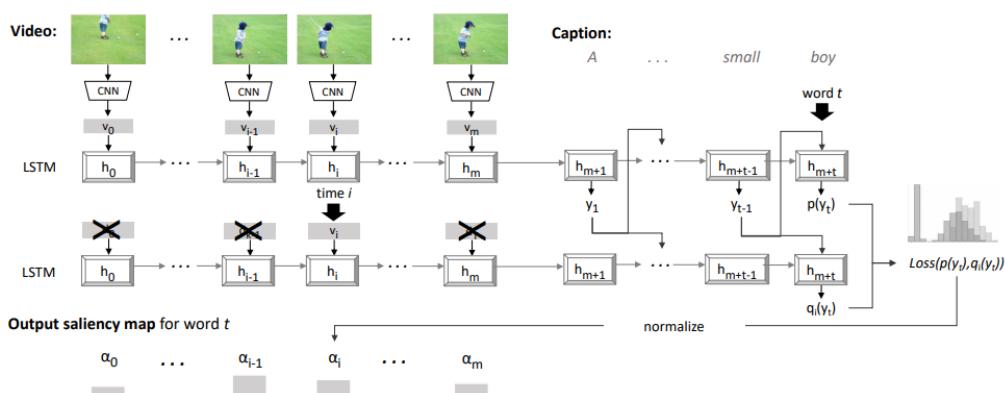
²²<http://ai.bu.edu/caption-guided-saliency/>



Obr. 12: Predikcia salientných oblastí na základe klúčových slov vo vete, hore po **a** predikcia pre obrázok, dolu po **b** predikcia pre video[34]

Autori použili prístup nazvaný titulkami vedená vizuálna pozornosť (z angl. Caption-Guided Visual Saliency), ktorý produkuje mapy výraznosti pre nepohyblivé obrázky alebo video. Ako základný model použili LSTM enkóder-dekóder (z angl. encoder-decoder), ktorý predikuje aj dočasné (z videa) aj priestorové (z obrázkov) salientné mapy na základe klúčových 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 ďalej ako vektor pre LSTM vrstvy reprezentujúce dekóder - ten rozhoduje o tom, ktoré časti použije

pomocou LSTM výstupných brán k predikcii mapy výraznosti pre klúčové slovo v čase t . Autori d'alej zvolili veľmi 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 klúčové slová vo vete.

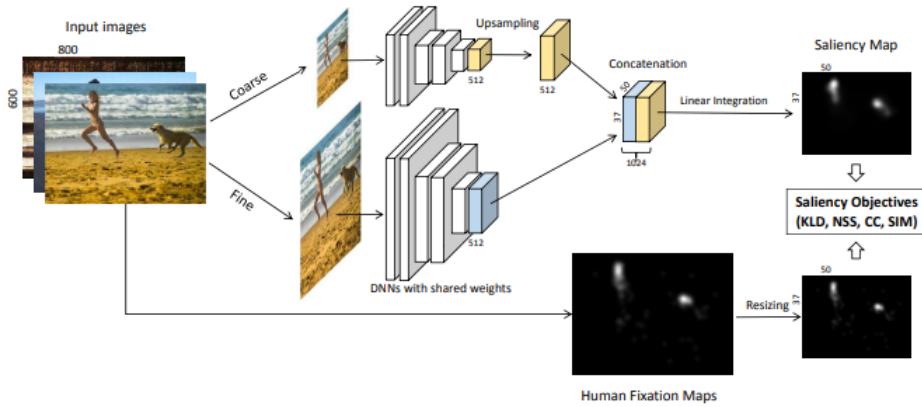


Obr. 13: Diagram modelu neurónovej siete[34], zhora vstup vďavo vo forme videa, vpravo titulky k nemu. V strede LSTM vrsty neurónovej siete, dolu výstupná mapa výraznosti pre klúčové slová.

Popisované 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. [20] 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:

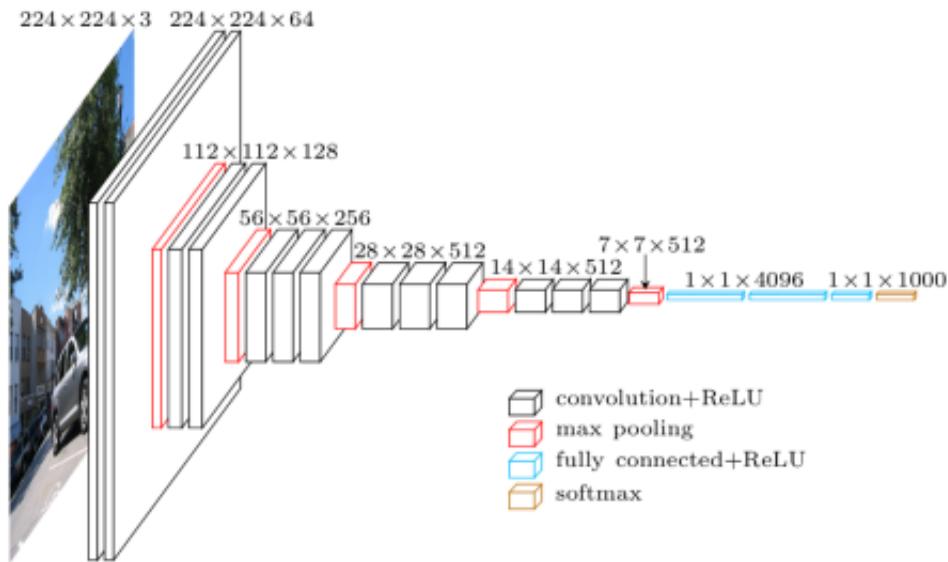
- tzv. "hrubozrnnnej"(z angl. coarse) - je zameraná na detekciu centier výrazných veľkých regiónov pozornosti, vstupom je menší obrázok
- tzv. "jemnozrnnnej"(z angl. fine) - je zameraná na detekciu salientných oblastí menšej veľkosti, vstupom je vačší obrázok

Autori experimentovali s architektúrami sietí pre detekciu objektov (s časti popisované v kapitole 4.3) ako napríklad AlexNet [25], VGG-16 [38] a GoogLeNet [42]). 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 uviest' konvolučnú sieť VGG16[39] 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.

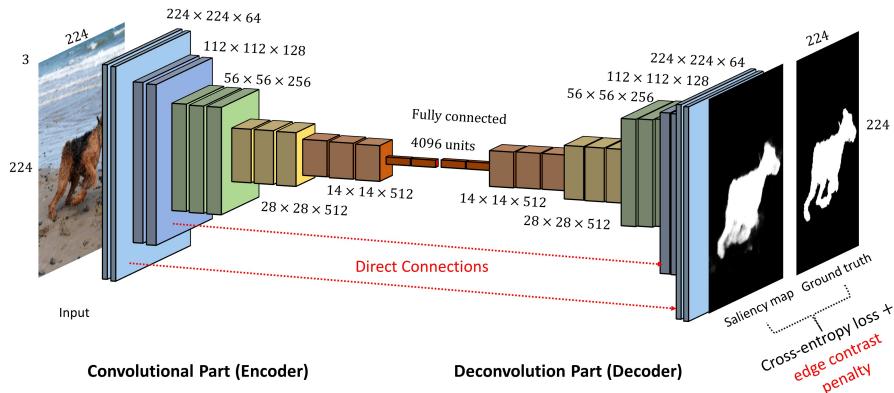


Obr. 15: Štruktúra neurónovej siete VGG-Net²⁴

Ďalším zaujímacím riešením detektie objektov v obrázku je autoenkóder od A. Meyer-a[2], ktorý využíva aj vyššie popisovanú 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 predikuje v podstate binárny obrázok, kde hodnota 1 reprezentuje pixel nájdeného objektu. Veľkou výhodou pri takejto sieti je možnosť použitia predtrenovaného modelu VGG16 pre enkóder.

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

5 Datasetsy vizuálnej pozornosti

TODO

6 Metriky pre modely vizuálnej pozornosti

Obvykle sa modely k predikcii vizuálnej pozornosti evaluujú 2 spôsobmi, a to buď vzhľadom na pohyb očí (resp. fixácie) alebo vzhľadom na originálnu mapu výraznosti. K tomu slúži veľké množstvo metrík [6][8], 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 vzorkov od negatívnych. Presnosť podľa tejto metriky je daná nasledovne:

- 0.90 - 1 = výborná
- 0.80 - 0.90 = dobrá
- 0.70 - 0.80 = priemerná
- 0.60 - 0.70 = slabá
- 0.50 - 0.60 = veľmi slabá

- sAUC - Zamiešaná 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[8], 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.

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

$$\text{kde } \sum_i P_i = \sum_i Q_i^D = 1$$

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 VGG-Net siete s autoenkóderom od A. Meyer-a (oba popisované v 4.3) na našom predpripravenom datasete.

V závislosti od toho, ktorý model, resp. varianta, sa ukáže byť najpresnejšia sa potom budeme venovať d'alošiemu experimentovaniu s pridávaním dodatočných (top-down) informácií o scéne, akými sú napr. poloha objektov konkrétnych objektov.

7.1 Dataset

Podarilo sa nám nájsť veľké množstvo datasetov, ktoré by sa potencionálne dali použiť pre našu neurónovú sieť, ako napr. CAT2000[5], NUSeF²⁶, či DUT-OMRON[46]. 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.).
- rozdielne zariadenia pre záchytenie fixácií s rôznou vzorkovacou frekvenciou

²⁶<http://mmas.comp.nus.edu.sg/NUSeF.html>

Z vyššie uvedených dôvodov sme sa preto rozhodli vybrať iba jeden dataset. Volba padla na DUT-OMRON[46], pretože:

- obsahuje viac než 5000 obrázkov
- obsahuje pohľady 5 ľudí za prvé 2 sekundy
- 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 centrované v tej istej oblasti
- veľkosť obrázkov je $400x300$
- má dobre štruktúrovane spracované dáta

Z vybraných dát sme si potom predpočítali mapy výraznosti (z angl. saliency maps) použitím fixácií a Gauss-ovho filtra. Mapy výrazností jednotlivých ľudí pre obrázky sme potom spojili dokopy. Takoto úpravou sme potom získali viac než 5000 vzoriek, na ktorých sme d'alej trénovali navrhnuté modely.

7.2 Konvolučná neurónová siet'

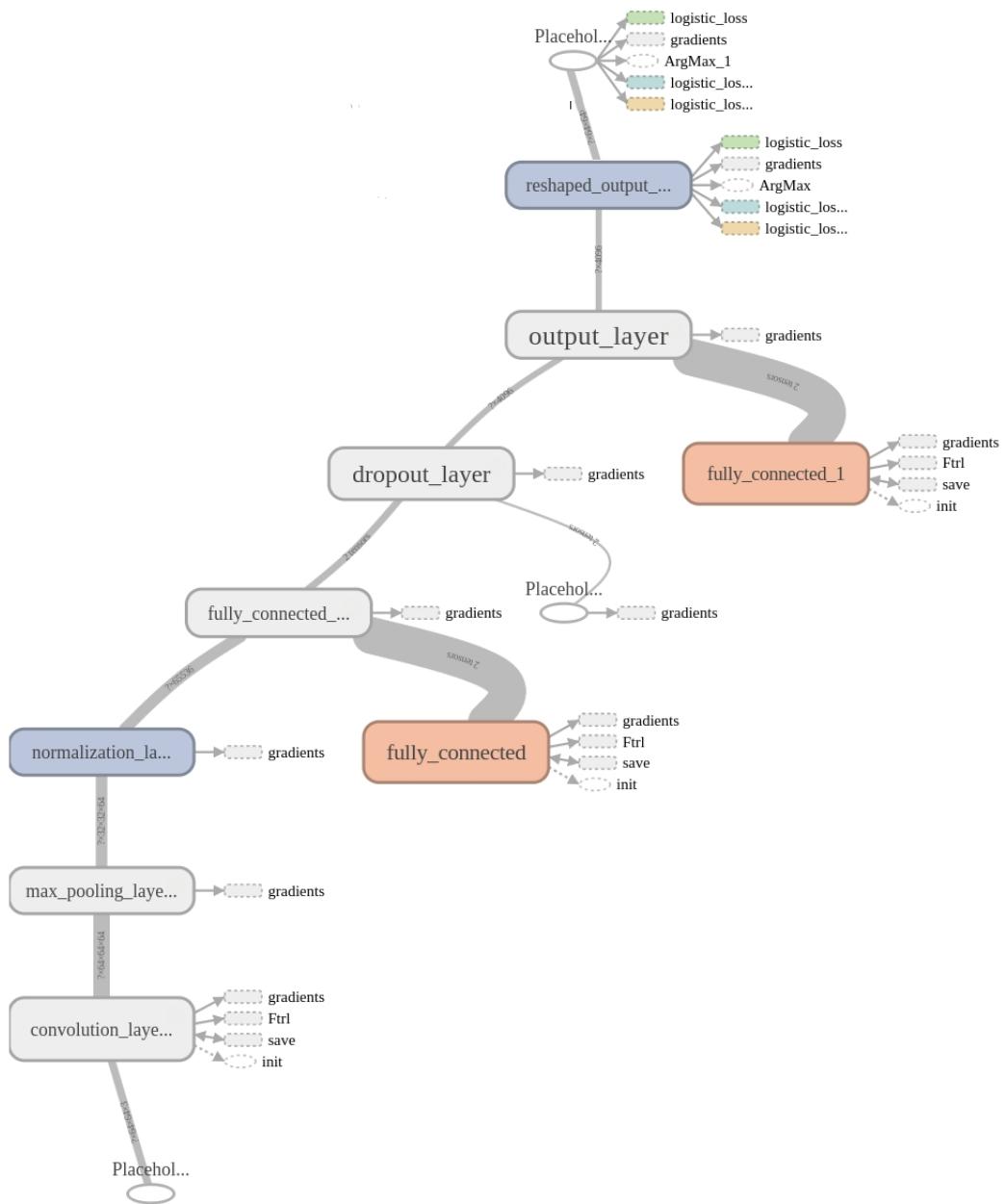
Celá architektúra je načrtnutá na schéme na obrázku 17 vytvorenej pomocou nástroja TensorBoard²⁷.

Jedná sa o jednoduchú siet' so vstupnou konvolučnou vrstvou pre spracovanie obrázkov. Táto vrstva obsahuje konvolučný filter (veľkosť $5x5$) s aktivačnou funkciou sigmoid. Výstup z nej d'alej pokračuje do vrstvy združovania, kde sa použije operácia MAX s filtrom o veľkosti $2x2$ a krokom tiež s veľkosťou 2. Po nich nasleduje vrstva normalizácie, kde je celý výstup zlúčený do jednej širokej vrstvy. Za ňou sa nachádza plne prepojená vrstva (z angl. fully-connected layer) s aktivačnou funkciou sigmoid a vrstva výpadku (z angl. dropout layer[40]), 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

²⁷https://www.tensorflow.org/get_started/summaries_and_tensorboard/

predchádzať pretrénovaniu. Za vrstvou výpadku už nasleduje iba výstupná vrstva a jej transformácia na 2D maticu, obrázok predstavujúci mapu výraznosti, ktorú chceme dostať.

Aktivačná funkcia sigmoid je použitá najmä preto, že mapa výraznosti je prakticky pravdepodobnosťné rozdelenie, t. j. siet' sa snaží predikovať pravdepodobnosti výraznosti každého pixelu. Ako algoritmus učenia sme zvolili štandardný algoritmus spätného šírenia chyby (z angl. backpropagation) s trochu extravagantným FTRL optimizérom.



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

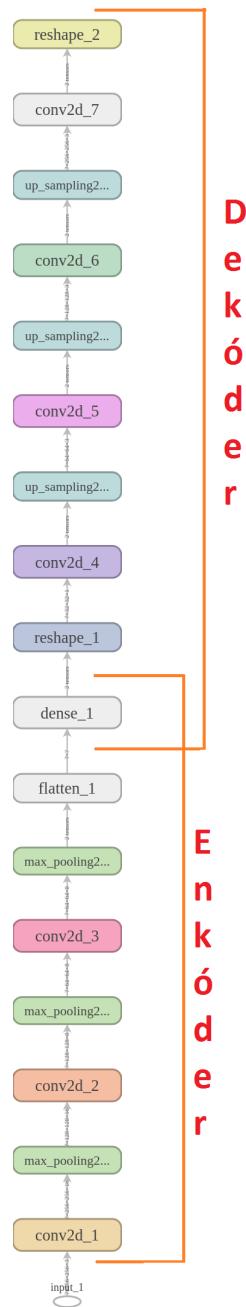
7.3 Autoenkóder

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 $3x3$ a aktivačnú funkciu ReLU (popísaná v kapitole 3.1), vrstvy združovania podobne ako pri predchádzajúcim type siete používajú operáciu MAX s filtrom o veľkosti $2x2$ 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 $3x3$ a aktivačnú funkciu ReLU. Každá z nich je ešte navýše 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 $5x5$ a aktivačnú funkciu sigmoid. Dátam na nej je ale ešte nutné predtransformovať na 2D maticu, reprezentujúcej nami

požadované mapy 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

TODO

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 vrámci predmetu Počítačové videnie²⁸ sú zachytené v kapitole 8.2. Ďalšie experimenty potom prebiehali postupne s konvolučnou neurónovou siet'ou (kapitola 8.3), konvolučným autoenkóderom (kapitola 8.4) a autoenkóderom s predtrénovanou siet'ou VGG-Net (kapitola 8.5). Pri každom modeli sme odskúšali nespočetné množstvo rôznych konfigurácií, pre popis sme ale vybrali len tie s najlepšími výsledkami pre jednotlivé časti.

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. Najdôležitejšími knižnicami sú:

- TensorFlow²⁹ - open-source knižnica pre prácu s neurónovými siet'ami a strojovým učením
- Keras³⁰ - vysoko úrovňová knižnica pre prácu s neurónovými siet'ami, beží na nižšie úrovňovými knižnicami ako Theano či TensorFlow
- Matplotlib³¹ - knižnica pre 2D vykresľovanie v Python-e
- OpenCV³² - open-source knižnica pre počítačové videnie a strojové učenie

Trénovanie modelov v experimentoch prebiehalo na GPU.

²⁸<http://vgg.fit.stuba.sk/teaching/computer-vision/>

²⁹<https://www.tensorflow.org/>

³⁰<https://keras.io/>

³¹<https://matplotlib.org/>

³²<https://opencv.org/>, <https://pypi.org/project/opencv-python/>

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 dataset-ov vizuálnej pozornosti (CAT2000[5], NUSeF³³, ...). 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 19. Nevyhovujúce časti datasetov (ako napr. abstraktné umenie, fraktály, cartoon obrázky, ...) boli odfiltrované.



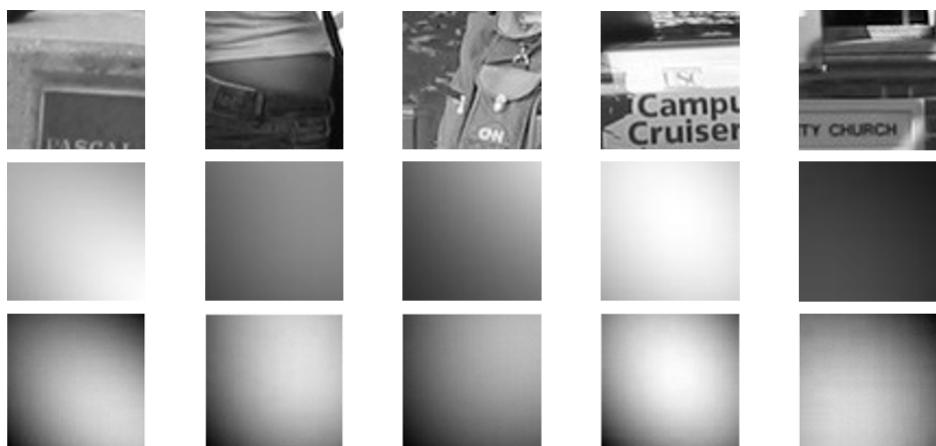
Obr. 19: *Vizualizácia extrakcie regiónov záujmu, vľavo obrázok, vpravo mapa výraznosti k nemu*

Takto získané dátá boli d'alej pre neurónovú sieť normalizované, dokopy sme si nakoniec zaistili zhruba 500 000 vzoriek.

³³<http://mmas.comp.nus.edu.sg/NUSeF.html>

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 20 možno vidieť porovanie predikovaných máp výraznosti s originálnymi a so vstupnými obrázkami.

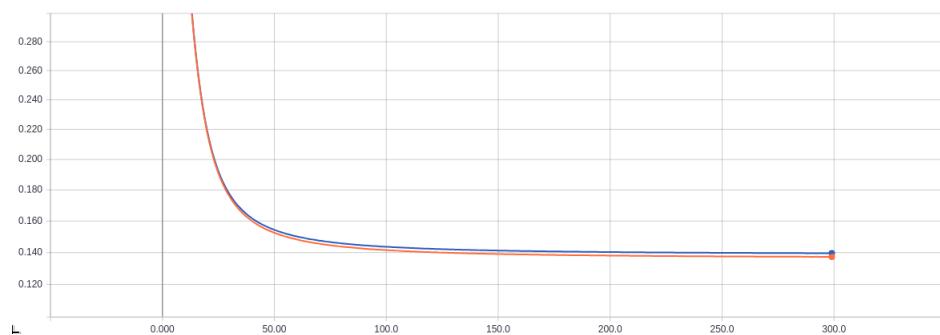


Obr. 20: Porovnanie predikcií (dolu) s originálnymi mapami výraznosti (v strede) voči vstupným obrázkom (hore)

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.

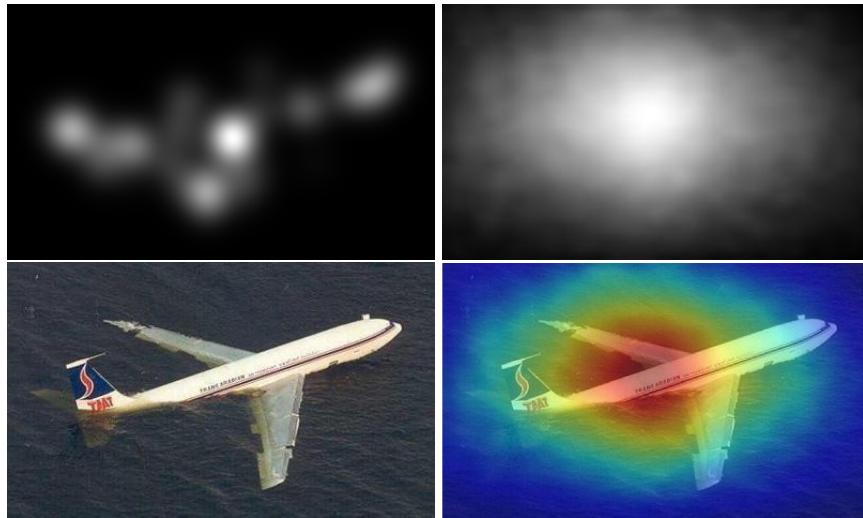
8.3 Konvolučná neurónová siet'

Ďalším experimentom bolo použitie klasickej konvolučnej neurónovej siete rovnako ako v predchádzajúcom prípade, tentokrát ale na celých obrázkoch. Ako dataset bol použitý spracovaný DUT-OMRON dataset (popísaný v kapitole 7.1) 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 stúpať⁷. Na obrázku nižšie možno vidieť vývoj jednotlivých chýb počas trénovania. Chyba bola počítaná ako priemier chýb predikcií v jednotlivých bodoch obrázka.



Obr. 21: 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 solídný 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 22.

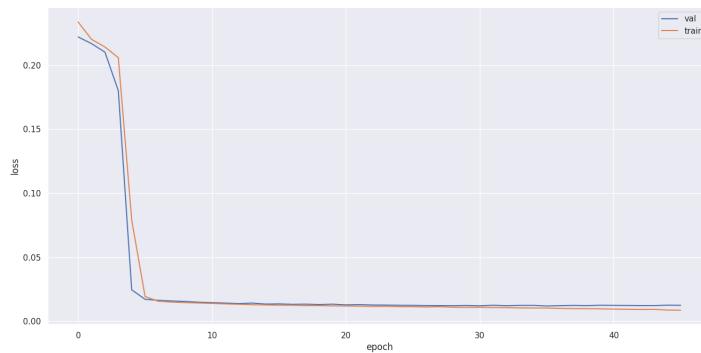


Obr. 22: Vl'avo hore originálna mapa vizuálanej pozornosti, vpravo hore predikovaná mapa, vl'avo dole obrázok pre prislúchajúce mapy výraznosti, vpravo dole zobrazenie predikcie na obrázku

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

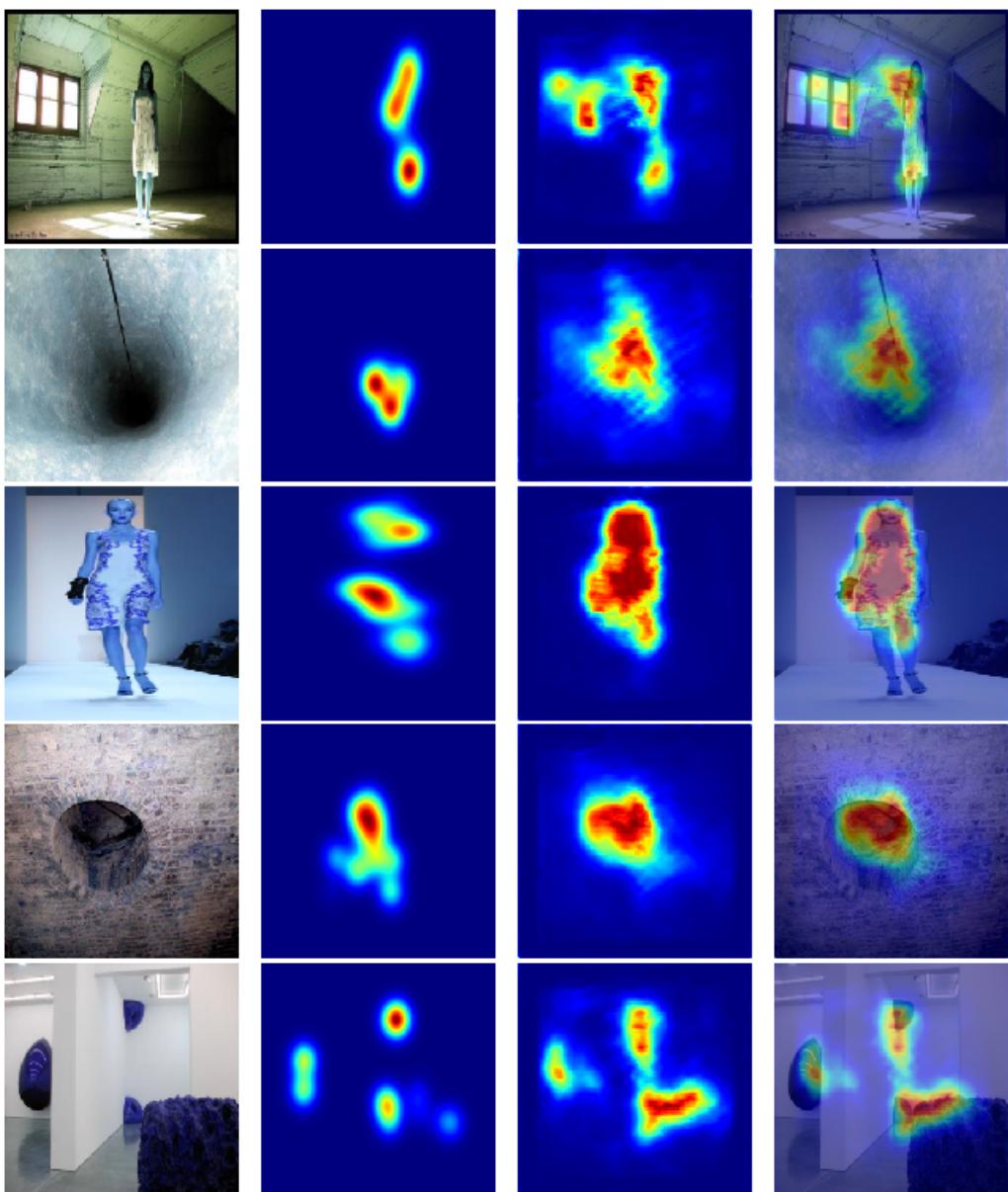
8.4 Konvolučný autoenkóder

Ďalšou architektúrou, ktorú sme sa rozhodli vyskúšať, bol konvolučný autoenkóder (navrhnutý v kapitole 7.3). Použitý dataset a jeho rozdelenie bolo rovnaké ako v predošлом 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 23.



Obr. 23: 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 24.



Obr. 24: 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

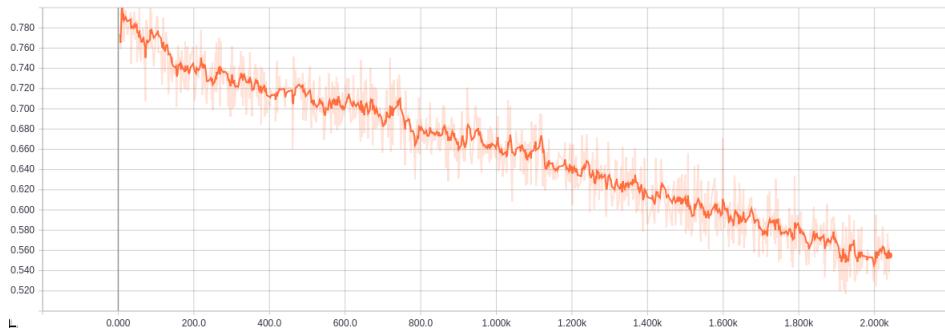
veľkosťou vstupných obrázkov. Nakol'ko autoenkóder si interne vstupné dátu zkomprimuje a zníži ich dimezionalitu, je v porovnaní zo spomínanou siet'ou z predchádzajúceho experimentu pri rovnako veľkých vstupných obrázkoch menej pamäťovo náročný.

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 trénovalia 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 8.7.

8.5 Autoenkóder s predtrénovaným modelom VGG16

Pre tento experiment sme si vybrali siet' od A. Meyer-a³⁴ popisovanú v kapitole 4.3, ktorá je voľne dostupná. Jedná sa o variáciu autoenkóderu využívajúceho siet' VGG16 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 trénovaliu na našom datasete, postupný vývoj chyby počas trénovalia možno vidieť na obrázku 25. Validácia prebiehala každých 100 epoch, trénovanie končilo v momente keď minimálna hodnota chyby za posledných *n* epoch neklesla.

³⁴https://github.com/arthurmeyer/Saliency_Detection_Convolutional_Autoencoder



Obr. 25: 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

8.7 Porovnanie výsledkov experimentov

Do výsledného porovnania sme nakoniec zobrali iba modely so zmysluplnými výsledkami - konvolučnú neurónovú siet' a autoenkóder. Konvolučná neurónová siet' pri rovnakej vstupnej veľkosti obrázkov zaberá značne väčšie množstvo pamäte ako autoenkóder, jeho trénovanie ale trvá o niečo dlhšie kvôli viacerým konvolučným vrstvám. Pri porovnaní sme sa riadili najmä metrikami vizuálnej pozornosti (kapitola 6), prehľadné zhrnutie ich hodnôt je v tabuľke 1.

Tabuľka 1: Porovnanie konvolučnej neurónovej siete a autoenkóderu spolu s hodnotami metrič vizuálnej pozornosti

	Konvolučná ne-urónová siet'	Autoenkóder
Velkosť vstupných obrázkov	64x64	256x256
Chyba na testovacích dátach	0.1459	0.01301
Metriky		
CC	0.3891	0.4634
AUC	0.6892	0.7967
sAUC	0.6631	0.7245
NSS	0.7334	1.1588

Z vyššie uvedenej tabuľky jasne vyplýva, že lepšie výsledky jednoznačne podáva práve autoenkóder, čo je koniec koncov pozorovateľné pri vizualizáciách predikcií. Predpokladáme, že to bude spôsobené najmä veľkosťou vstupných obrázkov. Keďže autoenkóder zaberá menšie množstvou pamäte, boli sme ním schopní spracovať väčšie vstupné obrázky. Pri ďalších experimentoch bude vychádzat práve z neho.

9 Zhrnutie

Vypracovaná druhá časť diplomového projektu pokračuje v nasadenom pláne z prvej časti a nadväzuje najmä na výsledky z prvotných experimentov. Ked'že sme sa tento semester rozhodli nájsť čo najlepší model pozornosti zdola nahor založený na princípe neurónových sietí, väčšina práce sa týkala najmä experimentovania s rôznymi sietami a ich parametrami. Netreba však zabúdať na problémy s datasetom, kedy nám trvalo celkom dlhý časť nájsť dostatočne vhodný. Nakoniec sme sa rozhodli pre dataset DUT-OMRON, z ktorého sa nám dokopy podarilo získať viac ako 5000 vzoriek pre modely neurónových sietí. Pri tých sme sa po niekoľkých pokusoch rozhodli zamerat' na 3 typy a to klasickú konvolučnú neurónovú siet', autoenkóder a autoenkóder s predtrénovaným model pre detekciu objektov. Práve s posledným spomínaným sme strávili značné množstvo času iba aby sme zistili, že sa jedná o slepú uličku a žiadne relevantné výstupy nedostaneme. Zostávajúce dva modely sme boli schopní natrénovať k predikcii máp výraznosti a po ich porovnaní nie len metrikami vizuálnej pozornosti nám ako lepší vyšiel autoenkóder, s ktorým sme sa rozhodli pracovať aj d'alej v nasledujúcom semestri.

Ako d'alší smer, ktorým by sa mala práca uberať, vidíme postupné dodávanie d'alších informácií o scéne neurónovej sieti. Pre začiatok by to mohli byť informácie o polohe rôznych objektov, neskôr by ale bolo ideálne prejsť aj na d'alšie faktory pozornosti zhora nadol, ako napríklad emočný kontext a iné. Bude však nutné premysliť, ako takéto informácie pre siet' reprezentovať. Ďalšou časťou práce by malo byť vykonanie vlastného experimentu pre získanie dát s dostatočným množstvom participantov, aby na nich bolo možné vyhodnotiť našu navrhnutú siet' a zároveň sa aj porovnať s už existujúcimi riešeniami.

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, 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.
- [7] J Brownlee. A gentle introduction to transfer learning for deep learning.
- [8] 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.
- [9] 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.
- [10] 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.

- [11] 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.
- [12] 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.
- [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] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.
- [15] E Bruce Goldstein. *The Blackwell handbook of sensation and perception*. John Wiley & Sons, 2008.
- [16] 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.
- [17] Richard Langton Gregory. *Concepts and mechanisms of perception*. Charles Scribner’s Sons, 1974.
- [18] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer, 2009.
- [19] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.
- [20] 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.

- [21] Laurent Itti. Visual salience. *Scholarpedia*, 2(9):3327, 2007.
- [22] Laurent Itti and Christof Koch. Computational modelling of visual attention. *Nature reviews neuroscience*, 2(3):194, 2001.
- [23] 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.
- [24] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [25] 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.
- [26] Fei-Fei Li, Andrej Karpathy, and J Johnson. Cs231n: Convolutional neural networks for visual recognition, 2015.
- [27] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*, 2015.
- [28] 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.
- [29] 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.
- [30] Fayyaz ul Amir Afsar Minhas and Asa Ben-Hur. Multiple instance learning of calmodulin binding sites. *Bioinformatics*, 28(18):i416–i422, 2012.

- [31] 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.
- [32] Michael A Nielsen. *Neural networks and deep learning*. Determination Press, 2015.
- [33] Patrik Polatsek. Saliency maps. Prezentácia, 2015.
- [34] 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.
- [35] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.
- [36] 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.
- [37] Chengyao Shen and Qi Zhao. *Webpage Saliency*, pages 33–46. Springer International Publishing, Cham, 2014.
- [38] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [39] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [40] 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.
- [41] Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39:43–62, 1997.

- [42] 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.
- [43] 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.
- [44] 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.
- [45] 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.
- [46] 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.
- [47] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

A Plán d'alšej práce

Plán a rozdelenie práce na skúškové obdobie a d'alší semester:

- refaktORIZÁCIA a upratanie doterajšej práce (kód, dokumentácia)
- experimentÁCIA s neurónovou siet'ou - autoenkóderom
- postupné dodávanie d'alších informácií o scéne sieti - pre začiatok poloha objektov
- pripravenie experimentu pre získanie dát pre pozornosť zhora nadol (top-down)
- vykonanie experimentu s čo najväčším počtom ľudí
- vyhodnotenie
- otestovanie d'alších faktorov pozornosti zhora nadol
- experimentovanie
- refactor kódu
- porovnanie výsledkov s inými riešeniami
- predbežné doplnenie dokumentácie

V závislosti od problémov a dosiahnutých výsledkov počas semestra sa bude plán predbežne meniť a upravovať.

B Obsah priloženého elektronického nosiča

Štruktúra dát na elektronickom nosiči:

- **dataset** - vzorka upraveného pripraveného datasetu pre neurónovú siet'
- **source_codes** - zdrojové kódy
- **document** - pdf verzia odovzdávanej práce