Google Landmark Retrieval Challenge

Petra Martinjak, Matija Šelendić, Stjepan Teskera,
Stjepan Zbiljski
PMF - Matematički odsjek
Sveučilište u Zagrebu
Zagreb, Hrvatska

Tomislav Šmuc
Strojno učenje

Uvod i motivacija

Problem dohvaćanja slike fundamentalan je problem u području računalnog vida. Taj problem sastoji se u sljedećem: možemo li za danu sliku pronaći sličnu u velikoj bazi podataka?
Sustav dohvaćanja slike računalni je sustav za pregledavanje, pretraživanje i dohvaćanje slike iz velike baze podataka u kojoj se nalaze digitalne slike. Prvi sustav za dohvaćanje slike iz baze podataka razvijen je na MIT-u 1987. od strane Banireddyja Prasaada, Amara Gupte, Hoo-mina Toonga i Stuarta Madnicka.
Većina tradicionalnih i uobičajenih metoda dohvaćanja slike koristi neku metodu dodavanja metapodataka kao što su naslovi, ključne riječi ili opisi da bi se dohvaćanje izvršavalo preko pribilježenih riječi. Ručno bilježenje slika vremenski je iscrpno, mukotrpno i skupo te se u svrhu rješavanja tog problema provela velika količina istraživanja nad automatskim pribilježavanjem slika. Međutim, razvojem tehnologije prednost se počela davati dohvaćanju slike temeljem sadržaja (CBIR). CBIR je primjena računalnog vida u dohvaćanju slike. CBIR ima za cilj izbjegavati korištenje tekstualnih opisa i u zamjenu dohvaća slike na temelju sličnosti u njihovom sadržaju (teksture, boje, oblici itd.) sa pruženom slikom upita ili određenim svojstvima slike.
Sve veća potreba za dohvaćanjem slike iz baze javlja se u raznim domenama, primjerice data-miningu, medicini, edukaciji, prevenciji zločina i meteorologiji. Jedno od područja gdje dohvaćanje slike nalazi široku primjenu je turizam, konkretnije, dohvaćanje slika znamenitosti, budući da su to objekti koje ljudi često fotografiraju.
Podaci koji se koriste preuzeti su sa Kaggle službene stranice natjecanja [1]. Radi se o podacima iz najvećeg svjetskog skupa podataka za dohvaćanje slika, na više od milijun fotografija prikazano je 15 tisuća jedinstvenih svjetskih znamenitosti.

Cilj projekta

Svaka slika u bazi podataka označena je svojim jedinstvenim id-em. Pretpostavlja se da za danu sliku ne postoje nikakvi dodatni podaci o njoj, poput geolokacije ili opisa. Ako za neke slike takvi podaci i postoje, oni nisu uzeti u obzir. Rješenje problema u potpunosti se oslanja na sadržaj slike. Cilj ovog projekta je izraditi program koji će za određenu sliku znamenitosti zadanu njenim id-em pronaći sve slike koje prikazuju istu znamenitost, ukoliko takve slike postoje i ispisati njihove id brojeve.
Očekujemo da će naše rješenje odgovoriti na dva ključna pitanja:

1. Za upit u obliku id-a slike znamenitosti, koje sve slike predstavljaju tu istu znamenitost?
2. S kolikom preciznošću znamo da te slike doista predstavljaju istu znamenitost, tj kolika je točnost našeg rješenja?

Budući da je pristup problema dohvaćanja slike isključivo na temelju njenog sadržaja relativno nov, ne postoji optimalna robusna metoda za rješenje ovog problema. Smatramo da ćemo se zbog toga susresti s mogućim poteškoćama vezanim uz uspješnost rješenja te da će njegova točnost biti značajno manja od one dobivene u state-of-the-art pristupima.

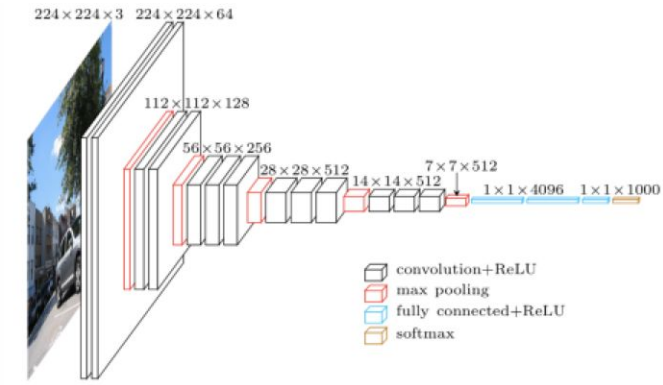
Pristup rješavanju problema

Podaci i obrada podataka

Na Kaggle stranici natjecanja dana su 3 skupa podataka: index.csv, test.csv i sample_submission.csv. U skupu index.csv (slike iz ovog skupa ćemo u daljnjem tekstu zvati slike iz baze) nalaze se slike koje predstavljaju bazu unutar koje ćemo tražiti slike slične našem upitu. Ovaj skup sadržavao je 1.091.756 slika. U skupu test.csv (u daljnjem tekstu query slike ili slike upita) nalaze se slike koje će predstavljati upite. U ovom skupu nalazilo se 114.943 slike. Oba skupa podataka imaju istu strukturu: za svaku sliku dan je njen id i url s kojeg se slika može preuzeti. Skup sample_submission.csv služi kao primjer formata u kojem treba predati rješenje. Za svaku query sliku treba navesti njen id i zatim id-eve slika iz baze koje su joj slične, počevši of najsličnije prema manje sličnima. Prvi korak našeg projekta bio je preuzimanje svih slika (i iz baze i iz queryja). Kaggle je u tu svrhu ponudio skriptu koju smo iskoristili i prepravili tako da su sve preuzete slike u konačnici bile u .jpg formatu i u RGB-u te su preuzete u rezoluciji 256x256. Ovaj postupak na našim računalima trajao je gotovo 3 dana. Osim toga, uzeli smo i već preuzete slike formata 128x128 ponuđene na Kaggle stranici od strane jednog od korisnika [6]. Razlog tome bila je ušteda vremena potrebnog za direktno preuzimanje slika preko linkova korištenjem skripte. Originalna ideja bila je raditi sa slikama najmanje 256x256 rezolucije, ali smo zbog ograničenja izazvanih tehnologijom koju smo koristili (privatni laptopi) u konačnici odabrali manje slike.

Feature extraction

Budući da slika s kojima radimo ima puno, rad s toliko slika bio bi prezahtjevan za većinu računala. Stoga je potrebno napraviti feature extraction kako bi se smanjilo opterećenje na memoriju. Odlučili smo se za korištenje već postojećih neuronskih mreža pretreniranih na ImageNet-u. ImageNet [5] je ogromna baza organiziranih i klasificiranih slika koja se koristi za pretreniranje modela za feature extraction. Naime kako bi se napravila konvolucijska neuralna mreža za feature extraction koja će davati zadovoljavajuće rezultate, potrebno je koristiti puno heuristika i ručnih provjera kako bi se iskombinirale sve bitne značajke slike. Osim što je nepraktično i zamorno, to često rezultira i overfittingom jer su skupovi slika, s kojima se uobičajeno radi, premali. Pretreniranjem mreže na velikoj bazi slika kao što je ImageNet, izbjegava se overfitting i poboljšava točnost modela. Istraživanjem različitih postojećih modela isprva smo se odlučili za ResNet. ResNet je bio prvi odabir jer se u postojećim radovima pokazalo da daje točnije rezultate, ima veću dubinu i manje parametara. Međutim za njega je potrebno koristiti veće slike (256x256) te računanje raditi na grafičkoj procesorskoj jedinici kako bi se izvršilo u razumnom vremenu. To se pokazalo teško izvedivim na opremi kojom smo raspolagali, stoga smo se ipak odlučili za manje točan, ali i manje zahtjevan model VGG16 kojeg smo mogli koristiti na manjim slikama. VGG16 je konvolucijska mreža koja koristi 3x3 konvolucijske layere u kombinaciji sa ReLu aktivacijskom funkcijom i 2x2 pooling. Shematski prikaz može se vidjeti na slici 1. ReLu funkcija je funkcija oblika $f(x) = max(0,x)$.



Slika 1: Shematski prikaz VGG16 modela

Koristeći VGG16 konvolucijsku neuralnu mrežu i dodatne optimizatore izradili smo model koji nam je omogućio da pohranimo slike samo kao njihove najvažnije značajke. Tehnički detalji ovog postupka biti će podrobnije opisani u odjeljku 'Korištene metode i algoritmi'. Na slici 2 može se vidjeti usporedba različitih modela kako je navedno u njihovoj dokumentaciji.

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.715	0.901	138,357,544	23
VGG19	549 MB	0.727	0.910	143,667,240	26
ResNet50	99 MB	0.759	0.929	25,636,712	168
InceptionV3	92 MB	0.788	0.944	23,851,784	159
InceptionResNetV2	215 MB	0.804	0.953	55,873,736	572
MobileNet	17 MB	0.665	0.871	4,253,864	88
DenseNet121	33 MB	0.745	0.918	8,062,504	121
DenseNet169	57 MB	0.759	0.928	14,307,880	169
DenseNet201	80 MB	0.770	0.933	20,242,984	201

Slika 2: Dokumentacija za pojedine modele

Usporedba značajki

Značajke za sve slike dobivene u prethodnom koraku spremljene su u obliku vektora značajki. Sada je potrebno za svaku query sliku pronaći 100 njoj najsličnijih slika i poredati ih po sličnosti od najsličnije do najmanje slične. To ćemo učiniti tako što ćemo gledati udaljenosti vektora značajki query slike od vektora značajki svih slika iz baze i tražiti one slike iz baze čiji vektori značajki su najmanje udaljeni od vektora značajki query slike. Isprobali smo dvije mjere za udaljenost, euklidsku udaljenost (iz norme 2) i kosinusovu sličnost. Euklidska udaljenost dala je značajno bolje rezultate. Ovaj postupak rađen je uz pomoć funkcija iz faiss biblioteke koje će biti opisane u sljedećem poglavlju, no ono što valja naglasiti je da se pri tome koristi bruteforce postupak traženja najmanje udaljenih vektora [4]. Rezultat je pohranjen u obliku traženom od strane Kaggle-a: u svakom retku .csv datoteke nalazi se id query slike i zatim id-evi slika iz baze počevši od najsličnije, pa do najmanje sličnih.

Korištene metode i algoritmi

Sve dijelove istraživanja izvodili smo u Anacondi 3.7, koristeći Spyder 3.8 sučelje. Rješenje je podijeljeno u dvije glavne skripte i jednu pomoćnu. Pomoćna skripta služi za preuzimanje slika sa linkova danim u podacima s Kaggle-a. Ova skripta je također preuzeta sa Kaggle stranice u sklopu materijala danih za natjecanje i promijenjena kako bi slike bile u odgovarajućoj rezoluciji. Skripta koristi *request* modul iz *urllib* biblioteke za preuzimanje sadržaja sa url-a i *Image* modul iz *PIL* biblioteke za rukovanje slikama (promjena u RGB boje, promjena rezolucije, spremanje u .jpg format). U prvoj glavnoj skripti izvedena je izrada modela za feature extraction i upotreba tog modela kako bi se slike pretvorile u vektore njihovih značajki. Kao što je opisano u prethodnom poglavlju, konačno rješenje koristi VGG16 pretrenirane neuronske mreže za izradu modela. U tu svrhu korištena je Keras biblioteka [3][2] koja sadrži sve potrebno za izradu željenog modela. Umjesto gotovog modela dostupnog u Keras-u, odlučili smo gotov model prepraviti drugačijim poolingom kako bismo smanjili mogući overfitting. U tu svrhu, nakon posljednjeg konvolucijskog sloja umjesto max poolinga radimo GlobalAverage pooling. Za optimizator smo koristili Adam algoritam sa stupnjem učenja od 0.0001, jer

smo istraživanjem saznali da taj algoritam nije memorijski zahtjevan. Funkcija gubitka (loss function) je *categorical_crossentropy*, često korištena funkcija u problemima sa više klasa. Tako konsturiran model stvara se pozivom *get_model* funkcije u skripti. Osim te funkcije, postoji i *test_generator* funkcija koja služi generiranju serija slika (batches). Te serije su input Kerasovoj *predict_generator* funkciji koja se poziva na modelu. Rezultat su značajke slika spremljene u vektore.
U drugoj skripti traže se međusobne udaljenosti značajki slika kako bi se dobila sličnost slika. Kao što je već spomenuto, u svrhu usporedbe vektora značajki koristi se faiss biblioteka i njezine metode *IndexFlatL2*, *IndexFlatIP* i *search* kako bi se pronašli najbliži vektori značajki. *IndexFlatL2* metoda služi tome da se za udaljenost uzme euklidska udaljenost. *IndexFlatIP* metoda gleda unutarnji produkt i kad se koristi na normaliziranim vektorima daje kosinusovu sličnost.
Osim spomenutih biblioteka, metoda i algoritama, koristili smo i brojne pomoćne biblioteke od kojih je najvažnije spomenuti Numpy i Pandas.

Rezultati

Kaggle za mjeru greške koristi sljedeću formulu:

$$mAP@100 = \frac{1}{Q}\sum_{q=1}^{Q}\frac{1}{min(m_q, 100)}\sum_{k=1}^{min(n_q,100)}P_q(k)rel_q(k)$$

- Q je broj query slika koje opisuju znamenitosti iz baze
- m_q je broj slika iz baze koje sadržavaju znamenitost zajedničku s query slikom q (vrijedi samo za upite koji označuju znamenitosti iz baze, $m_q \neq 0$)
- n_q je broj predikcija za upit q
- $P_q(k)$ je preciznost na k-tom rangu za q-ti upit
- $rel_q(k)$ označuje značajnost predikcije k za q-ti upit: 1 ako je k predikcija točna, 0 inače

Kaggle prikazuje dvije vrste rezultata: public i private. Razlika je u tome što public evaulira rezultate nad 34% podataka testnog skupa, a private nad ostalih 66%.
Prvi pokušaj rješavanja problema bio je računanjem kosinusove sličnosti između normaliziranih vektora značajki query slike te pojedine slike iz baze, metodom IndexFlatIP iz faiss biblioteke. Nakon uploadanja rezultata, public rezultat iznosio je 0.4%, a private 0.9%. Nakon toga, pokušali smo dobiti bolje rezultate metodom IndexFlatL2 što radi sličnu stvar samo za mjeru koristi euklidsku udaljenost. Nakon obrađenih 114.943 slika iz testnog skupa tom metodom dobili smo public rezultat od 4.3% što je znatno poboljšanje u odnosu na prvu metodu. Private rezultat nam se povećao na 6.6%.
Pretpostavljamo da bismo korištenjem ResNeta, što je prvotno bio naš odabir, dobili puno bolje rezultate, no njegovo korištenje zahtijevalo je preveliku količinu vremena i memorije zbog većih slika te korištenje GPU-a umjesto CPU-a.
Drugi projekti koji su imali bolje resurse mogli su koristiti i kompleksnije modele, što se vidi i na rezultatima. Najbolji rezultat na cijelom natjecanju iznosi 62.5% na public setu podataka i 62.7% na private setu.

Zaključak

Rezultati koje smo dobili nisu zadovoljavajući, ali smo u početku očekivali da bi se to moglo dogoditi. Razlog tome je prvenstveno zbog slabih računala, a i zbog činjenice da su i drugi rezultati većinom bili prilično niski, što samo dodatno ukazuje na netrivijalnost problema.
Iz gornjih rezultata možemo zaključiti:

- Za računanje udaljenosti između vektora značajki euklidska udaljenost daje daleko bolje rezultate od kosinusove sličnosti
- VGG mreža nije adekvatan model za ovakve probleme - ResNet ili neki drugi model bi bio puno bolji
- Računanje na GPU značajno ubrzava treniranje neuronskih mreža

Daljnja istraživanja

Zbog ograničenja u tehnologiji s kojima smo se susreli u izvedbi ovog istraživanja, bili smo primorani prijeći na jednostavniji model i manje slike te su dobiveni rezultati bili lošiji od očekivanog. U budućim istraživanjima trebalo bi ponoviti postupak koristeći kompleksnije ResNet neuronske mreže za izradu modela u kombinaciji s većim slikama kao što je prvotno bilo zamišljeno, ali nije bilo moguće ostvariti zbog već spomenutih ograničenja.
Većina projekata s boljim rezultatima koristili su ResNet neuronske mreže, k-najbližih susjeda s euklidskom udaljenosti i na kraju query ekspanziju difuzijom.
Projekt CVSPP & Visual Atoms [7] ostvario je najbolje rješenje za zadani problem. Iz ResNet i ResNeXt modela generirali su globalne deskriptore čiju su kombinaciju dalje koristili.Zatim su koristili k-najbližih susjeda, augmentaciju baze podataka i query ekspanziju te dobili točnost 62.5%.
Još jedno značajno poboljšanje bilo bi računanje na GPU, jer bi to značajno ubrzalo postupak treniranja mreža i omogućilo rad s boljim modelima.

Literatura

[1] Kaggle Challenge podaci, dostupni na: https://www.kaggle.com/c/landmark-retrieval-challenge/data

[2] Karen Simonyan, Andrew Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, verzija 6, 10. travnja 2015. https://arxiv.org/abs/1409.1556

[3] Keras dokumentacija, dostupna na: https://keras.io/

[4] Faiss dokumentacija, dostupna na: https://github.com/facebookresearch/faiss/wiki

[5] ImageNet baza podataka, dostupna na: http://www.image-net.org/

[6] Kaggle diskusija sa slikama, dostupna na: https://www.kaggle.com/c/landmark-retrieval-challenge/discussion/56194

[7] Kaggle diskusija s najboljim rješenjem, dostupna na: https://www.kaggle.com/c/landmark-retrieval-challenge/discussion/57855