FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

KIV/UPS

Hra oko bere

Martin Hamet
A14B0254P
hamet@students.zcu.cz

24. ledna 2018

Obsah

1 Zadání **2**

2 Úvod **2**

3 Analýza **3**
3.1 Vyhledávání obrázků na základě obsahu 3
3.2 Neuronová síť 3
3.2.1 Zpětná propagace 4
3.3 Konvoluční neuronová síť 4
3.3.1 Konvoluční operátor 5
3.3.2 Nelinearita a *pooling* 5
3.4 Autoencoder 6
3.5 Datové sady obrázků 7
3.6 Metrika 8
3.6.1 Přesnost 8
3.6.2 Úplnost 9
3.6.3 F-míra 9
3.6.4 Průměrná přesnost 9

4 Závěr **10**

1 Zadání

- Implementujte hru "Oko bere". Server bude implementován v jazyce C a klient bude v Java.
- Komunikace bude realizována nešifrovaným protokolem nad TCP protokolem.
- Výstupy serveru budou v alfanumerické podobě.
- Server bude běžet na operačním systému Linux.
- Server musí být schopen obsluhovat více klientů současně.
- Musí být možné zachytit proces komunikace na úrovni aplikačního a protokolu a výpis do souboru.
- Dále budou implementovány statistiky o činnosti serveru.

2 Úvod

S popularitou digitálních zařízení, která jsou schopna pořizovat fotografie, značně roste sdílení informací ve formě obrázků a to především přostřednictvím internetu. Přestože je, již od roku 1990[5], vyhledávání snímků poměrně dobře prozkoumanou disciplínou stále se objevují nová použití s novými technologiemi jako například počítačové vidění, expertní systémy, etc. Obvykle bylo vyhledávání založené na datech spojených se obrázkem jako název a tags. Kvalita těchto přidaných dat se značně liší a nemusí být konzistentní s obsahem obrázků, což vede k využití vyhledávání podle vlastního obsahu obrázku.

Problém při vyhledávání obrázků podle jejich obsahu vzniká na dvou hlavních místech a to schopnost uživatele exaktně vyjádřit svůj požadavek a schopnost systému rozpoznat globální vlastnosti objektu snímku z jeho reprezentace[9, 7]. Jinými slovy systém musí být schopen rozpoznat například vlastnosti auta z jednotlivých pixelů snímku. Jedná se tedy o problém rozpoznání záměru ze strany uživatele a vyšší sémantiky vzhledem ke snímku.

Kolem roku 2000 se výzkum zaměřil na dvě nové metody. Využití popisu pomocí lokáních invariantních vlastností *SIFT* a popisu pomocí *bag of words*.

K posledním trendům patří *deep learning*. Jedná se o soubor algoritmů, které se snaží modelovat vyšší úroveň abstrakce obsahu obrázku pomocí mnohavrstvé hierarchické struktury nelineárních transformací. Tato struktura se

snaží napodobovat funkci lidského vnímání. Dobrým příkladem *deep learning* jsou neuronové sítě s mnoha skrytými vrstvami. V našem případě budeme testovat využití takové neuronové sítě pro vyhledávání podobných obrázků vzhledem k jejich obsahu.

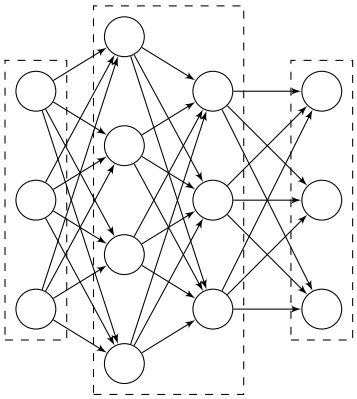
3 Analýza

3.1 Vyhledávání obrázků na základě obsahu

SIFT

3.2 Neuronová síť

Jedná se o prostředek počítačového učení, kde učíme počítač zadanou úlohu na připravených trénovacích datech. Například v případě rozpoznávání obrázků, kdy se systému předkládá velké množství snímků s předem známým obsahem. Systém tyto vstupy zpracovává a snaží se učit datové vzory odpovídající známým výsledkům.



Obrázek 1: Vrtvy sítě.

Samotná síť se skládá z mnoha uzlů (neuronů), které jsou mezi sebou hustě propojeny a tvoří tak vlastní neuronovou síť. Většinou jsou neurony organizovány do vrstev. Síť se obvykle skládá z jedné vstupní, výstupní a jedné nebo více skrytých vrstev. Data procházejí tedy sítí pouze jedním směrem od vstupní k výstupní vrstvě (viz obr.1).

Všechny vstupy neuronu mají přidělené váhy a jejich produkt jejich hodnot vstupem aktivační funkce neuronu (například sigmoida). Výstupem jednoho neuronu je tedy hodnota aktivační funkce. Tuto hodnotu O tedy spočítáme jako:

$$O = S\left(\sum_{v} {}_i \cdot w_i\right) \tag{1}$$

kde v_i je hodnotou vstupu i neuronu, w_i je váha příslušná vstupu i a S je aktivační funkce.

Na počátku jsou váhy všech vstupů jednotlivých neuronů nastaveny na náhodné hodnoty. V procesu trénování, kdy jsou do sítě zavedena vstupní data (na neurony ve vstupní vrstvě), transformací vstupních hodnot zatím náhodnými vahami získáme hodnoty aktivačních funkcí ve výstupní vrstvě. Dále je třeba provést samotný akt učení, kdy pomocí zpětné propagace (viz 3.2.1) upravujeme váhy neuronů tak, aby se výstup více blížil našemu požadovanému. Po dostatečném počtu opakování tohoto procesu bude síť schopná vrátit požadovaný výstup na podobný vstup.

Vzhledem k tomu že hledáme podobné snímky a nikoliv pouze klasifikaci do nějakých tříd je možné využít hodnoty jednotlivých neuronů v *FCL* jako popisový vektor, který budeme srovnávat s ostatními popisy v databázi, pro získání podobných obrázků.

3.2.1 Zpětná propagace

3.3 Konvoluční neuronová síť

Myšlenka konvoluční neuronové sítě *CNN*[1] spočívá ve využití předzpracování snímku, které poskytne unikátní pohled na scénu (například při detekci hran). Informace získané tímto způsobem použijeme jako vstup klasické neuronové sítě.

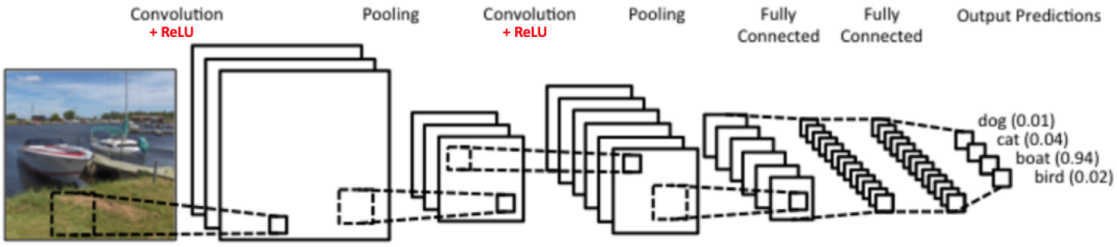
Název *CNN* je odvozen od konvolučního operátoru, který použijeme jako prostředek výše zmíněného předzpracování viz 3.3.1. Obecná CNN je založená na získávání informací z obrázku pomocí těchto konvolučních operátorů (fltrů), které si však sama vytvoří. Kombinací jednotlivých filtrů je síť schopná detekovat netriviální vlastnosti snímku. Jako příklad lze uvést kombinaci filtrů pro detekci horizontálních a vertikálních hran, která by umožnila síti rozpoznávat "rohy"například u dveří. Samozřejmě většinou filtrům, vytvořených vlastní sítí, a jejich kombinacím nelze snadno přiřadit význam srozumitelný člověkem, avšak princip přiřazování významu jednotlivým částem snímku zůstává stejný.

Celková struktura CNN je znázorněna na obr.2, kde tato síť obsahuje dvě konvoluční vrstvy za kterými vždy následuje *ReLU*[3] a *pooling* a tři plně

[1] Convolutional Neural Network

[2] Převzatý obrázek CNN `https://ujwlkarn.files.wordpress.com/2016/08/screen-shot-2016-08-07-at-4-59-29-pm.png?w=1493`

[3] Rectified Linear Unit



Obrázek 2: Struktura CNN[2]

propojené vrstvy *FCL*[4]. Tyto poslední vrstvy představují klasickou neuronovou síť, kde je každý neuron propojen s každým neuronem v následující i předchozí vrstvě jak bylo popsáno výše (3.2 a na obr. 1).

3.3.1 Konvoluční operátor

Konvoluce je matematická operace nad maticí pomocí jiné matice (jádra). První maticí je v našem případě samotný snímek respektive jeho pixely uspořádané do matice a jádrem bude samotný operátor. Výslednou hodnotu matice získáme ze vztahu:

$$F_{x,y} = \sum_{i=-k}^{k} \sum_{j=-k}^{k} M_{x-i,y-j} \cdot K_{i,j} \tag{2}$$

kde $F_{x,y}$ je hodnota výsledné matice na pozici $[x, y]$, k je dimenze jádra (jádro musí být čtvercová matice), M je zdrojová matice a K je matice jádra.

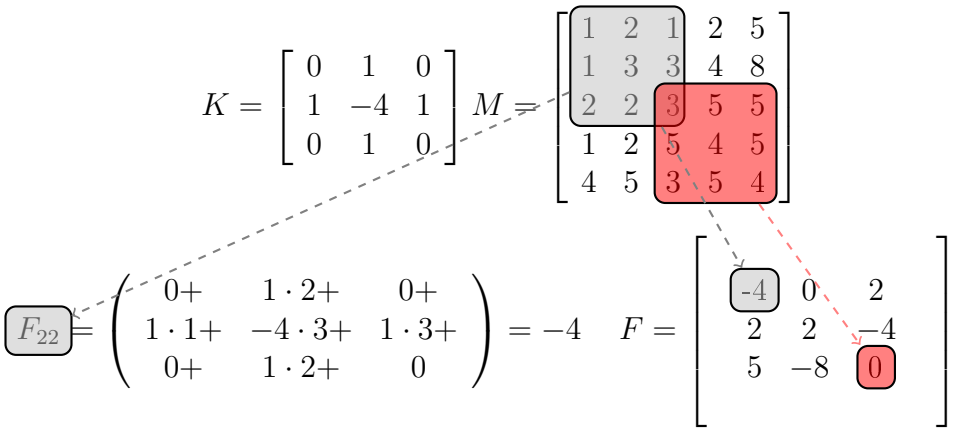
Samotný výpočet operace je lépe vidět na příkladu (viz obr.3).

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Obrázek 4: Konvoluční filtr pro detekci horizontálních hran

Vhodnou volbou hodnot jádra lze vytvořit filtr například pro detekci horizontálních hran (viz obr.4). Při použití takového filtru je třeba ošetřit kraje zdrojové matice buď bude výsledek menší v závislosti na velikosti filtru, nebo lze doplnit původní matici o nulové okraje potřebné pro filtr.

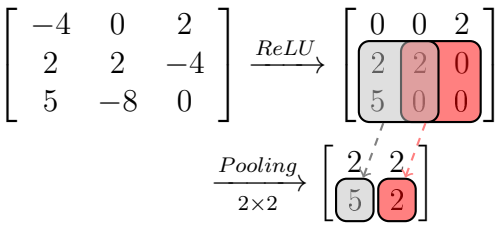
[4]Fully Connected Layer



Obrázek 3: Příklad výpočtu konvoluce

3.3.2 Nelinearita a *pooling*

Jednotlivé operace konvoluce jsou pouze lineární operace a proto s využitím $ReLU$ zavedeme do procesu zpracování nelinearitu, kdy nahradíme všechny záporné hodnoty ve výsledku hodnotou nulovou (viz obr.5). Toto nahrazení také zrychluje učení v případě nulové váhy daného neuronu tento neuron přestává mít jakýkoliv vliv na výstup (není potřeba počítat úpravu jeho váhy).

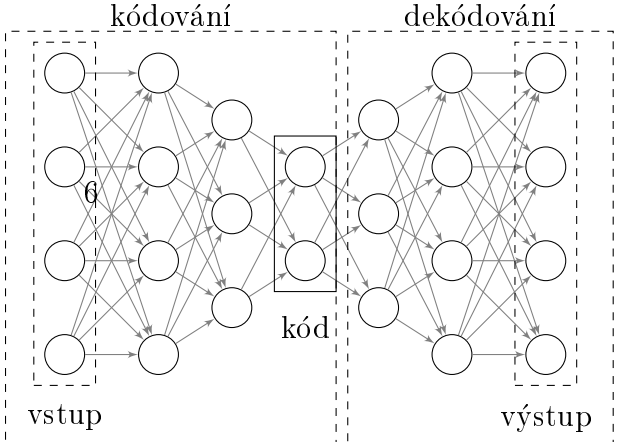


Obrázek 5: Příklad $ReLU$ a následný *pooling*

Proces *pooling* za každou konvolucí se stará o snížení dimenze jednotlivých výstupů z filtrů se zachováním jejich nejdůležitějších součástí. Jedná se tedy o zmenšení výsledné matice, kde definujeme okolí pro každý prvek a z daného okolí bereme pouze maximální hodnotu (viz obr.5).

3.4 Autoencoder

Jedná se o neuronovou síť určenou pro učení bez uči-



tele. Není tedy nutné předem znát správný výsledek při trénování. Princip funkce autoenkoderu spočívá v učení sítě co nejlépe replikovat vstupní hodnoty na svém výstupu. Tedy požadovaný výstup je vstupní vektor sítě a proto vstupní i výstupní vrstva musí mít stejný počet neuronů. Struktura autoenkoderu se obecně skládá z kódovací a dekódovací části jak je vidět na obr. 6. Zpracováním vstupu v kódovací části získáme kódové slovo, které se dekódovací část snaží převést zpět na původní vstup. Výstupem je vzniklý kód a nikoliv výstupní vrstva, která slouží pouze jako nástroj pro učení. Vytvářením "úzkého hrdla"(vrstvy s menším počtem neuronů) zajistíme ztrátovou kompresi vstupu a nutíme tím síť k vytvoření robustního kódování (odtud název autoenkoder) ze kterého bude možné co nejlépe sestavit původní vstup.

V kombinaci s konvolucí resp. CNN se autoenkoder snaží najít kódování snímku pomocí jednotlivých filtrů, tedy popis snímku pomocí jeho význačných vlastností. Takto vzniklý kód by měl obsahovat zobecněný popis snímku, který bude možné použít pro porovnání a výběr podobných snímků.

3.5 Datové sady obrázků

Caltech 101(256)[2][3] Jedná se o databázi čítající 9 000(30 000) obrázků dělených do 101(256) kategorií. Obrázky byly pořízeny s využitím *Google Image Search* a ručně roztříděny do kategorií. Kategorie průměrně obsahují 90 (119) prvků. Jednotlivé snímky jsou průměrně rozměrů 300×200pixelů.

Corel[1] Dataset obsahuje 10 000 obrázků rozdělených do 80 kategorií. Obrázky mají formát 192×128 nebo 128×192, který pevně daným počtem pixelů usnadňuje zpracování.

ImageNet Velmi rozsáhlá databáze s více než 15 miliony obrázků s vysokým rozlišením dělených do 22 000 kategorií. Snímky byly získány z webu a ručně řazeny do tříd. Pro zpracování je možné využít podmnožin určených

pro pro *ImageNet Large-Scale Visual Recognition Challenge*(ILSVRC)[6]. Nevýhodou jsou různé formáty rozlišení.

CIFAR10 Soubor se skládá z 60 000 obrázků rozdělených do 10 tříd. Každý snímek je ve formátu 32×32 pixelů. Set obrázků je rozdělen do trénovacích a testovacích podmnožin pro každou třídu. Bylo by možné použít rozšířenou verzi *CIFAR100*.

3.6 Metrika

Systémy v oblasti klasifikace a rozpoznávání je třeba nějakým způsobem hodnotit, aby bylo možné vyjádřit jejich úspěšnost. Takové hodnocení umožňuje systém porovnat s konkurencí, nebo zjistit efekt jeho modifikací při vývoji.

Hlavní otázkou je jakým způsobem systém hodnotit. Z principu rozpoznávání vyplývají dva hlavní měřitelné atributy, úplnost a přesnost (viz 3.6.2 resp. 3.6.1).

Protože systémy popsané dvěma atributy není tak snadné porovnávat, v praxi je vhodnější využít popisu pouze jednou hodnotou. Takovou hodnotu je třeba přizpůsobit konkrétní úloze. Pro popis se nabízí několik možností jako užití *F-measure*, průměrná přesnost, střední geometrická přesnost (GMAP), střední průměrná přesnost (MAP), etc.

V našem případě bude výsledkem dotazu seznam obrázků seřazený podle relevance určené systémem. Bude tedy vhodné využít průměrné přesnosti (viz 3.6.4), která zohledňuje pořadí nalezených výsledků. Jak je vidět na obr.7 průměrná přesnost značně zvýhodňuje prvky podle jejich pořadí v odpovědi na dotaz. Pro popis použijeme tedy metriku střední průměrné přesnosti, která zhodnotí výsledky všech použitých dotazů při testování s velkým důrazem na uspořádání získaného výsledku. Bylo by možné použít modifikaci GMAP, která vyžaduje dostatečný výkon ve všech dotazech. Náš systém nebude mít omezení na nejhorší povolenou úspěšnost a nebudeme tedy tuto metriku uvažovat.

3.6.1 Přesnost

Přesnost neboli *precision* hodnotí kolik relevantních výsledků systém poskytl. Jedná se o podíl relevantních a všech získaných výsledků (viz rovnice č. 3). Procentuální přesnost intuitivně ukazuje úspěšnost rozpoznávání pro

daný dotaz. V případě že systém vrátil všechny relevantní výsledky hodnota přesnosti bude 100%. Výpočet přesnosti $P@n$ provedeme dle vzorce:

$$P@n = \frac{R_Q}{Q} \tag{3}$$

kde Q je počet získaných výsledků na dotaz a R_Q je počet relevantních výsledků ze všech získaných. Samotnou přesnost značíme jako $P@n$, kde n udává počet požadovaných výsledků dotazu. Pokud by v datech existoval pouze jeden relevantní výsledek přesnost pro velké n by se značně snížila a bude tedy třeba tento fakt zohlednit.

3.6.2 Úplnost

Samotná přesnost nevypovídá o úplnosti dotazu. Jinými slovy chtěli by jsme zohlednit počet všech relevantních prvků ve zdroji. Pro tyto potřeby se používá hodnota úplnosti neboli *recall*. jedná se o poměr získaných relevantních výsledků a všech relevantních prvků v prohledávaném zdroji. Výpočet úplnosti $R@n$ pro dotaz na n prvků provedeme dle vzorce:

$$R@n = \frac{R_Q}{R} \tag{4}$$

kde R_Q je počet relevantních výsledků ze všech získaných, podobně jako v případě výpočtu *Precision*(viz 3.6.1), a R je počet všech relevantních prvků v databázi.

Hlavní nevýhodou tohoto atributu je nutnost znalosti všech relevantních prvků v databázi na daný dotaz.

3.6.3 F-míra

Hodnota *F-measure*[4] udává harmonický průměr úplnosti a přesnosti (viz rovnice č.5). Vztah pro výpočet by bylo možné dále upravit, pokud by jsme chtěli klást větší důraz na přesnost nebo úplnost. Přestože *f-measure* zahrnuje oba výše zmíněné atributy, její hodnotu nelze intuitivně interpretovat a sloužila by pouze jako prostředek pro porovnání s případnou předpojatostí vůči jednomu z atributů.

$$F_1 = 2 \cdot \frac{P \cdot R_Q}{P + R_Q} \tag{5}$$

3.6.4 Průměrná přesnost

Tato hodnota kombinuje úplnost a přesnost pro daný počet dotazovaných prvků. Průměrná přesnost[8] $AP@n$ je průměrem hodnot přesností $P@n$ ze všech relevantních výsledků n. Hodnotu vypočítáme ze vztahu:

$$AP@n = \frac{\sum_n P@n}{R_Q} \tag{6}$$

kde $P@n$ je přesnost pro n požadovaných prvků (viz vzorec č. 3) a R_Q je počet získaných relevantních prvků. Příklad výpočtu je vidět na obr. 7.

Střední průměrná přesnost Hodnotou mAP rozumíme pouze průměr jednotlivých průměrných přesností AP. Výpočet je znázorněn na obr. 7.

Předpokládáme že máme celkem 4 relevantní prvky pro oba dotazy.

Query #1	F	T	T	F	T	F
$R@n$	0	0.25	0.50	0.50	0.75	0.75
$P@n$	0	0.50	0.67	0.50	0.60	0.50

Query #2	T	F	T	F	F
$R@n$	0.25	0.25	0.50	0.50	0.50
$P@n$	1.00	0.50	0.67	0.50	0.40

střední průměrnou hodnotu mAP spočítáme jako:

$$AP_{Q1} = (0.50 + 0.67 + 0.60)/3 = 0.59$$
$$AP_{Q2} = (1.00 + 0.67)/2 = 0.83$$
$$mAP = (0.59 + 0.54)/2 = 0.71$$

Obrázek 7: Příklad dotazu

T značí relevantní prvek
F značí irelevantní prvek

4 Závěr

Nejpoužívanější metody pro klasifikaci obrázků využívají vlastností snímku. Jedná se o vnější znaky jako "tags", "bag of words", etc. a vnitřní související s vlastním obsahem. V našem případě se zabýváme pouze vnitřními. Problémem zůstává volba metody extrakce takových vlastností, která významně ovlivňuje úspěšnost klasifikace při konkrétní aplikaci systému. Protože hledáme podobné obrázky k danému vzoru v databázi, pouhé určení třídy daného snímku jako v běžném klasifikačním systému je krajně nedostačující popis. Hledáme tedy věrný popis scény, který je ovšem dostatečně obecný pro vyhledání podobných snímků.

Princip konvoluce se ukazuje jako vhodná cesta zkoumání obsahu. V kombinaci s neuronovou sítí dokážeme do jisté míry napodobovat lidské vidění ve smyslu identifikace objektů v závislosti na jejich významných rysech. Konvoluční filtry bude vytvářet samotná `CNN` ve fázi trénování, odpadá tím náročný úkol výběru vhodné sady a jejich kombinace.

Jako způsob trénování se jeví vhodné sestavit síť jako autoenkodér. Vstup (snímek) bude při trénování i požadovaným výstupem. Jinými slovy síť se bude učit vybírat vlastnosti nastavováním konvolučních filtrů tak, aby byla schopná co nejlépe replikovat stejný snímek na výstupu. Jako popis se tedy nevyužije samotný výstup `CNN`, ale její část ve, které jsou zakódované vlastnosti snímku. Vzhledem k tomu že se síť učí replikovat snímek na vstupu mělo by se jednat o jeho dobrou reprezentaci.

Výběr datových sad pro zpracování závisel především na rozměrech obrázku, které bude třeba upravit tak, aby odpovídal počet pixelů a počet neuronů vstupní vrstvy. Příliš zásadní úpravy by značně zhoršovali funkčnost rozpoznávání (příliš velké zploštění, roztažení etc.).

Jako metrika určování úspěšnosti systému byla zvolena střední průměrná přesnost, která velice dobře popisuje daný problém a to především vzhledem k pořadí získaných výsledků. Chceme vytvořit systém, který vrátí seznam snímků řazený podle jejich podobnosti vzoru.

Reference

[1] Yixin Chen, Jinbo Bi, and James Z Wang. Miles: Multiple-instance learning via embedded instance selection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(12):1931–1947, 2006.

[2] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. In *Pattern Recognition and Machine Intelligence*, 2004.

[3] Griffin G., Holub AD., and Perona P. The caltech 256. Technical report, California Institute of Technology, 2006.

[4] Powers and David M. W. Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, pages 37–63, 2011.

[5] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: A power tool for interactive content-based image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):644–655, 1998.

[6] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[7] Ji Wan, Dayong Wang, Steven C. H. Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li. Deep learning for content-based image retrieval: A comprehensive study. In *ACM Multimedia*, 2014.

[8] Ethan Zhang and Yi Zhang. *Average Precision*, pages 192–193. Springer US, Boston, MA, 2009.

[9] Wengang Zhou, Houqiang Li, and Qi Tian. Recent advance in content-based image retrieval: A literature survey. *CoRR*, abs/1706.06064, 2017.