SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 686

Klasifikacija prometnih scena u skupove oznaka

Ivan Horvatin

Zagreb, lipanj 2014.

Zahvala

Zahvaljujem se mentoru izv. prof. dr. sc. Siniši Šegviću na strpljenju i potpori tijekom izrade ovog rada.

Također se zahvaljujem dr. sc. Karli Brkić i Ivanu Sikiriću, dipl. ing. na stručnom vođenju kroz proces izrade diplomskog rada, te pomoći, savjetima i strpljenju tijekom ovog rada.

Konačno, zahvaljujem se svojoj obitelji i prijateljima na potpori i razumijevanju u periodu izrade diplomskog rada i općenito na potpori tijekom studiranja.

SADRŽAJ

1. Uvod

Ovaj rad se bavi problemom klasifikacije prometnih scena u skupove oznaka. Svakoj promentoj sceni se dodjeljuju oznake za objekte ili scene koje se nalaze na slici. Pod skupom oznaka podrazumijevamo oznake koje dodjeljujemo slici kojima određujemo što se na njoj nalazi, za razliku od klasičnog problema klasifikacije gdje svaka slika ima po jednu oznaku. Cilj rada je proučavanje i vršenje eksperimenta koristeći klasifikaciju u skupove oznaka i razne klasifikatore i značajke, te usporedba i prikaz rezultata istih kako bismo mogli dobiti informaciju o tome koliko je pouzadno raditi klasifikaciju u skupove oznaka na predložen način.

U radu se razmatra skup slika FM2 [17]. Primjeri prometnih scena iz skupa FM2 prikazani su na slici 1.1. Skup FM2 sadrži 6237 slika prometnih scena sa različitim obilježjima na svakoj slici (npr. ceste, auti ili nadvožnjaci, vidi sliku 1.1). Više o skupu FM2 se može pročitati u poglavlju 3.

Slika 1.1: Primjer slika FM2 skupa

Tradicionalna klasifikacija s jednom oznakom se bavi učenjem uzoraka iz skupa podataka koji je označen sa jednom oznakom l iz skupa disjunktnih oznaka L, za $|L| > 1$. Ako je $|L| = 2$, tada se problem naziva binarni klasifikacijski problem, ali ako je $|L| > 2$ tada to postaje višeklasni problem klasifikacije. Klasifikacija sa više oznaka se bavi učenjem uzoraka iz skupa podataka označenih sa više oznaka $Y \subseteq$

L. Klasifikacija u skup oznaka se često koristi kod kategorizacije teksta i medicinske dijagnoze. Na primjer tekstualni dokumenti mogu često pripadati u više klasa (kao npr. časopis koji govori o nogometu i politici može biti nogometni časopis ili politički). Klasifikacija u skupove oznaka nalazi primjenu i u medicini, jer čovjek može bolovati od više bolesti odjednom. Sve češće se metode klasifikacije u skup oznaka koriste u modernim aplikacijama, na primjer kod kategorizacije glazbe (pjesma može pripadati u više od jednog žanra), pa tako i kod klasifikacije scena u prometu. Slike prometnih scena se mogu klasificirati kao npr. cesta, auto, gužva, nadvožnjak, ali to ne znači da je samo jedna od tih stvari na slici. Ako smo nešto klasificirali kao cestu to ne znači da na toj slici mora biti auto, ali ne znači ni da auta nema. Zato koristimo klasificiranje u skup oznaka, pošto svaka slika može sadržavati više stvari koje su nam bitne za razumijevanje scene.

Za postupak klasifikacije u skupove oznaka u ovom radu koristi se *Bag of Words* model za prikaz slike vektorima značajki. Najprije se iz skupa slika izlučuju *SIFT* značajke. One se koriste za izradu slikovnog rječnika, te se zatim grade histogrami slikovnih riječi. Ti histogrami koriste se kao ulaz u klasifikator. Dio skupa slika se koristi za učenje klasifikatora dok se drugi dio skupa koristi za evaluaciju uspješnosti. Ovi (disjunktni) podskupovi slika se zovu skup za učenje i skup za testiranje. U ovom radu koriste se klasifikatori *SVM* (stroj s potpornim vektorima, engl. Support Vector Machine) i *Random Forest* (nasumične šume) sa različitim parametrima.

Za izradu programskog rješenja i vršenja eksperimenata korišten je program iz [14]. Od značajki korištene su različite varijante SIFT značajki, generirane pomoću programskog rješenja [14].

Za izradu programskog rješenja korišten je programski jezik *Python* te biblioteka otvorenog tipa *Scikit-learn* koja je u zadnje vrijeme jako popularna, a sadrži korisne i jednostavne alate za analizu i obradu podataka. Uz izradu programskog rješenja, modificiran je kod iz [14] kako bi se mogao iskoristiti u sklopu mojeg programskog rješenja.

Razvijeno programsko rješenje se sastoji od učitavanja značajki, njihove obrade te ispisivanja i usporedbe rezultata svakog eksperimenta. Eksperimenti su prilično procesno zahtjevni zbog količine slika, izrade rječnika kod *Bag of Words* modela te isprobavanja različitih parametara kako bi rezultati bili optimalni.

Ovaj radi je strukturiran kako slijedi. U poglavlju 2 su ukratko opisani algoritmi koji se koriste u eksperimentima, u poglavlju 3 je dana detaljnija informacija o skupu slika s kojima se radi te postupak označavanja slika u skupove oznaka, u poglavlju 4 su opisane biblioteke koje su bile korištene i neke od funkcija koje se koriste u pro-

gramskom rješenju, u poglavlju 5 se mogu vidjeti rezultati i komentari eksperimenata te u poglavlju 6 dajem zaključak na kraju rada na osnovu eksperimenata.

2. Korišteni algoritmi

U ovom radu se koristi model *Bag of Words* za prikaz pojedinačnih slika, te klasifikatori *SVM* i *Random Forest* za klasifikaciju tako prikazanih slika u skupove oznaka. Kao značajke u modelu *Bag of Words* sam koristio nekoliko različitih varijanti generiranih *SIFT* značajki mijenjajući im pragove. *SIFT* značajke su bile generirane progamskim rješenjem [14]. Za postupak *Bag of Words SIFT* značajke su korištene kako bi izgradili rječnik vizualnih riječi pomoću kojega se onda dobivaju histogrami pojavljivanja vizualnih riječi za svaku sliku. Ti su histogrami bili ulaz u klasifikacijski postupak. Prije nego što je započeta izrada programskog rješenja bilo je potrebno označiti svaku sliku oznakama koje će pokazivati što se nalazi na slici. Kao što je već bilo rečeno broj slika u skupu FM2 je 6237, a za svaku sliku je razmatrano 5 oznaka: cesta, auto, nadvožnjak, tunel i naselje.

2.1. Model *Bag of Words*

Jednostavan pristup klasificiranju slika je da ih tretiramo kao skup interesantnih regija opisujući njihov izgled i zanemarujući njihovu prostornu strukturu. Slični modeli se uspješno koriste nad analizom teksta i zovu se *Bag of Words* modeli, pošto je svaki tekst reprezentiran distribucijom riječi preko rječnika. Postupak je takav da se iz teksta koji želimo obraditi stvara histogram pojavljivanja riječi, ne uzimajući u obzir gramatiku ili red pojavljivanja riječi. Brojevi pojavljivanja svih riječi se tada koriste u vektoru značajki za učenje klasifikatora. Na temelju pojavljivanja određenih riječi moglo bi se tada zaključiti da li se u tekstu radi o sportu, politici, da li je riječ o neželjenoj pošti ili o nečemu drugome.

2.1.1. Primjena u računalnom vidu

Bag of words model se također može koristiti i za klasifikaciju u računalnom vidu. U ovom slučaju značajke na slici se tretiraju kao riječi, nazovimo ih slikovnim riječima.

Da bi ostvarili prikaz tim modelom potrebno je pratiti 3 koraka: detektirati značajke na slici koje ćemo koristiti kao slikovne riječi, reprezentirati ih na način koji bi bio upotrebljiv za obradu (primjer SIFT, koji je i jedan od značajki koji se koristi u ovom radu), te stvaranje rječnika slikovnih riječi preko kojeg će se moći stvoriti histogram pojavljivanja svake slikovne riječi za pojedinu sliku. U ovom radu se koristi algoritam *k*-srednjih vrijednosti za generiranje rječnika, algoritam SIFT za detektiranje i izvlačenje značajki u obliku deskriptora iz slike, te *SVM* i *Random Forest* zajedno sa rječnikom za klasifikaciju slika. Iako jednostavan, postupak može donijeti dobre rezultate. Jedan od velikih nedostataka *Bag of Words* modela je to što ne uzima u obzir prostornu udaljenost između dijelova slike na kojima su detektirane značajke, što je u nekim slučajevima vrlo bitno u reprezentaciji slike.

2.1.2. Kategorizacija slika histogramima slikovnih riječi

Kako bismo slike kategorizirali primjenom histograma slikovnih riječi (*Bag of Words*) potrebno je primijeniti sljedeća tri algoritma: algoritam *SIFT* za detektiranje i izvlačenje značajki, algoritam k-srednjih vrijednosti za generiranje rječnika te *SVM* i *Random Forest* za klasifikaciju slika. Za početak, potreban je skup slika koje će se koristiti kao skup za stvaranje slikovnog rječnika te skup slika koje će se koristiti kao skup za učenje klasifikatora. Isti skup slika može se koristiti u oba slučaja. Prvi korak kategorizacije je izračun lokalnih značajki slika koje će se koristiti za stvaranje slikovnog rječnika. U tu svrhu može poslužiti algoritam SIFT. Drugi korak je stvaranje slikovnog rječnika. Sve lokalne značajke koje su prethodno izlučene grupiraju se u slikovne riječi. Prethodno grupiranju potrebno je odrediti željeni broj slikovnih riječi. Prilikom odabira broja slikovnih riječi potrebno je uzeti u obzir broj slika te prosječan broj značajki koji se dobiva iz jedne slike. Značajke će se grupirati u riječi pomoću nekog algoritma za grupiranje, poput k-srednjih vrijednosti. Nastojat će se minimizirati suma kvadrata pogreške grupiranja, kako bi lokalne značajke koje su slične bile grupirane kao jedna slikovna riječ. Razlog zašto koristimo slikovne riječi, a ne samo SIFT opisnike lokalnih značajki je što se značajka na slici može pojaviti gledana pod različitim kutem, može biti djelomično zaklonjena ili se može razlikovati od objekta do objekta. Npr. svaki bicikl neće imati sjedalo potpuno istog oblika. Vrijednosti opisnika lokalnih značajki lagano će varirati, ali će oni ipak predstavljati isti objekt na slici. Ako pod jednom slikovnom riječi, npr. bude 100 različitih modela sjedala za bicikl (odnosno njihovih opisnika) te određujemo za neku novu značajku (koja predstavlja dosad neviđeno sjedalo za bicikl) kojoj slikovnoj riječi pripada, vrlo je vjerojatno da ćemo za tu značajku

odrediti da pripada slikovnoj riječi koja predstavlja sjedalo. Nakon stvaranja slikovnog rječnika, za SIFT opisnik značajke moguće je odrediti kojoj riječi pripada zato što će značajka biti jako slična riječi koju opisuje. Ponavljanjem tog postupka nad svakim opisnikom slike moguće je stvoriti histogram slikovnih riječi te slike. Histogram će imati isti broj komponenata koliko i rječnik ima riječi. Svakoj pojedinoj komponenti biti će pridružen jedan cijeli broj koji označuje koliko puta se ta slikovna riječ pojavila na slici. Za svaku sliku iz skupa slika za učenje potrebno je izgraditi histogram slikovnih riječi. Ti histogrami koriste se za učenje klasifikatora.

Prilikom klasifikacije slike, postupak je sljedeći:

1. Generiranje opisnika lokalnih značajki slike
2. Stvaranje histograma slikovnih riječi pridruživanjem lokalnih opisnika slikovnim riječima slikovnog rječnika
3. Kategorizacija pomoću dobivenog histograma i klasifikatora

2.2. Klasifikacija slika u skup oznaka

U strojnom učenju klasifikacija slike sa jednom oznakom je uobičajen problem učenja gdje je cilj učiti iz skupa podataka, gdje svaki skup ima svoju jedinstvenu oznaku iz skupa disjunktnih oznaka L. U ovisnosti o ukupnom broju neovisnih oznaka problem može biti identificiran kao binarni (kada je $|L| = 2$) ili višeklasni (kada je $|L| > 2$). Za razliku od takvih problema, klasifikacija u skup oznaka podržava slučaj pojavljivanja više oznaka na podacima. Cilj u klasifikaciji sa skupom oznaka je da se uči na skupu podataka gdje svaki podatak može pripadati jednoj ili više klasa iz L. Iako je potreba za klasifikacijom u skupove oznaka nastala prvenstveno zbog automatske kategorizacije teksta i medicinskih dijagnoza, sveprisutnost problema koji sadrže skupove oznaka privlači sve više i više zanimanja za ovu domenu. Tradicionalni binarni i višeklasni problemi se mogu iskoristiti kao posebni slučajevi problema klasifikacije slika u skup oznaka, ali općenitost klasifikacije u više oznaka čini taj problem težim od njih. Postoji više načina rješavanja tog problema koje dijelimo u dvije skupine: transformiranje problema u jednostavniji i metode adaptacije [18].

U našem slučaju se koristi metoda pod nazivom *Binary Relevance (BR)* (binarna značajnost). Ovo je jedna od najpopularnijih transformacijskih metoda koja zapravo stvara k skupova podataka ($k = |L|$, ukupni broj klasa), svaki skup za jednu oznaku, i uči klasifikator nad svakim od tih skupova. Svaki od tih k skupova sadrži isti broj

podataka kao i originalni skup podataka, ali svaki od tih skupova $D_{\lambda_j}, 1 \leq j \leq k$ označava da li podatak pripada ili ne pripada oznaci (labeli) λ_j. Na slici 2.1 možemo vidjeti podjelu u transformirane skupove podatak.

Brojevi 1-4 predstavljaju slike, a λ_J predstavlja oznake, što konkretno u ovom slučaju na slici znači da se naš početni skup sastojao od 4 slike i 4 oznake $\lambda_{1,2,3,4}$. Za svaku od oznaka je napravljen zasebni skup identičan početnom, samo što se za svaki skup gleda samo jedna oznaka (to se podrazumijeva kod transformiranih skupova podataka). Stoga možemo vidjeti npr. u slici a) da se oznaka λ_1 ne nalazi na 1. i 4. slici ali se nalazi na 2. i 3.

Ex.	Label
1	$\neg\lambda_1$
2	λ_1
3	λ_1
4	$\neg\lambda_1$

(a)

Ex.	Label
1	λ_2
2	$\neg\lambda_2$
3	λ_2
4	λ_2

(b)

Ex.	Label
1	λ_3
2	$\neg\lambda_3$
3	λ_3
4	$\neg\lambda_3$

(c)

Ex.	Label
1	$\neg\lambda_4$
2	$\neg\lambda_4$
3	$\neg\lambda_4$
4	λ_4

(d)

Slika 2.1: Transformirani skupovi podataka sa oznakama $\lambda_{1,2,3,4}$ i slikama 1, 2, 3 i 4 dobiveni metodom binarne značajnosti (BR) [18]

Kada imamo transformirane skupove podataka, jednostavno je naučiti po jedan binarni klasifikator za svaki od njih. Za svaki novi primjer *BR* kao odluku daje uniju labela λ_j koje postoje na slici. Iako se *BR* koristi u puno praktičnih sustava često je kritiziran zbog implicitne pretpostavke o neovisnosti oznaka, što nije uvijek slučaj [18].

2.3. SIFT

Kako bismo slike uopće mogli kvalitetno uspoređivati potreban nam je algoritam koji će iz slike moći izlučiti stabilne lokalne značajke (značajke su točke od interesa koje na neki način ističu jedinstvenost sadržaja slike). Značajke će biti stabilne ukoliko će biti relativno invarijantne na promjene u mjerilu, orijentaciji, osvjetljenju ili na pojavu šuma. U posljednje vrijeme vrlo su popularne značajke SIFT (engl. Scale Invariant Feature Transform). SIFT je predložio David G. Lowe 1999. godine [9]. Glavna karakteristika SIFT značajki je to da su izuzetno robusne na prethodno spomenute promjene (rotacija, mjerilo, osvjetljenje, afine transformacije).

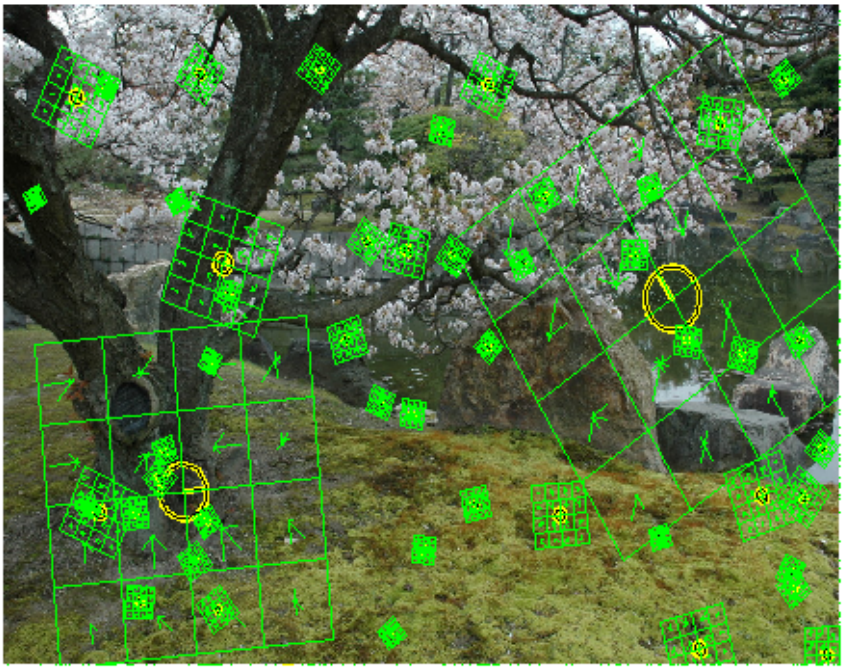
Određivanje lokalnih značajki slike vrši se kroz nekoliko koraka. Prvo početnu

sliku zaglađujemo Gaussovim filtrom koristeći pritom neki početni iznos standardne devijacije. Gaussov filtar koristi se kako bi se eliminirao šum na slici. Zatim se mjerilo slike povećava za neki faktor koristeći pritom linearnu interpolaciju za određivanje vrijednosti novih slikovnih elemenata. Isto kao i kod početne slike, dobivenu sliku također zaglađujemo Gaussovim filtrom koristeći pritom početni iznos standardne devijacije uvećan za neki faktor. Postupak povećavanja mjerila i zaglađivanja novodobivene slike može se primijeniti više puta. Sve slike jednake veličine po svim razinama mjerila čine jednu oktavu. Nakon što se izračunaju sve slike jedne oktave, rezolucija početne slike smanjuje se za faktor 2 i to tako da se uzima svaki drugi element iz svakog retka odnosno stupca slike te se ponavlja prethodno opisani postupak izračunavanja svih slika oktave čime se dobiva nova oktava.

Nakon izračuna potrebnog broja oktava (npr. 4 oktave s po 5 razina mjerila), unutar svake oktave promatra se razlika slika susjednih mjerila, odnosno razlika Gaussiana (engl. Difference of Gaussians, ili popularno skraćeno DoG). Traže se lokalni maksimumi i minimumi te se vrši odbacivanje značajki koje se proglase nestabilnima (nestabilne značajke su one koje nemaju dovoljno veliki kontrast ili su loše lokalizirane uz rub). Jednom kada se značajka proglasi stabilnom, izračunava se njena dominantna orijentacija (dominantan smjer promjene gradijenta) te joj se dodjeljuje lokacija na slici (x i y koordinate) i mjerilo. Nakon što značajka ima definiranu lokaciju, mjerilo i orijentaciju potrebno je odrediti opisni vektor značajke. Opisni vektor gradi se tako da se promatra regija oko značajke. Regija se dijeli na 4x4 podregije te se za svaku podregiju gradi opisnik koji se sastoji od osam komponenata (brojeva). Vrijednosti komponenata ovise o smjerovima gradijenata promatrane podregije (gradi se histogram smjerova gradijenta sa 8 pretinaca). Nakon izračuna opisnika za svaku podregiju svi dobiveni opisnici spajaju se u jedan 128-komponentni opisnik. Dobiveni opisnik je SIFT opisnik značajke koji možemo vidjeti na slici 2.2. Žuti kružići označavaju centar interesne točke SIFT-a (drugačije su veličine jer su u različitim skalama), a crtice unutar njih dominantnu orijentaciju pojedine ćelije. Oko njih u zelenim kockicama nalaze se opisnici za svaku podregiju u obliku crtica u 8 različitih smjerova gdje dužina crtice označava jačinu gradijenta u tom smjeru.

2.4. Algoritam k-srednjih vrijednosti

Algoritam k-srednjih vrijednosti u programskom rješenju se koristi za stvaranje slikovnog rječnika na način da centroidi određeni algoritmom predstavljaju slikovne riječi, od kojih se onda tvori slikovni rječnik. Svrha algoritma k-srednjih vrijednosti je par-



Slika 2.2: Detektirane značajke i njihovi opisnici (preuzeto sa [22])

ticijsko grupiranje N vrijednosti u K grupa gdje je K unaprijed određen. Prilikom grupiranja svaki podatak smješta se u onu grupu čijoj je srednjoj vrijednosti najbliži (euklidska udaljenost).

Funkcija pogreške se računa se kao:

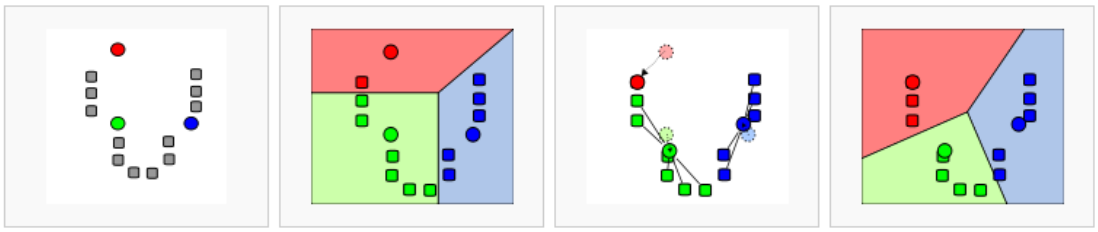
$$J = \sum_{k=1}^{K} \sum_{x_i \in S_k} ||x_i - \mu_k||^2 \tag{2.1}$$

Parametar x_i predstavlja vrijednost koja je svrstana u neku grupu k dok μ_k predstavlja srednju vrijednost k-tog centroida, a S_k predstavlja skup svih x_i iz grupe k. Podatke je potrebno grupirati na način koji minimizira funkciju pogreške J. Prilikom inicijalizacije centroida, njihove srednje vrijednosti odabiru se nasumično. Nakon početne raspodjele svih vrijednosti po centroidima, računaju se nove vrijednosti centroida na temelju podataka koji se trenutno u njima nalaze. Zatim se vrši preraspodjela svih vrijednosti koje više ne pripadaju centroidu u kojem se trenutno nalaze nego zbog novih srednjih vrijednosti centroida pripadaju nekom drugom centroidu. Postupak se iterativno ponavlja tako dugo dok se srednje vrijednosti centroida mijenjaju. Iteracija bez promjena srednjih vrijednosti centroida označuje da je pronađeno lokalno optimalno rješenje.

Algoritam ne garantira da je rješenje koje pronađe optimalno jer ovisi o početnim srednjim vrijednostima centroida koji se odabiru slučajnim odabirom (pa algoritam

može „zaglaviti“ u lokalnom optimumu). Zato se najčešće algoritam provodi više puta uz različiti odabir početnih srednjih vrijednosti centroida te se kao rješenje uzima ono s najmanjom vrijednosti pogreške J. Prilikom određivanja kojem centroidu pripada ispitni podatak, potrebno je izračunati euklidsku udaljenost ispitnog podatka i svakog centroida te odabrati onaj s najmanjom udaljenošću.

Na slici 2.3 možemo u par sličica vidjeti ilustrativni primjer rada algoritma k-srednjih vrijednosti. Na prvoj sličici se generiraju početne srednje vrijednosti (u boji) u ovom slučaju $k = 3$, na sljedećoj sličici se podaci grupiraju u skupine s obzirom na udaljenost najbližeg centroida, na trećoj sličici možemo vidjeti da srednja vrijednost podataka u svakoj skupini postaje novi centroid skupine. Koraci na sličici 2 i 3 se ponavljaju dok se srednja vrijednost svake skupine prestane mijenjati kao u sličici 4.



Slika 2.3: Demonstracija rada algoritma k-srednjih vrijednosti [12]

2.5. SVM (Stroj s potpornim vektorima)

Stroj s potpornim vektorima (engl. Support Vector Machine – popularno SVM) je binarni klasifikator, što znači da je u mogućnosti na temelju nekih značajki odijeliti skup uzoraka u dvije grupe. Stroj s potpornim vektorima na temelju skupa za učenje konstruira hiperravninu ili skup hiperravnina u visokodimenzionalnom prostoru (moguće je i u beskonačnodimenzionalnom prostoru) koja odjeljuje dvije grupe uzoraka. Prilikom učenja algoritmu je potrebno dati informaciju u koju od dvije skupine spada pojedini uzorak, što ovaj algoritam čini nadziranim algoritmom strojnog učenja. U slučaju binarne klasifikacije, jednu grupu uzoraka možemo označiti s "1", a drugu grupu s "-1".

2.5.1. Pouzdanost klasifikacije

Neka postoji funkcija $f(x) : R^n \rightarrow R$ koja je funkcija odluke o klasifikaciji. Ta funkcija prima n – dimenzionalni vektor značajki x nekog uzorka te ga preslikava u realan broj, $y = f(x)$. U slučaju da je $y \geq 0$ uzorak klasificiramo tako da pripada

grupi uzoraka s oznakom "1", a ukoliko je $y < 0$ uzorak svrstavamo u grupu uzoraka s oznakom "-1". Funkcija $f(x)$ mora biti što pouzdanija, što znači da ako uzorak svrsta u jednu od grupa, on zaista i pripada toj grupi.

Idealna funkcija $f(x)$ ispravno odjeljuje sve uzorke danog skupa za učenje, odnosno vrijedi

$$y^{(i)} = 1 \implies f\left(x^{(i)}\right) \geq 0 \quad \wedge \quad y^{(i)} = -1 \implies f\left(x^{(i)}\right) < 0 \tag{2.2}$$

2.5.2. Odvajanje linearno razdvojivih grupa objekata

Pretpostavimo da imamo skup za učenje koji se sastoji od nekog broja uzoraka. Svakom uzorku pridružen je vektor značajki $x_i \in R^n$ gdje n označava broj značajki (dimenzionalnost vektora značajki), te svaki uzorak još ima i pridruženu vrijednost $y_i \in \{-1, 1\}$ koja označava kojoj grupi uzoraka dotični uzorak pripada. Ulazni podaci formirat će se kao skup uređenih parova $\{x_i, y_i\}$ te će se proslijediti algoritmu *SVM* koji će pronaći hiperravninu koja odjelju te dvije grupe. Hiperravninu koja odjeljuje dvije grupe uzoraka formuliramo izrazom $w \cdot x + b = 0$

w predstavlja vektor normale razdvajajuće ravnine

x predstavlja vektor značajki nekog objekta

b ukoliko je vektor normale hiperravnine normaliziran b predstavlja udaljenost hiperravnine od ishodišta

Potporni vektori su uzorci iz obje grupe objekata koji se nalaze najbliže razdvajajućoj hiperravnini kao što je prikazano na slici 2.4. Zadaća stroja s potpornim vektorima je da pronađe hiperravninu koja će biti maksimalno udaljena od oba potporna vektora.

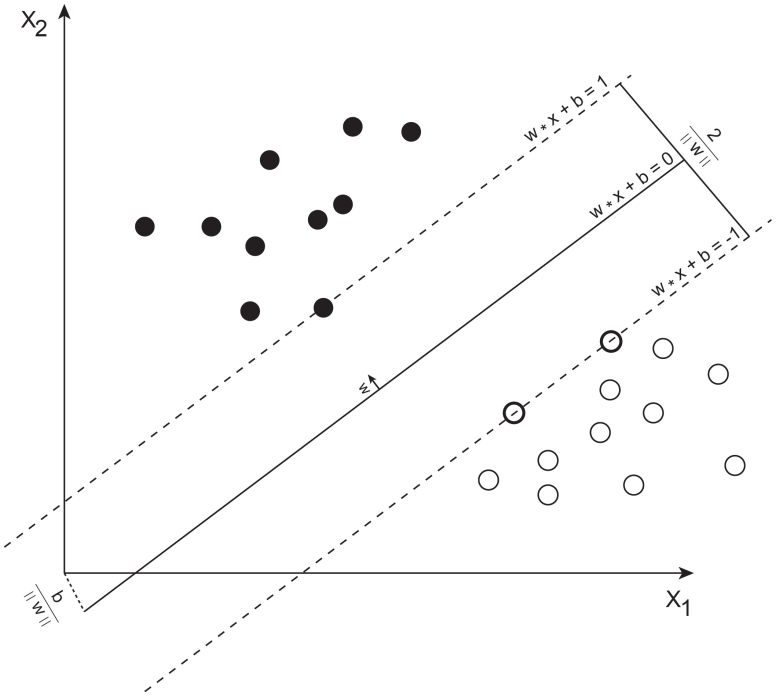
Za hiperravnine u kojima leže potporni vektori vrijedit će :

$$w \cdot x + b = 1 \text{ ako je to granica grupe s oznakom "1"} \tag{2.3a}$$

$$w \cdot x + b = -1 \text{ ako je to granica grupe s oznakom "-1"} \tag{2.3b}$$

gdje je x točka koja leži u pripadajućoj hiperravnini.

Udaljenost tih dviju hiperravnina naziva se margina te je jednaka $\frac{2}{||w||}$. Stroj s potpornim vektorima mora riješiti optimizacijski problem koji se može predstaviti kao pronalazak $\min ||w||$ uz uvjet da vrijedi $y_i(w \cdot x + b) \geq 1$. Ako za neki objekt i vrijedi $y_i = 1$ tada će funkcija klasifikacije vratiti vrijednost za koju će vrijediti $f(x_i) \geq 1$



Slika 2.4: Dvije grupe linearno odjeljivih uzoraka i njihova granica te potporni vektori na njoj

te će gornja jednadžba biti zadovoljena. Jednadžba će također biti zadovoljena i za objekte za koje vrijedi $y_i = -1$.

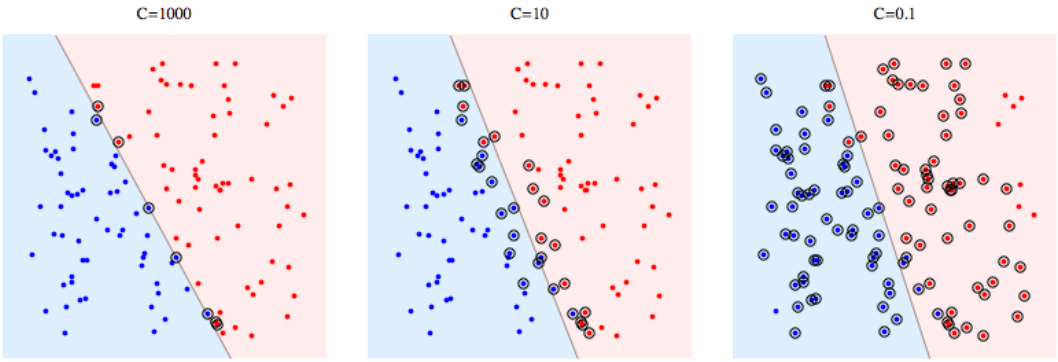
2.5.3. Meka margina

Corinna Cortes i Vladimir N. Vapnik su 1995. godine predložili modifikaciju graničnih margina koja je manje stroga pa dopušta loše klasificrane uzorke [4]. U našem slučaju u programskom rješenju to se radi promjenom parametra C. Parametar C utječe na postavljanje decizijske ravnine i to na način da što je C bliži nuli ne obraćamo toliku pažnju na točke koje krše ograničenje naše margine. To je ekvivalent stvaranju velikog prostora ili jako razmaknutih graničnih margina oko decizijske funkcije. Ako je C bliže beskonačnosti onda obraćamo veliku pažnju na točke koje krše ograničenje margine i bliže smo sa marginama decizijskoj funkciji. Na slici 2.5 možemo vidjeti utjecaj promjene parametra C. Veći kružići simboliziraju potporne vektore. Možemo vidjeti da smanjivanjem parametra C žrtvujemo linearnu seperabilnost radi stabilnosti.

Bez dodatka parametra C margine su bile definirane strogom marginom (2.4)

$$y_i \times (x_i^T \times w + b) \geq 1 \quad y_i \in \{-1, 1\} \tag{2.4}$$

Međutim ovdje se koristi manje stroga margina (2.5)



Slika 2.5: Primjer margina za različiti paramaetar C [3]

$$y_i \times (x_i^T \times w + b) \geq 1 - \xi_i \quad \xi_i \geq 0 \tag{2.5}$$

Pri tome ξ_i predstavlja nenegativnu varijablu koja mjeri razinu pogrešne klasifikacije uzorka x_i. Novo ograničenje dopušta marginu koja je manja od 1, i sadrži pogrešku $C\xi_i$ za svaku točku koja pada unutar prave strane margine razdvajajuće hiperravnine (npr. kada je $0 < \xi_i \leq 1$), ili kada pada s krive strane razdvajajuće hiperravnine (npr. kada je $\xi_i > 1$). Parametar C kontrolira relativnu težinu između toga da margina bude dovoljno mala i da se pobrine da većina točaka ima funkcionalnu marginu koja je barem 1. Stoga se minimizira ukupna suma greške ξ_i preko svih i. Naš novi problem optimizacije postaje:

$$\begin{aligned} \min_w \quad & \frac{1}{2}||w||^2 + C\sum \xi_i \\ t.d. \quad & y_i \times (x_i^T \times w + b) \geq 1 - \xi_i \quad \xi_i \geq 0 \\ & \xi_i \geq 0 \end{aligned} \tag{2.6}$$

Taj problem se može napisati i kao jedna minimizacija bez ograničenja:

$$\min_w \quad \frac{1}{2}||w||^2 + C\sum_i \max\left(0, \quad 1 - y_i\left(w^T x_i + b\right)\right) \tag{2.7}$$

Svaka točka x_i spada u jednu od 3 kategorije. Točka ili leži iza margine $y_i\left(w^T x_i + b\right) > 1$, u kojem slučaju ne doprinosi pogrešci u 2.7. Može biti direktno na margini $y_i\left(w^T x_i + b\right) = 1$, u kojem slučaju ne doprinosi pogrešci direktno, ali sudjeluje u optimizaciji kao potporni vektor, slično kao i kod strogo definirane margine. Ako se točka nalazi unutar margine dodajemo pogrešku koja je proprocionalna za onoliko koliko svaka točka prelazi stroga ograničenja. Slobodan parametar C u nijansama kontrolira minimizaciju norme w (što je ekvivalentno maksimiziranju margine) i zadovoljava ograničenje margine za svaku danu točku.

2.6. Nasumične šume (Random Forest)

Nasumična šuma (*Random Forest*) je klasifikacijski algoritam koji su razvili Leo Breiman i Adele Cutler [8]. Osnovna ideja algoritma je korištenje mnoštva klasifikatora umjesto samo jednoga. Prilikom klasifikacije podataka, svaki klasifikator (stablo) donosi zasebnu odluku o klasi (stabla glasaju za klasu). Nakon što sva stabla glasaju, podatak će biti klasificiran klasom koja dobije najviše glasova. Kako bi uopće mogli pričati o tome kako se gradi *Random Forest* trebalo bi reći i nešto o *stablu odlučivanja* (engl. *clasiffication tree*). *Stablo odlučivanja* jest klasifikacijski algoritam u formi stablaste strukture (slika 2.6), u kojoj se razlikuju dva tipa čvorova povezanih granama:

krajnji čvor (engl. leaf node) kojim završava određena grana stabla. Krajnji čvorovi definiraju klasu kojoj pripadaju primjeri koji zadovoljavaju uvjete na toj grani stabla,

čvor odluke (engl. decision node) koji definira određeni uvjet u obliku vrijednosti određenog atributa (varijable), iz kojeg izlaze grane koje odgovaraju određenim vrijednostima tog atributa.

Stablo odlučivanja može se koristiti za klasifikaciju uzorka tako da se krene od prvog čvora odlučivanja u korijenu stabla i kreće po onim granama stabla koja sa svojim vrijednostima zadovoljavaju sve uvjete do krajnjeg čvora koji klasificira uzorak u jednu od postojećih klasa problema.

Osnovni preduvjeti za korištenje tehnike stabla odlučivanja su:

Opis u obliku parova vrijednosti atributa. Podaci o uzorku moraju biti opisani u obliku konačnog broja atributa.

Prethodno definiran konačan broj klasa (vrijednosti ciljnog atributa). Kategorije kojima uzorci pripadaju moraju biti definirane unaprijed i treba ih biti konačan broj.

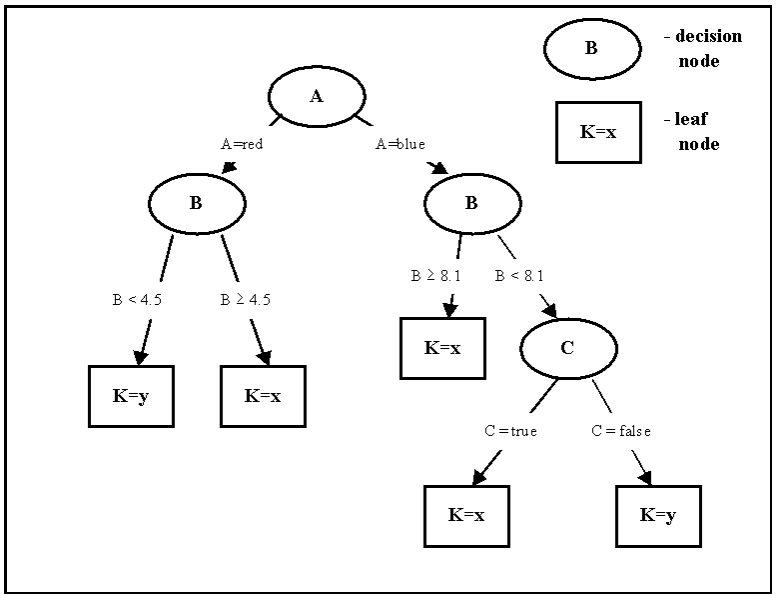
Klase moraju biti diskretne. Svaki uzorak mora pripadati samo jednoj od postojećih klasa, kojih mora biti znatno manje nego broja uzoraka.

Značajan broj uzoraka. Obično je poželjno da u skupu uzoraka za generiranje stabla odlučivanja postoji barem nekoliko stotina uzoraka.

Kriterij kvalitete u algoritmu *stabla odlučivanja* vezan je uz selekciju atributa koji će poslužiti kao kriterij za razdvajanje primjera u određenom čvoru odlučivanja stabla. Cilj je odabrati atribut koji je najupotrebljiviji s obzirom na osnovni cilj, klasifikaciju

uzorka. Dobra kvantitativna mjera vrijednosti atributa u tom smislu je statistička vrijednost nazvana informacijski dobitak (engl. information gain), kojom se mjeri koliko dobro dani atribut razdvaja primjere. Ova se mjera koristi da bi se odabrao najbolji kandidat (atribut) u svakom novom koraku stvaranja *stabla odlučivanja*.

Primjer jednostavnog *stabla odlučivanja* možemo vidjeti na slici 2.6.



Slika 2.6: Primjer jednostavnog *stabla odlučivanja* [6]

Nasumična šuma se sastoji od mnogo *stabala odlučivanja*. Kada bismo željeli klasificirati neki uzorak, pustimo ga niz stablo, svako stablo zasebno donosi odluku o klasi (kažemo da stabla glasaju za klasu), a uzorak će biti klasificiran klasom koja dobije najviše glasova. Konstrukcija nasumične šume se radi na sljedeći način:

- neka je n broj uzoraka, k broj stabala i m zadani parametar
- za svako od k stabala:
  - odaberemo na slučajan način n primjera za učenje s preklapanjem
  - konstruiramo stablo tako da na svakom čvoru odaberemo m atributa i radimo najbolju podjelu po tih m atributa
  - stablo raste do maksimalne veličine – nema rezanja

Greška nasumične šume ovisi o koreliranosti između stabala (povećanje koreliranosti povećava grešku) te jačini svakog pojedinog stabla u šumi (stablo s niskim stupnjem pogreške ima veliku jačinu). Povećanje jačine svakog pojedinog stabla smanjuje se greška cijele šume. Smanjenjem varijable m smanjuje se koreliranost i jakost, a povećavanjem varijable m se one povećavaju. Negdje na sredini se nalazi optimalan

interval za m (obično dosta širok). To je jedini podesivi parametar (osim broja stabla) na koji je nasumična šuma osjetljiva.

3. Skup slika FM2

Za ispitivanje implementiranog postupka klasifikacije slike u skupove oznaka korišten je skup slika prometnih scena FM2 [17]. Skup sadrži 6237 slika. Slike su snimljene u vožnji po hrvatskim cestama, iz vozačeve perspektive. Veći dio slika je snimljen na autocesti od Zagreba do Zadra. Na njima imamo razne scene na koje možemo naići na tom putu. Skup FM2 je inicijalno imao slike označene tako da je svakoj slici bila pridružena jedna oznaka, te je bilo osam klasa. Ukupan broj oznaka je bio jednak broju slika, a to je 6237. Podjelu oznaka možemo vidjeti u tablici 3.1.

Oznaka	Broj slika
autocesta	4337
cesta	516
tunel	601
izlaz iz tunela	64
naselje	464
nadvožnjak	86
naplatna kućica	75
promet	94

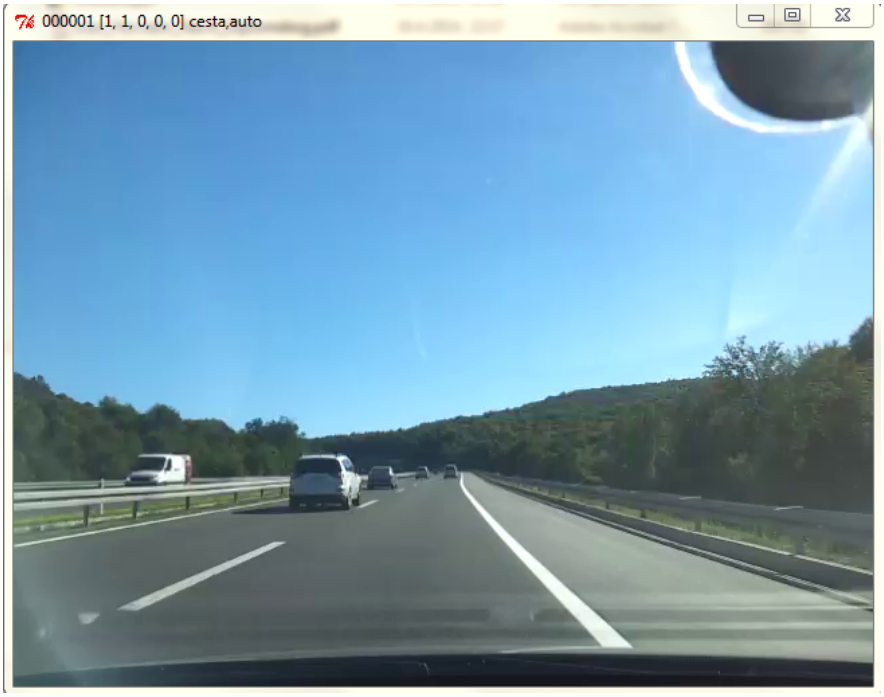
Tablica 3.1: Inicijalni broj oznaka po klasi u skupu FM2

3.1. Označavanje slika

Izvorni skup FM2 ima pridruženu jednu oznaku po slici, stoga je bilo potrebno i označiti svaku sliku različitim oznakama koje će predstavljati što se na slici nalazi. Slike su označavane tako da je svakoj pridružena jedna ili više oznaka iz skupa od 5 oznaka od kojih svaka predstavlja objekt ili pojavu na slici. Tih 5 oznaka su *cesta*, *auto*, *nadvožnjak*, *tunel* i *naselje*. Kod označavanja koristimo konvenciju da svaku sliku označimo nizom od 5 brojeva, pri čemu svaki broj može biti 0 ili 1, te označava prisutnost ili

odsutstvo određene oznake na slici. Primjer označavanja svake slike je npr. 1, 0, 1, 0, 0 što bi značilo da je na slici cesta i nadvožnjak. Neke kombinacije nisu moguće, tj. ako smo u tunelu, to ne klasificiramo kao i cestu, tako da nikada se neće dogoditi kombinacija 1, 0, 0, 1, 0, tj. da je na slici i cesta i tunel.

Za označavanje oznaka je stvoreno programsko rješenje *labeler.py* koji otvara sve slike sa ekstenzijom .png u zadanoj mapi, jednu po jednu, te ih možemo cirkularno izmjenjivati korištenjem strelica na tipkovnici.



Slika 3.1: Prikaz jedne slike pomoću labeler.py

Na slici 3.1 u gornjem lijevom kutu, možemo vidjeti broj slike, formata 000001, u uglatim zagradama vidimo jedinice i nule, koje predstavljaju da li se pojedina oznaka (objekt) nalazi na slici ili ne. Desno od toga je popis oznaka trenutno pridruženih slici. Postavljanje 1 ili 0 se vrši pritiskom na numeričke tipke 1-5, zbog toga što imamo 5 oznaka, a svaki broj predstavlja jednu oznaku. Kada postavimo broj 1 za određenu oznaku na slici se pojavljuje i ime te oznake kako bismo mogli vidjeti što se nalazi na slici. Prva oznaka je *cesta*, druga je *auto*, treća je *nadvožnjak*, četvrta je *tunel* i peta je *naselje*. Trenutna slika je prikazana na ekranu, a pritiskom tipke Esc program se prekida i u komandnoj liniji ispisuje koliko slika je još ostalo neoznačeno, kako bismo mogli znati nastaviti sa radom. Označavanje slika ovim programom trajalo je otprilike 3 sata.

S obzirom da je skup slika FM2 velik nekih objekata na slici ima i više nego do-

voljno za dobru klasifikaciju. Sveukupni broj oznaka po slikama označenih sa *labeler.py* programskim rješenjem možemo vidjeti u tablici 3.2.

Broj labela po klasi				
cesta	auto	nadvožnjak	tunel	naselje
5238	2415	417	725	346

Tablica 3.2: Broj oznaka za svaku pojedinu oznaku u sveukupnom skupu slika

Primjer slika sa jednom i više oznaka možemo vidjeti na slici 3.2. Slike od (a) do (d) možemo označiti jednom oznakom. Slike 3.2 (a) i (b) su primjeri slika s oznakom *cesta*, (c) sa oznakom *tunel* i (d) sa oznakom *nadvožnjaka*. Slike od (e) do (g) bi mogli označiti jednom oznakom ali to ne bi bilo jednoznačno. U mom slučaju te slike će imati više oznaka i to, slici (e) i (g) će biti pridružene oznake *ceste*, *auta* i *nadvožnjaka*, slici (f) će biti pridružene oznake *auto* i *tunel*, a slici (h) će biti pridružene oznake *auta* i *naselja*.

(a) cesta **(b)** cesta **(c)** tunel

(d) nadvožnjak **(e)** cesta, nadvožnjak i auto **(f)** tunel i auto

(g) auto i nadvožnjak **(h)** auto i tunel

Slika 3.2: Primjer slika prometnih scena FM2 skupa

Za razliku od originalne podjele FM2 ukupni broj oznaka u našem slučaju je 9141, što je puno više, ali ne i neočekivano, jer za svaku sliku kojoj je prije bila pridružena

jedna oznaka sada je moguće da će ih biti više. Označavanje je bilo subjektivno, pa da je netko drugi označavao možda i ne bi bile identične oznake za svaku sliku. Auto i nadvožnjak sam označavao na temelju udaljenosti, tj. ako su auto ili nadvožnjak predaleko onda im nisam pridružio oznaku. Oznaku naselje sam pridruživao slikama koje su za mene imale dovoljno objekata tipa zgrada, zidova. Tunele sam označavao osim kada se auto fizički nalazio u tunelu i kada je ulaz u tunel bio dovoljno blizu. Za oznake *cesta*, *naselje* i *tunel* sam još i uzeo kriterij da pridruživanje jedne od tih oznaka isključuje mogućnost pridruživanja ostale dvije na istoj slici. Npr. ako je nešto cesta, ne može biti naselje ni tunel. Primjer nekih kriterija za označavanje možemo vidjeti na slici 3.3.

(a) Primjer slike gdje je nadvožnjak predaleko i ne pridružujemo mu oznaku nadvožnjaka

(b) Primjer slike gdje je nadvožnjak dovoljno blizu i pridružujemo mu oznaku nadvožnjaka

(c) Primjer slike gdje je auto dovoljno blizu te slici pridružujemo oznaku auto

(d) Primjer slike gdje je auto predaleko i ne pridružujemo mu oznaku auto

(e) Primjer slike gdje je naselje i pridružujemo mu oznaku

(f) Primjer slike gdje ne bi pridružili oznaku naselja

Slika 3.3: Primjer označavanja slika

4. Programska izvedba i vanjske biblioteke

Za ostvarenje programskog rješenja ovog rada korišten je Python. Python je programski jezik visoke razine, podržava objektno orijentiranu, imperativnu i funkcijsku paradigmu, a jedna od glavnih karakteristika mu je mogućnost velike ekspresivnosti unutar malo linija koda. Također, odlika mu je i veoma lako čitljiv kod, koji izgledom može podsjećati na neki pseudokod, a ne konkretan kod nekog programskog jezika. Za Python postoji veliki broj besplatno dostupnih biblioteka koje sadrže funkcije za obradu i rad sa slikama, što je i bio jedan od glavnih razloga za odabir Pythona kao jezika za ostvarenje programskog rješenja.

4.1. Korištene biblioteke

Za izradu programskog rješenja, korištena je biblioteka *Scikit-learn* [15] otvorenog koda koja se temelji na bibliotekama *NumPy*, *SciPy* [7] i *matplotlib*. Da bi biblioteka uopće funkcionirala treba prvo instalirati sve ostale biblioteke o kojima *Scikit-learn* ovisi. Prvo i osnovno što je potrebno je Python, jer je to biblioteka striktno napravljena za njega, i to Python 2.6 ili noviji. Od dodatnih bibilioteka potrebane su *NumPy* 1.3 ili više, *SciPy* 0.7 ili više, *setuptools* i *C/C++* prevodilac. U kratko vrijeme *scikit* biblioteka se naglo razvila i sada sadrži mnoge stvari koje olakšavaju rad sa podatcima kao što su *train_test_split*, razne funkcije za više vrsta klasifikatora, unakrsnu validaciju. Svaka od funkcija je detaljno objašnjena na službenim stranicama *Scikita* i uz primjere i ilustracije lako se protumači o čemu se radi.

Važno je pri programiranju voditi računa da većina podataka mora biti *numpy.array*. Ako se na to ne pazi, lako se može pogriješiti i dobiti greške, ili još gore netočne rezultate, a da se to ni ne primijeti.

Python Imaging Library (PIL) je besplatno dostupna biblioteka namijenjena za Python koja pruža metode za obradu i rad sa slikama. U okviru ovoga rada korištene su

metode za učitavanje slike te pretvorbu slika koje su u boji u slike koje su u tonovima sive. *NumPy* je biblioteka otvorenog koda namijenjena za Python koja sadrži pregršt metoda za rad s brojevima u matričnom obliku te implementacije raznih matematičkih funkcija. *SciPy* je, također kao i *NumPy*, biblioteka otvorenog koda namijenjena za Python. Sadrži veliki broj matematičkih funkcija poput funkcija za deriviranje, integriranje, Fourierove transformacije, grupiranje podataka, itd. *SciPy* koristi *Python Imaging Library* i *NumPy* biblioteke za funkcije koje nudi. *VLFeat* [21] je biblioteka otvorenog koda koja sadrži implementacije popularnih algoritama koji se koriste u računalnom vidu kao što su *SIFT*, algoritam k-srednjih vrijednosti, *SVM* i drugi. Biblioteka je zbog efikasnosti napisana u programskom jeziku C te pruža sučelje za rad s MATLAB-om. Ograničenu funkcionalnost biblioteke moguće je dobiti i preko naredbenog retka operacijskog sustava što će biti i korišteno u nastavku.

4.2. SIFT značajke

S obzirom da sam koristio program [14] ukratko ću objasniti bitne dijelove koda kojeg sam koristio i što on radi. Također ću detaljnije opisati onaj dio programa koji je bio izmijenjen i podešen da odgovara ostatku mog programskog rješenja. Dodatni podatci i pojašnjenja dostupni su u tekstu završnog rada [14].

Dakle, za izračun SIFT značajki, kao što je i rečeno korištena je biblioteka *VLFeat* koja pruža sučelje za *MATLAB*, ali ne i *Python*. Međutim do ograničene funkcionalnosti se može doći preko naredbenog retka, što je ovdje i korišteno. *VLFeat* očekuje da slika bude u PGM formatu. Prikaz u PGM formatu znači da je slika prikazana u nijansama sive boje. Svaki element slike prikazan je jednim brojem koji predstavlja intenzitet sive razine u pojedinom elementu. Kako bi se došlo do naredbenog retka iz Pythona iskorišten je standardni modul *os*. Modul sadrži metodu *os.system* (naredba za operacijski sustav) koja kao argument prima znakovni niz koji predstavlja naredbu koju je potrebno izvršiti. Naredba otvara privremenu ljusku te nakon odrade zadanog posla tu ljusku zatvara.

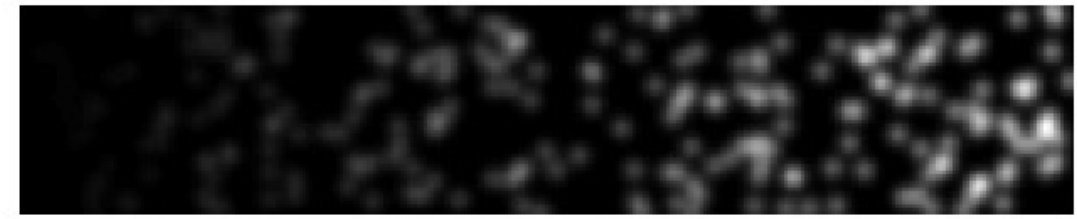
Sama naredba izgleda *sift* slika -**output =** datoteka **parametri**

Slika označava apsolutni put do slike koja se nalazi na disku.

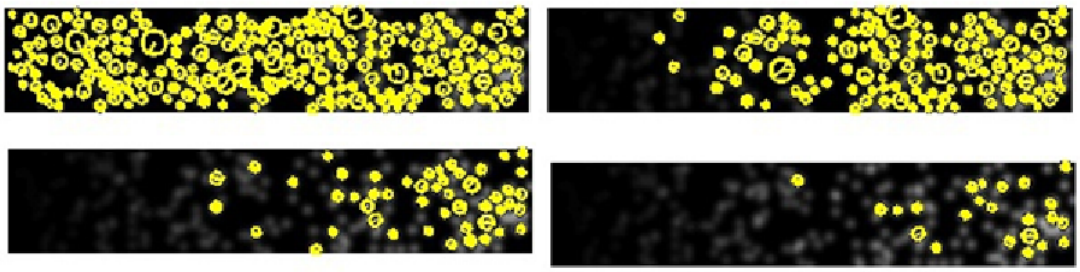
Datoteka označava apsolutni put do datoteke za pohranu rezultata izračuna SIFT značajki. Ukoliko se ne navede izlazna datoteka rezultat izračuna će se ispisati na standardni izlaz.

Parametri predstavljaju mogućnost kalibracije SIFT algoritma tako da se postave neki od parametara algoritma. U našem konkretnom slučaju mijenjali su se parame-

tri *edge-thresh* i *peak-thresh*, koji su pragovi koji najviše utječu na generiranje *SIFT* značajki. Prag *peak-thresh* filtrira točke interesa čija je vrijednost premala (u apsolutnoj vrijednosti) u prostoru razlike Gaussiana, tj. odbacuju se detektirane točke koje imaju premali kontrast. Što je veći prag *peak-thresh* dobiti ćemo manji broj točaka. Primjer jedne slike možemo vidjeti na slici 4.1, a rezultantne SIFT značajke dobivene mijenjanjem praga na slici 4.2.



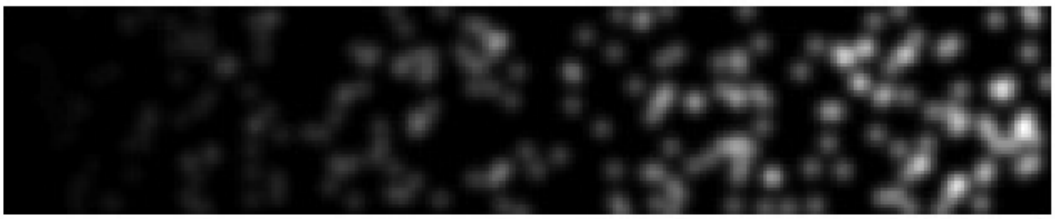
Slika 4.1: Testna slika za određivanje praga kontrasta u prostoru razlike Gaussiana *peak-thresh* [21]



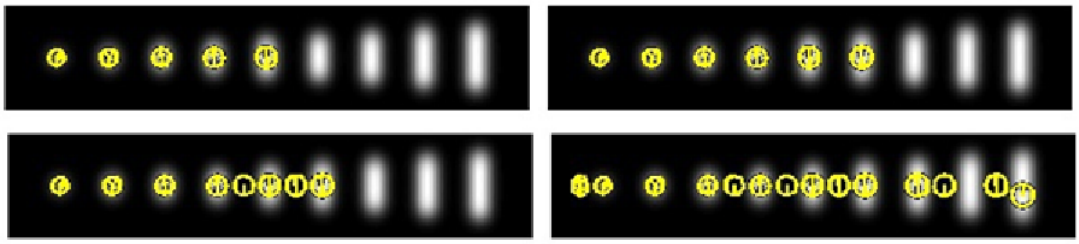
Slika 4.2: Lokacije detektiranih SIFT značajki za parametar praga kontrasta u prostoru razlike Gaussiana *peak-thresh* počevši do gornje lijeve slike (*peak_thresh* = 0, 10, 20, 30)

Prag *edge-thresh* eliminira točke interesa koje imaju jak rubni odaziv samo u jednom smjeru. Npr. rub stola ima jak rubni odaziv u samo jednom smjeru i on bi bio odbačen kao točka, a na primjer kut stola ima jak rubni odaziv u oba smjera i time bi ta točka bila zadržana. Za razliku od praga *peak-thresh* prag *edge-thresh* djeluje suprotno, tj. ako imamo veći prag, naći ćemo više točaka. Cilj nam je naći što više jedinstvenih točaka. Primjer testne slike možemo vidjeti na slici 4.3, a utjecaj mijenjanja praga *edge-thresh* na slici 4.4.

U korištenom programskom rješenju *SIFT* značajke se generiraju u istoj mapi kao i izvorne slike i to na način da ime dokumenta bude isto kao i kod slike, samo sa drugom ekstenzijom. Na primjer, ako se slika zove 000001.png, stvorit će se *SIFT* dokument pod nazivom 000001.sift u kojem će se nalaziti *SIFT* opisnici te slike. S obzirom da



Slika 4.3: Testna slika za određivanje praga rubnog odaziva u prostoru razlike Gaussiana *edge-thresh* [21]



Slika 4.4: Detektirane točke za parametar praga kontrasta u prostoru razlike Gaussiana *edge-thresh* počevši do gornje lijeve slike (*edge_thresh* = 3.5, 5, 7.5, 10)

stvaranje *SIFT* opisnika za svaku sliku traje dosta dugo korisno ih je spremiti kao dokumente, kako bi im se brže moglo pristupiti u bilo kojoj fazi eksperimenata, umjesto da se svaki puta ispočetka generiraju.

Format zapisa u datoteke prikazan je na slici 4.5. Svaki redak sadrži potpuni opis pojedine značajke. Prva dva broja određuju x i y koordinatu značajke na slici. Treći broj određuje mjerilo u kojem je značajka pronađena, dok četvrti broj određuje orijentaciju značajke izraženu u radijanima. Preostalih 128 brojeva predstavlja SIFT opisni vektor značajke (na slici nije prikazano svih 128 komponenata). Slike pragova *edge-thresh* i *peak-thresh thresh* su preuzete od [21].

```
1  626.349 63.8189 1.01768 3.9979 51 12 0 0 0 9 11 22 144 75 0 0 0 10 5 6 144 53 0 0 1 1
2  621.455 69.4067 1.07426 4.23915 102 3 0 0 0 0 24 139 65 3 0 0 0 45 47 82 37 0 0 0 0 4
3  619.999 75.7001 0.948795 3.77092 40 26 1 1 9 7 41 35 47 68 4 4 28 13 15 21 86 132 8 0
4  601.452 83.2073 0.905635 5.15289 6 0 0 0 0 18 122 96 99 18 0 0 0 21 76 45 93 76 0 0 0
5  604.615 83.3625 0.939035 4.27801 48 14 11 7 0 0 0 6 42 104 31 4 0 0 0 3 46 134 67 0 0
6  563.023 84.0423 0.899123 4.77854 13 32 61 76 8 0 0 9 21 31 77 16 0 1 8 20 103 24 36 8
7  563.023 84.0423 0.899123 6.2815 0 0 0 0 0 0 36 127 24 63 0 0 0 0 4 63 127 41 0 0 0 0 0 (
8  635.814 84.2296 0.884703 0.952376 14 1 0 0 118 134 60 15 98 22 3 3 8 45 46 31 17 8 4 1
9  594.595 84.5449 1.1089 4.26216 21 20 4 10 25 7 11 35 127 14 0 0 7 2 5 127 86 16 0 0 0
10 543.453 86.0988 0.963139 5.08215 0 0 0 0 19 114 65 5 75 2 0 0 6 65 54 62 47 1 0 0 0 3
11 605.148 86.3301 0.936091 3.70223 39 15 3 0 0 0 2 19 42 70 10 0 0 2 3 6 21 53 17 17 16
12 563.236 87.0957 1.11447 4.77213 2 3 32 118 19 1 15 77 13 19 43 118 18 1 3 23 30 28 55
13 593.224 87.3372 0.902472 4.20885 21 20 2 2 2 11 23 34 57 10 0 1 3 4 15 69 127 62 0 0 (
14 607.491 87.6034 1.02757 3.63011 4 4 9 10 11 0 0 2 59 24 4 0 2 1 3 32 40 90 15 0 0 6 8
```

Slika 4.5: Format zapisa SIFT opisnika u datoteci (zbog duljine SIFT opisnika prikazan je samo jedan dio)

Za potrebe kategorizacije slike potreban je samo 128-dimenzionalni opisni vektor

koji će se koristiti pri stvaranju slikovnog rječnika. Koordinate, skalu i orijentaciju ne koristimo pri daljnjem izračunu, pa njih ni ne uzimamo u obzir prilikom učitavanja SIFT značajki iz datoteka. Za potrebe izračuna i dohvata *SIFT* opisnika u okviru programskog rješenja napisan je modul *sift*. Taj modul sadrži tri metode od kojih se dvije koriste za obradu slike dok se jedna koristi za dohvat opisnika s diska. Metoda *sift.processAllImages* kao argument prima niz mapa u kojima se nalaze slike za koje je potrebno izračunati *SIFT* opisnike. Povratna vrijednost ove metode je lista koja sadrži apsolutni put do svake datoteke s opisnicima koja je stvorena tijekom obrade. Ova metoda oslanja se na metodu *sift.processImage* kako bi obradila pojedinu sliku. Metoda *sift.readFeaturesFromFile* kao argument prima put do datoteke koja sadrži *SIFT* opisnike te ih učitava i vraća kao listu s dva člana. Prvi član te liste je lista koja sadrži x i y koordinate, mjerilo i orijentaciju značajke. Drugi član liste je lista koja sadrži 128 elemenata, odnosno *SIFT* opisni vektor. Za razliku od originalne funkcije iz [14], *sift.ReadFeaturesFromFile* je malo izmijenjena i prilagođena tako da se sada može kao dodatni opcionalni parametar upisati i broj koji bi tada predstavljao koliko značajki želimo da nam funkcija *readFeaturesFromFile* odabere iz pojedinog dokumenta i to na način da su *SIFT* opisnici iz skupa svih opisnika tog dokumenta uzeti nasumično, radi bolje općenitosti. Ako želimo sve značajke taj dio ostavimo praznim, jer je to opcionalan parametar.

4.3. Slikovni rječnik

Nakon što smo pronašli *SIFT* značajke za svaku sliku u skupu, skup slika potrebno je razdvojiti na skupove za učenje i testiranje. To je postignuto funkcijom

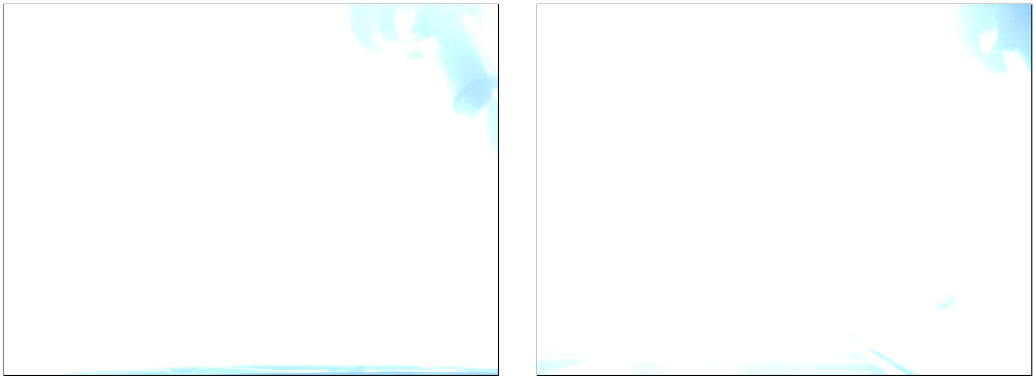
sklearn.cross_validation.train_test_split(arrays,test_size,random_state)

iz biblioteke *scikit-learn* koja za ulazne parametre prima sekvencu polja ili matrica istog oblika (*arrays*), broj uzoraka za učenje i testiranje u rasponu od 0.0 do 1.0 koji predstavlja postotak (*test_size*), te parametar algoritma nasumičnog odabira (*random_state*). Ulazni skup dijeli se na 2 manja, od kojih prvi sadrži veći postotak skupa (skup za učenje), a drugi manji postotak skupa (skup za testiranje). Nakon što smo dobili svoj skup za učenje i testiranje sada napokon sa *SIFT* značajkama možemo početi raditi rječnik. *SIFT* opisnike iz slika za učenje potrebno je grupirati u slikovne riječi pomoću algoritma k-srednjih vrijednosti. To je ostvareno pomoću biblioteke *SciPy*, odnosno njenog *scipy.cluster.vq* potpaketa. Grupiranje pomoću *k*-srednjih vrijednosti radi se pomoću metode *scipy.cluster.vq.kmeans2* koja kao argumente prima skup ulaznih podataka, broj centroida, te može primiti i željeni broj iteracije algoritma (svaki

puta se ponovno nasumice određuju početne srednje vrijednosti centroida). Povratna vrijednost metode biti će skup srednjih vrijednosti centroida i vrijednost funkcije pogreške. Vrijednost funkcije pogreške nije značajna za daljnji postupak pa se slobodno može odbaciti. Dobivene srednje vrijednosti centroida zapravo predstavljaju slikovne riječi. Skup svih centroida čini slikovni rječnikom. Nakon što imamo slikovne riječi, moguće je za svaku sliku izgraditi histogram ponavljanja slikovnih riječi. Slikovni rječnik izveden je kao poseban razred Vocabulary unutar modula vocabulary. Razred Vocabulary sadrži dvije metode, *createVocabulary* i *histogram*. Metoda *createVocabulary* kao argument prima skup apsolutnih puteva do datoteka koje sadrže *SIFT* opisne vektore (izlazna vrijednost metode *sift.processAllImages*) te željeni broj centroida (slikovnih riječi). Ova funkcija je za razliku od originalne implementacije [14] promijenjena kako bi mogli raditi rječnik od određenog broja nasumično izabranih *SIFT* opisnika, kako bi spriječili predugo i vremenski zahtjevno obavljanje *kmeans2* operacije za prevelik broj *SIFT* opisnika. Metoda stvara slikovni rječnik kojega interno pohranjuje. Metoda *histogram* služi za stvaranje histograma slikovnih riječi. Kao ulazni argument prima skup *SIFT* opisnih vektora neke slike. Prije obrade ulaznih podataka stvara se vektor od n komponenata gdje su sve vrijednosti početno postavljene na 0; taj vektor predstavlja histogram slikovnih riječi. Parametar n odgovara broju slikovnih riječi u rječniku. Za svaki opisni vektor iz ulaznog skupa podataka određuje se kojoj slikovnoj riječi pripada te se komponenta histograma koja predstavlja tu slikovnu riječ uvećava za jedan. Metoda vraća stvoreni histogram. Stvoreni histogrami koristiti će se kao ulazni podaci u razmatrane klasifikatore.

4.4. Klasifikacija slike

Iz skripte *SVMCreator* koja je dio programskog rješenja iz [14] jedine dvije metode koje koristim u programskom rješenju su *createHistogramList* i *createVisualWordHistogramListFromFiles*. Funkcija *createVisualWordHistogramListFromFiles* je u odnosu na orginalni oblik promijenjena tako da joj je dodan još jedan dodatan parametar, a to je argument "removed". On predstavlja listu koja će sadržavati indekse slika koje su izbačene iz skupa za učenje jer nisu imale niti jednu SIFT značajku. Te indekse dalje koristimo kako bi iz liste oznaka izbacili one oznake koje odgovaraju indeksu slika koje smo izbacili. Primjer slika u kojima se ne može naći niti jedna *SIFT* značajka možemo vidjeti na slici 4.6. S obzirom na to kakve su slike, i mijenjanjem parametara izračuna *SIFT* značajki teško je moguće dobiti drugačiji rezultat na ovim slikama zato što je nemoguće naći bilo kakve interesne točke.



Slika 4.6: Slike bez *SIFT* značajki, 004790.png (lijevo) i 005675.png(desno)

Metoda *createVisualWordHistogramListFromFiles* iz svake datoteke sa SIFT značajkama pojedine slike radi histogram slikovnih riječi pojedine slike i stavlja ih u listu histograma. Upravo ti histogrami i labele se koriste kao ulaz u klasifikator. Za klasificiranje sam u programskom rješenju koristio klasifikatore *sklearn.svm.SVC* (Support Vector Classification) i *sklearn.ensemble.RandomForestClassifier* (nasumične šume) iz biblioteke *Scikit-learn*. Oba klasifikatora je moguće koristiti u klasifikaciji sa skupom oznaka, a to je ostvareno preko funkcije *OneVsRestClassifier* (jedan protiv svih). *OneVsRestClassifier* svodi problem klasifikacije sa skupom oznaka na binarnu klasifikaciju, tj. *BR* (Binary relevance) metodu (opisane u poglavlju 2). Koristi se na način da se unutar njega ugnježđuje klasifikator nad kojim želimo koristiti "*jedan-protiv-svih*" strategiju, npr.:

multi = OneVsRestClassifier(LinearSVC()).

Klasa *RandomForestClassifier* je klasifikator koji se temelji na stablima i njihovom odlučivanju kako bi se klasificirali podaci. Koristim ga u obliku:

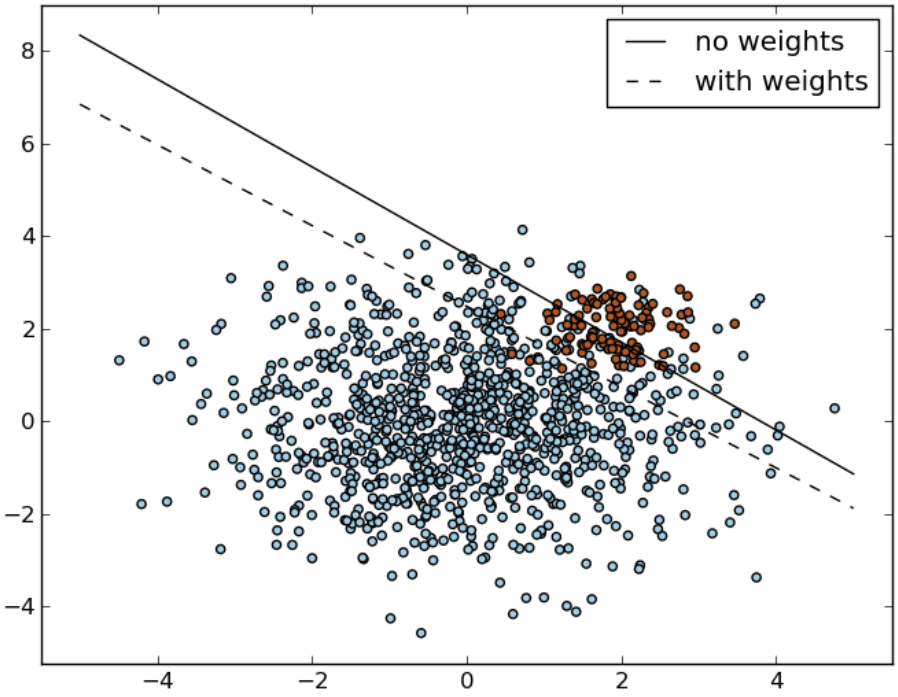
RF = RandomForestClassifier(n_estimators)

gdje je parametar *n_estimators* broj stabala. Klasa *LinearSVC* (Linear Support Vector Classification) je linearni *SVM* klasifikator. Koristio sam ju u obliku

SVM = LinearSVC(class_weight, C)

Parametar C, čija je početna vrijednost 1, mijenjanjem utječe na margine decizijske funkcije, povećavajući ili smanjujući ju, i time dopušta da se neke od točaka koje su blizu margine drugačije klasificiraju. Parametar *class_weight*, koji je opcionalni parametar, utječe na parametar C tako da mijenja parametar C klase i u oblik *class_weight[i]*C*. To predstavlja težinu, tj. kada imamo puno manji broj nekih oznaka, želimo da one imaju veći utjecaj na postavljanje decizijske ravnine. Ako je *class_weight = auto* tada se broj pojavljivanja pojedine oznake koristi za automatsko namještanje težine tako da se težina određuje kao broj čija je vrijednost u obrnuto

proporcionalnom odnosu sa frekvencijom pojavljivanja svake oznake. Primjer odluke decizijskih ravnina sa i bez korištenja *class_weight* možemo vidjeti na slici 4.7. Plavi kružići predstavljaju uzorke jedne klase, a crveni uzorke druge klase.



Slika 4.7: Primjer podjele plavih i crvenih uzoraka bez korištenja težine za uzorke i sa korištenjem težina [15]

5. Eksperimentalni rezultati

Eksperimenti u ovom radu su rađeni nad skupom slika *FM2* [17] koji sadrži slike raznih prometnih scena. Slike su označene u skupove oznaka (poglavlje 3) i eksperimente smo izvodili s više klasifikatora i različito generiranim SIFT značajkama. Cilj je bio vidjeti kakvi se rezultati dobiju klasifikacijom u skupove oznaka. U prvom dijelu eksperimenata ukupni skup slika je bio podijeljen nasumično na skup za učenje koji je sadržavao 70% ukupnog skupa i skup za testiranje koji je sadržio 30% ukupnog skupa slika. Također se od svih *SIFT* značajki pojedine slike koristilo 30 nasumično izabranih za svaku sliku kako bi zadržali općenitost, a smanjili broj operacija koje bi se trebale obaviti tijekom cijelog postupka. Taj broj je moguće mijenjati prije pokretanja programskog rješenja. U svrhu ispitivanja stvoreno je više slikovnih rječnika, te za svaki pojedini rječnik više različitih klasifikatora. Za svaki klasifikator probano je više različitih parametara te su bili međusobno uspoređeni da bismo vidjeli koje postavke i koji klasifikator je efikasniji. Postojat će mala odstupanja u rezultatima svakog pojedinog ispitivanja jer se kod izrade rječnika inicijalizacija algoritma *k*-srednjih vrijednosti odabire nasumično, pa podjele kod izrade slikovnog rječnika neće uvijek biti identične. Kako bi algoritam *k*-srednjih vrijednosti obavio svoj posao potrebno mu je dati sve podatke na raspolaganje. Sve se to mora držati u radnoj memoriji, pa se kod rada s velikim skupom podataka može dogoditi da računalo nema dovoljno memorije.

Stvaranje rječnika, ovisno o veličini rječnika trajalo je dosta dugo. Stvaranje rječnika od 1000 riječi na računalu sa procesorom *AMD Phenom II X4 840* sa radnim taktom od 3.2 GHz trajalo je oko 48 minuta.

Mjerena je preciznost (5.1) i odziv (5.2) te F1 mjera (5.3).

$$\textit{Preciznost} = \frac{T_p}{T_p + F_p} \tag{5.1}$$

$$\textit{Odziv} = \frac{T_p}{T_p + F_n} \tag{5.2}$$

Preciznost označava omjer broja ispravno klasificiranih uzoraka i svih uzoraka koji

su klasificirani kao primjeri te klase, a odziv je omjer broja ispravno klasificiranih uzoraka i ukupnog broja uzoraka koje je trebalo klasificirati kao pozitivne, gdje T_p označava ispravno klasificirane uzorke (True Positive), F_p označava uzorke koji su klasificirani kao ispravni a to nisu (False Positive), a F_n označava uzorke koji su klasificirani kao neispravni a to nisu (False Negative). $F1$ mjera je još jedna mjera s kojom određujemo koliko su eksperimenti bili uspješni ali je stroža od same preciznosti i odziva, ona je u biti harmonička sredina preciznosti i odziva i računa se na sljedeći način:

$$F1 = \frac{2T_p}{2T_p + F_n + F_p} \tag{5.3}$$

5.1. Eksperimenti s ručnim podešavanjem parametara

Za prvi eksperiment koristili smo linearni *SVM* klasifikator sa parametrom C za par vrijednosti ($0.001,\ 0.1,\ 1,\ 100,\ 10000,\ 1 \cdot 10^{10}$). Cilj je bio vidjeti na koji način bi promjenom parametra C utjecali na rezultate. U prvoj grupi eksperimenata je korišten rječnik od 1000 slikovnih riječi. U tablici 5.1 možemo vidjeti da je prepoznavanje cesti iznimno dobro, tunela malo lošije, ali recimo nadvožnjaka iznimno loše. To bi moglo biti zbog toga što su pragovi za izlučivanje značajki previše strogi ili jednostavno zato što u odnosu na veličinu skupa nemamo dovoljno uzoraka nadvožnjaka, također jedan od problema bi mogao biti zato što u skupu za učenje ima preteških primjera nadvožnjaka. Također sam primijetio da se bez obzira na variranje parametra C dobivaju isti rezultati. Kada imamo više tablica, podebljani su oni rezultati koji su najbolji u danoj kategoriji.

	Preciznost	Odziv	F1 mjera	Broj labela po oznaci
Cesta	0.98	0.97	0.98	1597
Auto	0.73	0.55	0.63	719
Nadvožnjak	0.28	0.40	0.33	124
Tunel	0.88	0.89	0.89	196
Naselje	0.70	0.73	0.72	98
Prosjek/Ukupno	0.86	0.82	0.84	2734

Tablica 5.1: Rezultati *SVM* klasifikatora sa različitim parametrom C ($0.001,\ 0.1,\ 1,\ 100,\ 10000,\ 1 \cdot 10^{10}$). Bez obzira na variranje C, dobivaju se isti rezultati

U drugom ispitivanju se koriste *SVM* i *Random Forest* klasifikator sa automatskim postavkama. Od bitnijih parametara izdvajam $C = 1$ za *SVM* i $n = 10$ za *Random Forest* koji označava broj stabala. Iz tablice 5.2 možemo vidjeti kako je *Random Forest* klasifikator malo bolji u raspoznavanju tunela, ali lošiji za ostale oznake, a za raspoznavanje nadvožnjaka značajno lošiji.

Razlika između *SVM-a* i *Random Foresta* nije velika, ali je primjetna, pogotovo kod nadvožnjaka. Tako recimo za oznaku *auto* kod *SVM-a* dobivamo rezultat od $F1 = 0.68$ što i nije tako dobro, a za *Random Forest* taj rezultat je manji i iznosi $F1 = 0.55$. *Random Forest* u drugom ispitivanju jedino daje bolje rezultate za tunel, gdje $F1$ iznosi $F1 = 0.91$ a za *SVM* iznosi $F1 = 0.87$. To je možda zbog premalog broja stabala ili zbog toga što je skup za učenje nadvožnjaka pretežak za naučiti. Promjenom načina generiranja značajki možemo dobiti više značajki, pa možda tako dobijemo bolje rezultate. Također možemo probati promijeniti oznake za slike, tako da olakšamo skup za učenje da se slike mogu lakše klasificirati.

SVM			
	Preciznost	Odziv	F1 mjera
Cesta	0.98	0.97	**0.98**
Auto	0.73	0.63	**0.68**
Nadvožnjak	0.28	0.40	**0.32**
Tunel	0.86	0.88	0.87
Naselje	0.75	0.72	**0.74**
Prosjek/Ukupno	0.87	0.84	0.85
Random Forest			
	Preciznost	Odziv	F1 mjera
Cesta	0.96	0.99	0.97
Auto	0.71	0.45	0.55
Nadvožnjak	1.00	0.03	0.06
Tunel	0.98	0.84	**0.91**
Naselje	0.89	0.42	0.57
Prosjek/Ukupno	0.90	0.77	0.80

Tablica 5.2: Rezultati linearnog *SVM* i *Random Forest* klasifikatora sa parametrima $C = 1$ za *SVM* i $n = 10$ za *Random Forest*

U trećem ispitivanju se koristi linearni *SVM* i *Random Forest* klasifikator ali smo koristili opciju *class_weight*, s obzirom da se na puno slika pojavljuje oznaka cesta,

neke oznake poput nadvožnjaka i naselja ne, uvodimo težinu, kako bi veće značenje imali one oznake kojih manje ima. U tablici 5.3 možemo vidjeti rezultate redom za linearni *SVM*, *SVM* kojemu je omogućena opcija *class_weight* i to na način da se veća težina dodaje onim labelama koje se najmanje pojavljuju, te *Random Forest*. Može se primijetiti da se razlika vidi tek u nijansama.

SVM ($C = 1$)			
	Preciznost	Odziv	F1 mjera
Cesta	0.98	0.97	0.97
Auto	0.66	0.75	**0.70**
Nadvožnjak	0.29	0.40	0.34
Tunel	0.85	0.90	0.87
Naselje	0.69	0.84	0.76
Prosjek/Ukupno	0.85	0.88	0.86
SVM ($C = 1$ i *weight = 'auto'*)			
	Preciznost	Odziv	F1 mjera
Cesta	0.98	0.97	**0.98**
Auto	0.69	0.71	**0.70**
Nadvožnjak	0.30	0.42	**0.35**
Tunel	0.85	0.91	0.88
Naselje	0.73	0.84	**0.78**
Prosjek/Ukupno	0.86	0.87	0.86
Random Forest ($n = 10$)			
	Preciznost	Odziv	F1 mjera
Cesta	0.96	0.99	0.97
Auto	0.67	0.43	0.53
Nadvožnjak	1.00	0.05	0.09
Tunel	0.99	0.86	**0.92**
Naselje	0.82	0.46	0.59
Prosjek/Ukupno	0.88	0.77	0.80

Tablica 5.3: Rezultati linearnog *SVM-a* ($C = 1$), težinskog *SVM-a* ($C = 1$) i *Random Forest* klasifikatora ($n = 10$)

5.1.1. Detaljna usporedba klasifikatora s ručno podešenim parametrima

U četvrtom ispitivanju imamo 6 rezultata koje možemo vidjeti na tablici 5.4. Redom se koristilo bestežinski linearni *SVM* sa parametrom C = 1, bestežinski *SVM* sa parametrom C = 0.0001, težinski *SVM* sa parametrom C = 1, težinski *SVM* sa parametrom C = 0.0001, *Random Forest* klasifikator sa brojem stabala n = 100, *Random Forest* klasifikator sa brojem stabala n = 500. Možemo vidjeti da je težinski *SVM* sa parametrom C = 0.0001 za nijansu bolji od bestežinskog *SVM-a*, a *Random Forest* klasifikator je u potpunosti zakazao što se tiče raspoznavanja nadvožnjaka.

SVM ($C = 1$)

	Preciznost	Odziv	F1 mjera
Cesta	0.97	0.97	0.97
Auto	0.67	0.70	0.68
Nadvožnjak	0.26	0.39	**0.31**
Tunel	0.87	0.89	0.88
Naselje	0.73	0.76	0.74
Prosjek/Ukupno	0.84	0.86	0.85

SVM ($C = 0.0001$)

	Preciznost	Odziv	F1 mjera
Cesta	0.97	0.97	0.97
Auto	0.67	0.70	0.68
Nadvožnjak	0.26	0.39	**0.31**
Tunel	0.87	0.89	0.88
Naselje	0.73	0.76	0.74
Prosjek/Ukupno	0.84	0.86	0.85

SVM ($C = 1$ i *weight = 'auto'*)

	Preciznost	Odziv	F1 mjera
Cesta	0.97	0.97	0.97
Auto	0.72	0.65	0.69
Nadvožnjak	0.26	0.39	**0.31**
Tunel	0.88	0.89	0.89
Naselje	0.74	0.76	**0.75**
Prosjek/Ukupno	0.86	0.85	0.85

SVM ($C = 0.0001$ i *weight = 'auto'*)

	Preciznost	Odziv	F1 mjera
Cesta	0.99	0.97	**0.98**
Auto	0.69	0.75	**0.72**
Nadvožnjak	0.20	0.66	**0.31**
Tunel	0.91	0.89	0.90
Naselje	0.46	0.98	0.63
Prosjek/Ukupno	0.85	0.89	0.86

Random Forest ($n = 100$)

	Preciznost	Odziv	F1 mjera
Cesta	0.96	0.99	**0.98**
Auto	0.79	0.56	0.66
Nadvožnjak	1.00	0.02	0.03
Tunel	0.99	0.86	**0.92**
Naselje	0.90	0.44	0.59
Prosjek/Ukupno	0.92	0.81	0.83

Random Forest ($n = 500$)

	Preciznost	Odziv	F1 mjera
Cesta	0.96	1.00	**0.98**
Auto	0.79	0.56	0.66
Nadvožnjak	1.00	0.02	0.05
Tunel	0.99	0.85	**0.92**
Naselje	0.92	0.48	0.63
Prosjek/Ukupno	0.92	0.81	0.84

Tablica 5.4: Rezultati linearnog *SVM* ($C = 1, 0.0001$), linearnog *SVM-a* s težinama ($C = 1, 0.0001$) i *Random Forest* klasifikatora ($n = 100, 500$) sa rječnikom od 1000 riječi

5.2. Promjene veličine rječnika

Isti eksperiment kao i prijašnji je bio ponovljen nad rječnikom od 5000 riječi i rezultati su skoro identični. Malo su bolji za linearni *SVM* i nadvožnjak ali s obzirom da je stvaranje rječnika od 5000 riječi trajalo 5 puta duže nego rječnika sa 1000 riječi, ne isplati se koristiti tako veliki rječnik i utrošiti tako puno vremena na taj proces. Rezultate možemo vidjeti u tablici 5.5.

SVM ($C = 1$)

	Preciznost	Odziv	F1 mjera
Cesta	0.98	0.98	**0.98**
Auto	0.66	0.69	0.68
Nadvožnjak	0.50	0.44	**0.47**
Tunel	0.90	0.88	0.89
Naselje	0.81	0.77	**0.79**
Prosjek/Ukupno	0.86	0.86	0.86

SVM ($C = 0.0001$)

	Preciznost	Odziv	F1 mjera
Cesta	0.98	0.98	**0.98**
Auto	0.66	0.69	0.68
Nadvožnjak	0.50	0.44	**0.47**
Tunel	0.90	0.88	0.89
Naselje	0.81	0.77	**0.79**
Prosjek/Ukupno	0.86	0.86	0.86

SVM ($C = 1$ i *weight = 'auto'*)

	Preciznost	Odziv	F1 mjera
Cesta	0.98	0.98	**0.98**
Auto	0.66	0.69	0.68
Nadvožnjak	0.49	0.43	0.46
Tunel	0.91	0.88	0.89
Naselje	0.74	0.76	0.75
Prosjek/Ukupno	0.80	0.77	0.78

SVM ($C = 0.0001$ i *weight = 'auto'*)

	Preciznost	Odziv	F1 mjera
Cesta	0.99	0.97	**0.98**
Auto	0.70	0.76	**0.73**
Nadvožnjak	0.24	0.69	0.35
Tunel	0.94	0.88	**0.91**
Naselje	0.46	1.00	0.63
Prosjek/Ukupno	0.86	0.89	0.87

Random Forest ($n = 100$)

	Preciznost	Odziv	F1 mjera
Cesta	0.95	1.00	0.97
Auto	0.77	0.56	0.65
Nadvožnjak	1.00	0.02	0.05
Tunel	0.99	0.85	**0.91**
Naselje	0.93	0.29	0.44
Prosjek/Ukupno	0.91	0.80	0.82

Random Forest ($n = 500$)

	Preciznost	Odziv	F1 mjera
Cesta	0.95	1.00	0.97
Auto	0.79	0.60	0.68
Nadvožnjak	1.00	0.02	0.03
Tunel	0.98	0.85	**0.91**
Naselje	0.90	0.28	0.42
Prosjek/Ukupno	0.91	0.81	0.83

Tablica 5.5: Rezultati linearnog *SVM* ($C = 1, 0.0001$), linearnog *SVM-a* s težinama ($C = 1, 0.0001$) i *Random Forest* klasifikatora ($n = 100, 500$) sa rječnikom od 5000 riječi

Također je eksperiment ponovljen i za rječnik sa manjem brojem riječi, konkretno 500. Iako se rječnik napravi brže i cijeli proces bude prije završen, rezultati su tek malo lošiji te ih možemo vidjeti u tablici 5.6.

SVM ($C = 1$)

	Preciznost	Odziv	F1 mjera
Cesta	0.97	0.97	0.97
Auto	0.62	0.78	0.69
Nadvožnjak	0.30	0.44	**0.36**
Tunel	0.81	0.89	0.84
Naselje	0.63	0.76	0.69
Prosjek/Ukupno	0.83	0.88	0.85

SVM ($C = 0.0001$)

	Preciznost	Odziv	F1 mjera
Cesta	0.97	0.97	0.97
Auto	0.62	0.78	0.69
Nadvožnjak	0.30	0.44	**0.36**
Tunel	0.81	0.89	0.84
Naselje	0.63	0.76	0.69
Prosjek/Ukupno	0.83	0.88	0.85

SVM ($C = 1$ i *weight = 'auto'*)

	Preciznost	Odziv	F1 mjera
Cesta	0.97	0.97	0.97
Auto	0.71	0.62	0.66
Nadvožnjak	0.23	0.54	0.33
Tunel	0.80	0.88	0.84
Naselje	0.67	0.76	**0.71**
Prosjek/Ukupno	0.85	0.84	0.84

SVM ($C = 0.0001$ i *weight = 'auto'*)

	Preciznost	Odziv	F1 mjera
Cesta	0.99	0.97	**0.98**
Auto	0.69	0.74	**0.71**
Nadvožnjak	0.21	0.68	0.32
Tunel	0.90	0.89	0.89
Naselje	0.47	0.99	0.64
Prosjek/Ukupno	0.85	0.89	0.86

Random Forest ($n = 100$)

	Preciznost	Odziv	F1 mjera
Cesta	0.97	0.99	**0.98**
Auto	0.79	0.53	0.64
Nadvožnjak	1.00	0.03	0.06
Tunel	0.99	0.84	**0.91**
Naselje	0.84	0.55	0.67
Prosjek/Ukupno	0.92	0.80	0.83

Random Forest ($n = 500$)

	Preciznost	Odziv	F1 mjera
Cesta	0.97	0.99	**0.98**
Auto	0.81	0.54	0.65
Nadvožnjak	1.00	0.02	0.05
Tunel	0.99	0.85	**0.91**
Naselje	0.89	0.57	0.70
Prosjek/Ukupno	0.93	0.81	0.84

Tablica 5.6: Rezultati linearnog *SVM* ($C = 1, 0.0001$), linearnog *SVM-a* s težinama ($C = 1, 0.0001$) i *Random Forest* klasifikatora ($n = 100, 500$) sa rječnikom od 500 riječi

5.3. Ugniježđena unakrsna validacija

Za sljedeći eksperiment napravljena je ugniježđena unakrsna validacija za *SVM* sa linearnim i *RBF* kernelom i raznim parametrima kako bi mogli vidjeti koja kombinacija nam daje najbolji estimator. To se radilo uz pomoć funkcije

Grid = GridSearchCV(OneVsRestClassifier(SVC()), tuned_parameters, cv=5, n_jobs = 4, verbose = 3)

gdje prvi parametar predstavlja klasifikator nad kojim radimo unakrsnu validaciju, *tuned_parameters* predstavlja listu svih parametara kroz koje želimo proći (npr. više kernela, parametara poput C, γ), $cv = 5$ što označava koliko puta ćemo razdijeliti skup na skup za učenje i testiranje te optimizirati parametre na skupu za učenje (čineći to onda ugniježđenom unakrsnom validacijom), $n\_jobs = 4$ koji predstavlja broj poslova koje će se obavljati u paraleli i $verbose = 3$ čiji parametar označava koli-

činu povratne informacije koju ćemo dobiti prilikom obavljanja unakrsne validacije. Unakrsna validacija je postupak koji se provodi kako bi generalizirali rezultate nekog skupa podataka. Koristimo ga kad pokušavamo predvidjeti kako će se određeni model ponašati u praktičnom radu. Cilj unakrsne validacije je testirati podatke za učenje sa raznim parametrima kako bi izbjegli ili smanjili problem kao što je prenaučenost (engl. overfitting), te da vidimo kako će se određeni model ponašati na novim podacima. Taj proces je dosta iscrpan i zna iznimno dugo trajati. U ovom slučaju provjeravao se linearni *SVM* kernel i *RBF* kernel. Primjer ispisa za *GridSearchCV* možemo vidjeti na slici 5.1

```
[Parallel(n_jobs=1)]: Done  90 out of  90 | elapsed: 1760.5min finished
Best parameters set found on development set:
()
OneVsRestClassifier(estimator=SVC(C=1, cache_size=200, class_weight=None, coef0=
0.0, degree=3, gamma=0.001,
  kernel=rbf, max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False),
          estimator__C=1, estimator__cache_size=200,
          estimator__class_weight=None, estimator__coef0=0.0,
          estimator__degree=3, estimator__gamma=0.001,
          estimator__kernel=rbf, estimator__max_iter=-1,
          estimator__probability=False, estimator__random_state=None,
          estimator__shrinking=True, estimator__tol=0.001,
          estimator__verbose=False, n_jobs=1)
()
Grid scores on development set:
()
0.714 (+/-0.000) for {'estimator__kernel': 'rbf', 'estimator__C': 0.001, 'estima
tor__gamma': 0.001}
0.714 (+/-0.000) for {'estimator__kernel': 'rbf', 'estimator__C': 0.001, 'estima
tor__gamma': 0.0001}
0.714 (+/-0.000) for {'estimator__kernel': 'rbf', 'estimator__C': 0.01, 'estimat
or__gamma': 0.001}
0.714 (+/-0.000) for {'estimator__kernel': 'rbf', 'estimator__C': 0.01, 'estimat
or__gamma': 0.0001}
0.857 (+/-0.000) for {'estimator__kernel': 'rbf', 'estimator__C': 1, 'estimator_
_gamma': 0.001}
0.714 (+/-0.000) for {'estimator__kernel': 'rbf', 'estimator__C': 1, 'estimator_
_gamma': 0.0001}
0.857 (+/-0.000) for {'estimator__kernel': 'rbf', 'estimator__C': 10, 'estimator
__gamma': 0.001}
0.857 (+/-0.000) for {'estimator__kernel': 'rbf', 'estimator__C': 10, 'estimator
__gamma': 0.0001}
0.857 (+/-0.000) for {'estimator__kernel': 'rbf', 'estimator__C': 100, 'estimato
r__gamma': 0.001}
0.857 (+/-0.000) for {'estimator__kernel': 'rbf', 'estimator__C': 100, 'estimato
r__gamma': 0.0001}
0.857 (+/-0.000) for {'estimator__kernel': 'rbf', 'estimator__C': 1000, 'estimat
or__gamma': 0.001}
0.857 (+/-0.000) for {'estimator__kernel': 'rbf', 'estimator__C': 1000, 'estimat
or__gamma': 0.0001}
0.857 (+/-0.000) for {'estimator__kernel': 'linear', 'estimator__C': 0.001}
0.857 (+/-0.000) for {'estimator__kernel': 'linear', 'estimator__C': 0.01}
0.857 (+/-0.000) for {'estimator__kernel': 'linear', 'estimator__C': 1}
0.857 (+/-0.000) for {'estimator__kernel': 'linear', 'estimator__C': 10}
0.857 (+/-0.000) for {'estimator__kernel': 'linear', 'estimator__C': 100}
0.857 (+/-0.000) for {'estimator__kernel': 'linear', 'estimator__C': 1000}
()
Detailed classification report:
()
The model is trained on the full development set.
The scores are computed on the full evaluation set.
()
```

Slika 5.1: Primjer ispisa funkcije *GridSearchCV* iz biblioteke Scikit-learn

Počevši od vrha možemo vidjeti koliko kombinacija je *GridSearchCV* ispitao i koliko je to sveukupno trajalo. Nakon toga se ispisuje najbolji klasifikator koji je

pronađen u testiranju, zajedno sa svim parametrima. Ispod toga imamo ocjenu za svaku moguću kombinaciju parametara te ispis na koji način se radila provjera. Iz priloženog se vidi da je unakrsna validacija bila rađena nad linearnim i *RBF SVM* kernelom, koristili su se parametri C koji su bili 0.001, 0.01, 1, 10, 100 i 1000, te parametar γ koji je bio 0.001 i 0.0001.

Rezultati koji su dobiveni iz unakrsne ugniježđene validacije se mogu vidjeti u tablici 5.7. Najbolji klasifikator po ugniježđenoj unakrsnoj validaciji je *SVM* sa *RBF* kernelom, $\gamma = 0.001$ i $C = 1$.

SVM (*RBF kernel*)

	Preciznost	Odziv	F1 mjera
Cesta	0.97	0.99	0.98
Auto	0.76	0.55	0.64
Nadvožnjak	0.00	0.00	0.00
Tunel	0.96	0.89	0.92
Naselje	0.95	0.54	0.69
Prosjek/Ukupno	0.87	0.81	0.83

Tablica 5.7: Rezultati najboljeg klasifikatora dobivenog ugniježđenom unakrsnom validacijom

5.4. Promjene oznaka na skupu FM2

S obzirom da čak ni unakrsna validacija nije uspjela pronaći kombinaciju koja bi bolje uspjela klasificirati nadvožnjake, postavlja se pitanje da li su podaci za učenje možda bili preteški. Možda bi promjena oznaka za učenje dovela do boljeg skupa za učenje, a time i bolje klasifikacije. Prije sljedećeg eksperimenta promijenjen je način označavanja nadvožnjaka kako bi detektiranje nadvožnjaka bilo jednostavnije. Dodatno, na drugačiji način su generirane SIFT značajke, tako da bi ih bilo više. Na slici 5.2 možemo vidjeti primjer slika kojima su oznake bile takve da sadrže nadvožnjak, a sada će samo zadnja ući u tu kategoriju pošto su ranije odabrani nadvožnjaci bili previše udaljeni za dobru detekciju značajki.

Za razliku od prijašnjeg broja nadvožnjaka kojih je bilo 417, nakon promjene oznaka imamo ih 194. Uz nadvožnjake, promijenjeni su još i neki teški primjeri pa imamo drugačiji broj oznaka za svaku kategoriju. To je utjecalo i na ukupan broj oznaka kojih više nije 9141, već se taj broj smanjio na 8937. U tablici 5.8 možemo vidjeti novi broj oznaka za svaku kategoriju.

Slika 5.2: Primjer slika koje su inicijalno imale oznake nadvožnjaka

Broj labela po klasi				
cesta	auto	nadvožnjak	tunel	naselje
5239	2411	194	681	412

Tablica 5.8: Broj oznaka za svaku pojedinu oznaku u skupu FM2 nakon promjene označavanja

Također su promijenjeni i parametri za izračun *SIFT* značajki kako bi dobili veći broj značajki, što bi moglo pomoći u dobivanju boljih rezultata. Na slikama 5.3 možemo vidjeti gdje su nađene SIFT značajke sa pragovima *edge-thresh = 10* i *peak-thresh = 5*,a na slikama 5.4 se mogu vidjeti pronađene SIFT značajke sa pragovima *edge-thresh = 10* i *peak-thresh = 2*.

Slika 5.3: SIFT značajke slika nadvožnjaka za pragove $edge-thresh = 10$ i $peak-thresh = 5$

Vidimo da su točke koje se ističu na nadvožnjacima one na ogradi. Ostale točke

nađene blizu nadvožnjaka i ne pomažu u klasifikaciji nadvožnjaka (eventualno možda one na stupovima). Kao što možemo vidjeti na primjeru jedne od slika, SIFT značajke su nađene na brdu iza nadvožnjaka.

Slika 5.4: SIFT značajke slika nadvožnjaka za pragove $edge-thresh = 10$ i $peak-thresh = 2$

U tablici 5.9 možemo vidjeti rezultate klasifikacije nakon promjene pragova kod računanja SIFT značajki i izmjena oznaka. Korišten je rječnik od 1000 slikovnih riječi, a sa svake slike je uzeto po 30 nasumično odabranih SIFT značajki.

Kada uspoređujemo tablice 5.4 i 5.9 ne vidi se neko poboljšanje. Iako dobivamo više SIFT značajki mijenjanjem pragova nisu sve značajke reprezentativne za ono što nam je interesantno na slici, pa time ne doprinose boljoj klasifikaciji. Mijenjanje broja SIFT značajki koje koristimo za izradu rječnika je dalo mala poboljšanja kod oznaka tunela, nadvožnjaka i auta (kod SVM-a), ali ne u tolikoj mjeri da bi se za klasificiranje nadvožnjaka moglo reći da je uspješno. Rezultate možemo vidjeti u tablici 5.10.

5.5. Usporedba modela Bag of Words s opisnikom GIST

Radi usporedbe rezultata modela *Bag of Words* s drugim popularnim metodama za prikaz slike sam još primijenio GIST opisnike iz [17] kao vektore značajki u postupku klasifikacije. GIST opisnik je globalni opisnik i promatra scenu kao cjelinu, ima vrlo

SVM ($C = 1$)

	Preciznost	Odziv	F1 mjera
Cesta	0.98	0.97	0.97
Auto	0.73	0.43	0.54
Nadvožnjak	0.32	0.33	0.33
Tunel	0.92	0.94	0.93
Naselje	0.73	0.73	0.73
Prosjek/Ukupno	0.88	0.80	0.83

SVM ($C = 0.0001$)

	Preciznost	Odziv	F1 mjera
Cesta	0.98	0.97	0.97
Auto	0.73	0.43	0.54
Nadvožnjak	0.32	0.33	0.336
Tunel	0.92	0.94	0.93
Naselje	0.73	0.73	0.73
Prosjek/Ukupno	0.88	0.80	0.83

SVM ($C = 1$ i *weight = 'auto'*)

	Preciznost	Odziv	F1 mjera
Cesta	0.98	0.97	**0.98**
Auto	0.65	0.56	0.60
Nadvožnjak	0.38	0.39	**0.38**
Tunel	0.93	0.94	0.93
Naselje	0.74	0.75	**0.74**
Prosjek/Ukupno	0.86	0.84	0.85

SVM ($C = 0.0001$ i *weight = 'auto'*)

	Preciznost	Odziv	F1 mjera
Cesta	0.99	0.97	**0.98**
Auto	0.69	0.72	**0.71**
Nadvožnjak	0.21	0.74	0.33
Tunel	0.94	0.94	**0.94**
Naselje	0.54	0.96	0.69
Prosjek/Ukupno	0.87	0.90	0.88

Random Forest ($n = 100$)

	Preciznost	Odziv	F1 mjera
Cesta	0.97	1.00	**0.98**
Auto	0.77	0.51	0.61
Nadvožnjak	1.00	0.06	0.11
Tunel	0.99	0.89	**0.94**
Naselje	0.90	0.53	0.67
Prosjek/Ukupno	0.91	0.82	0.85

Random Forest ($n = 500$)

	Preciznost	Odziv	F1 mjera
Cesta	0.97	1.00	**0.98**
Auto	0.80	0.52	0.63
Nadvožnjak	1.00	0.09	0.17
Tunel	0.99	0.89	**0.94**
Naselje	0.90	0.53	0.66
Prosjek/Ukupno	0.92	0.82	0.85

Tablica 5.9: Rezultati linearnog *SVM* ($C = 1, 0.0001$), linearnog *SVM-a* s težinama ($C = 1, 0.0001$) i *Random Forest* klasifikatora ($n = 100, 500$) sa rječnikom od 1000 riječi i SIFT značajkama sa pragovima $edge = 10$ i $peak = 2$

malenu dimenzionalnost i ne koristi rječnik. Svojstva značajna ljudskom promatraču su očuvana u opisniku. Dobiva se tako da se razmatra samo intenzitet, a kontrast se lokalno normalizira. Slika se podijeli na mrežu 4×4 ćelija te se za svaku ćeliju računa prosječna energija 32 orijentacijska filtra (na 4 skale, po 8 orijentacija pri svakoj). Veličina opisnika je $(4 \cdot 4) \cdot 32 = 512$. Rezultate možemo vidjeti u tablici 5.11.

Iz rezultata se može vidjeti puno bolje raspoznavanje nadvožnjaka kod SVM-a i nešto malo bolje kod Random Forest klasifikatora, ali i dalje nedovoljno da bi se klasificiranje nadvožnjaka moglo smatrati uspješnim. Kod SVM-a sa korištenjem *weight* opcije i parametrom $C = 0.0001$ dobivamo jako loše rezultate, pa iz toga možemo vidjeti koliko parametar C zajedno sa dodijeljenim težinama može utjecati na rezultate.

SVM ($C = 1$)

	Preciznost	Odziv	F1 mjera
Cesta	0.98	0.97	**0.98**
Auto	0.72	0.52	0.60
Nadvožnjak	0.41	0.37	0.39
Tunel	0.94	0.93	0.94
Naselje	0.75	0.79	**0.77**
Prosjek/Ukupno	0.88	0.83	0.85

SVM ($C = 0.0001$)

	Preciznost	Odziv	F1 mjera
Cesta	0.98	0.97	**0.98**
Auto	0.72	0.52	0.60
Nadvožnjak	0.41	0.37	0.39
Tunel	0.94	0.93	0.94
Naselje	0.75	0.79	**0.77**
Prosjek/Ukupno	0.88	0.83	0.85

SVM ($C = 1$ i *weight = 'auto'*)

	Preciznost	Odziv	F1 mjera
Cesta	0.98	0.97	**0.98**
Auto	0.62	0.70	0.66
Nadvožnjak	0.47	0.39	**0.42**
Tunel	0.94	0.94	0.94
Naselje	0.76	0.78	**0.77**
Prosjek/Ukupno	0.86	0.88	0.87

SVM ($C = 0.0001$ i *weight = 'auto'*)

	Preciznost	Odziv	F1 mjera
Cesta	0.99	0.97	**0.98**
Auto	0.70	0.75	**0.72**
Nadvožnjak	0.20	0.72	0.31
Tunel	0.95	0.95	**0.95**
Naselje	0.54	0.96	0.69
Prosjek/Ukupno	0.88	0.90	0.88

Random Forest ($n = 100$)

	Preciznost	Odziv	F1 mjera
Cesta	0.97	1.00	**0.98**
Auto	0.78	0.54	0.64
Nadvožnjak	1.00	0.04	0.07
Tunel	0.99	0.88	0.93
Naselje	0.93	0.56	0.70
Prosjek/Ukupno	0.92	0.83	0.86

Random Forest ($n = 500$)

	Preciznost	Odziv	F1 mjera
Cesta	0.97	1.00	**0.98**
Auto	0.79	0.53	0.64
Nadvožnjak	1.00	0.06	0.11
Tunel	0.99	0.89	0.94
Naselje	0.96	0.56	0.71
Prosjek/Ukupno	0.92	0.83	0.86

Tablica 5.10: Rezultati linearnog *SVM* ($C = 1, 0.0001$), linearnog *SVM-a* s težinama ($C = 1, 0.0001$) i *Random Forest* klasifikatora ($n = 100, 500$) sa rječnikom od 1000 riječi i SIFT značajkama sa pragovima $edge = 10$ i $peak = 2$ (kod izrade rječnika korišteno 50 SIFT značajki po slici)

SVM ($C = 1$)

	Preciznost	Odziv	F1 mjera
Cesta	0.98	0.99	**0.99**
Auto	0.79	0.70	0.74
Nadvožnjak	0.79	0.41	**0.54**
Tunel	0.96	0.92	0.94
Naselje	0.86	0.87	0.86
Prosjek/Ukupno	0.92	0.89	0.90

SVM ($C = 0.0001$)

	Preciznost	Odziv	F1 mjera
Cesta	0.98	0.99	**0.99**
Auto	0.79	0.70	0.74
Nadvožnjak	0.79	0.41	**0.54**
Tunel	0.96	0.92	0.94
Naselje	0.86	0.87	0.86
Prosjek/Ukupno	0.92	0.89	0.90

SVM ($C = 1$ i *weight = 'auto'*)

	Preciznost	Odziv	F1 mjera
Cesta	0.99	0.98	**0.99**
Auto	0.71	0.79	0.75
Nadvožnjak	0.36	0.87	0.51
Tunel	0.92	0.96	0.94
Naselje	0.59	0.97	0.73
Prosjek/Ukupno	0.88	0.92	0.90

SVM ($C = 0.0001$ i *weight = 'auto'*)

	Preciznost	Odziv	F1 mjera
Cesta	1.00	0.79	0.88
Auto	0.41	0.98	0.58
Nadvožnjak	0.03	1.00	0.06
Tunel	0.48	0.96	0.64
Naselje	0.06	1.00	0.12
Prosjek/Ukupno	0.74	0.87	0.73

Random Forest ($n = 100$)

	Preciznost	Odziv	F1 mjera
Cesta	0.99	0.99	**0.99**
Auto	0.87	0.74	0.80
Nadvožnjak	1.00	0.26	0.41
Tunel	0.99	0.91	**0.95**
Naselje	0.90	0.84	**0.87**
Prosjek/Ukupno	0.95	0.90	0.92

Random Forest ($n = 500$)

	Preciznost	Odziv	F1 mjera
Cesta	0.99	0.99	**0.99**
Auto	0.89	0.75	**0.81**
Nadvožnjak	1.00	0.30	0.46
Tunel	0.99	0.91	**0.95**
Naselje	0.90	0.82	0.86
Prosjek/Ukupno	0.96	0.90	0.92

Tablica 5.11: Rezultati linearnog *SVM* ($C = 1, 0.0001$), linearnog *SVM-a* s težinama ($C = 1, 0.0001$) i *Random Forest* klasifikatora ($n = 100, 500$) uz korištenje opisnika GIST)

6. Zaključak

U okviru ovog rada prikazan je postupak kategorizacije u skupove oznaka na primjeru prometnih scena (skup slika FM2 [17]). Lokalne značajke slika izlučene su i opisane pomoću algoritma *SIFT* te su grupirane u slikovne riječi pomoću algoritma *k*-srednjih vrijednosti. Za prikaz slika histogramima slikovnih riječi korišten je model *Bag of Words*. Kao klasifikator korišteni su *SVM* i *Random Forest* koji su bili dostupni u biblioteci *Scikit-learn*, koja je vrlo pogodna za rješavanje problema iz strojnog učenja. Ovakav način klasificiranje u mom slučaju daje iznimno dobre rezultate za oznaku ceste (uspjeli smo doći do $F1 = 98\%$) i jako dobre za tunele ($F1 = 94\%$). Za ostale klase ovaj postupak se i nije pokazao prevelikim uspjehom, za oznaku auto i naselje je $F1$ varirao između 65-75 %, no najveći problem je bilo klasificirati nadvožnjak (rezultati su bili jako loši, $F1 < 50\%$). Zbog same prirode nadvožnjaka, na njemu je bilo teško pronaći *SIFT* značajke. U skupu za učenje oznaka *nadvožnjak* je bila pridružena jako teškim slikama, kojima ni golim okom nije jednostavno odrediti oznake.

Kroz provođenja ispitivanja primijetilo se da je stvaranje rječnika jako zahtjevna operacija. Mijenjanjem rječnika mijenjali su se i rezultati ali u malom rasponu. Za rječnik od 5000 riječi koji se radio skoro 3 puta duže od rječnika od 1000 riječi rezultati su bili tek marginalno bolji. Za rječnik od 500 riječi rezultati su bili tek nešto lošiji, što samo govori da i nije potreban ogromni rječnik da bi klasifikacije bile dobre.

Kad uspoređujemo *SVM* i *Random Forest* klasifikator, oba imaju slične rezultate, ali se *Random Forest* klasifikator više muči sa klasifikacijom težih primjera nego *SVM*. Korištenje previše značajki nad toliko slika lako može prepuniti memoriju, pa smo i time ograničeni prilikom rada na prosječnom računalu. U daljnjem istraživanju bilo bi interesantno isprobati druge vrste klasifikatora, te probati značajke koje bi sa manje informacija dale bolje rezultate kako bi se cijeli postupak ubrzao (npr. detaljnije GIST, Spatial Fisher Vector).

LITERATURA

[1] Andrew Y. Ng Adam Coates. Learning feature representations with k-means. *Neural Networks: Tricks of the Trade*, 2012. URL http://web.stanford.edu/~acoates/papers/coatesng_nntot2012.pdf.

[2] Dan Benyamin. A gentle introduction to random forests, ensembles, and performance metrics in a commercial system, June 2012. URL http://citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics/.

[3] Yaroslav Bulatov. Svm - hard or soft margins?, 2011. URL http://stackoverflow.com/questions/4629505/svm-hard-or-soft-margins.

[4] Corinna Cortes i Vladimir Vapnik. Support-vector networks. U *Machine Learning*, stranice 273–297, 1995.

[5] Pornpat Nikamanon Dennis Park. Machine learning, lecture 11, 2008. URL http://www.ics.uci.edu/~dramanan/teaching/ics273a_winter08/lectures/lecture11.pdf.

[6] D. Gamberger i T. Šmuc. DMS poslužitelj za analizu podataka, 2001. URL http://dms.irb.hr/.

[7] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL http://www.scipy.org/. [Online; accessed 2014-06-27].

[8] Adele Cutler Leo Breiman. Random forests. URL http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#intro.

[9] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, stranice 91–110, 2004. URL http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

[10] Group of authors. Latex, . URL http://en.wikibooks.org/wiki/LaTeX/.

[11] Group of authors. Precision_and_recall, . URL http://en.wikipedia.org/wiki/Precision_and_recall.

[12] Group of authors. K-means_clustering, . URL http://en.wikipedia.org/wiki/K-means_clustering.

[13] Group of authors. Support vector machine, . URL http://en.wikipedia.org/wiki/Support_vector_machine.

[14] Dino Pačandi. Kategorizacija slika histogramima slikovnih riječi. Technical report, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, June 2013.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, i E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[16] Jesse Read. Scalable multi-label classification, September 2010. URL http://users.ics.aalto.fi/jesse/talks/Charla-UC3M.pdf.

[17] Ivan Sikirić, Karla Brkić, Josip Krapac, i Siniša Šegvić. Image representations on a budget: Traffic scene classification in a restricted bandwidth scenario. *IEEE Intelligent Vehicles Symposium*, 2014.

[18] M. S. Sorower. A literature survey on algorithms for multi-label learning. 2010.

[19] Chih-Fong Tsai. Bag-of-words representation in image annotation: A review. *ISRN Artificial Intelligence*, 2012. URL http://downloads.hindawi.com/journals/isrn.artificial.intelligence/2012/376804.pdf.

[20] Grigorios Tsoumakas i Ioannis Katakis. Multi-label classification: An overview. *International Journal Data Warehousing and Mining*, 2007:1–13, 2007.

[21] A. Vedaldi i B. Fulkerson. VLFeat - an open and portable library of computer vision algorithms. U *ACM International Conference on Multimedia*, 2010.

[22] Brendt Wohlberg. Visual words: Text analysis concepts for computer vision, 2009. URL `http://www.ima.umn.edu/2008-2009/MM8.5-14.09/abstracts.html`.

Klasifikacija prometnih scena u skupove oznaka

Sažetak

U ovom radu razmatra se klasifikacija prometnih scena u skupove oznaka. Za izlučivanje značajki korišten je algoritam *SIFT*. Za grupiranje lokalnih značajki slika iz skupa za učenje korišten je algoritam *k-srednjih* vrijednosti. Grupe koje smo dobili i njihove srednje vrijednosti tvore slikovni rječnik, gdje svaka srednja vrijednost predstavlja jednu slikovnu riječ. Izgrađeni su histogrami slikovnih riječi iz svih slika skupa za učenje i ti histogrami su korišteni za učenje klasifikatora, stroja sa potpornim vektorima i nasumične šume. Problem klasifikacije u skupove oznaka je sveden na binarni problem klasifikacije korištenjem *jedan-protiv-svih* strategijom. Provedeno je ispitivanje uspješnosti klasifikacije sa raznim postavkama nad više oznaka. Računala se *F1* mjera. Rezultati za pojedine oznake su bili jako dobri (*cesta*, $F1 = 98\%$ i *tunel*, $F1 = 94\%$) dok za neke su bili jako loši (*nadvožnjak*, $F1 < 50\%$).

Ključne riječi: Klasifikacija prometnih scena, kategorizacija slike, histogram slikovnih riječi, slikovne riječi, slikovni rječnik, SIFT, stroj s potpornim vektorima, nasumične šume, skup oznaka, k-srednje vrijednosti, F1 mjera, izlučivanje značajki

Multi-label classification of traffic scenes

Abstract

This work deals with multi-label classification of traffic scenes. Extraction of local features from an image is done with the *SIFT* algorithm. Grouping of local feature vectors from the training set of images is done by the *k-means* algorithm. Centroids produced by the k-means and their corresponding mean values are used as visual words, and all the visual words put together form a visual vocabulary. Using the visual vocabulary, for each image, a histogram is built, which is then used as training data for the classifiers, support vector machine and random forest classifier. Multi-label classification problem was simplified to a binary problem using a *one-vs-all* strategy. Categorization success is evaluated over multiple labels using a variety of parameters. F1 measure is calculated. Results for some labels were very good (*road*, $F1 = 98\%$ and *tunnel*, $F1 = 94\%$), while for others, not so well (*overpass*, $F1 < 50\%$).

Keywords: Multi-label classification, traffic scenes, SIFT, k-means, Support vector machine, Random Forest, image classification, image categorization, histogram, visual words, visual vocabulary, multi-label, F1 measure, feature point extraction