**UNIVERZITET SINGIDUNUM**

**Tehnički Fakultet**

Upotreba metoda dubokog učenja u sintezi sistema za prepoznavanje lica

- diplomski rad -

**Mentor: Kandidat:**

**prof. dr  *Milan Milosavljević Davor Jordačević***

**Beograd, *2020*.**

Sadržaj

[Uvod - 3 -](#_Toc49689240)

[Detekcija lica - 5 -](#_Toc49689241)

[1.1 RetinaNet - 5 -](#_Toc49689242)

[1.2 RetinaFace - 6 -](#_Toc49689243)

[1.2.1 MobileNet v2 - 7 -](#_Toc49689244)

[Poravnanje lica - 8 -](#_Toc49689245)

[Ekstrakcija vektora obeležja - 10 -](#_Toc49689246)

[Pretraga vektora obeležja (prepoznavanje) - 12 -](#_Toc49689247)

[Implementacija u programskom jeziku Python - 15 -](#_Toc49689248)

[Zaključna razmatranja - 16 -](#_Toc49689249)

[Literatura - 17 -](#_Toc49689250)

# Uvod

Glavni problem koji razmatram u ovom radu jesu komponente koje čine jedan sistem za prepoznavanje lica. Još od ranih dana sa pojavom kamera, pojavila se potreba za sistemima koji bi mogli da identifikuju osobe na njima. Tokom vremena su se razne metode smenjivale, od prepoznavanja pokreta, otisaka prstiju, do prepoznavanja lica. U ovom radu će biti reči upravo o tehnologijama koje se koriste za prepoznavanje lica, konrektno, fokusiraćemo se na tehnike dubokog učenja u sintezi sistema za prepoznavanje lica. Ovakvi sistemi su složeni i predstavljaju sintezu raznih tehnologija i metoda kako bi uspešno radili. Sam razvoj ovih sistema je zahtevan, i zahteva tim ljudi koji su sposobni za rešavanje kako matematičkih, tako i računarskih problema. Ovi problemi proizilaze iz potrebe za visokom tacnošću sistema, kao i od varijacija usled koriscenja razne opreme i alata. Svakoga dana se sve vise softvera zasniva upravo na ovoj tehnologiji, a to podrazumeva njenu ispravnost, robusnost i tačnost. Jedna od tehnika koja je omogućila nagli razvoj ove oblasti jesu duboke neuronske mreže. Pored ovoga, ubrzan razvoj tehnologija i računarskih komponenti omogućili su znatno brži razvoj i treniranje ovakvih mreža, o čemu će biti reči u nastavku ovog rada.

Glavni cilj mog istraživanja u ovom radu jeste sinteza sistema za prepoznavanje lica koristeći već postojeće metode za detekciju, ekstrakciju vektora obeležja, i njihovo upoređivanje radi dobijanja željenih rezultata.

U svom radu koristiću analitičke metode kako bi svaku celinu razlozio na delove i bolje objasnio. Sam sistem za prepoznavanje lica je jedna složena celina koja uključuje delove koji su se godinama razvijali i istrazivali. Ovo znači da se kombinacijom ovih podsistema mogu dobiti novi sistemi sa specifičnom namenom ili performansama. Ovi delovi u produkucionom sistemu ne mogu da rade jedan bez drugog, dok je prilikom istraiživanja moguće preskočiti neke od njih. Uzmimo za primer detekciju lica. Ukoliko je cilj sistema samo prepoznavanje, a za testiranje, razvoj i upotrebu se koriste slike koji sadrže samo detektovana lica, onda se sam korak detekcije moze preskočiti. S obzirom na to da ovo često nije slučaj, fokusiraćemo se na sintezu kompletog sistema. Treba uzeti u obzir da ovakav sistem mogu činiti i dodatne komponente, poput dela za predikciju da li je osoba stvarna ili lažna (face anti-spoofing).

U cilju testiranja performansi koristićemo podatke koji nisu često korišćeni, ali su javno dostupni. Reč je o slikama koje na prvi pogled čoveku deluju tesko za prepoznavanje.

Analiza lica predstavlja jedan od bitnih procesa u našim životima. Ljudi analizom lica prikupljaju bitne podatke o drugim osobama. Ovo uključuje podatke o broju godina, polu, rasnoj pripadnosti. Takodje možemo prepoznati da li je osoba srećna ili tužna, ili pak neku drugu emociju. Pokreti usana su vazni u oblasti prepoznavanja govora, kao i sve popularnijoj oblasti kao sto je generisanje lažnih snimaka. Metode analize lica nam mogu reći gde je usmeren pogled neke osobe, odnosno šta privlači njenu pažnju, i ovo može biti posebno interesantno u marketingu i šopovima, kazinima. U medicini ove metode mogu biti od koristi za prepoznavanje nekih bolesti, poput autizma koji se odlikuje time što osobe imaju poteškoće da iskazu svoje emocije. Sve navedene metode koriste kako ljudi, tako i računari.

U ovom radu ćemo se fokusirati na metode koje se koriste u procesu detekcije lica, njegove ekstrakcije, obrade i zatim prepoznavanja.

Na početku ovog rada je prvo bitno da uvrdimo sta je zapravo prepoznavanje lica. Svaka osoba ima karakteristično lice, i to je ono što nas čini unikatnima. Većna metoda se bazira upravo na ovoj činjenici, i njihov cilj je ekstrakcija ovih obelezja (features) za svaku osobu, a zatim i njihova klasifikacija na osnovu određenih parametara.

Treba razlikovati algoritme za prepoznavanje po vise kriterijuma. Osnovna klasifikacija je na algoritme zasnovane na geometriji (Geometry based) i algoritme zasnovane na šablonima (template based). Geometrijski zasnovani algoritmi analiziraju određena područja i geometrijske veze na njima. Zbog ovoga su i poznati kao algoritmi zasnovani na obelezjima. Sa druge strane su algoritmi zasnovani na šablonima, i u ovu grupu spadaju: metoda nosećih vektora (SVM), analiza glavnih komponenti (PCA), linearna diskriminantna analiza (LDA), kernel metode i još mnogo drugih.

# Detekcija lica

Prvi korak u procesu prepoznavanja lica je njegova detekcija. Osnovna ideja procesa detekcije je pronalaženje svih lica na slici, odnosno njihovih koordinata. Ove koordinate služe za ekstrakciju lica sa slike, te se u nastavku radi samo sa slikama koje sadrže lice. Takođe se vrlo često koordinate čuvaju ukoliko je reč o radu sa video snimcima radi iscrtavanja. Pored ovog, čuvanjem originalne slike i koordinata (bounding box-a ili bbox-a) imamo mogucnost ponovne ekstrakcije lica sa raznim scale faktorima, što moze biti korisno kao ulaz u neke druge neuronske mreže. Primer ovoga su anti-spoofing mreže koje imaju za cilj predikciju da li je osoba na slici ili snimku prava (real) ili lažna (fake).

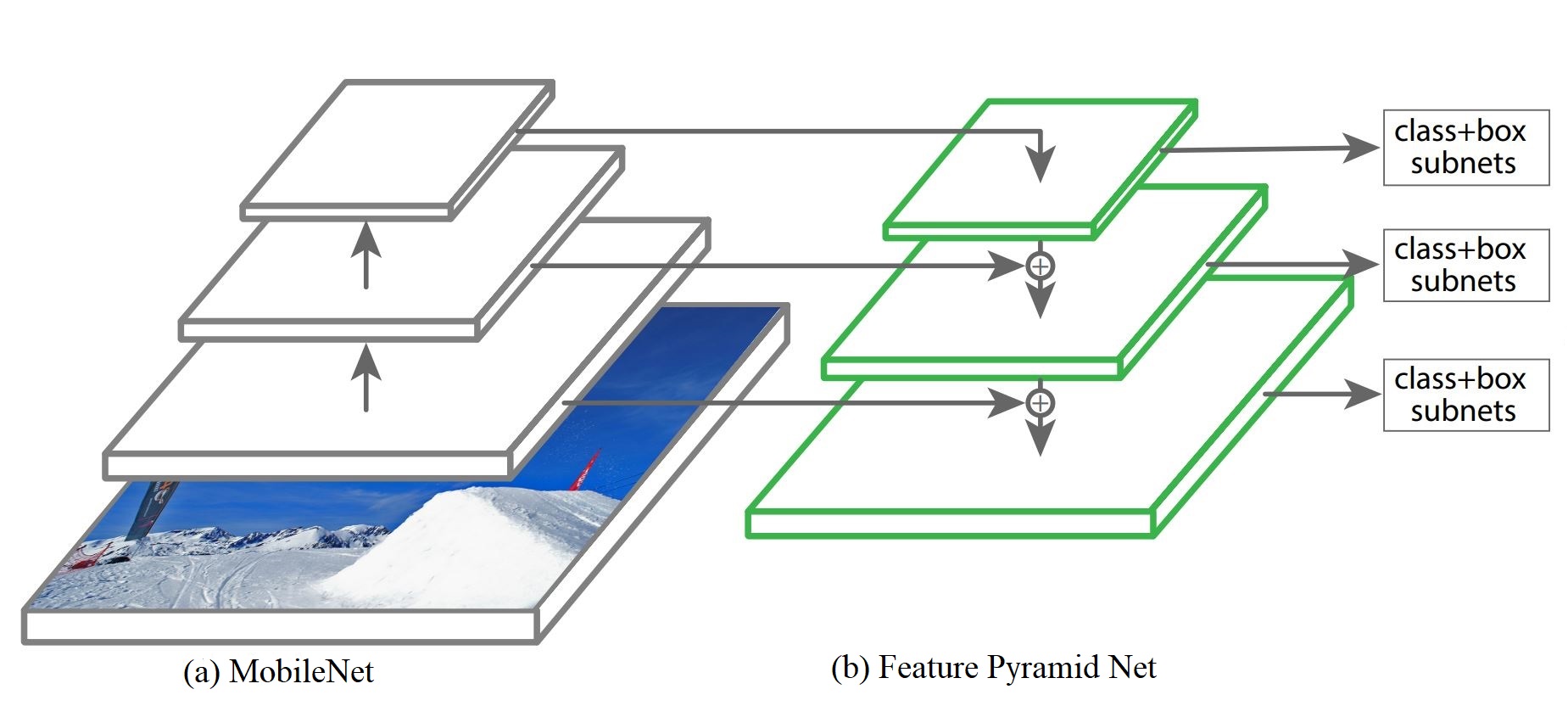
## RetinaNet

Postoji zaista veliki broj razvijenih ideja i projekata na temu detekcije, kako objekata, tako i lica. Jedna implementacija se izdvaja od drugih, reč je o RetinaNet mreži (arXiv:1708.02002v2).

Osnovna ideja sa RetinaNet detektorom je bila u uraditi sve predickije u jednoj fazi. U 2019. godini, state-of-the-art (SOTA) detektori su bili bazirani na mehanizmu sa dve faze (takozvani two-stage). Prva faza je podrazumevala generisanje skupa kandidata za lokaciju na kojoj je moguće naći traženi objekat, dok se druga faza sastoji iz klasifikacije svakog kandidata u jednu od klasa.

RetinaNet je jednofazni (single stage) detektor koji se sastoji iz jedne mreže (bacbone) i dve mreze sa specifičnom funkcijom. Kao backbone mreža se moze koristiti bilo koja od poznathih arhitektura (VGG, MobileNet, ResNet) i osnovni cilj ove mreže je računanje konvolucione feature mape. Nakon toga, prva podmreža radi klasifikaciju na osnovu izlaza backbone mreže, dok druga podmreža proračunava bounding box regresiju.

Pored toga što je ovaj detektor single stage, podrzava koncept piramida. RetinaNet je zasnovan na konceptu piramida obeležja (Feature Pyramid Network - FPN) (arXiv:1803.07737v2). Ovo je omogućeno korišćenjem FPN mreže kao backbone mreže. Osnovni princip rada je da FPN mreža radi augmentaciju standardne konvolucione mreže sa vrha ka dnu sa lateralnim konekcijama (bočne veze).  Ovo omogućava mreži da efikasno konstruise piramidu sa različitim scale faktorima iz jedne slike. Svaki nivo piramide se može koristiti za detektovanje objekata u različitoj razmeri.



Slika 1 –RetinaNet arhitektura koja koristi FPN backbone mrežu preko MobileNet arhitekture (arXiv:1708.02002v2)

## RetinaFace

Jedan od velikih problema koji se pojavljivao je bila detekcija lica različitih veličina u nekotrolisanim uslovima (prirodi, gužvama u gradu). Pošto već poznati koncept piramida može reštiti probleme detektovanja sa različitim scale faktorima, sledeća mreža koristi iste principe. RetinaFace (arXiv:1905.00641v2) je mreža nešto novijeg porekla i zasnovna je na istim principima na kojima se zasniva i RetinaNet, ali je namenjena isključivo detekciji lica.

RetinaFace predstvalja single stage detektor lica. Tvorci ovog modela su uneli nove ili unapredili već postojeće metode korišćene u ovu svrhu, kao sto su multi-task obucavanje za istovremenu predikciju uverenja, bounding box-a, 5 ključnih tačaka na licu, i 3D poziciju (u originalnoj implementaciji).

Ono što je novo je 5 ključnih tačaka (keypoint-a) koji ce se kasnije koristiti za poravnanje lica.

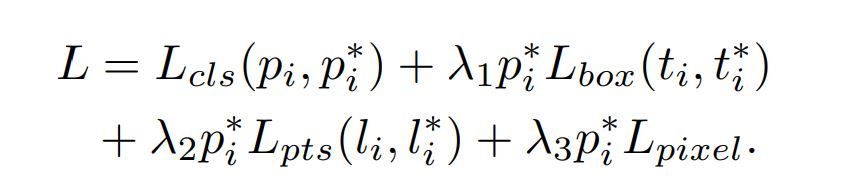
Kako bi se poboljšala detekcija lica, ili takozvanih Hard detekcija, korišćen je koncept modelovanja konteksta. Takozvana Hard lica su teska za detekciju zbog nedostatka vizualne konzistentosti, pozicije ili konteksta [arXiv:1803.07737v2]. Osnovna ideja je da mreža može da nauči ne samo obeležja koja su karakteristična za lice, već i kontekstualni deo kao što su vrat ili telo.

Kako bi se povećali efekti modelovanja nelinearnih (ne-rigidnih) transformacija (scaling, shearing) korišćeni su kontekst moduli. Geometrijske varijacije predstavljaju jedan od velikih problema u oblasti detekecije i prepoznavanja. Metoda koja se pokazala korisnom u prevazilazenju ovih problema je koriscenje deformabilne (eng. deformable)  konvolucije (arXiv:1703.06211v3).

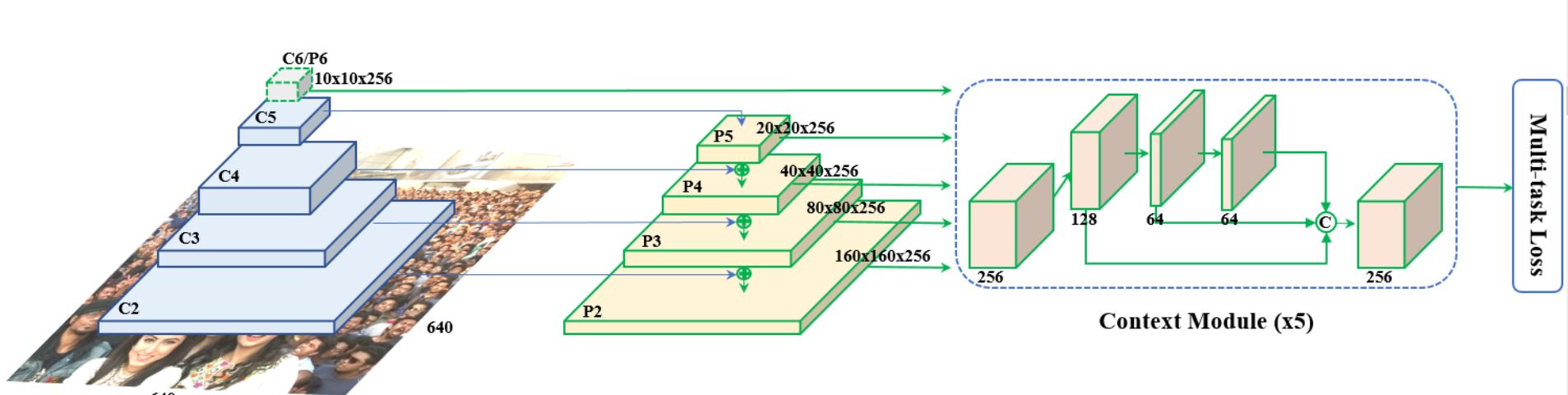
Matemaika o deformable conv vs obicna conv.

i DCN v2 [<https://arxiv.org/pdf/1811.11168.pdf>]

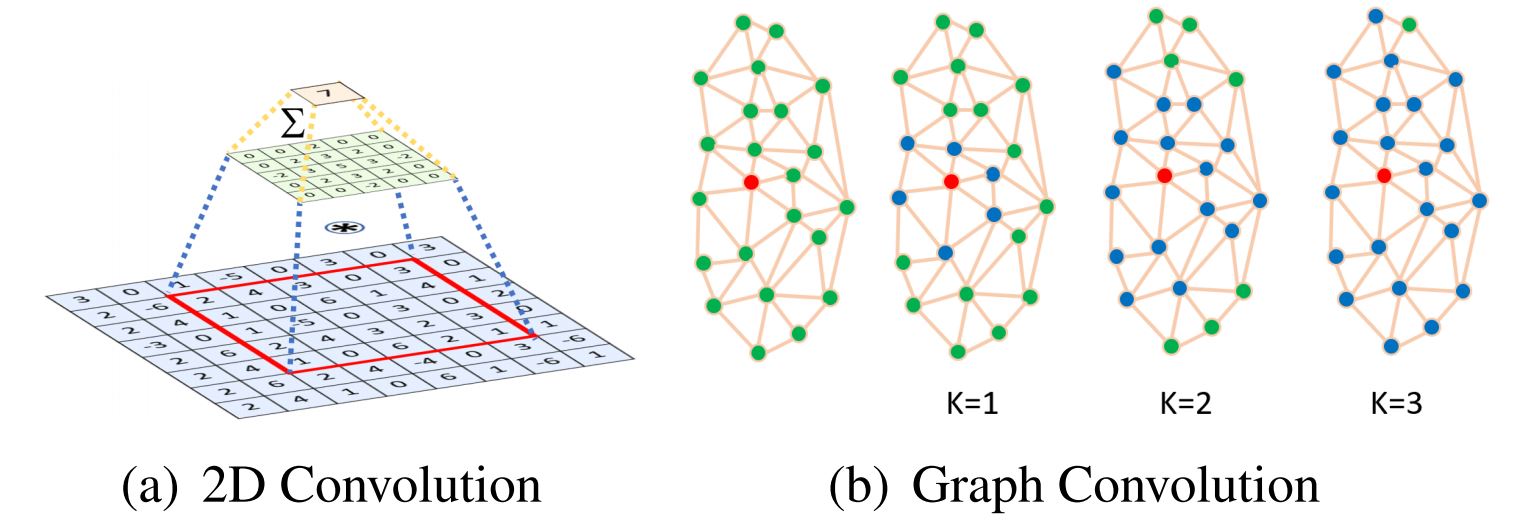
Kao što je već pomenuto, RetinaFace mreža je zasnovana na pricipu multi-tast obučavanja. Samim tim se nameće koriscenje drugacije funkcije gubitka (loss funkcije). Loss funkcija koriscena u ovom slucaja je multi-task loss:



Ova funkcija se sastoji iz više delova, gde prvi deo *Lcls* predstavlja softmax gubitak binarne klasifikacije (ima lica/nema lica), drugi sabirak je gubitak regresije bounding box-ova, zatim sledi regresioni gubitak za predikciju 5 ključnih tačaka i dense regresioni gibitak.



Kako bi ubrzao proces detekcije korišćen je takozvani mesh dekoder (mesh konvolucija i up-sampling). Ovo predstavlja vid graf konvolucionog metoda.



Dodati jos o implementaciji ovde…

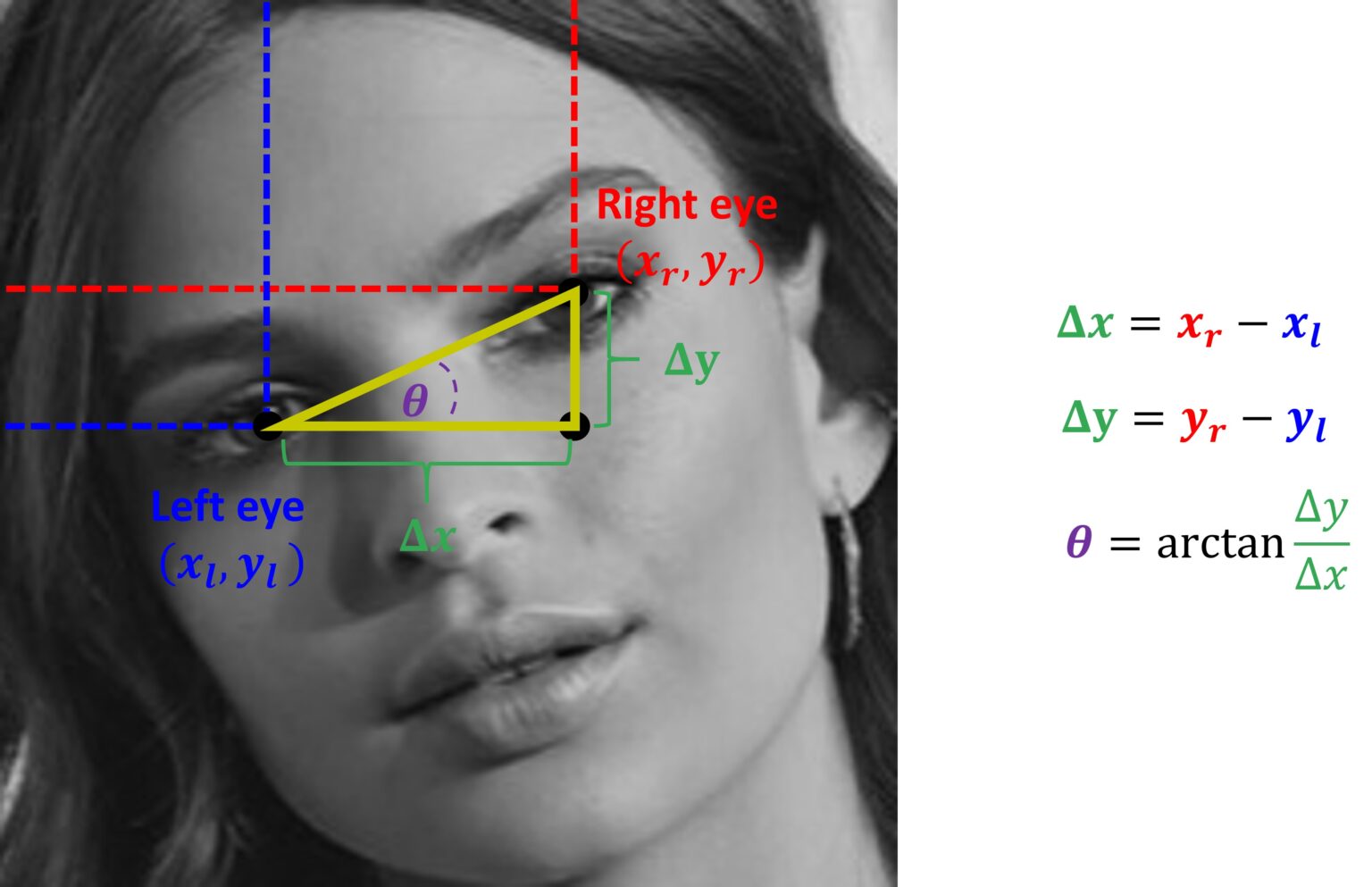
### MobileNet v2

Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst Tekst.

# Poravnanje lica

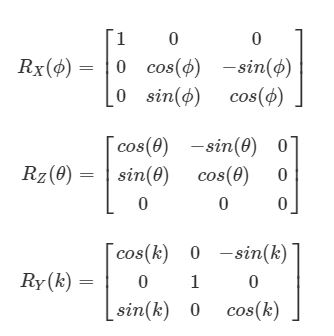
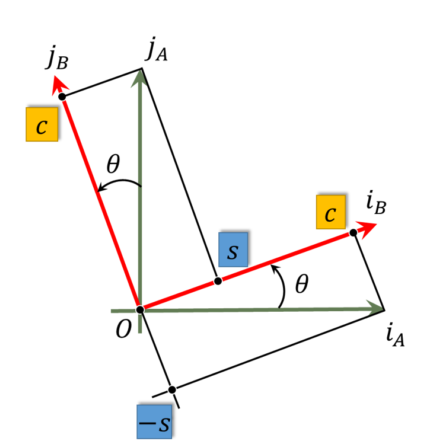
Nakon procesa detekcije i ekstrakcije lica i 5 ključnih tačaka, kako bi proces prepoznavanja bio što uspešniji, potrebno je uraditi poravnanje lica.

Jedan od jednostavnijih ali uspešnih metoda za ovo je pronalaženje arcus tangensa između dva oka (odnosno ugla između dva oka).

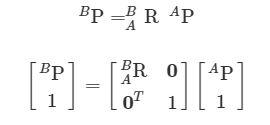


Ovde je bitno napomenuti da arctan funkcija u Numpy paketu vraća ugao u radijanima. Za dalju upotrebu je potrebno pretvoriti ga u stepene. Za ovo je samo potrebno pomnožiti sa 10 i podeliti sa PI.

Nakon ovoga, potrebno je izracunati rotacionu matricu.



Sledeći korak je upotreba afinih transformacija kako bi postigli željeni efekat. Ovakva vrsta transformacija, odnosno preslikavanja preslikva tačke u tačke, prave u prave, ravni u ravni. Kod ovakvih transformacija, par paralelnih pravi ostaje paralelan i nakon preslikavanja, ali uglovi između pravih ili razdaljine između tačaka ne moraju nužno da ostanu isti.



# Ekstrakcija vektora obeležja

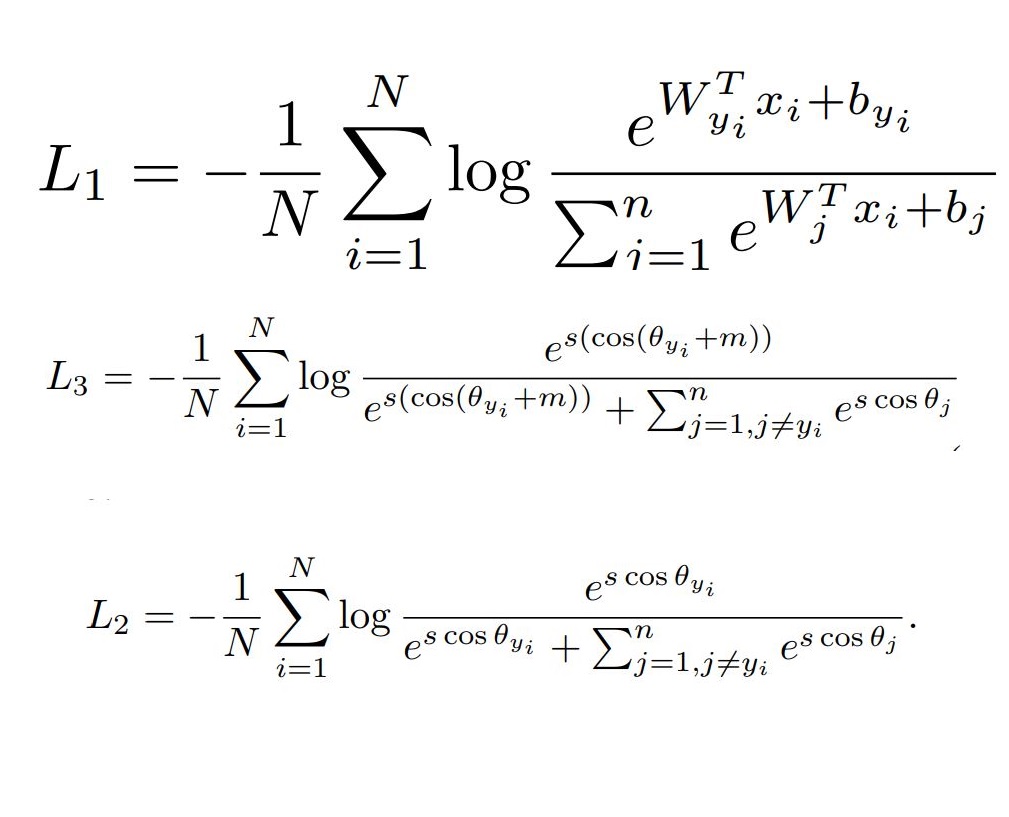
Nakon što imamo sliku poravnanog lica, sledeći korak u procesu prepoznavanja je ekstrakcija vektora obelezja (feature vektora). Tokom godina su se smenjivale razne implementacije sa istom namenom, ali sa različitim metodama ekstrakcije vektora, kao  i veličine vektora. Ranije SOTA implementacije poput FaceNet mreže su bile zasnovane na 128-dimenenzionalnim vektorima. Novije metode imaju mogućnost ekstrakcije 512-dimenzionalnih obelezja, pa čak i 1024. Kao optimalan broj obeležja se pokazao 512.

Dodati poredjenje…..

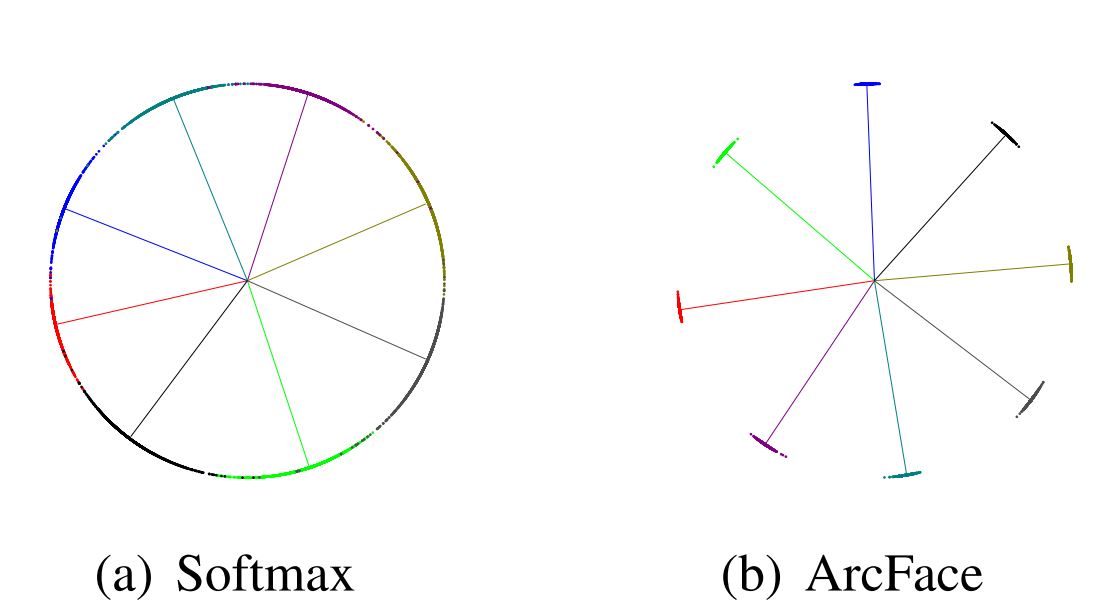
Metode poput FaceNet [https://arxiv.org/pdf/1503.03832.pdf] mreže su za cilj imale direktno učenje vektora obeležja zasnovano na triplet loss funkciji. Ideja je minimiziranje distance između vektora iste osobe (positive), dok se vektori druge osobe (negative) udaljavalju. Ovo je podrazumevalo da u svakom trenutku tokom treninga imamo tri vektora (anchor, positive, negative). Proces odabiranja tripleta je jako zahtevan i spor, što je bio prvi nedostatak ovakvog i sličnih metoda.

Jedna od implementacija koja je postigla SOTA rezultate je ArcFace [<https://arxiv.org/pdf/1801.07698.pdf>]. Osnovna razlika i ideja je bila napraviti klasifikator koji može da razdvoji različite identitete u trening setu na osnovu određene loss funkcije, kao i korišćenje 512-dimenzionalnih vektora obeležja. Problem sa triplet loss obučavanjem je i eksponencijalni skok broja kombinacija kod velikih setova podataka, dok je problem sa tradicionalnim funkcijama poput Softmax funkcije sto se linearna transformaciona matrica povecava linearno, što nije problem kod manjih setova podataka, ali je kod velikih setova ili produkcionih sistema neupotrebljivo.

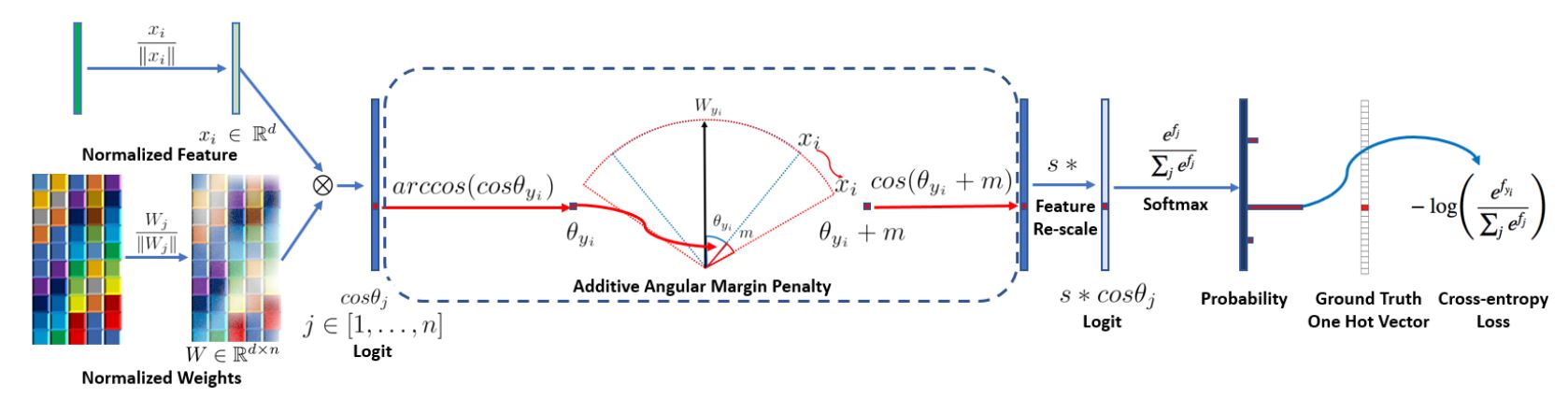
Kako bi se povećala margina između klasa, predstavljena je nova funkcija gubitka. Rec je o ArcFace funkciji.



Na sledećoj slici je prikazano kako izgleda separacija 8 klasa korišćenjem Softmax i ArcFace funkcjie.



Kao backbone mreza je u originalnom radu koriscen ResNet, ali u nasem slucaju je izabrana lightweight arhitektura pod nazivom MobileNet.



# Pretraga vektora obeležja (prepoznavanje)

Ranije imeplemetnacije sličnih sistema su koristile razne metode pretrage. Od najjednostavnihij poput brute force pretrage, k najblizih suseda (KNN), metode nosećih vektora (SVM), kd-trees, LSH, pa i neuronskih mreža za pretragu vektora i predickiju identitieta.

Pomenute metode su se pokazale kao spore, ili nedovoljno precizne. Stoga, bilo je potrebe za novim i bržim metodama pretrage velikih skupova podataka. Metoda koja se izdvaja je pretraga aproksimiranih k najblizih suseda korišćenjem Hierarchical Navigable Small World grafova (HNSW) (arXiv:1603.09320v4).

KNN algoritam za svaki element iz skupa podataka prvo definiše razdaljinu od susednih elemanta. Ta razdaljina može da predstavlja rastojanje između dva vektora u n dimenzionalnom prostoru i može se koristiti zeljena metrika. Parametar k biramo sami i on predstavlja broj suseda sa minimalnom razdaljinom od željene tačke, ili u našem slučaju vektora. Setovi podataka i njihove razmere u ovoj oblasti su veliki, kao i njihova dimenzionalnost. Stoga su metode kao što je linearno skeniranje spore. Treba napomenuti da sve novije metode ne garantuju vracanje korektnog rezultata osim takozvane exhaustive pretrage. Ovo je poznato kao “curse of dimensionality” problem.

Kako bi se ovaj problem rešio, pojavljuje se metoda pod nazivom aproksimirani k najblizih suseda (Approximate Nearest Neighbors - ANN). Ova metoda dozvoljava mali broj grešaka, a kvalitet pretrage je definisan kao odnos tačno pronađenih suseda i parametra k.

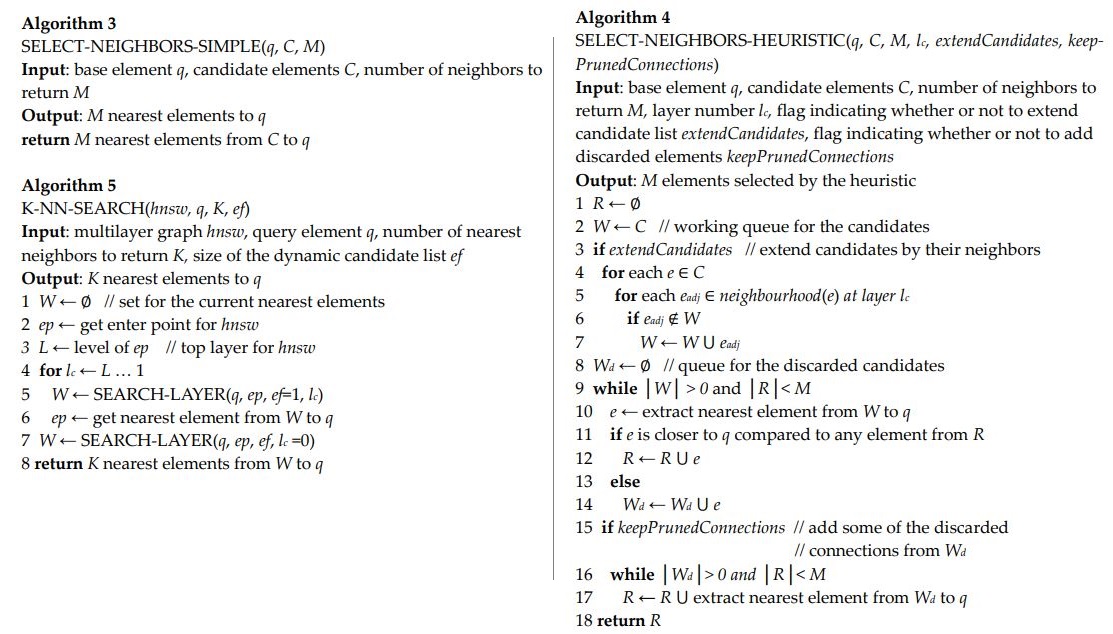
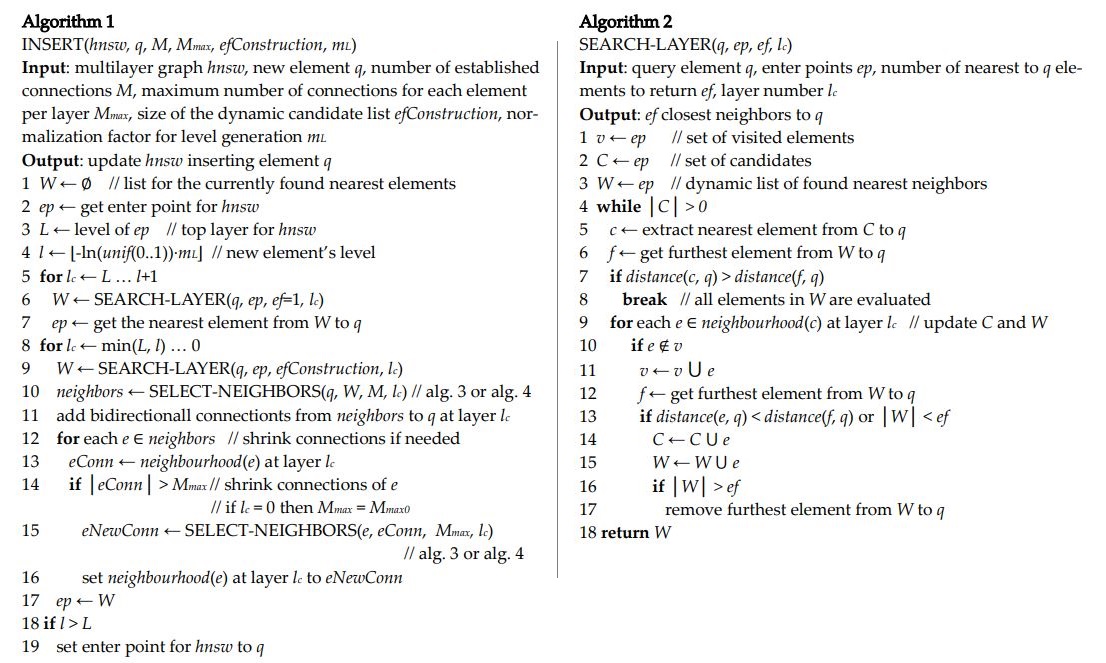
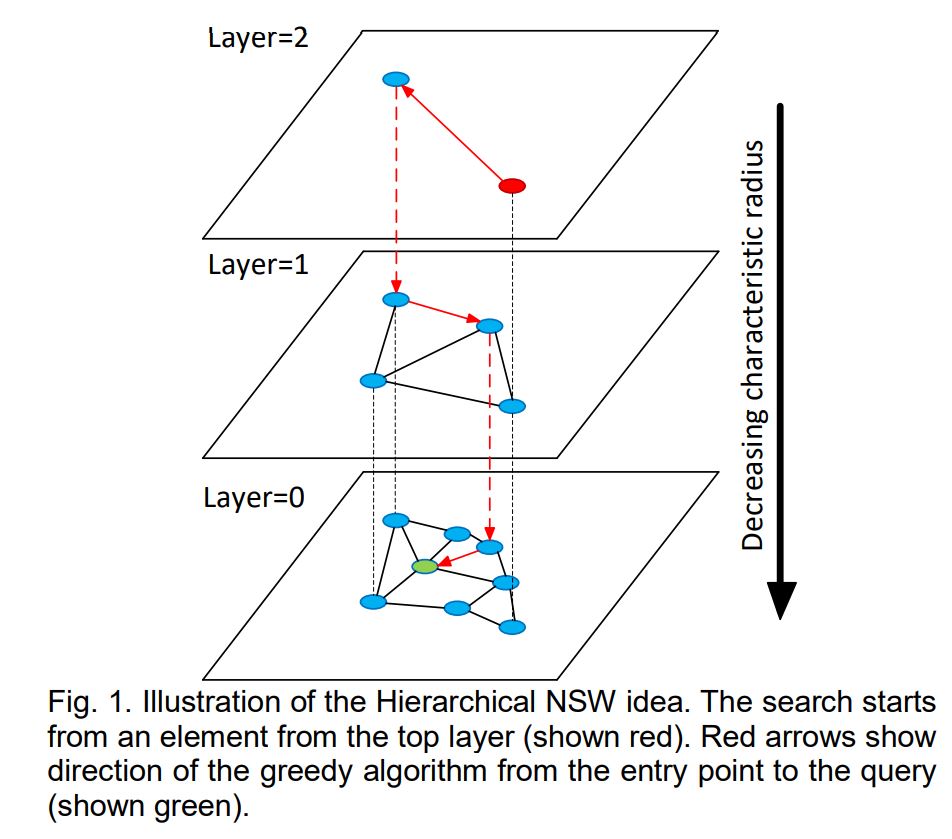
Kako bi se ubrzala pretraga ANN algoritmom korišćene su tehnike transformacije vektora (smanjenje dimenzija, rotacija) pre indeksiranja, ali i enkodovanje vektora.

Vektori se mogu enkdovati koristeći stabla (Annoy), LSH, kvantizaciju, ali i grafove.

**Dodati teoriju o grafovima**

HNSW predstavlja implementaciju graf inkrementalnog ANN algoritma.

Teorija iza NSW i HNSW algoritma je previše obimna za ovaj rad, stoga će u nastavku ukratko biti predstavljeni algoritami.



Jedna od osnovnih stvari prilikom pretrage je izbor odgovarajuće distance. S obzirom na to da je ArcFace model treniran korišćenjem ArcFace funkcije, automatski se nameće korišćenje cos-inusne distance ili angular distance.

# Implementacija u programskom jeziku Python

Za implementaciju sistema za preponzavanje lica je korišćen programski jezik Python. Za upravljanje paketima je korišćena Conda.

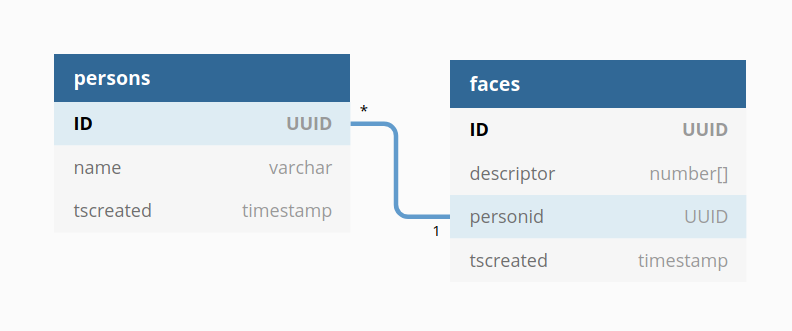
U nastavku se nalazi lista paketa koji su korišćeni za implementaciju:

1. build-essential
2. python=3.7
3. tensorflow-gpu
4. opencv=4.4.0
5. psycopg2
6. pyfiglet
7. mtcnn
8. pillow
9. flask
10. flask-cors
11. Cython
12. n2
13. googledrivedownloader

Za kompletnu instalaciju sistema potrebno je instalirati tri odvojena dela, bazu, Python service i frontend servis. Instalaciju je moguće uraditi ručno, ili uz pomoć dostupnih skripti. Sistem je testiran samo na Linux operativnom sistemu, tačnije na Ubuntu 20.04 LTS.

U nastavku neće biti prikazan ceo kod zbog svoje obimnosti, ali će biti dostupan. Fokisiraćemo se na delove koji su ključni i povezuju sve u jednu celinu. Treba napomenuti da se u sistemu nalaze i moduli koji nisu uključeni, na primer SSD detektor lica, MTCNN detektor lica, anti-spoofing modul. Promenom paramtera u JSON konfiguracionom fajlu moguće je menjati detektor, ili uključiti i isključiti eksperimentalni deo za anti-spoofing. Ovime je dobijeno na modularnosti sistema, novi moduli se mogu lako dodavati, kao i podesavati parametri vec postojećih. Ovo omogućava jednostavnije testiranje.

Za početak je bitno kreirati bazu podataka. U ovom slučaju je korišćena PostgreSQL baza, ali je vrlo jednostavno zameniti je. U bazi imamo samo dve tabele, tabelu persons gde čuvamo informaciju o imenu osobe, datum kreiranja i njen ID (UUIDv4) i drugu tabelu faces gde čuvamo encoding osobe (descriptor), ID (UUIDv4) zapisa, personid (FK na tabelu persons) i datum kreiranja zapisa.



Za čitanje podataka iz baze kreirana je funkcija read\_descriptors. Ona čita sve podatke iz tabele faces koji se kasnije koriste za kreiranja indeksa za pretragu.

|  |
| --- |
| def read\_descriptors(db):     """     read\_descriptors     The function for reading all face descriptors and ids from database     :param db: cursor     :return: {} or [], [], []     """      try:         db     except NameError:         print('Problem with the database connection')          return -1      query = 'SELECT "ID", descriptor, personid FROM public.faces'     db.execute(query)     records = db.fetchall()      ids = []     descriptors = []     persons\_ids = []      if records:          for r in records:             ids.append(r[0])             descriptors.append(r[1][0])             persons\_ids.append(r[2])          return ids, descriptors, persons\_ids     else:         return {'status': 'ERROR'} |

Nakon čitanja vektora iz baze, potrebno je dodati ih u HNSW graf. Ovo radimo funkcijom make\_base. Ideja je dodati samo vektore i kreirati indeks. Ovo možemo sačuvati radi kasnijeg učitavanja ukoliko dođe do neke neželjene situacije. N2 paket podržava multithreading, tako da je ova opcija korišćena radi bržeg kreiranja indeksa.

|  |
| --- |
| def make\_base(self, descriptors: []) -> dict:         """         make\_base         The function used for adding data and building the index         :param descriptors: []         :return: img: numpy.array()         """          self.recognizer = HnswIndex(512, 'angular')          # add vectors to the ann         for d in descriptors:             self.recognizer.add\_data(np.array(d))          # build ann         self.recognizer.build(m=5, max\_m0=10, n\_threads=4)          self.recognizer.save('index.hnsw')         return {'status': 'SUCCESS'} |

# Zaključna razmatranja

U svom istraživanju u ovom radu došao/la sam do sledećih važnih nalaza.

RetinaFace, kao i ArcFace mreže su u ovom radu bazirane na MobileNet arhitekturi. Ova arhitektura pruža bolje performanse od prethodno pomenutog ResNet-a, ali to nije reč sa preciznošću. Prvi korak u postižanju tačnijeg sistema bi bila zamena mreže. N2 paket koji je korišćen u ovom radu podržava jako mali broj dostupnih distanci. Premda angular distanca pruža bolje performanse, i može prceiznije da razvrsta klase koje su blizu jedna drugoj, kod velikih setova podataka može doci do gresaka, pa je bolja opcija koristiti pakete koji podržavaju cos-inusnu distance, ili je implementirati i dodati u N2 paket.

Iako je MobileNet arhitektura brza, implementacija korišćena u ovom radu ne postiže SOTA rezultate, te se performanse mogu dodatno poboljšati korišćenjem originalne implementacije koja koristi dodatne trikove za optimizaciju.

Za jedan produkcioni sistem je bitno da koristi i anti-spoofing metode kako bi se zaštitili od napada. Sistem poput ovog je modularan, pa se ubacivanjem dodatnog modula između dela za detekciji i dela za prepoznavanje ovo može postići. Treba imati na umu da su implementacije anti-spoofing sistema kompleknse, i većina SOTA modela koristi RGB, depth i IR slike.

Delove koda koji se često ponavljaju, kao i delovi koji isključivo rade sa Numpy nizovima, mogu se prepraviti da koriste CuPy i Numba kompajler.

Delovi koda se mogu prepraviti po konkurentnom principu kako bi se izbeglo zaključavanje procesa i čekanje odgovora. Takodje, premda lak za implementaciju, Flask razvojni okvir nije namenjen za komercijalne sisteme, te bi njegova zamena bila neophodna.

Sistemi dubokog učenja su zahtevni za deployment i odrzavanje. Rešenje koje je idealno za ovakve sisteme je korišćenje Docker-a, odnosno principa kontejnera. U ovom slučaju ne samo da bi se samo Python deo sistema morao prebaciti u kontejner, već i cela baza.

Do sada je bilo reči o radu sa slikama. Ukoliko ima potrebe raditi sa video snimcima, situacije postaje komplikovanija. Premda video snimak jeste samo niz slika, i moguće je uraditi prepoznavanje lica na svakoj slici (frejmu), ovo je prilično zahtevan posao za računar. U ovim slučajevima je potrebno koristiti tehnike praćenja (eng. tracking) i reidentifikacije.

# Literatura

1. Holland, M. (1996) *HarvardSystem* [on-line]. Poole, Bournemouth Universitv. <http://bournemouth.ac.uk/service-depts/lis/LIS_Pub/harvardsyst.html> (dostupno dana: 15.05.2014.)
2. Petrović, Z., Vićentijević, K. i Stanišić, N. (2008) *Zbirka zadataka iz računovodstva*, drugo izdanje, Univerzitet Singidunum, Beograd
3. Pfeffer, J. (1996) When it comes to "best practices" - why do smart organizations do dumb things? *Organizational Dvnamics,* 25, 33-44
4. Stanišić, M. (2014) *Revizija*, Univerzitet Singidunum, Beograd
5. Zakon o računovodstvu i reviziji (Službeni glasnik Republike Srbije *br/godina*)
6. Zakon o autorskim i srodnim pravima (Službeni glasnik Republike Srbije *br/godina*)