

Nenadzirana klasifikacija slika

Tehnička dokumentacija

Zagreb, Svibanj 2017.

Sadržaj

1 Uvod i opis dataseta	2
2 Motivacija i intuitivni opis algoritma za klasifikaciju	4
3 Preprocesiranje	5
4 Određivanje bitnih značajki	7
5 Klasteriranje	9
5.1 Algoritam k-sredina	9
5.2 Uklanjanje anomalija prije klasteriranja	9
6 Analiza rezultata	11
7 Zaključak	15
8 Literatura	16

1 Uvod i opis dataseta

Ovaj dokument predstavlja tehničku dokumentaciju vezanu uz rješenje problema nenađizirane klasifikacije zadanog skupa podataka od 6889 slika prema njihovom sadržaju. Većina slika je u *.jpg* formatu (6668), dok su formati *.png*, *.tiff* i *.gif* manje zastupljeni. Za početak je dobro vizualizirati podake kako bismo se *sprijateljili* sa zadanim datasetom. Letimičnim pregledom danog dataseta uočavamo sljedeće klastere koji se ističu: auti, avioni, cvijeće, katedrale, kolači, kuće, leopardi, lica, motori i ovce. No, budući da se radi o nenađiranom učenju, ne smijemo pretpostaviti točan broj klasa, što znatno otežava rješavanje problema. Stoga je česta vizualizacija podataka korisna za izgradnju subjektivne procjene uspješnosti pojedinih algoritama.

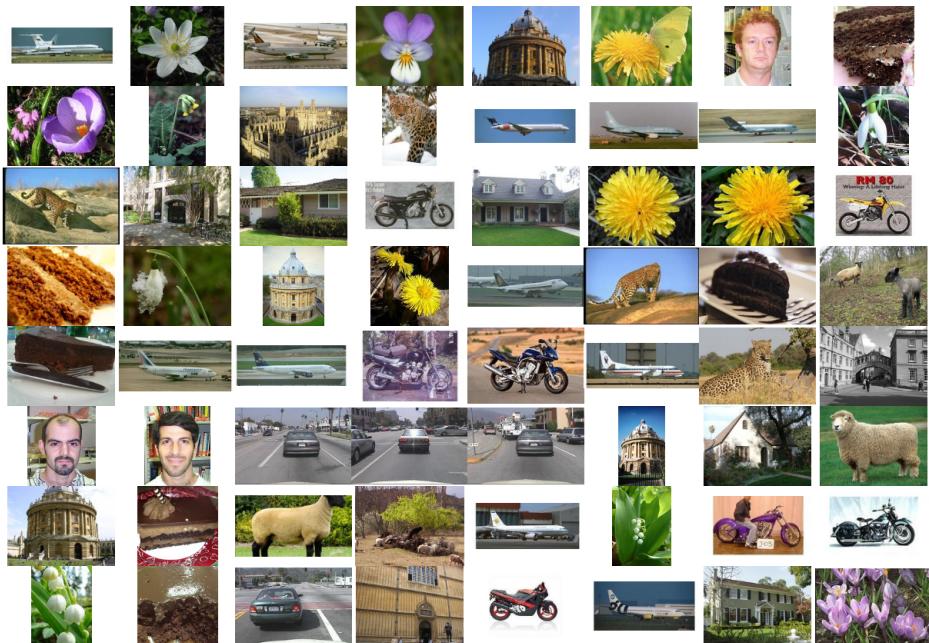


Figure 1: Primjeri slika iz danog skupa podataka

Također, u datasetu postoje brojne slike koje su gotovo identične:



Figure 2: Primjeri gotovo identičnih slika iz danog skupa podataka

Nadalje, uspoređujući naš skup podataka sa skupovima podataka koji su tipični za testiranje algoritama za klasifikaciju slika, primjećujemo da je dani skup podataka izuzetno malen.

ime skupa podataka	broj klasa	veličina slike
STL-10	10	96x96
CIFAR-10	10	32x32
MNIST	10	28x28
CIFAR-100	100	32x32

Figure 3: Primjeri poznatih skupova slika

Uz problem određivanja broja klastera usko je vezan i problem evaluacije točnosti klasteriranja. Naime, često nije jasno trebamo li neki klaster podijeliti na podklastere ili ne. Primjerice, avione možemo promatrati kao zaseban klaster, ali ih možemo podijeliti i na avione na tlu i avione u zraku; ljudska lica je prirodno klasificirati na muška i ženska, ali se nameće pitanje je li to uopće nužno. Iz toga zaključujemo da je ljudski faktor od osobite važnosti u analizi nenađiranog klasteriranja. Ovim i sličnim problemima bavit ćemo se u nastavku ovog rada.

Prvo ćemo dati kratak opis i motivaciju za korištenje našeg algoritma. Zatim ćemo dati tehničke detalje koji se tiču preprocesiranja, određivanja značajki i klasteriranja. Na kraju ćemo analizirati rezultate,

procijeniti utjecaj pojedinih dijelova algoritma na konačan rezultat te iznijeti zaključak.

2 Motivacija i intuitivni opis algoritma za klasifikaciju

Razmišljajući o ovom zadatku, prvo smo promatrali do kakvih sve problema može doći u klasifikaciji slika. Jedna od ključnih karakteristika ljudskog vida je mogućnost uočavanja objekta od interesa bez obzira na to gdje se on nalazi na slici, koje je boje, je li rotiran i kako izgleda njegov okoliš. Stoga bismo htjeli postići da naša klasifikacija bude invarijantna na takve promjene. Da bi to postigli, napraviti ćemo niz augmentacija na postojećim slikama (poput translacije, rotacije i promjene kontrasta) i na njima trenirati konvolucijsku neuronsku mrežu kako bismo pomoću nje naučili značajke koje dobro opisuju pojedinu klasu objekata. Naime, ta vrsta neuronskih mreža postiže impresivne rezultate u nadziranom klasteriranju pa ih stoga nije loše pokušati prilagoditi na naš problem. Takav pristup baziran je na [1]. Na kraju, pomoću dobivenih značajki i algoritma k -sredina napravimo klasteriranje slika, po uzoru na [2].

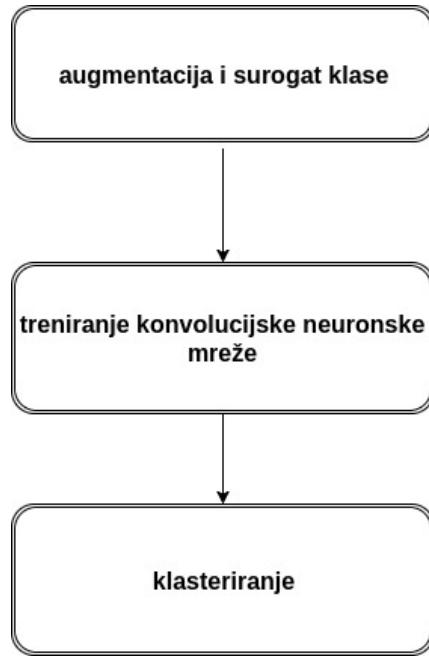


Figure 4: Kratki shematski prikaz algoritma

3 Preprocesiranje

Prije korištenja konvolucijskih neuronskih mreža, od određenog broja nasumičnih slika napravljene su nove, pomoćne (eng. surrogate) klase. Uzimali smo po 100,200 i 500 pomoćnih klasa. Broj pomoćnih klasa odredili smo pomoću [1], gdje se koristi sličan princip za nenadzirano klasificiranje slika. Budući da je u [1] uzeto 5-8% dataseta da bi se stvorile pomoćne klase, dolazimo do gornje ograde od 500 klasa za naš problem.

Sve slike su reskalirane na veličinu 96×96 piksela te je nakon augmentacije iz svake slike izrezano centralnih 64×64 piksela. Augmentacije su kompozicija promjena $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4, \mathcal{T}_5, \mathcal{T}_6$ i \mathcal{T}_7 napravljene u *RGB* prostoru boja (osim ako nije drugačije navedeno), gdje su navedene promjene redom:

1. rotacija za kut $\alpha \in [-20^\circ, 20^\circ]$,
2. translacija po x i y -osi u oba smjera za najviše 20% odgovarajuće dimenzije slike,
3. skaliranje slike brojem iz intervala $[0.7, 1.4]$,
4. dodavanje broja iz intervala $[-0.1, 0.1]$ H komponenti u *HSV* prostoru boja,
5. *Kontrast1*: potenciranje komponenti zasićenja (S) i vrijednosti (V) u *HSV* prostoru boja brojem između $[0.5, 2]$, množenje s brojem između $[0.7, 1.4]$ te zbrajanjem s brojem između $[-0.1, 0.1]$,
6. *Kontrast2*: množenje svake *RGB* komponente s različitim nasumično odabranim brojevima iz intervala $[0.5, 2]$.

Većina promjena napravljena je pomoću Python biblioteke *imgaug* (image augmenter library) koja ima brojne funkcije za augmentacije, a manji dio je napravljen pomoću *numpy* biblioteke. Pomoću tih promjena napravili smo 500 pomoćnih klasa. Kod promjena kontrasta, morali smo paziti da navedene kontraste pažljivo komponiramo, kako međusobno, tako i s ostalim transformacijama, kako navedene augmentacije ne bi rezultirale presvjetlim ili pretamnim slikama. Zatim smo iz svake dobivene slike izrezali centralnih 64×64 piksela, uzimajući u obzir rotaciju i translaciju. Odgovarajućim računskim operacijama spriječili smo da se na augmentiranim slikama vidi *padding*, odnosno dio pozadine koji je nadopunjjen crnom bojom zbog translacije ili rotacije. Primjeri umjetnih klasa prikazani su na sljedećim slikama:



Figure 5: Primjer pomoćnih klasa korištenjem *kontrasta* 2

4 Određivanje bitnih značajki

Nakon što smo napravili pomoćne klase i njima pridružili odgovarajuće oznake, konstruirali smo i istrenirali konvolucijsku neuronsku mrežu. Iz mreže smo izvukli nove značajke za svaku sliku. Koristili smo dvije mreže, koje ćemo nazivati *mala* i *velika* mreža. Kovolucijske neuronske mreže posebno su dobre za klasifikaciju slika. Sastoje se od konvolucijskih, potpuno povezanih i aktivacijskih slojeva te sloja sažimanja. Konvolucijski slojevi sastoje se od receptivnih polja neurona, tj. skupine neurona koje procesiraju dijelove ulazne slike koje nazivamo jezgrom. Rezultati tih skupina služe za dobivanje reprezentacije originalne slike u višoj rezoluciji. Iza konvolucijskog sloja slijedi sloj sažimanja, čija je svrha smanjenje dimenzije rezultata nastalog u konvolucijskom sloju. Taj postupak ponavlja se kroz sve takve slojeve. To omogućava konvolucijskoj neuronskoj mreži invarijantnost na translacije. Nakon slojeva s receptivnim poljima neurona slijedi potpuno povezani sloj. Poptuno povezani slojevi su oni čija je jezgra veličine 1×1 . Posljednji sloj u konvolucijskoj neuronskoj mreži je aktivacijski sloj, koji prema unaprijed zadanoj pravilu računa krajnji rezultat.

Malu mrežu možemo kompaktno zapisati kao $64c5 - 64c5 - 128f$, čime želimo reći da se radi o konvolucijskoj neuronskoj mreži s dva konvolucijska sloja koji sadrže 64 filtra za svaku grupu od 5×5 piksela, nakon kojih dolazi potpuno povezani sloj dimenzije 128. Zadnji sloj, softmax sloj, klasificira primjerke iz pomoćne klase. Primijenili smo $2 \times 2 max - pooling$ na rezultate iz prvog i drugog konvolucijskog sloja. Sukladno prethodnom, veliku mrežu poistovjećujemo s $64c5 - 128c5 - 256c5 - 512f$. Dakle, imamo 3 konvolucijska sloja redom s 64, 128 i 256 filtra za 5×5 piksela, nakon kojih dolazi potpuno povezani sloj dimenzije 512. Nakon njega slijedi softmax sloj u kojem se računa krajnji rezultat. Ovdje je također primijenjen $2 \times 2 max - pooling$.

Tablica u nastavku prikazuje rezultate i parametre treniranja, pri čemu je S oznaka za malu mrežu, B oznaka za veliku mrežu, num_surr broj pomoćnih klasa, $train_steps$ broj koraka treniranja, $learning_rate$ stopa učenja, $batch_size$ količina podataka pomoću kojih se ažuriraju težine u neuronskoj mreži u danom trenutku, T i V točnost klasifikacije na slučajnim podskupovima za treniranje, odnosno validaciju veličine $batch_size$.

Kao nove značajke za svaku sliku u datasetu uzeli smo rezultat u potpuno povezanom sloju konvolucijske neuronske mreže te smo na taj način slike prebacili u 128, odnosno 512-dimenzionalni vektorski prostor. Ovako dobivene značajke imaju nekoliko prednosti. Naime, u dalnjim slojevima konvolucijske neuralne mreže dobivamo značajke puno veće razine apstrakcije od samih piksela na slici i koje imaju puno manju dimenziju pa su algoritmi na njima brži. U analizi rezultata, vidjet ćemo da takve značajke zbilja obuhvaćaju bitne informacije o pojedinom objektu te da su invarijantne s obzirom na elementarne transformacije poput rotacije i translacije. Prikaz male mreže izgleda ovako:

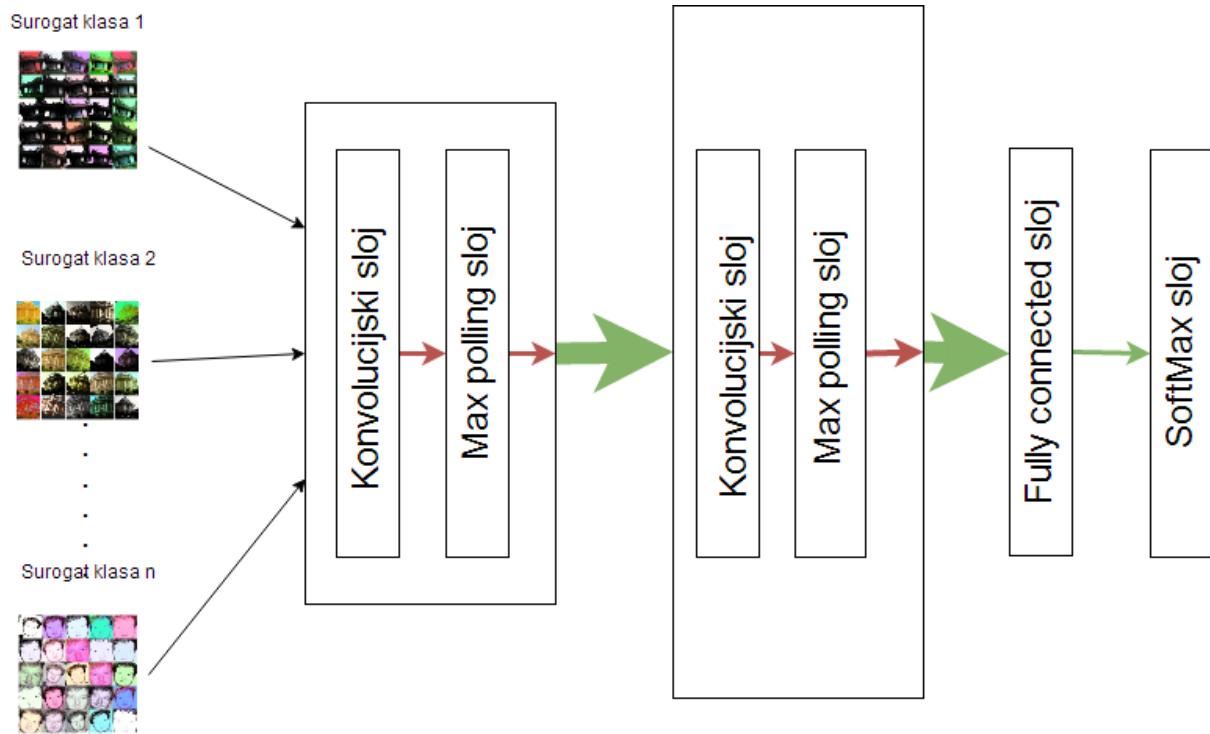


Figure 6: Mala konvolucijska neuronska mreža

Treniranje konvolucijske neuronske mreže i dobivanje značajki implementirano je u *TensorFlowu*. Obje mreže smo trenirali na 100, 200 i 500 pomoćnih klasa. Detalji treniranja dani su u sljedećoj tablici:

CNN	num_surr	train_steps	learning_rate	batch_size	train_accuracy	valid_accuracy
S	100	2000	0.0005	100	100	99
S	200	3500	0.0005	100	100	97
S	500	5000	0.0005	100	99	96
B	100	4000	0.0001	100	99	96
B	200	5000	0.0001	250	99	96.8
B	500	7000	0.0001	500	99	97.8

Figure 7: Detalji treniranja male i velike konvolucijske neuronske mreže

5 Klasteriranje

Nakon što smo naučili značajke, potrebno je slike određene tim značajkama svrstati u klastere. Isprobavali smo brojne algoritme s različitim metrikama. Algoritam k -sredina s euklidiskom udaljenosti dao je najbolje rezultate.

5.1 Algoritam k -sredina

Neka su (x_1, x_2, \dots, x_n) d -dimenzionalni podaci te neka su (S_1, S_2, \dots, S_k) klasteri kojima ti podaci mogu pripadati. Cilj algoritma k -sredina je svrstati podatke u klastere na način da se minimizira suma udaljenosti pojedinih elemenata od centroida klastera.

$$\arg \min_{\mathbf{S}} \sum_{i=1}^k \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \arg \min_{\mathbf{S}} \sum_{i=1}^k |S_i| \text{Var } S_i \quad (1)$$

Algoritam u praksi kao ulazni argument prima broj iteracija i prepostavljeni broj klastera te se sastoji od dva koraka:

1. Dodjeljivanje elemenata pojedinim klasterima na temelju početno generiranih centroida.
2. Izračun novih centroida na temelju elemenata koji su dodijeljeni pojedinim klasterima.

Za klasteriranje je korištena klasa *KMeans* iz paketa *sklearn* unutar koje je već implementirano optimalno uzimanje početnih centroida.

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=num_clusters, random_state=0).fit(X)
```

5.2 Uklanjanje anomalija prije klasteriranja

Pošto su algoritmi za klasteriranje poprilično osjetljivi na outliere, potrebno ih je prije samog klasteriranja izbaciti. U tu svrhu korišten je *Isolation Forest* algoritam iz *sklearn.cluster* paketa.

```
from sklearn.cluster import IsolationForest
clf = IsolationForest(max_samples=100, random_state=1, n_estimators = 7,
                       contamination = 0.15)
clf.fit(column(X,1))
anomaly_table = clf.predict(column(X,1))
```

Navedeni algoritam temelji se na konceptu izolacije određene točke od ostatka skupa podataka. Izolacija određenih točaka radi se na način da se skup ulaznih podataka (x_1, x_2, \dots, x_n) rekurzivno particionira na temelju nasumično odabranih značajki i nasumično odabranih granica, uz prepostavku da će rezultantna stablasta struktura biti izgrađena tako da će se ulazni podaci koje je veoma lako partacionirati nalaziti na samom vrhu stablaste strukture. To se može vidjeti na sljedećoj slici:

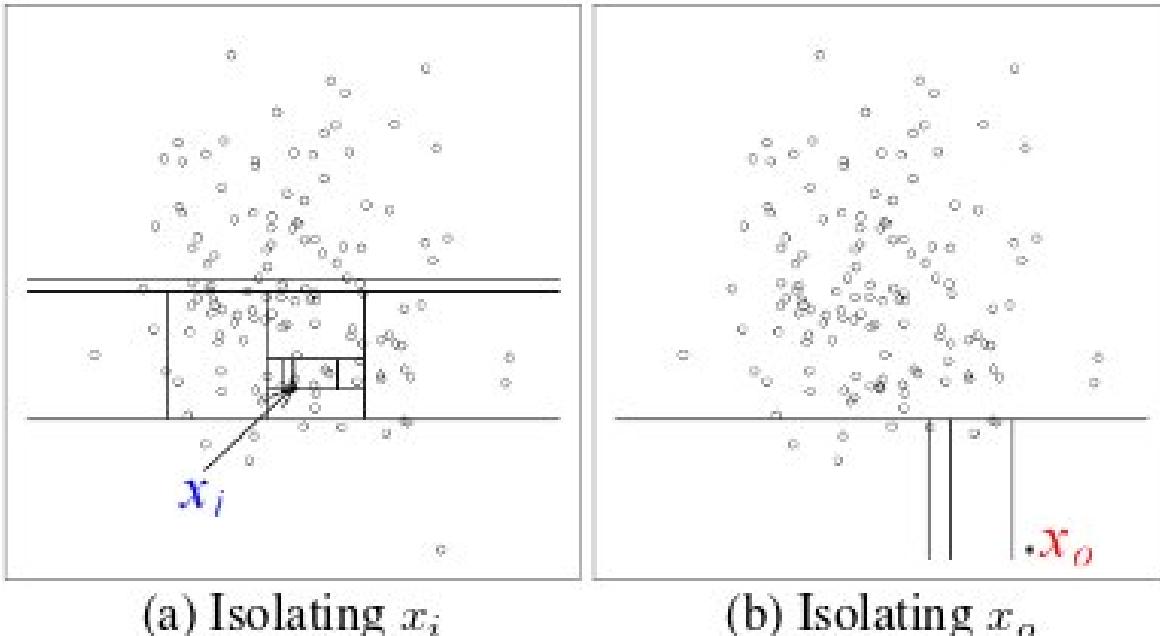


Figure 8: Određivanje outliera pomoću Isolation Forest algoritma

Kao što je vidljivo na slici, točku x_o je puno jednostavnije odvojiti od točke x_i te se ona može smatrati kandidatom za outlier. Slično kao i kod stabla odluke kod klasifikacije, točnost određivanja outliera povećava se korištenjem više stabala gdje se zadani broj ili postotak ulaznih podataka koji su najbliže udaljeni korijenu stabala može smatrati outlierima.

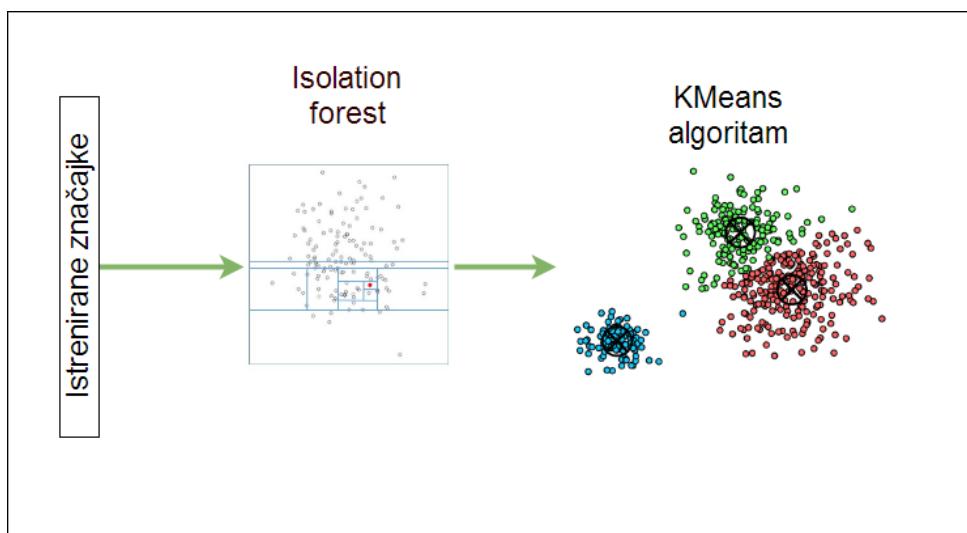


Figure 9: Tijek klasteriranja

6 Analiza rezultata

U uvodnom poglavlju smo spomenuli da je evaluacija rezultata tipičan problem nenadzirane klasifikacije. Stoga ćemo ovom odjeljku posvetiti posebnu pažnju. Za početak, klaster jednostavno poistovjećujemo s najzastupljenijom klasom slika iz dataseta. To znači da ćemo pod pojmom točnosti unutar pojedinog klastera smatrati udio slika iz najzastupljenije klase. Točnost klasteriranja je onda jednostavno težinska suma po svim klasterima, pri čemu su težine proporcionalne veličini klastera. Lako se vidi da sukladno toj definiciji točnosti vrijedi:

Povećavanjem broja klastera gubimo osjetljivost (udio dobro klasificiranih slika u tom klasteru od ukupnog broja slika iz te klase), ali povećavamo preciznost (udio dobro klasificiranih slika u tom klasteru od ukupnog broja slika u klasteru).

Zbog same prirode problema (klasifikacija slika), odlučujemo da nam je preciznost važnija od osjetljivosti. Smatramo da smo time opravdali gornju definiciju točnosti.

Egzaktno utvrđivanje točnosti je spor i naporan proces jer bi se od čovjeka zahtjevalo da prođe po svim slikama u svim klasterima. Iako je to izvedivo za dani dataset od nekoliko tisuća slika, zbog neskalabilnosti na veće datasetove odlučujemo se na korištenje heuristike. Naime, iz svakog klastera ćemo nasumično izabrati 100 slika, pronaći među njima najzastupljeniju klasu te izračunati njen udio u klasteru. Zbog nasumičnog biranja slika, očekujemo da tako dobivena preciznost odgovara preciznosti cijelog klastera. Preostaje procijeniti broj klastera. U tu svrhu ćemo koristiti metodu lakta (eng. elbow method). Ova metoda mjeri udio u kojem matematički model opisuje varijaciju (disperziju) danog skupa podataka. Preciznije, ako se nacrtava postotak objašnjene varijance po klasterima prema broju klastera, prvi klasteri dati će puno informacija (objasnit će puno varijance), ali nakon neke točke promjena objašnjene varijance će pasti te će to rezultirati kutom (laktom) na grafu, a točka pregiba označavat će broj traženih klastera. Slika objašnjene varijance s obzirom na broj klastera izgleda ovako:

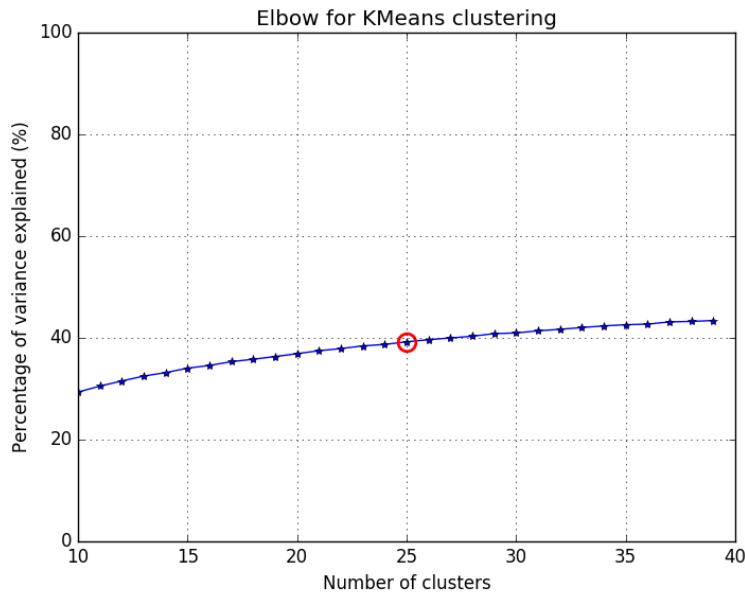


Figure 10: Opis objašnjene varijance

Vidimo da "lakat" nije jasno naznačan pa nije očito da je sugeriranih 25 odgovarajući broj klastera. Mogući razlog tome je prevelika dimenzionalnost prostora značajki i izražena varijabilnost našeg skupa podataka. Iako iz samog grafa ne možemo sa sigurnošću reći, 25 nije toliko loša procjena s obzirom na broj vidljivo različitih uočenih klasa.
Još jedna mjera koju smo koristili za određivanje broja klastera je promatranje prosječne sume kvadrata unutar pojedinog klastera. Procijenjeni broj klastera tom metodom je 10, kao što se vidi i na sljedećoj slici:

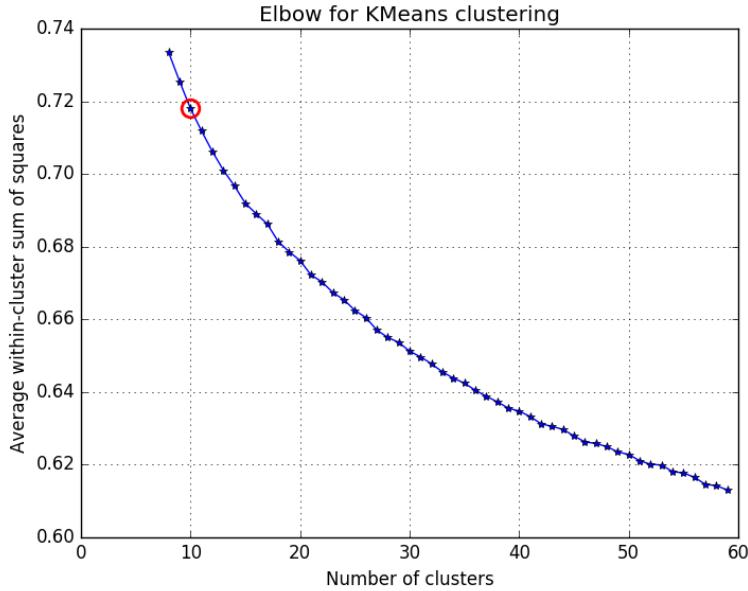
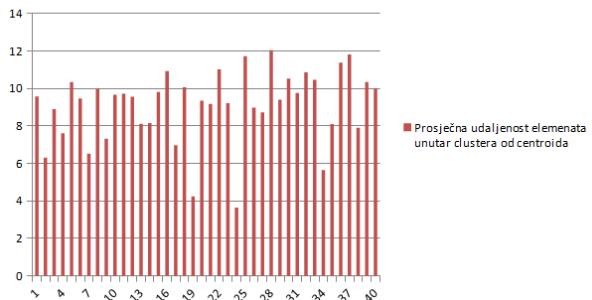


Figure 11: Metoda lakta uz računanje prosječne sume kvadrata unutar klastera

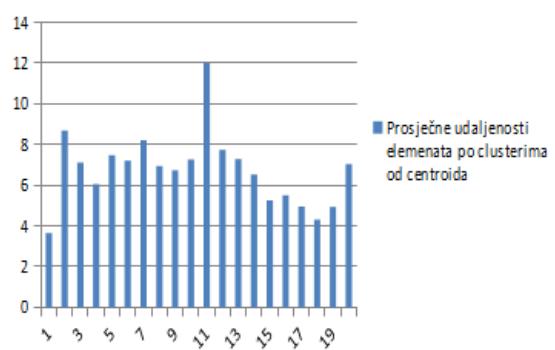
Poštujući sugestije ovih dviju metoda kao i subjektivne procjene, koristit ćemo metodu k -sredina za sve k između 10 i 25.

Sada smo spremni za iznošenje konačnih rezultata. Prisjetimo se, varirali smo složenost neuronske mreže (velika, mala), broj pomoćnih klasa (100, 200, 500), metode klasteriranja (k -sredina, *Isolation Forest*) te upravo spomenuti broj klastera (10, ..., 25). Dobiveni rezultati su bolji kada koristimo veliku mrežu u odnosu na malu, ponešto bolji kada koristimo veći broj pomoćnih klasa te ponešto bolji kada koristimo *Isolation Forest* u odnosu na algoritam k -sredina. Što se tiče broja klastera, nije toliko lako donesti zaključak, ali smatramo da su rezultati u tom intervalu najstabilniji kada koristimo 20 klastera. Važno je napomenuti da metode lakta služe za približnu procjenu broja klastera. Zbog toga smo dodatno napravili klasteriranje s 40 klastera te analizu centroida, prosječnih udaljenosti za 40 i 20 klastera od odgovarajućih centroida te udaljenosti pojedinih klastera od preostalih. Sljedeći dijagrami prikazuju rezultate te analize:

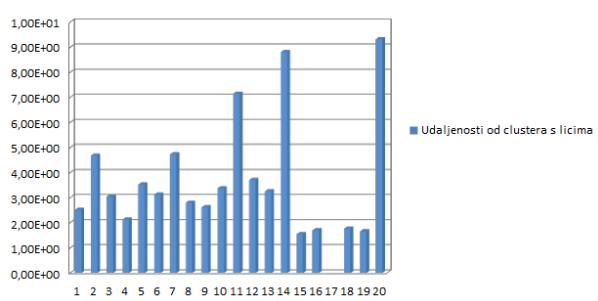
Prosječna udaljenost elemenata unutar cluster-a od centroida



Prosječne udaljenosti elemenata po clusterima od centroida



Udaljenosti od cluster-a s licima



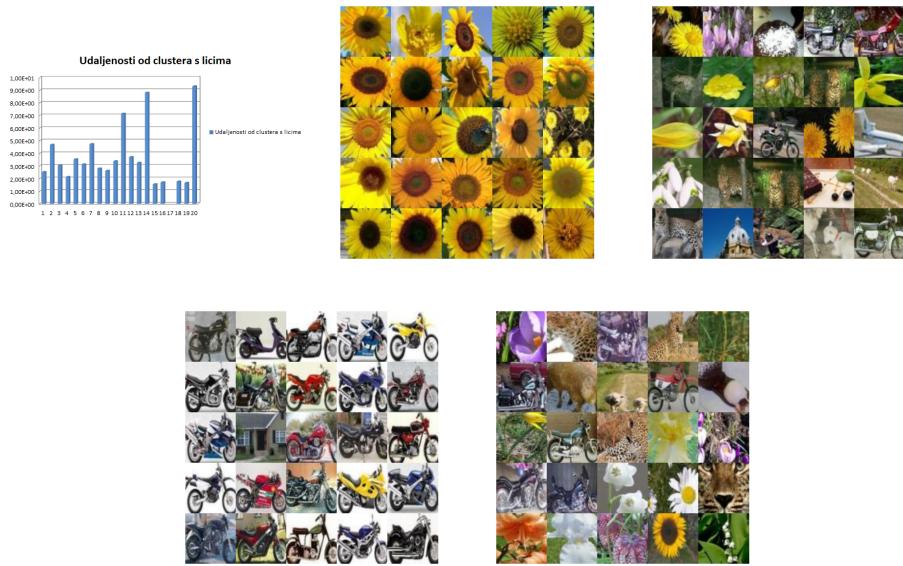


Figure 12: Primjeri nekih klastera

7 Zaključak

Uočili smo da dobivene značajke daju podosta informacija o slikama na apstraktnoj razini, što se vidi iz nekoliko dobivenih klastera s visokom preciznosti. Međutim, smatramo da je razlog zašto klasifikacija nije bolja taj što nedostaje više informacija o objektu sa slike s niže razine apstrakcije. Također, treba napomenuti da problem stvaraju i nebalansirane klase pa zbog toga konvolucijska neuronska mreža ne nauči značajke jednako dobro za sve klase.

Za kraj ćemo istaknuti klase koje se gotovo uvijek dobro klasificiraju (obzira na broj klastera): lica, žuto cvijeće, motori, avioni na tlu, avioni u zraku, arhitektura te kolači.

8 Literatura

- [1] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, Thomas Brox, *Discriminative Unsupervised Feature Learning with Exemplar Convolutional Neural Networks*
- [2] K. Y. Hui, *Direct modeling of complex invariances for visual object features*, in ICML, 2013.
- [3] Ian Goodfellow, Yoshua Bengio, Aaron Courville, *Deep Learning*
- [4] Fei Tony Liu, Kai Ming Ting, Zhi-Hua Zhou, *Isolation Forest*