Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/08BE34Nikolic

SISTEM ZA PRIKUPLJANJE I ANALIZU FOTOGRAFIJA SA ULIČNIH KAMERA UPOTREBOM MAŠINSKOG UČENJA

SYSTEM FOR COLLECTING AND ANALYZING PHOTOS FROM STREET CAMERAS USING MACHINE LEARNING

Aleksandar Nikolić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Rad opisuje distribuiranu arhitekturu i implementaciju sistema za prikupljanje i analizu fotografija sa uličnih kamera upotrebom modela mašinskog učenja za detekciju objekata baziranog na konvolucionim neuronskim mrežama.*

Ključne reči: *Elasticsearch, mašinsko učenje, mikroservisi, detekcija objekata*

Abstract – *This paper describes the distributed architecture and implementation of the system for collecting and analyzing photos from street cameras using machine learning model for object detection based on convolutional neural networks.*

Keywords: *Elasticsearch, machine learning, microservices, object detection*

1. UVOD

Sa rastom računarskih sistema, količina podataka koje ti sistemi prikupljaju takođe raste. Jedan deo ovih podataka čine fotografije o čijoj semantici nije moglo mnogo da se zaključi na osnovu samih fotografija, osim podataka kao što su anotacije fotografija koje su ljudi manuelno anotirali. Pojava robusnih modela mašinskog učenja i dovoljne količine podatka za njihovo obučavanje omogućava automatizovanu ekstrakciju informacija iz fotografija. U kontekstu DIKW piramide [1], fotografije predstavljaju sirove podatke, čijom transformacijom u korisne informacije je moguće doći do znanja. Sistem opisan u ovom radu ima zadatak da fotografije prikupljanje sa uličnih kamera širom sveta analizira uz pomoć modela mašinskog učenja, koji služi za detekciju objekata, analizira i time ove podatke pretvori u korisne informacije. Ove informacije je zatim moguće staviti u geo-prostorni kontekst koji daje uvid u dešavanja na ulicama, kao što je kretanje ljudi i vozila.

2. DETEKCIJA OBJEKATA

Detekcija objekata je oblast računarske vizije koja se bavi identifikovanjem i lociranjem objekata određenih klasa na slikama. Probleme koje rešava detekcija je moguće predstaviti pomoću četiri kategorije [2]:

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dragan Ivanović, vanr. prof.

Semantička segmentacija: Ovo je oblast koja se bavi problemom klasifikacije svakog pojedinačnog piksela u određenu kategoriju.

Klasifikacija i lokalizacija: Proces koji se, pored uspešne klasifikacije slike u određenu kategoriju, sastoji iz određivanja graničnog okvira oko klasifikovanog objekta. Pretpostavka je da se na slici nalazi fiksan broj objekata (uglavnom tačno jedan).

Detekcija objekata (u užem smislu): Ovo je generalniji slučaj klasifikacije i lokalizacije u kojem broj objekata nije fiksan, već se na slikama može nalaziti više objekata različitih kategorija.

Segmentacija instance: Umesto određivanja graničnog okvira određuje se koji pikseli pripadaju detektovanom objektu. Razlikuje se od semantičke segmentacije po tome što bi u slučaju da na slici postoje dve instance iste klase semantička segmentacija detektovala obe instance kao jednu instancu.

Klasifikacija i lokalizacija posmatra problem detekcije kao regresiju. Cilj je klasifikovati objekat na slici i detektovati njegov granični okvir. Pošto je izlaz ovog procesa fiksan broj parametra, tj. tačno četiri (klasa detektovanog objekta, x i y koordinate gornjeg levog ugla graničnog okvira , širina i visina graničnog okvira) moguće je model obučavati na principu regresije gde se računaju dve greške: *Softmax* za klasifikaciju i L2 udaljenost za granični okvir.

Kada je u pitanju detekcija proizvoljnog broja fotografija, javlja se problem predlaganja regija fotografije koje su verovatne da sadrže objekte. Na osnovu načina na koji se ove regije predlažu postoji nekoliko načina za rešavanje ovog problema.

R-CNN model, koji je opisan u radu [3], za predlaganje regija koristi eksterni algoritam pod nazivom Selektivna pretraga. Selektivna pretraga je algoritam koji kao osnovu koristi segmentaciju slike baziranu na grafova čije segmente zatim iterativno spaja po sličnosti (boja, teksture, veličine i oblik). U svakoj iteraciji manji segmenti se grupišu u veće segmente koji se dodaju u listu predloženih regija. Kao rezultat ovog hijerarhijskog procesa dobija se oko 2000 predloženih regija.

***Fast* R-CNN**, koji je opisan u radu [4], takođe koristi selektivnu pretragu. Jedina razlika je što se ne propusta svaka regija kroz CNN, već se regije direktno projektuju na mapu obeležja koja je rezultat konvolucije cele slike. Ovo znatno ubrzava algoritam.

***Faster* R-CNN**, koji je opisan u radu [5], u potpunosti zamenjuje selektivnu pretragu sa Mrežom sa predlaganje regija (eng. *Region proposal network*) koju je moguće obučavati zajedno sa ostatkom modela. Na ovaj način je deo algoritma za predlaganje regija znatno ubrzan u odnosu na ostale modele.

3. KORIŠĆENE TEHNOLOGIJE

Spring je radni okvir koji predstavlja rešenje za brz i jednostavan razvoj poslovnih aplikacija. Zamišljen je kao lagan i neinvazivan okvir koji teži tome da se aplikacija što manje prilagođava *Spring* API-u. Sadrži veb modul koji pruža podršku za razvijanje veb aplikacija po MVC modelu. *Spring Boot* je radni okvir koji olakšava konfiguraciju *Spring* aplikacija.

RabbitMQ je softver otvorenog koda koji ima ulogu brokera poruka. Originalno, *RabbitMQ* je implementirao *Advanced Message Queuing Protocol* (AMQP), ali vremenom je proširen sa još protokola (STOMP, MQTT). AMQP je protokol koji omogućava komunikaciju između klijentskih aplikacija i brokera poruka koji implementiraju protokol. Brokeri poruka primaju poruke od izdavača poruka (eng. *publishers*) i rutiraju ih do potrošača (eng. *consumers*) odnosno aplikacija koje procesiraju poruke.

Elasticsearch je softver otvorenog koda za pretraživanje i analizu koji se zasniva na *Lucene* biblioteci i razvijen u Java programskom jeziku. Omogućava pristup informacijama u realnom vremenu preko REST API-a u JSON formatu. Takođe poseduje mnoštvo klijentskih biblioteka u svim trenutno popularnim programskim jezicima.

Kibana je platforma za analizu i vizualizaciju koja je dizajnirana za rad sa Elasticsearch-om. Pomoću Kibane je moguće pretraživati, pregledati i interagovati sa *Elasticsearch* indeksima. Kibana pruža korisniku mogućnost da lako izvršava napredne analize i vizualizuje podatke. Poseduje funkcionalnosti kao što su histogrami, linijski grafikoni, pita grafikoni, toplotne mape, kao i ugrađenu podršku za geo-prostornu vizualizaciju.

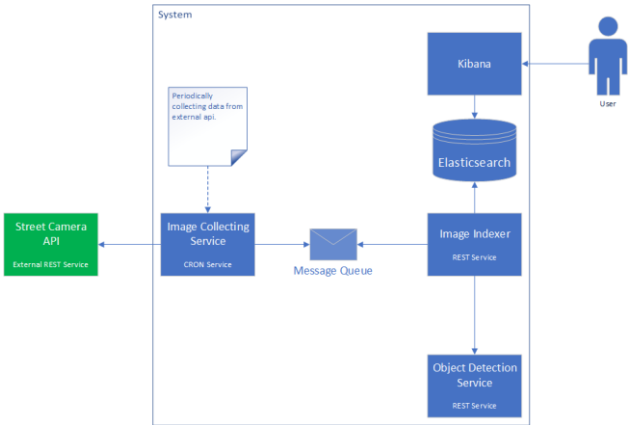
Flask je radni okvir za veb napisan u *Python* programskom jeziku. Klasifikuje se kao mikro radni okvir, jer ne zahteva nikakve dodatne alate i biblioteke. Ne sadrži sloj za apstrakciju baze podataka, validaciju formi i ostale funkcionalnosti za koje već postoje biblioteke koje ih pružaju. Zasniva se na dva projekta, a to su *Werkzeug* i *Jinja*.

Docker je je skup „platforma kao servis“ (eng. *platform as a service*) (PaaS) alata koji koriste virtualizaciju na nivou operativnog sistema da obezbede korisnicima softver koji je upakovan u pakete pod nazivom kontejneri. Kontejneri su izolovani jedni od drugih i sadrže sve potrebne resurse kao što su biblioteke i konfiguracioni fajlovi potrebni za uspešno izvršavanje željenog softvera unutar kontejnera. Na ovaj način programeri mogu da se pouzdaju u to da će njihove aplikacije raditi isto na svim mašinama koje podržavaju izvršavanje *Docker* kontejnera.

Azure je Majkrosoftova platforma za računarstvo u oblaku (eng. *cloud computing*) koja omogućava pravljenje, upravljanje i raspoređivanje aplikacija korišćenjem raznovrsnih alata, programskih jezika i radnih okvira.

4. ARHITEKTURA SISTEMA

Sistem opisan u ovom radu se sastoji iz više mikroservisa gde je svaki servis zadužen za izvršavanje određenog zadatka. Arhitektura sistema je predstavljena na drugom nivou C4 modela za vizualizaciju [6] i prikazana na slici 1.



Slika 1 Arhitektura sistema prikazana na drugom nivou C4 modela

Servis za detekciju objekata je veb aplikacija realizovana kao REST API koja omogućava detekciju objekata na fotografijama dobijenih od strane eksternog servisa.

Servis za prikupljanje fotografija je veb aplikacija koja ima zadatak da periodično šalje zahteve eksternom servisu i na taj način prikuplja fotografije sa uličnih kamera. Nakon toga fotografije preko brokera poruka šalje u red sa kojeg dalje servis za indeksiranje pristupa fotografijama.

Servis za indeksiranje fotografija je zadužen da preuzima poruke iz reda čije podatke dalje šalje na obradu servisu za detekciju objekata. Nakon što servis dobije odgovor od servisa za detekciju, podatke indeksira u Elasticsearch pretraživač.

5. IMPLEMENTACIJA SISTEMA

Servis za detekciju objekata je implementiran kao veb aplikacija po REST principima u *Flask* radnom okviru koristeći *Python* programski jezik. API sadrži *endpoint* za detekciju koji prima *POST* HTTP zahteve. Telo *POST* zahteva je JSON koji se sastoji iz jednog polja pod nazivom *data*. Ovo polje sadrži fotografiju u *base64* formatu radi jednostavnosti samog API-a. Nakon što funkcija koja je mapirana na rutu za detekciju primi zahtev, njegov sadržaj zatim isparsira i konvertuje u rečnik objekat. Iz ovog rečnika je potrebno iščitati polje *data* koje sadrži fotografiju u *base64* formatu. Fotografija se dalje šalje komponenti za detekciju koja je implementirana pomoću *GluonCV* skupa alata za duboko učenje u kompjuterskoj viziji. *GluonCV* obezbeđuje *Faster* R-CNN model koji je obučen na *Comon objects in context* (COCO) [7] skupu podata. Ovaj skup podataka sadrži 80 klasa među kojima su ljudi, automobili, itd. Model se koristi za detekciju objekata na fotografijama koje prima servis. Odgovor servisa je JSON objekat koji sadrži polja koja odgovaraju svakoj klasi COCO skupa

podataka i čije vrednosti predstavljaju broj instanci objekta odgovarajuće klase.

Servis za prikupljanje fotografija je implementiran kao veb servis pomoću *Spring Boot* radnog okvira. Realizovan je kao servis koji periodično automatski prikuplja fotografije sa eksternog REST API-a pod nazivom *Webcams.travel* [8]. *Webcams.travel* omogućava besplatan pristup kamerama širom sveta preko svog API-a. Trenutno sadrži više od 75 hiljada kamera čiji broj raste svakog dana. Za periodično preuzimanje fotografija sa API-a koristi se *Scheduled* anotacija kojom se specificira kada da se pokreće anotirana metoda. U ovom slučaju podešeno je da se metoda poziva svakih 30 minuta.

Da bi preuzimanje fotografija moglo da se distribuira na više računara potrebno je specificirati redni broj kamere od koje počinje preuzimanje i broj kamera. Na taj način moguće je pokrenuti dva ili više servisa od kojih će svaki preuzimati fotografije sa različitih kamera.

Nakon što se prikupe svi potrebni podaci, za svaku fotografiju se kreira poruka koja se šalje na *RabbitMQ* razmenu koja dalje rutira poruku na odgovarajući red. *RabbitMQ* je konfigurisan kroz *Spring* konfiguracioni fajl gde su definisani svi potrebni parametri.

Servis za indeksiranje fotografija je implementiran kao *Spring Boot* veb servis koji komunicira sa *RabbitMQ* brokerom sa čijeg reda preuzima poruke, šalje ih servisu za detekciju objekata i rezultate indeksira u *Elasticsearch*. Sastoji se iz komponente za primanje poruka, komponente za detekciju i komponente za indeksiranje.

Komponenta za primanje poruka je implementirana kao *Spring Boot* komponenta koja poseduje metodu anotiranu sa *RabbitListener* anotacijom. Ova anotacija govori *Spring*-u sa kog reda da čitaju poruke. Komponenta za detekciju komunicira sa servisom za detekciju pomoću *Spring RestTemplate* klase, i omogućava slanje fotografije na analizu. Komponenta za indeksiranje indeksira podatke u *Elasticsearch*. Za komunikaciju sa *Elasticsearch*-om se koristi *Spring Data* repozitorijum interfejs za *Elasticsearch* pod nazivom *ElasticsearchRepository*.

Ovaj interfejs zahteva da se specificira klasa koja predstavlja jedinicu indeksiranja. Da bi aplikacija znala da komunicira sa *Elasticsearch* instancom potrebno je u *Spring* konfiguracionom fajlu obezbediti potrebne vrednosti kao što su *host* i *port* na kom se nalazi *Elasticsearch*, kao i naziv klastera.

Sistem je konfigurisan tako da ga je moguće pokrenuti preko *Docker*-a. Svaki servis poseduje sopstveni *Dockerfile* koji definiše korake za pokretanje servisa. Da bi se svi servisi pokrenuli odjednom koristi se *Docker Compose*. U ovom fajlu su specificirani svi ostali servisi koji su potrebni da bi sistem radio kako treba, kao što su *Elasticsearch*, *RabbitMQ* i *Kibana*. Ovo znatno olakšava ceo proces pokretanja sistema pošto sve sto je potrebno da bude instalirano na računaru je *Docker*.

U ovoj konfiguraciji su svakom servisu prosleđeni svi potrebni parametri kao sto u promenljive okruženja, portovi koje koriste, kao i URL na koji skripta koja čeka da se servis pokrene može da proveri status servisa koji čeka. U slučaju servisa za detekciju kreiran je poseban *endpoint* koji vraća HTTP status 200 kada servis počne da radi. Takođe kreiran je volumen koji skladišti *Elasticsearch* indekse tako da u slučaju ponovnog pokretanja sistema prethodno prikupljeni podaci ostaju sačuvani.

Jedan od servisa koji pruža *Azure* je pokretanje aplikacija u *Docker* kontejnerima. Ova funkcionalnost omogućava da se prethodno kreirani *Docker Compose* fajl iskoristi da se ceo sistem pokrene na *Azure* platformi. Azure pruža dva načina da se koristi platforma. Prvi način je kroz njihov *online* portal koji daje grafički interfejs za upravljanje svim resursima na platformi. Drugi način je kroz *Azure* CLI alat za komandnu liniju. Ovo omogućava kreiranje skripte koja automatizuje ceo proces pokretanja sistema na Azure platformi.

6. REZULTATI I DISKUSIJA

U ovom poglavlju su prikazani rezultati sistema nakon dva dana rada, nedelja 24.11.2019. i ponedeljak 25.11.2019. radi poređenja stanja tokom radnih i neradnih dana. Sistem je pokrenut na računaru sa 16 GB rama, *Intel Core i5-4670K CPU @ 3.40GHz* i *NVIDIA GeForce GTX 1060 6GB.*

Prikupljeni su podaci sa 1000 najpoznatijih kamera na *Webcams.travel* servisu. Fotografije su preuzimane svakih pola sata i analizirane. Ukupno je prikupljeno 91617 indeksiranih fotografija.

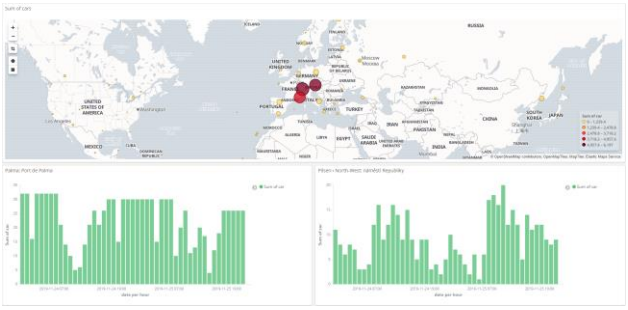
U Kibani su kreirane dve komandne table, jedna na kojoj su prikazani podaci vezani za detektovane ljude na fotografijama i druga na kojoj su prikazani podaci o broju automobila.

Kontrolna tabla sa podacima o broju ljudi se sastoji od tri vizualizuje. Na jednoj je prikazana mapa koordinata gde se vidi suma ljudi na svakoj koordinati. Druge dve vizualizacije su histogram broja ljudi na dve lokacije. Prva lokacija je trg u Novom Sadu u Srbiji, a druga lokacija je skijalište u Švajcarskoj. Kontrolna tabla se može videti na slici 2.

Slika 2 Kibana kontrolna tabla sa podacima o broju ljudi

Ovde se može uočiti šablon kretanja ljudi na datim lokacijama. U Novom Sadu se vidi da najviše ljudi ima u posle podnevnim časovima oko 16 časova, što je moguće primetiti na fotografiji prikazanoj na slici. Kada je u pitanju ponedeljak broj ljudi se povećava tek u kasnim posle podnevnim časovima oko 19 časova, što se može objasniti radnim vremenom većine ljudi koji je od 9 do 17 časova. Skijalište u Švajcarskoj je aktivnije nedeljom.

Na drugoj kontrolnoj tabli gde su prikazani podaci o broju automobila mogu se videti tri vizualizacije. Prva je mapa koordinata sa sumom automobila, dok su druge dve histogrami sa sumom automobila sa dve lokacije: luka u gradu Palma, Majorka i grad Pilsen u Češkoj. Ova tabla je prikazana na slici 3.



Slika 3 Kibana kontrolna tabla sa podacima o broju automobila

Na ove dve lokacijama je moguće primetiti časove gde je broj detektovanih automobila manji, što može da se protumače kao periodi kada je manja gužva na putevima.

7. ZAKLJUČAK

U ovom radu predstavljena je arhitektura sistema za prikupljanje i analizu fotografija sa uličnih kamera upotrebom mašinskog učenja. Opisane su svi servisi koji čine ovaj sistem, što podrazumeva komponente od kojih su sačinjeni, kao i komunikaciju između pojedinačnih servisa.

Predstavljeni su osnovni koncepti detekcije objekata na kojima se zasniva ovaj sistem kao i poređenje različitih modela za detekciju objekata. Takođe su opisane tehnologije koju su korišćeni za implementaciju sistema i prikazani su detalji implementacije svakog servisa. Pored toga, predstavljena je konfiguracija koja omogućava jednostavno pokretanje čitavog sistema.

Na kraju su predstavljeni rezultati u obliku vizualizacija podataka prikupljenih od strane sistema. Ove vizualizacije daju uvid u način kretanja ljudi i vozila koji omogućavaju donošenje zaključaka koji mogu biti od značaja.

Može se primetiti da servis za detekciju na većini primera gde postoji veliki broj instanci objekata nije precizan, pogotovo ako su ti objekti udaljeni zato što su same fotografije veoma male (400 x 224 px). Ovo je moguće lako unaprediti korišćenjem premijum pretplate na *Webcams.travel* API koja obezbeđuje fotografije visoke rezolucije. Servis za prikupljanje je moguće optimizovati tako što će se iskoristiti informacija o intervalu osvežavanja kamera i na taj način uz pomoć redova prioriteta optimalnije prikupljati fotografije.

Proces donošenja zaključaka je takođe moguće automatizovati kroz različite modele mašinskog učenja. Na osnovu podatka može se predvideti broj ljudi na određenim lokacijama, broj automobila na važnim raskrsnicama i na osnovu toga bolje planirati javni prevoz, odnosni procenu kada stiže naredni autobus, taksi, itd. Takođe moguće je iskoristiti ove podatke pri organizaciji raznoraznih promocija gde je veći broj ljudi poželjan. Modeli koji su dobri za rad sa istorijskim podacima su rekurentne neuronske mreže (RNN) koje mogu služiti za predviđanje budućih podataka.

8. LITERATURA

[1] J. Rowley, „The wisdom hierarchy: representations of the DIKW hierarchy,“ *Journal of Information Science*, 2007.

[2] „Object Detection: An End to End Theoretical Perspective,“ [Na mreži]. Available: https://towardsdatascience.com/object-detection-using-deep-learning-approaches-an-end-to-end-theoretical-perspective-4ca27eee8a9a. [Poslednji pristup Novembar 2019].

[3] R. Girshick, „Rich feature hierarchies for accurate object detection and semantic segmentation,“ 2014.

[4] R. Girshick, „Fast R-CNN,“ 2015.

[5] S. Ren, „Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,“ 2016.

[6] „The C4 model for visualising software architecture,“ [Na mreži]. Available: https://c4model.com/. [Poslednji pristup Novembar 2019].

[7] „Common objects in context,“ [Na mreži]. Available: http://cocodataset.org/. [Poslednji pristup Novembar 2019].

[8] „Webcams.travel,“ [Na mreži]. Available: https://www.webcams.travel/. [Poslednji pristup Novembar 2019].

Kratka biografija:

Aleksandar Nikolić je rođen 14.03.1995. godine u Novom Sadu. Osnovnu školu „Jožef Atila“ u Novom Sadu završio je 2010. godine kao učenik generacije. Gimnaziju „Jovan Jovanović Zmaj“ u Novom Sadu završio je 2014. godine. Iste godine upisao se na Fakultet tehničkih nauka, smer Softversko inženjerstvo i informacione tehnologije i diplomira 2018. godine sa prosekom 9,85. Iste godine upisuje master studije, smer Softversko inženjerstvo i informacione tehnologije, modul Elektronsko poslovanje. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Softversko inženjerstvo i informacione tehnologije odbranio je 2019. godine.