Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/15BE41Bucan

MAŠINSKO UČENJE NA IVICI UPOTREBOM NVIDIA JETSON TX2 UREĐAJA

MACHINE LEARNING ON THE EDGE USING NVIDIA JETSON TX2 DEVICE

Dušan Bućan, Fakultet tehničkih nauka, Novi Sad

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu opisana je primena konvolucionih neuronskih mreža u oblasti računarske vizije, kao i tehnike obrade toka podataka i arhitektura sistema za demografsku analitiku u realnom vremenu upotrebom mašinskog učenja na ivici. Opisan sistem je implementiran i izloženi su mogući pravci unapređenja i proširenja sistema.*

Ključne reči: *mašinsko učenje na ivici, računarska vizija, obrada toka podataka, demografska analitika u realnom vremenu*

Abstract – *In this work we present use of convolution neural networks in computer vision, stream processing techniques and an architecture of system for real-time demography analytics using machine learning on the edge. Implementation of described system and future improve-ments of system.*

Keywords: *machine learning on the edge, computer vision, stream processing, real-time demography analytics*

1. UVOD

Svakodnevno se kreira velika količina video materijala sa kamera postavljenih na javnim i privatnim mestima u gradu. Obrada ovih podataka bi mogla da unapredi kvalitet života i bezbednost građana, što predstavlja i jedan od motiva za razvoj sistema koji koriste mašinsko učenje na ivici. Mašinsko učenje na ivici (engl. *Machine learning on the edge*) [1] predstavlja novu oblast primene mašinskog učenja koja ima za cilj široku primenu modela mašinskog učenja na ugrađenim sistemima (engl. *embedded system*) [2]. Uređaji na ivici obrađuju ulazne podatke upotrebom modela mašinskog učenja i šalju procesirane podatke na centralni server, čime se prenosi manje podataka uz veću bezbednost.

Prenos manjeg obima procesiranih podataka čini sisteme sa mašinskim učenjem na ivici pogodnim za obradu velike količine podataka u realnom vremenu, zbog čega ovi sistemi često imaju arhitekturu zasnovanu na tokovima podataka. Kod arhitektura zasnovane na tokovima podataka podaci sa ugrađenih uređaja se upisuju u tok podataka u realnom vremenu nad kojim se zatim radi analiza i prikaz rezultata obrade u realnom vremenu.

U ovom radu izložena je ideja demografske analitike u realnom vremenu na osnovu video snimka upotrebom mašinskog učenja na ivici. Sistem se sastoji iz:

- Podsistema za obradu izvornih podakata na ivici
- Podsistema kontrolne jedinice
- *MLOps* [3] podsistema.

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

Podsistem za obradu izvornih podataka na ivici je sistem implementiran na *Nvidia Jetson TX2* uređaju [4], u nastavku *Jetson*. Ovaj podsistem na ulazu prima video snimak nad kojim obavlja procesiranje, odnosno radi demografsku analizu ljudi.

Podsistem kontrolne jedinice ima za cilj obradu tokova podataka koji sadrže informacije o demografskim odlikama ljudi u realnom vremenu, prikaz rezultata obrade tokova podataka kao i upravljanje ugrađenim sistemima na ivici.

Cilj postojanja *MLOps* podsistema i upotrebe *MLOps* praksi je brži razvoj sistema koji koriste modele mašinskog učenja, brža adaptacija na promenu zahteva, jednostavnije beleženje eksperimenata i poređenje modela mašinskog učenje, kao i uvođenje istih u produkciju.

2. PRETHODNA REŠENJA

U ovom poglavlju opisan je jedan od razmatranih sistema koji ima sličnu arhitekturu kao sistem mašinskog učenja na ivici za demografsku analitiku stanovništva. Sistemi za obradu velikih skupova podataka na ivici zasnovanom na klasteru ugrađenih sistema.

Rad [5] predstavlja ideju obrade velikih skupova podataka na ivici u približno realnom vremenu. Predloženo rešenje implementira sistem koji se sastoji od klastera *Raspberry Pi* [5] uređaja. Prednost predloženog rešenja u odnosu na tradicionalne tehnike obrade velikih skupova podataka je to što se podaci obrađuju na *Raspberry Pi* uređajima bliže mestu prikupljanja čime se obrada ubrzava. Obrada podataka u sistemu izloženom u radu [5] je realizovana upotrebom *Spark* [5] alata i *HDFS* [5] tehnologije. U poređenju sa radom [5] naš sistem koristi *Kafka* klaster [6] umesto *HDFS* što ga čini pogodnijim za obradu velikog skupa podataka u realnom vremenu kad je *Spark* alat zamenjen *Flink* [7] alatom za obradu toka podataka. Sistem izložen u radu [5] koristi *Docker* [5] i *Docker Swarm* [5] kako bi sistem učinio otpornijim na otkaze. Upotreba navedenih tehnologija uparena sa *Prometheus* i *Grafana* alatima [5] doprinosi lakšem nadgledanju i upravljanju celokupnim sistemom što bi moglo predstavljati pravac daljeg unapređenja našeg sistema. Naš sistem trenutno koristi *Docker* samo kao izvršno okruženje alata za procesiranje toka podataka i prikaz rezultata analitike toka podataka ali ne i za praćenje stanja ugrađenih uređaja.

3. SPECIFIKACIJA I IMPLEMENTACIJA REŠENJA

Ovo poglavlje posvećeno je korišćenim alatima (poglavlje 3.1), specifikaciji rešenja i arhitekture sistema (poglavlje 3.2), i skupovima podataka korišćenim za treniranje i testiranje modela (poglavlje 3.3).

3.1 Korišćeni alati

Pri implementaciji podsistema za obradu izvornih podataka na ivici korišćen je *Python* programski jezik, *Python OpenCV* biblioteka [8] za obradu slike kao i *Python* biblioteke za interakciju sa *Kafka message broker*-om [6], *MinIO* [9] skladištem podataka i *TensorRT* [10] okruženjem za izvršavanje modela. Modeli za detekciju ljudi i lica, klasifikaciju pola kreirani su, trenirani i testirani upotrebom *Tensorflow* biblioteke [11].

Pri implementaciji podsistema kontrolne jedinice za potrebe obrade toka podataka korišćen je alat *Flink* i programski jezik *Java.* Centralnu komponentu podsistema čini *Kafka message broker* koja pruža podršku za upis, čuvanje i čitanje poruka. *Za* smeštanje rezultata analize toka podataka korišćeni su *Kafka connect* [6] softverski alat i *PostgreSQL* [12] baza podataka. Vizualizacija uskladištenih rezultata obrade toka podataka implementirana je upotrebom *Superset* alata [13] i *SQL*-a. *MLOps* podsistem implementiran je upotrebom *MLFlow* alata [14], *PostgreSQL* relacione baze podataka, *MinIO* objektnog skladišta podataka i *Tensorflow Docker* kontejnera za treniranje modela. Infrastruktura *MLOps* podsistema i infrastruktura kontrolne jedinice su implementirane upotrebom *Docker* tehnologije, tačnije alati su korišćeni kao *Docker* kontejneri.

3.2 Arhitektura rešenja

Rešenje se sastoji iz tri odvojena podsistema – podsistema za obradu izvornih podataka na ivici (poglavlje 0), podsistema kontrolne jedinice (poglavlje 0) i *MLOps* podsistema (poglavlje 0). Arhitektura celokupnog sistema predstavljena je grafički na Slika 1.

3.2.1 Podsistem za obradu izvornih podataka na ivici

Komponente koje čine podsistem za obradu izvornih podataka su:

- Komponente za učitavanje videa
- Komponente za čuvanje videa
- Komponente za slanje poruka
- Komponente za detekciju ljudi
- Komponente za detekciju lica
- Komponente za klasifikaciju pola
- *Pipeline* komponenta

Komponente za učitavanje videa pružaju funkcionalnost učitavanja videa frejm po frejm sa različitih izvora podataka. Trenutna implementacija podržava učitavanje videa sa:

- *Web* kamere
- Ugrađene kamere na *Jetson* uređaju
- Lokalnog fajl sistema

Komponente za čuvanje videa pružaju funkcionalnost čuvanja videa na različitim tipovima skladišta. Trenutna implementacija podržava čuvanje videa na:

- Lokalnom fajl sistemu
- *MinIO* objektnom skladištu podataka

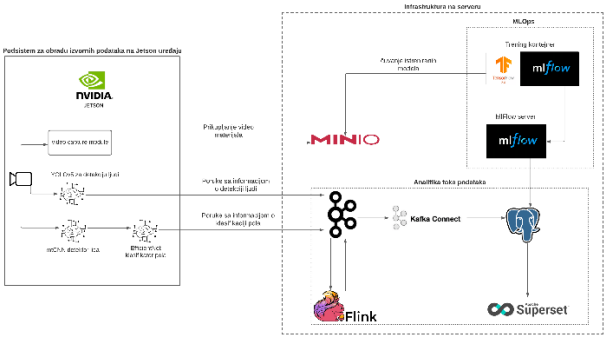
Komponente za slanje poruka pružaju funkcionalnosti kreiranja poruka na osnovu ulaznih, obrađenih podataka kao i slanja istih na *Kafka message broker.*

Komponente za detekciju ljudi nad ulaznim frejmovima detektuju ljude, tj. njihove granične okvire. Postoje dve implementacije komponente za detekciju ljudi i obe u osnovi koriste pretreniran *YOLOv5m* detektor [15]. Jedna komponenta za detekciju ljudi je prilagođena efikasnijem izvršavnju na *Jetson* uređaju, dok se druga koristi u svrhe testiranja na lokalnom računaru. Komponenta za detekciju ljudi na *Jetson*-u je konvertovana u *TensorRT* format, dok je komponenta za upotrebu na lokalnom računaru kovertovana u *Tensorflow* format iz *PyTorch* [16] formata, koji je izvorni format detektora.

Komponente za detekciju lica nad ulaznim frejmovima detektuju lica, tj. granične okvire lica. Postoje dve implementacije komponente za detekciju lica i obe u osnovi koriste pretreniran *mtCNN* detektor [17]. Jedna komponenta je prilagođena efikasnijem izvršavanju na *Jetson* uređaju, dok se druga koristi u svrhe testiranja na lokalnom računaru. Komponenta za detekciju lica na *Jetson* uređaju konvertovana je u *TensorRT* format, dok je komponenta za upotrebu na lokalnom računaru konvertovana u *Tensorflow* format iz *PyTorch* formata.

Komponente za klasifikaciju pola na ulazu primaju izdvojene regione od interesa koje je komponenta za detekciju lica prepoznala kao lice a na izlazu kreiraju skup predikcija pola za svaki od ulaza. Postoje dve implementacije komponente za detekciju lica i obe u osnovi koriste ručno kreiranu neuronsku mrežu. Ručno kreiranu mrežu čine *EfficientNet B7* [18] konvolutivna neuronska mreža bez potpuno povezanih slojeva sa dodatim potpuno povezanim slojem od 128 neurona i izlaznim slojem koji sadrži jedan neuron.

Pipeline komponenta se sastoji od drugih komponenti i ima za cilj koordinaciju drugih komponenti.



Slika 1. Arhitektura sistema

3.2.2 Podsistem kontrolne jedinice

Podsistem kontrolne jedinice se sastoji iz više komponenti:

- Kontrolne table na kojoj je moguće pratiti demografsku analitiku u realnom vremenu
- Modula za analitiku u realnom vremenu (engl. *real-time analytics*)
- Modula za kontrolu ugrađenih sistema na ivici

Kontrolna tabla je implementirana uz pomoć *Apache Superset* alata, odnosno koristi se *Apache Superset Docker* kontejner. Kontrolna tabla vizualizuje uskladištene podatke iz *PostgreSQL* baze podataka. Podaci su prikazani kao dva grafika od kojih jedan prikazuje prosečan broj detektovanih ljudi u minutu, a drugi prosečan broj detektovanih muškaraca i žena u minutu.

Modul za analitiku u realnom vremenu predstavlja aplikaciju za procesiranje toka podataka u realnom vremenu. *Apacke Flink* alat je korišćen u implementaciji modula zbog jednostavnosti upotrebe, integracije sa *Docker* alatom, lakoće skaliranja i ugrađenih strategija oporavka od

greške. *Apache Flink* predstavlja distribuirano izvršno okruženje za aplikacije obrade toka podataka. Osnovnu arhitekturu *Apache Flink* klastera čine dva čvora, jedan *JobManager* [7] i jedan *TaskManager* [7].
Uloga *JobManager*-a je organizacija i nadgledanje izvršavanja aplikacije za obradu toka podataka, dok je uloga *TaskManager*-a izvršavanje delova aplikacije. Pri implementaciji aplikacije za obradu toka podataka korišćena je *Flink API* biblioteka za *Java* programski jezik. U sistemu su implementirane dve aplikacije za obradu toka podataka:

- aplikacija za obradu toka podataka detekcije ljudi
- aplikacija za obradu toka podataka klasifikacije pola ljudi

Aplikacija za obardu toka podataka detekcije ljudi na ulazu prima poruke sa *peopleDetections Topic*-a iz *Kafka message broker*-a, prikazano na Slika 1. Aplikacija grupiše podatke koji pristižu u realnom vremenu u vremenske intervale od po minut. Nad svakim vremenskim intervalom određuje prosečan broj ljudi u tom vremenskom intervalu. Izlaz aplikacije za obradu toka podataka detekcije ljudi su poruke sa informacijma o prosečnom broju detektovanih ljudi kao i početak i kraj vremenskog intervala u kojem je prosečan broj ljudi računat. Izlazne poruke se upisuju u *peopleAvgCount Topic Kafka message broker*-u.
Aplikacija za obradu toka podataka klasifikacije pola ljudi radi po istom principu, jedina razlika je u čitanju podataka iz drugog toka podataka, *genderClassifaction Topic* iz *Kafka message broker*-a, kao i pisanje u drugi izlazni tok podataka, *avgGenderClassifaction*. Implementirane aplikacije za obradu tokova podataka se postavljaju na *JobManager* čvor *Flink* klastera i nakon čega počinje njihovo izvršavanje.
Upotrebom *Kafka Connect* alata poruke iz *Kafka Topic*-a, *peopleAvgCount* i *avgGenderClassifaction* se skladište u *PostgreSQL* bazi podataka, kao na Slika 1.
Modul za kontrolu ugrađenih sistema na ivici je implementiran u vidu *Python* skripte. U trenutnoj verziji sistema moguće je konfigurisati ugrađeni sistem samo pre početka njegove upotrebe, odnosno moguće je odabrati jednu od četiri opcije:

- Prikupljanje video materijala sa kamere
- Detekcija ljudi na videu
- Klasifikacija ljudi na videu
- Detekcija ljudi i klasifikacija ljudi na videu

3.2.3 *MLOps* podsistem

MLOps podsistem je implementiran upotrebom *MlFlow* platforme kao na Slika 1. *MLOps* podsistem čine:

- repozitorijum modela mašinskog učenja
- baza podataka sa informacijama o treniranju modela
- baza podataka sa prikupljenim video materijalom za treniranje modela
- *MlFlow* server koji predstavlja kontrolnu tablu za prikaz informacija o treniranim modelima

Repozitorijum modela mašinskog učenja i baza podataka sa video materijalima za treniranje modela mašinskog učenja su realizovane kao objektne baze upotrebom *MinIO* tehnologije. Baza podataka koja sadrži informacije o treniranju modela je realizovana kao relaciona baza podataka upotrebom *PostgresSQL* baze podataka. *MlFlow* platforma poseduje *Python API* biblioteku koju je potrebno koristiti pri interakciji sa *MLFlow* serverom. Treniranje modela mašinskog učenja se najčešće odvija unutar kontejnera za trening, obično *Tensorflow* ili *PyTorch* kontejner. Za vreme treninga potrebno je da trening kontejner šalje informacije o konfiguraciji treninga kao i informacije generisane tokom treninga upotrebom *MLFlow Python API* biblioteke. Implementirano je slanje *F1 score* [19] mere kao i slanje preciznosti, tačnosti nad validacionim skupom po epohama, vreme trajanja svake od epoha.

3.3 Skupovi podataka

Korišćen *YOLOv5m* detektor ljudi pretreniran je na *COCO* [20] skupu podataka od strane autora detektora. Detektor lica, *mtCNN*, je pretreniran na *WIDER FACE* [21] i *CelebA* [22] skupovima podataka od strane autora detektora. Za treniranje i testiranje modela za prepoznavanje pola ljudi korišćen je *UTKFace* [23] skup podataka, dok je osnova modela, *EfficientNet B7* bez potpuno povezanih slojeva, pretrenirana na *ImageNet* [24] skupu podataka. *COCO*, *WIDER FACE*, *CelebA*, *ImageNet* predstavljaju široko korišćene i poznate skupove podataka pa je njihov pregled izostavljen. U narednom poglavlju je detaljnije opisan *UTKFace* skup podataka.

3.3.1 Skup podataka za prepoznavanje pola ljudi

UTKFace skup podataka je javno dostupan i sadrži preko dvadeset tri hiljade šesto sedamdeset osam (23678) uzoraka sa anotacijama o polu, godištu i etničkoj pripadnosti. Uzorci unutar skupa podataka su raznovrsni, odnosno postoje slike lica različitog osvetljenja, rezolucije, ugla slikanja lica, ljudi različitih starnosnih grupa. Skup podataka pokriva starostni raspon ljudi od nula (0) godina do sto šesnaest (116). Skup podataka je namenjen za raznovrsnu upotrebu, neke od primena su prepoznavanje lica, prepoznavanje pola, procena godina ljudi. Od ukupnog broja uzoraka, 23678, za trening je korišćeno 17048 uzoraka koji su nasumično odabrani, dok je za validacioni skup nasumično odabrano 1894 uzorka, ostalih 4735 uzoraka čine test skup.

4. EVALUACIJA REŠENJA I REZULTATI

U ovom poglavlju biće reči samo o evaluciji podsistema za obradu izvornih podataka na ivici. Evaluacija podsistema kontrolne jedinice je izostavljena jer su pri implementaciji korišćeni najbolji alati za domenski problem. Poznavajući činjenicu da je maksimalan broj poruka koje je moguće upisati u *Kafka messagr broker* u jednoj sekundi dva miliona poruka, dok podsistem kontrolne jedinice upisuje svega dve poruke po sekundi može se zaključiiti da podsistem kontolne jedinice koristi alate daleko ispod maksimalne granice, što čini evaluaciju ovog podsistema suvišnom. Kao što je opisano u poglavlju 0 podsistem za obradu izvornih podataka na ivici poseduje više komponenti koje sadrže modele mašinskog učenja koji su evaluirani.

4.2. Evaluacija i rezultati *YOLOv5m* detektora ljudi

Za evaluaciju korišćenog *YOLOv5m* detektora odabrana je *mAP* [19] metrika. Vrednost za *mAP* metriku sa 0.5 graničnom vrednošću *IoU* nad *COCO* validacionim skupom

je izračunata od strane autora *YOLOv5m* detektora i iznosi 63.1%. Dodatna evaluacija *YOLOv5m* detektora je urađena nad ručno kreiranim skupom podataka koji se sastoji od slika prikupljenih upotrebom ranije pomenute komponente za čuvanje videa. Ručno kreirani skup podataka sadrži 21 sliku i ukupno 71 stvarnih graničnih regiona ljudi. Izračunata vrednost za *mAP* metriku sa 0.5 graničnom vrednošću *IoU* nad ručno kreiranim skupom podataka je 0.845%. Zbog visoke vrednosti *mAP* pri detekciji ljudi dodatni trening *YOLOv5m* detektor na ručno kreiranim skupom podataka je izostavljen, jer kreiranje trening skupa zahteva dosta resursa a ne bi značajno unapredilo performanse sistema.

4.3. Evaluacija i rezultati *mtCNN* detektora lica

Za evaluaciju korišćenog *mtCNN* detektora korišćena je *AP* metrika i funkcija preciznosti po odzivu, opisana ranije na *WIDER FACE* skupu podataka od strane autora detektora. Postignuti rezultati svrstavaju korišćeni *mtCNN* detektor među najbolje detektore lica testirane na *WIDER FACE* test skupu. *WIDER FACE* skup podataka poseduje veliki diverzitet, pa prikupljeni video materijal sa kamere *Jetson*-a na kojem su zabeležena lica ljudi čini njegov podskup. Iz ovoga je moguće zaključiti da dodatni trening *mtCNN* detektora nije potreban.

4.4 Evaluacija i rezultati testiranja klasifikatora pola

Model kreiran za klasifikaciju pola je evaluiran upotrebom *F1 score* metrike na test delu *UTKFace* skupa podataka. Test deo *UTKFace* skupa se sastoji od 4735 uzoraka nasumično odabranih pre treninga modela. Model postiže 0.855 *F1 score* pri predviđanju ženskog pola i 0.856 *F1 score* pri predviđanju muškog pola na kreiranom test skupu.

5. ZAKLJUČAK

U ovom radu predstavljen je sistem za demografsku analitiku u realnom vremenu na osnovu video snimka upotrebom mašinskog učenja na ivici. Jedna od mogućih primena je bolje određivanje ciljne grupe kupaca u prodavnicama. Ako bi se posmatrao broj i osobine ljudi koji prilaze određenom artiklu moguće bi bilo odrediti ciljnu grupu za taj artikal. U malim prodavnicama ovo je moguće realizovati bez upotrebe ovakvog sistema, ali u slučaju velikih prodavnica ili za vreme intenzivnih perioda kupovine sistem bi zasigurno nadmašio učinak prodavaca i u tom slučaju bi bio doprineo razvoju prodavnice.

Moguća unapređenja implementacije sistema opisanog u poglavlju 3.2 Arhitektura rešenjasu poboljšanje performansi podsistema za obradu izvornih podataka na ivici kao razvoj modula za inteligenciju mnoštva (engl. *Swarm intelligence*) [25]. Navedeni podsistem se oslanjaju na konvolutivne neuronske mreže pri određivanju pola osoba čije bi se performanse mogle poboljšati povećanjem obima skupa podataka za trening. Upotreba ansambla klasifikatora pola bi takođe doprinela porastu performansi sistema. Razvoj modula za inteligenciju mnoštva omogućio bi koordinisanu i istovremenu upotrebu više ugrađenih uređaja i samim tim unapredio i proširio primenu celog sistema.

6. LITERATURA

[1] Chang, Zheng, et al. "Learn to cache: Machine learning for network edge caching in the big data era." *IEEE Wireless Communications* 25.3 (2018): 28-35.
[2] https://en.wikipedia.org/wiki/Embedded_system (pristupljeno 15.08.2021.)
[3] https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning (pristupljeno 15.08.2021.)
[4] https://developer.nvidia.com/embedded/jetson-tx2 (pristupljeno 16.08.2021.)
[5] Scolati, Remo, et al. "A Containerized Big Data Streaming Architecture for Edge Cloud Computing on Clustered Single-board Devices." *Closer*. 2019.
[6] https://kafka.apache.org/documentation/ (pristupljeno 19.08.2021.)
[7] https://flink.apache.org/, poslednji pristup: 23.08.2021.
[8] https://docs.opencv.org/4.5.2/d6/d00/tu-torial_py_root.html (pristupljeno 24.08.2021.)
[9] https://docs.min.io/docs/minio-quickstart-guide.html (pristupljeno 23.08.2021.)
[10] https://docs.nvidia.com/deeplearning/tensorrt/develo-per-guide/index.html (pristupljeno 25.08.2021.)
[11] https://www.tensorflow.org (pristupljeno 25.08.2021.)
[12] https://www.postgresql.org/ (pristupljeno 23.08.2021.)
[13] https://superset.apache.org/ (pristupljeno 22.08.2021.)
[14] https://mlflow.org/docs/latest/index.html, (pristupljeno 23.08.2021.)
[15] https://github.com/ultralytics/yolov5 (pristupljeno 16.08.2021.
[16] https://pytorch.org/ (pristupljeno 24.08.2021.)
[17] Zhang, Kaipeng, et al. "Joint face detection and alignment using multitask cascaded convolutional networks." *IEEE Signal Processing Letters* 23.10 (2016): 1499-1503.
[18] Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." *International Conference on Machine Learning*. PMLR, 2019.
[19] https://towardsdatascience.com/on-object-detection-metrics-with-worked-example-216f173ed31e (pristupljeno 26.08.2021.)
[20] https://cocodataset.org/ (pristupljeno 26.08.2021.)
[21] http://shuoyang1213.me/WIDERFACE/ (pristupljeno 26.08.2021.)
[22] https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html (pristupljeno 26.08.2021.)
[23] https://susanqq.github.io/UTKFace/ (pristupljeno 26.08.2021.)
[24] https://www.image-net.org/ (pristupljeno 26.08.2021.)
[25] Chamoso, Pablo, et al. "Swarm agent-based architecture suitable for internet of things and smartcities." *Distributed Computing and Artificial Intelligence, 12th International Conference*. Springer, Cham, 2015.

Kratka biografija:

Dušan Bućan rođen je u Zrenjaninu 1997. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva odbranio je 2021.god.
kontakt: dusanbzr@gmail.com