Razvoj sustava za prilagodljivu ugradnju i izvođenje postupaka i modela strojnog učenja

Ivić, Vedran

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

RAZVOJ SUSTAVA ZA PRILAGODLJIVU UGRADNJU I IZVOĐENJE POSTUPAKA I MODELA STROJNOG UČENJA

Diplomski rad

Vedran Ivić

Osijek, 2019.

SADRŽAJ

1. UVOD

Strojno učenje je postalo jedna od najvažnijih tema unutar razvoja sustava i poslovanja koji traže inovativne načine upravljanja podacima s ciljem postizanja određenih spoznaja i unaprjeđenja. Kako su u današnjem svijetu uobičajene pojave sve većih, ogromnih količina podataka i informacija, to je i potreba za strojnim učenjem sve veća. Različitim postupcima strojnog učenja se nastoje dobiti odgovarajući modeli kako bi se iz velikih količina podataka mogle izvući relevantne informacije, ispravni zaključci i predviđanja budućeg ponašanja. Stoga, danas postoji jako velik broj različitih sustava koji implementiraju i izvode neke od postupaka i modela strojnog učenja.

Ovaj rad podrazumijeva idejno i programski izvedeno rješenje (sustav) zadanog zadatka diplomskog rada, gdje je cilj, korištenjem različitih odgovarajućih tehnologija, razviti sustav za prilagodljivu ugradnju i izvođenje postupaka i modela strojnog učenja, analizirajući pri tome same odabrane postupke i modele, kao i rad realiziranog sustava uz razradu arhitekture i funkcionalnih razina sustava. Rad je usredotočen na konkretnu primjenu sustava strojnog učenja za analizu sentimenta sa slikovnih podataka.

U drugom poglavlju rada se ponajprije pruža teorijska podloga o strojnom učenju potrebna za razumijevanje ostatka rada – korištenih pojmova, principa djelovanja odabranih postupaka te način vrednovanja modela strojnog učenja. Zatim se u trećem poglavlju općenito obrađuju sustavi za implementaciju postupaka i modela strojnog učenja, opširnije se opisuje područje konkretne primjene sustava (analiza sentimenta) te se objašnjavaju neki od postojećih sustava za implementiranje strojnog učenja u tu primjenu. U četvrtom poglavlju opisan je koncept odabranog rješenja, korišteni algoritam strojnog učenja i njegovo konfiguriranje te princip rada i arhitektura cijelog sustava. Nadalje, u petom su poglavlju ukratko opisane korištene tehnologije i okruženja potrebna za programsku realizaciju sustava, a u šestom se poglavlju detaljno obrađuje način implementacije tog sustava korištenjem opisanih tehnologija. Konačno, u sedmom poglavlju, obrađeno je testiranje modela i rada sustava uz pružanje analize dobivenih rezultata i osvrta na cjelokupan sustav.

1.1. Zadatak diplomskog rada

U diplomskom radu potrebno je opisati zahtjeve na razvoj sustava koji omogućuje prepoznavanje pozitivnog ili negativnog osjećanja sa slike s ciljem korištenja tih pristupa u

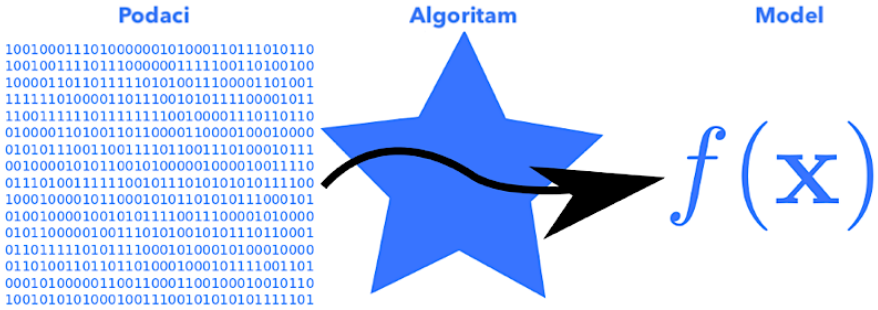
različitim primjenama (npr. za ocjenjivanje zadovoljstva kupaca pri kupovini i za slične primjene). Također, potrebno je analizirati prikladne postupke i modele strojnog učenja koji omogućuju analizu osjećaja subjekta (engl. *sentiment analysis*) sa slika. Nadalje, treba razraditi razine i funkcionalne komponente arhitekture sustava uključujući korisničko sučelje, sučelje za razmjenu podataka, kontejner koji omogućuje izvođenje primijenjenih postupaka i modela strojnog učenja, te rad s bazom podataka. Koristeći prikladne metodologije razvoja poslovnih programskih rješenja, tehnologije, okoline i jezike (Python, JavaScript, Jupyter Notebooks, Node.js), potrebno je programski ostvariti, ispitati i analizirati sustav primijenjen za vizualnu analizu sentimenta.

2. STROJNO UČENJE

U ovom poglavlju je obrađena osnovna teorijska podloga strojnog učenja potrebna za razumijevanje problematike rada (s naglaskom na metode i postupke korištene u ovom diplomskom radu). Pobliže je pojašnjen sam pojam strojnog učenja i njegov princip rada, obrađena je tematika umjetnih neuronskih mreža i dubokog učenja te korištenih konvolucijskih neuronskih mreža.

2.1. Pojam strojnog učenja

Prema [1], strojno učenje (engl. *Machine Learning*, ML) se definira kao oblik umjetne inteligencije koji omogućuje sustavu da uči iz podataka, a ne kroz eksplicitno programiranje. Strojno učenje je, dakle, jedan od podskupova umjetne inteligencije, uz, primjerice, obradu prirodnog jezika (engl. *Natural Language Processing*, NPL). Podrazumijeva kompleksan proces koji koristi skup različitih algoritama koji iterativno uče iz podataka kako bi se poboljšali, opisali podatke i predvidjeli ishode. Kako prikazuje slika 2.1, tri su osnovna elementa na kojima se temelji princip strojnog učenja – podaci, algoritam (postupak) te model.



Slika 2.1 *Osnovni koncept strojnog učenja* [2]

Algoritam općenito podrazumijeva niz naredbi koje je potrebno izvršiti kako bi se određen ulaz pretvorio u odgovarajući izlaz. Te naredbe mogu biti eksplicitno zadane ako je poznato kako se i po kojem pravilu ulaz transformira u izlaz. Međutim, u slučaju da isto nije poznato, nedostatak znanja moguće je nadomjestiti podacima. **Algoritam** strojnog učenja je algoritam koji je sposoban učiti iz podataka. Na osnovu ulaznih **podataka**, odnosno skupova podataka (engl. *dataset*) algoritmi oblikuju i daju određeni model. Taj se proces strojnog učenja naziva treniranje.

Model strojnog učenja, kako je definirano u [1], je upravo proizvod generiran algoritmom (postupkom) strojnog učenja kada se on istrenira podacima. Model može biti predviđajući (engl.

predictive) koji radi predviđanja u budućnosti, odnosno, daje predikciju, te opisni (engl. *descriptive*) koji stječe znanje o podacima. Nakon treniranja, kada se generiranom modelu pruži određen ulaz, dobit će se određeni izlaz. To je proces testiranja modela. Cilj je izgraditi modele koji iskazuju dobra svojstva (engl. *features*), odnosno, na temelju naučenih podataka, mogu uspješno predvidjeti svojstva novih podataka.

Kako je navedeno u [3], strojno učenje je programiranje računala kako bi se optimizirao određeni kriterij izvedbe koristeći podatkovne primjere ili prethodna iskustva. Raspolaže se modelom koji je definiran do na neke parametre, a sam proces učenja podrazumijeva optimizaciju tih parametara modela na temelju podataka (iskustva). Strojno učenje, kao takvo, pronalazi svoju primjenu u širokom spektru djelatnosti koje obično karakteriziraju kompleksni problemi (ne postoji ljudsko znanje ili objašnjenje procesa), sustavi koji se mijenjaju dinamički (potrebne su prilagodbe) ili ogromna količina podataka (uočavanje pravila i uzoraka te izvlačenje znanja odnosno relevantne informacije). U skladu s tim razvijene su i koriste se različite vrste algoritama i modela strojnog učenja kao što su strojevi s potpornim vektorima (engl. *Support Vector Machine*, SVM), Bayesove mreže (engl. *Bayesian networks*), stabla odluke (engl. *decision trees*), umjetne neuronske mreže (engl. *Artificial Neural Networks*, ANN) i drugi.

Neovisno o kojoj vrsti algoritma i modela se radi, nekoliko je osnovnih pristupa strojnom učenju, ovisno o vrsti i količini podataka. Tako razlikujemo:

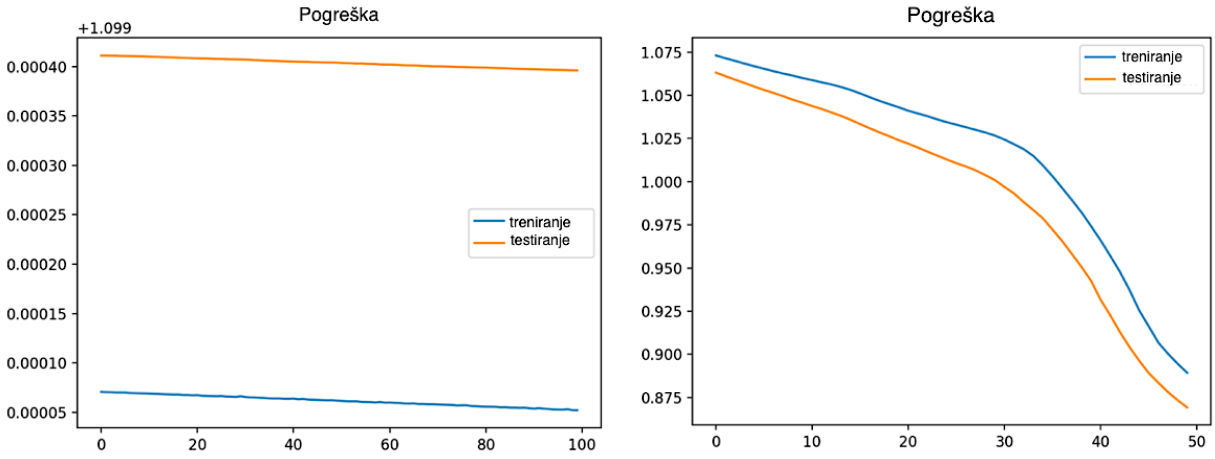
- **Nadzirano učenje** (engl. *supervised learning*) – cilj je naučiti mapiranje ulaza u izlaz čije ispravne vrijednosti daje nadzornik. Model se gradi na temelju označenih skupova podataka koji sadrže i ulaz (ulazni značajke koji se analiziraju) i odgovarajući željeni izlaz (oznaka, engl. *label*). Podaci su, dakle, uređeni parovi ulaza i izlaza te je potrebno pronaći funkciju koja preslikava ulaz u izlaz. Ako su izlazi diskretne ili nebrojčane vrijednosti (klase koje se pridjeljuju ulazima), odnosno, ograničeni su na točno određen skup vrijednosti, radi se o klasifikaciji (engl. *classification*), a ako su izlazi kontinuirane brojčane vrijednosti, odnosno mogu imati bilo koju brojčanu vrijednost unutar zadanog raspona, radi se o regresiji (engl. *regression*).
- **Nenadzirano učenje** (engl. *unsupervised* learning) – ne postoji nadzornik kao kod nadziranog učenja, postoje samo ulazni podaci. Cilj je, na osnovnu velikih količina neoznačenih ulaznih podataka, pronaći određene pravilnosti u podacima. Neke od takvih vrsta algoritama su grupiranje (engl. *clustering*), kojem je cilj pronaći klastere odnosno grupe uzoraka i grupirati ulazne podatke po njihovim značajkama, procjena gustoće (engl.

density estimation), kojom algoritam pokušava odrediti funkciju gustoće vjerojatnosti uzoraka, i drugi.

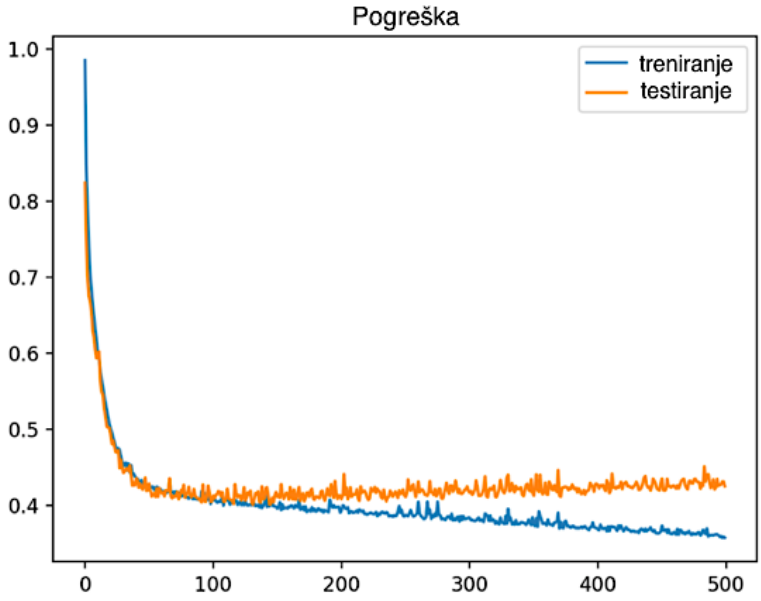
2.2. Vrednovanje modela i algoritma strojnog učenja

Središnji izazov u strojnom učenju je da algoritam iskazuje dobra svojstva na novim, prethodno neviđenim podacima – ne samo na onim na kojima je model treniran. Takva se sposobnost algoritma da radi dobro na prethodno nepromatranim ulazima naziva **generalizacija**. [4] Prilikom treniranja, nastoje se poboljšati svojstva i rezultati algoritma nad treniranim podacima – određuje se mjera pogreške na skupu za treniranje (**pogreška treniranja**, engl. *training error/loss*) te se ona nastoji minimizirati. To je proces **optimizacije** algoritma odnosno modela strojnog učenja. S druge strane, za ispitivanje sposobnosti generaliziranja algoritma, koristi se generalizacijska pogreška ili **pogreška testiranja** (engl. *test error/loss*, u nekim izvorima i *validation error/loss*) za koju je također poželjno da bude što manja kako bi model što bolje generalizirao. Definira se kao očekivana vrijednost pogreške modela na novim ulaznim podacima, a obično se procjenjuje mjerenjem izvedbe algoritma na odvojenom, testnom (validacijskom) skupu uzoraka. Podaci za treniranje i testiranje generirani su razdiobom vjerojatnosti nad skupom podataka uz pretpostavke da su uzorci u svakom skupu podataka međusobno neovisni te da su skup za treniranje i skup za testiranje podjednako raspodijeljeni. Navedeno omogućuje proučavanje i povezanost pogreške treniranja i pogreške testiranja.

Pogreška treniranja obično je uvijek manja od pogreške testiranja te postoji određen raskorak između te dvije vrijednosti. Izvedba algoritma strojnog učenja ogledat će se upravo u njegovoj sposobnosti da učini pogrešku treniranja što manjom te minimizira raskorak između pogreške treniranja i pogreške testiranja. Ta dva principa odgovaraju dvjema bitnim pojavama u strojnom učenju – **podnaučenost** (engl. *underfitting*) i **prenaučenost** (engl. *overfitting*). Podnaučenost se pojavljuje kada model ne može postići dovoljno nisku vrijednost pogreške treniranja odnosno nije sposoban učiti iz skupa podataka za treniranje (Slika 2.2). Uzroci su obično nepostojanje adekvatnog kapaciteta modela za složenost skupa podataka ili preuranjeno obustavljanje procesa treniranja. Prenaučenost se pojavljuje kada je raskorak između pogreške treniranja i pogreške testiranja prevelik (Slika 2.3), odnosno, kada je model previše vezan i naučen na skup podataka za treniranje – počinje usvajati i statistički šum i slučajna kolebanja u skupu te je sve manje sposoban generalizirati odnosno dati dobre rezultate na novim podacima. Obično se javlja kada je kapacitet modela veći nego je potreban za zadanu složenost skupa podataka ili kada se model trenira predugo.

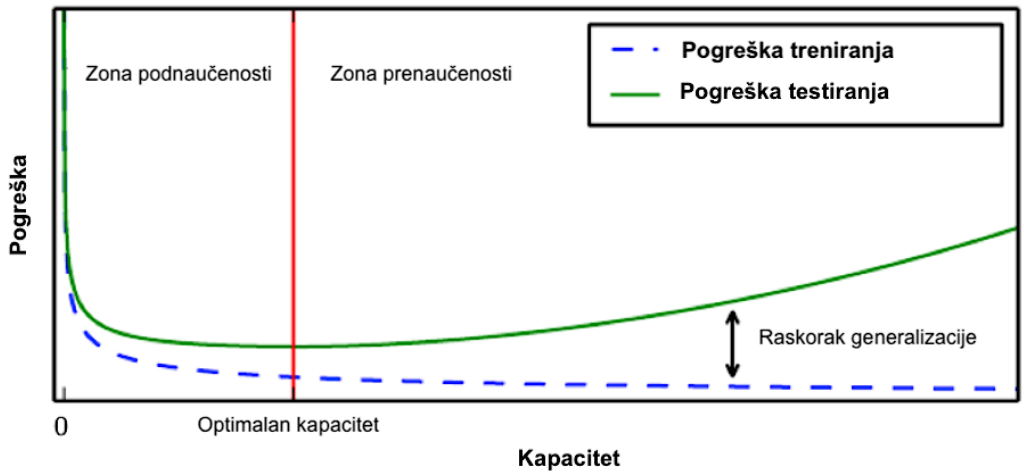


Slika 2.2 *Primjeri podnaučenosti modela – izgled pogreški* [5]



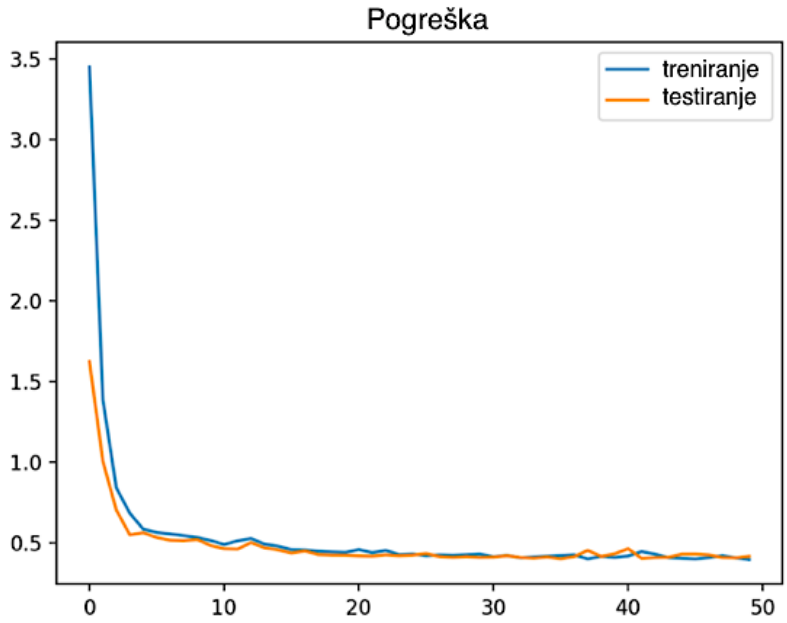
Slika 2.3 *Primjer prenaučenosti modela – izgled pogreški* [5]

Da se uočiti da na izvedbu algoritma i svojstva modela uvelike utječe spomenuti kapacitet modela. To je sposobnost podešavanja modela pri učenju koja predstavlja ključni faktor koji određuje hoće li model biti podnaučen ili prenaučen, kako je i prikazano na slici 2.4. Optimalan izbor modela leži upravo u odabiru odgovarajućeg kapaciteta modela – on treba biti najjednostavniji mogući kako razlika između pogreški testiranja i treniranja ne bi bila prevelika, odnosno, kako bi se izbjegla prenaučenost, ali opet dovoljno kompleksan kako bi se postigla dovoljno niska pogreška treniranja.



Slika 2.4 *Odnos kapaciteta i pogreški modela* [4]

Optimalnim kapacitetom se postiže dobro podešeni (engl. *good fit*) model kod koga se pogreška treniranja i pogreška testiranja snižavaju do točke stabilnosti s minimalnim raskorakom između konačnih vrijednosti pogreški treniranja i testiranja (raskorak generalizacije).



Slika 2.5 *Primjer dobro podešenog modela - izgled pogreški* [5]

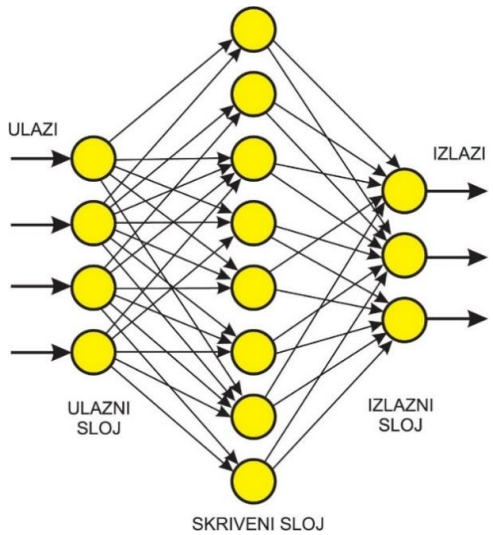
Postoje različite metode kojima je moguće izraziti preferiranje drugačijeg rješenja odnosno poboljšanja svojstava i rezultata modela. Algoritam se izmjenjuje s ciljem da se reducira pogreška generalizacije odnosno pogreška testiranja, ali ne i greška treniranja. To je proces **regularizacije** i skupa s procesom optimizacije čini okosnicu strojnog učenja.

Pri vrednovanju algoritma i modela strojnog učenja, bitno je obratiti pažnju i na **hiperparametre** algoritma – postavke algoritma koje se koriste za kontrolu ponašanja algoritma, a čije vrijednosti sam učeći algoritam ne može prilagođavati. Tu pripadaju primjerice kapacitet modela (složenost modela, broj iteracija), veličina serije (engl. *batch size*), veličina validacijskog skupa (engl. *validation set*) i drugi.

2.3. Umjetne neuronske mreže i duboko učenje

Jedan od oblika strojnog učenja su **umjetne neuronske mreže** (engl. *Artificial Neural Networks*, ANN) koje se zasnivaju na načinu na koji ljudski mozak obrađuje informacije i uči, točnije, na neuronima (živčanim stanicama) – sav proces se temelji na vezama između neurona i njihovim relativnim snagama (težinama). Na tom su se principu izgradile i umjetne neuronske mreže kojima se pokušavaju oponašati takvi procesi učenja. Struktura neuronske mreže neovisna je o zadatku koji treba obaviti – jedina stvar koja se mijenja su parametri međusobnih veza različitih neurona. Stoga učenje kod umjetnih neuronskih mreža zapravo podrazumijeva adekvatno podešavanje tih parametara veza (težina) dok se ne postigne zadovoljavajuća relacija ulaza i izlaza.

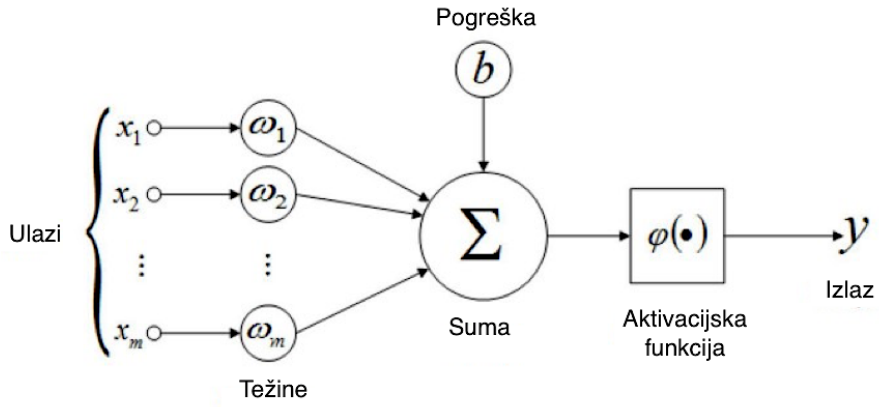
Umjetna neuronska mreža posjeduje slojevitu strukturu – sastoji se od uzastopnih slojeva neurona. Kako je prikazano i na slici 2.6, obično se radi o tri ili više slojeva: ulazni sloj, jedan ili više skrivenih slojeva te izlazni sloj.



Slika 2.6 *Osnovna struktura umjetne neuronske mreže* [6]

Krugovi na navedenoj slici predstavljaju neurone, a strjelice veze između pojedinih neurona. Broj skrivenih slojeva te broj neurona po sloju definirat će složenost modela i izvedbu mreže pa se odabire u skladu sa zadanim zadatkom mreže i ulaznim podacima. Prvi, ulazni sloj

oblikuju ulazni podaci, njihovi elementi i dimenzije, a krajnji, izlazni sloj je obično potpuno povezani (engl. *fully connected*) sloj koji daje željeni izlaz (broj neurona odgovara broju izlaza). Svakoj od veza između neurona pridružen je određen broj koji se naziva težina veze, a svakom neuronu pridružen je broj koji se naziva vrijednost praga te funkcija koja se naziva aktivacijska funkcija. Navedeni elementi skupa čine parametre neuronske mreže. Slika 2.7, po uzoru na [4], prikazuje „unutrašnjost" jednog **neurona** odnosno njegov model. Neuron za svaku od ulaznih veza korigira ulaznu vrijednost, odnosno, množi ulaz s težinom veze, sumira sve korigirane ulazne vrijednosti (množenje matrica ulaza i težina) te dobivenu sumu prosljeđuje kroz aktivacijsku funkciju koja definira izlaz neurona (primjerice „1", ako je suma veća ili jednaka definiranoj vrijednosti praga, a „0" u suprotnom).



Slika 2.7 *Model umjetnog neurona*

Proces treniranja se u svojoj osnovi odvija na idući način: mreža uzima ulazne podatke skupa s odgovarajući izlazima, šalje ih i obrađuje kroz svaki neuron svakog od sloja mreže, pri čemu, koristeći spomenute parametre mreže, svaki od neurona na određen način preoblikuje ulaz te ga šalje sljedećem sloju, sve do krajnjeg, izlaznog sloja. Rezultati koje mreža dobije na izlaznom sloju se uspoređuju s početnim očekivanim izlazima, određuje se odstupanje te se u skladu s tim podešavaju parametri (težine i pragovi) svakog od neurona kroz posebne algoritme (primjerice silazni gradijent ili povratno rasprostiranje) koji za cilj imaju što veće približavanje očekivanih i proizvedenih izlaza. Taj se proces ponavlja veliki broj puta te model uči.

U svrhu učenja jako složenih problema odnosno treniranja kompleksnih modela za predviđanje i klasifikaciju informacija koje ovisi o tisućama ili pak milijunima značajki, koriste se složenije umjetne neuronske mreže većeg kapaciteta koje sadrže veći broj skrivenih slojeva te veći broj neurona po skrivenom sloju. To su duboke neuronske mreže (engl. *Deep Neural Networks*, DNN) koje sadrže ukupno više od tri sloja neurona (dva ili više skrivena sloja), a proces učenja takvih mreža se naziva **dubokim učenjem** (engl. *Deep Learning*). Kako je i predočeno na slici

2.8, duboko učenje je podskup strojnog učenja kod kojeg se primjenjuju duboke neuronske mreže za učenje složenih zadataka. Nadalje, strojno učenje je podskup puno većeg skupa umjetne inteligencije (engl. *Artificial Intelligence*, AI), točnije uske (slabe) umjetne inteligencije (engl. *Narrow/Weak AI*) koja je usredotočena na jedan specifičan zadatak, za razliku od opće (jake) umjetne inteligencije (engl. *General/Strong AI*) koja podrazumijeva primjenu inteligencije na bilo koji problem odnosno zadatak (po uzoru na čovjeka).

Slika 2.8 *Hijerarhijski odnos umjetne inteligencije, strojnog i dubinskog učenja (napomena: razmjeri veličina podskupova na slici se ne odnose na stvarne omjere i udjele)*

Duboko učenje pronalazi svoju primjenu u širokom spektru aktivnosti, naročito s razvojem tehnologije i stvaranjem sve većih količina podataka – primjerice za predviđanja u sustavima interneta objekata, za automatsko prepoznavanje govora velikih razmjera, u području računalnog vida i prepoznavanja slike, gdje čak nadilaze i ljudske sposobnosti u rezultatima, u medicini, vojsci i sličnom. U skladu s područjem primjene, zadatkom koji trebaju obaviti i podacima koje trebaju obraditi, postoje i različite arhitekture dubokog učenja poput povratnih neuronskih mreža (engl. *Recurrent Neural Networks*, RNN), dubokih mreža vjerovanja (engl. *Deep Belief Networks*), konvolucijskih neuronskih mreža (engl. *Convolutional Neural Networks*, CNN) i drugih.

2.4. Konvolucijske neuronske mreže

Za razliku od klasičnih dubokih neuronskih mreža, umjesto množenja matrica ulaza i težina veza, konvolucijske neuronske mreže koriste operaciju konvolucije u barem jednom od slojeva.

Konvolucija podrazumijeva posebnu vrstu linearne matematičke operacije nad dvije funkcije realnih varijabli koja kao rezultat daje treću funkciju kojom se izražava kako se oblik jedne izmjenjuje drugom funkcijom. Definicija konvolucije prikazana je u izrazu (2-1).

$$h(t) = (f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau \tag{2–1}$$

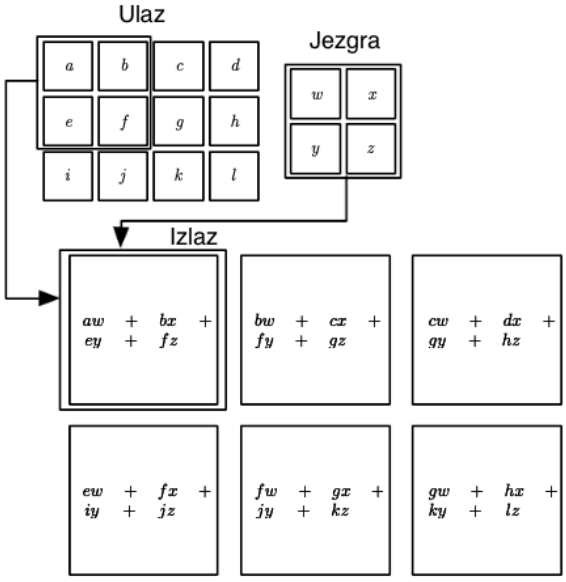
Prvi argument konvolucije (prva funkcija) se obično naziva ulaz, drugi jezgra (engl. *kernel*), a funkcija dobivena kao rezultat – mapa značajki (engl. *feature map*). U slučaju diskretnih funkcija koristi se izraz (2-2).

$$h(t) = (f * g)(t) = \sum_{n=-\infty}^{\infty} f(n) g(t - n) \tag{2–2}$$

Budući da je u strojnom učenju ulazna funkcija obično višedimenzionalni niz podataka (tenzor), a i sama konvolucijska jezgra najčešće višedimenzionalni niz parametara podešavanja algoritma (tenzor), često se koristi i višedimenzionalni oblik konvolucije. Izraz (2-3) definira konvoluciju za dvodimenzionalan ulaz i jezgru (primjerice slike).

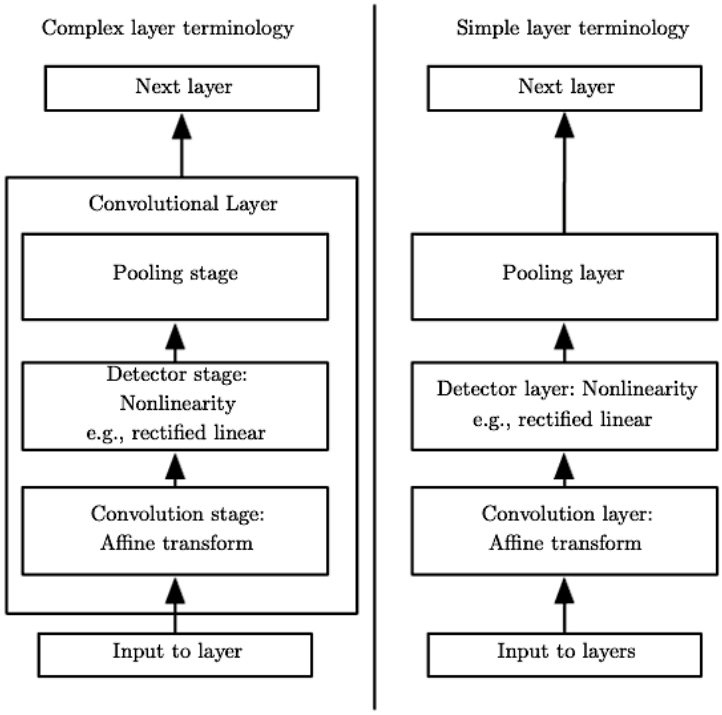
$$H(i,j) = (I * K)(i,j) = \sum_{m} \sum_{n} I(m,n)\, K(i - m, j - n) \tag{2–3}$$

Slika 2.9 prikazuje princip rada konvolucije na jednostavnom dvodimenzionalnom ulazu i dvodimenzionalnoj jezgri.



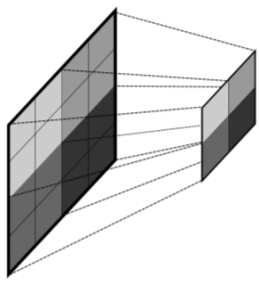
Slika 2.9 *Princip rada konvolucije (na dvodimenzionalnom ulazu i jezgri)* [4]

Konvolucijska neuronska mreža, dakle, sadrži barem jedan sloj u kojem se primjenjuje operacija konvolucije. Takav sloj nazivamo **konvolucijski sloj**. Prema [4], tipičan konvolucijski sloj sastoji se iz tri faze (Slika 2.10).



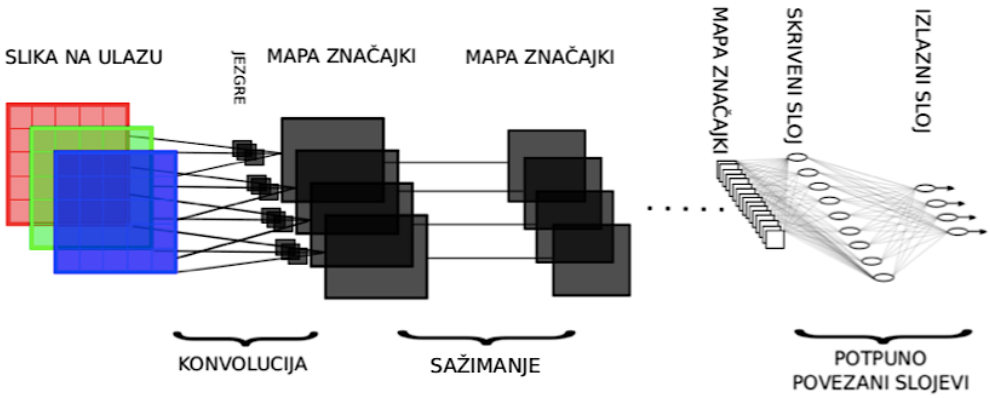
Slika 2.10 *Tri faze tipičnog konvolucijskog sloja* [4]

Prva je konvolucijska faza u kojoj sloj izvodi nekoliko konvolucija usporedno kako bi se proizveo skup linearnih aktivacija te se dobivaju mape značajki. Druga faza je faza detekcije u kojoj se svaka od linearnih aktivacija iz prethodne faze propušta kroz nelinearnu aktivacijsku funkciju (primjerice ispravljena linearna aktivacijska funkcija, engl. *Rectified Linear Unit,* ReLU). U trećoj fazi, fazi **sažimanja**, koristi se neka od funkcija sažimanja kako bi se izmijenio izlaz sloja – mape značajki se reduciraju (primjer na slici 2.11).



Slika 2.11 *Operacija sažimanja značajki* [7]

Funkcije sažimanja preslikavaju skup prostorno bliskih značajki ulaza (grupirani skup podataka zadane veličine, primjerice 2 × 2 kako je prikazano na slici 2.11) u jednu značajku izlaza – time se postiže invarijantnost na male translacije i rotacije ulaza jer će translatirani i rotirani ulazi dati istu značajku na izlazu, a uz to se smanjuju i memorijski kapaciteti potrebni za realizaciju takve konvolucijske mreže. Neke od funkcija sažimanja su, na primjer, sažimanje maksimalnom vrijednosti (engl. *max pooling*), sažimanje srednjom vrijednosti (engl. *average pooling*) te sažimanje L2 normom. Na slici 2.12 prikazana je klasična struktura konvolucijske mreže.



Slika 2.12 *Klasična struktura konvolucijske neuronske mreže* [7]

Dakle, konvolucijski sloj uzima ulazne podatke (sliku u prikazanom primjeru), nad njima obavlja konvoluciju konvolucijskom jezgrom (koja se u ovom slučaju ponaša kao filter koji filtrira ulaznu sliku i izvlači korisne značajke, primjerice rubove ili oblik) te proizvodi mapu značajki koja se sažimanjem reducira. Proces se ponavlja za svaki konvolucijski sloj sve do krajnjih slojeva koji su potpuno povezani slojevi i omogućuju dobivanje skalarnih izlaza. Veći broj konvolucijskih slojeva omogućuje i veću razinu apstrakcije značajki pa se mogu detektirati složeniji oblici i objekti. Konvolucijske neuronske mreže vrlo su efikasne u analizi dvodimenzionalnih podataka (slika), manje su kompleksne te su procesorski manje zahtjevne od, primjerice, potpuno povezanih mreža. Zbog svoje arhitekture dijeljenih težina i translacijske invarijantnosti, imaju karakteristike prostorno invarijantnih umjetnih neuronskih mreža te mogu uz relativno malu količinu predobrade (engl. *preprocessing*) naučiti filtere za analizu i klasifikaciju slika. Stoga, nailaze na široku primjenu u području računalnog vida, prepoznavanja sa slike i videa, klasifikacije slika, analize medicinskih slika te obrade prirodnog jezika.

3. SUSTAVI ZA IMPLEMENTACIJU STROJNOG UČENJA

Sustavi za implementaciju strojnog učenja podrazumijevaju različita okruženja, biblioteke, programe i programske pakete koji uspješno primjenjuju neke od brojnih modela i postupaka strojnog učenja s određenim ciljem – najčešće analiza podataka i izvlačenje relevantnih informacija, detekcija različitih pojava i potencijalnih korelacija unutar i između skupova podataka, prepoznavanje objekata, računalni vid ili slično. U ovom je poglavlju obrađena jedna od mogućih primjena (analiza sentimenta), prodiskutirana su neka od sličnih postojećih rješenja odnosno takvih sustava te su opisane značajke i zahtjevi na traženi sustav.

3.1. Analiza sentimenta

Analiza sentimenta (engl. *sentiment analysis*), često nazivana i rudarenje mišljenja (engl. *opinion mining*) ili umjetna inteligencija emocija (engl. *emotion AI*), prema [8], područje je proučavanja koje analizira ljudska mišljenja, osjećaje, procjene i stavove prema subjektima kao što su proizvodi, usluge, organizacije, pojedinci, pitanja, događaji, teme i njihove osobine. Podrazumijeva računalno proučavanje ljudskih mišljenja, osjećaja, raspoloženja, procjena i emocija te je jedno od najaktivnijih područja istraživanja i primjene obrade prirodnog jezika, rudarenja podataka, informacija i web sadržaja, kao što je navedeno u [9]. Iako se u konačnici svodi na problem klasifikacije sadržaja na pozitivne ili negativne sentimente (osjećanja) ili mišljenja, analiza sentimenta puno je složeniji problem koji za sobom povlači kompleksne metode rješavanja. Kako je opisano u [10], analiza sentimenta se definira kao proces računalnog klasificiranja i kategoriziranja sentimenta izraženog u određenom multimedijskom sadržaju (tekstualnom ili netekstualnom) s ciljem određivanja orijentiranosti stava vlasnika sadržaja prema specifičnoj temi, proizvodu, usluzi i sl. Pokušava se utvrditi je li izražaj poželjan, nepoželjan ili neutralan.

Dolazak i razvoj interneta, društvenih mreža i aplikacija drastično je izmijenio način na koji ljudi izražavaju mišljenja, osjećaje i stavove te je zbog toga analiza sentimenta, sa sve većim porastom takvog sadržaja, i više nego aktualna i uvelike korištena u poslovanju. Tako, na primjer, tvrtke i organizacije žele saznati mišljenja javnosti, korisnika i kupaca o vlastitim proizvodima i uslugama, a i sami individualni korisnici žele znati mišljenja drugih, postojećih korisnika proizvoda ili usluge prije same kupnje ili korištenja. Navedeno je moguće ostvariti upravo analizom sentimenta na osnovu ogromnih količina podataka koji su dostupni na društvenim

mrežama (objave, slike, komentari, recenzije i slično). Analiza sentimenta, u svojoj osnovi, većinom podrazumijeva obradu tekstualnih informacija (riječi, rečenica i dokumenata), ali sve više uključuje i ispitivanje glasovnih informacija i zvuka te analizu vizualnog sentimenta, obradu vizualnog sadržaja i slike.

Budući da se analiza sentimenta bavi proučavanjem mišljenja, bitno je definirati ključne pojmove na kojima se zasniva. **Mišljenje** (engl. *opinion*), u kontekstu analize sentimenta, podrazumijeva široki koncept koji uključuje osjećaje, procjene, stavove i slično, a sastoji se iz dva osnovna pojma – samog **sentimenta** odnosno pozitivnog ili negativnog osjećanja kojeg podrazumijeva mišljenje te koje ima svoj tip, orijentaciju i intenzitet te **mete** mišljenja (engl. *target*) odnosno predmeta mišljenja prema kojem je sentiment usmjeren. Mete su obično svojstva različitih entiteta primjerice proizvoda, usluga, tema, osoba, organizacija, problema događaja i sličnoga. Dva su osnovna tipa sentimenta – racionalni sentiment, koji dolazi iz racionalnog zaključivanja, opipljivih uvjerenja i utilitarnih stavova te ne izražavaju emocije, te emocionalni sentiment, koji dolazi iz neopipljivih i emocionalnih reakcija i ljudskih psiholoških stanja. Orijentacija sentimenta može biti pozitivna, negativna te neutralna koja obično podrazumijeva odsustvo sentimenta ili mišljenja, a intenzitet sentimenta obično se u primjenama izražava diskretnim ocjenama (primjerice 1 – 5 zvjezdica). Uz spomenute osnovne pojmove se često nadovezuju i pojmovi **nositelja mišljenja** (engl. *opinion holder*) odnosno subjekta, organizacije ili osobe koje je vlasnik izraženog mišljenja te **vrijeme mišljenja** (engl. *time of opinion*) u kojem je mišljenje izraženo te za koje vrijedi. Mišljenje se tada može promatrati kao uređena četvorka mete mišljenja, sentimenta, nositelja mišljenja i vremena, kako je definirano u [9]. Dakle, pri proučavanju i analizi sentimenta, za praktične primjene, bitno je poznavati ne samo sentiment već i njegov kontekst.

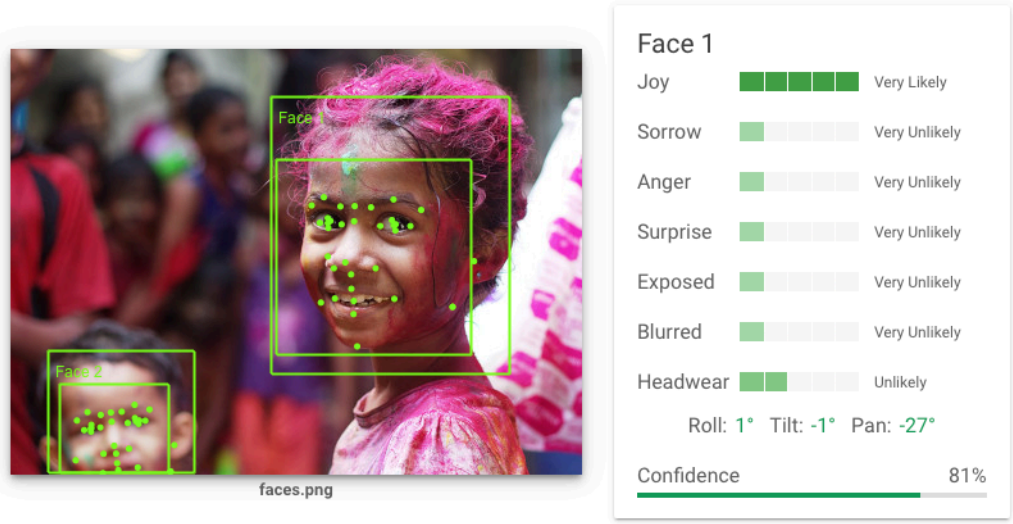
3.2. Postojeći sustavi za implementaciju strojnog učenja

Kako je opisano u prethodnom poglavlju, analiza sentimenta ima veoma rasprostranjenu primjenu u svijetu tehnologije te je razvijen velik broj sustava za implementaciju strojnog učenja u tu svrhu. Iako se većina bazira na obradi tekstualnih informacija, u novije vrijeme sve je više sustava koji se primjenjuju za analizu vizualnog sentimenta. U ovom potpoglavlju su ukratko opisani neki od popularnih sustava koji implementiraju različite oblike strojnog učenja za obradu vizualnih podataka.

3.2.1. Google Vision AI

Google Vision AI projekt je razvijen od strane Googlea te nudi nekoliko opcija odnosno proizvoda koji se mogu koristiti za integriranje strojnog učenja vizualnih modela u vlastite aplikacije ili web stranice te koji se grupiraju u dvije osnovne usluge – AutoML Vision i Vision API. AutoML Vision podrazumijeva različite proizvode i alate koji omogućuju treniranje vlastitih modela strojnog učenja te ih optimiziraju ih po pitanju preciznosti, kašnjenja i veličine. Vision API nudi pristup snažnim već istreniranim modelima strojnog učenja kroz REST i RPC aplikacijska programska sučelja. Omogućava jednostavnu implementaciju strojnog učenja u sustav te integriranje značajki vizualnog detektiranja u aplikacije (označavanje slika, detekcija lica i znamenitosti, optičko prepoznavanje znakova, lokalizacija objekata, označavanje eksplicitnog sadržaja i slično). Pri usporedbi s traženim sustavom u diplomskom radu, od važnosti je upravo značajka detekcije lica Vision API-ja. Navedeni sustav detekcije lica (engl. *face detection*) omogućuje detekciju više lica na slici skupa s povezanim ključnim svojstvima lica (primjerice emocionalno stanje osobe). Prepoznavanje lica nije podržano.

Za upotrebu i implementaciju u postojeći sustav potrebno je stvoriti Google Cloud Platform (GCP) projekt te u postojeći klijentski sustav (aplikaciju, program) uključiti Google Cloud Vision biblioteku. Funkcionalnosti se ostvaruju kao poziv klijentskog sustava prema Google Vision serveru koji vraća potrebne značajke u određenom formatu koje klijent može dalje obrađivati. U odgovoru servera (JSON format) se nalaze detektirana lica skupa s njihovim pozicijama, pozicijama svake od karakterističnih značajki lica (primjerice oči, obrve, nos, uši i slično) te procjenom emocionalnog stanja lica i njenom pouzdanošću. Slika 3.1 prikazuje izgled analize lica u Googleovom API Exploreru.



Slika 3.1 *Izgled korištenja Google Vision API-ja u Google API Exploreru*

Procjena emocionalnog stanja podrazumijeva opisne vjerojatnosti da osoba iskazuje emocije radosti, tuge, ljutnje i iznenađenja (vrlo vjerojatno, vjerojatno, malo vjerojatno i vrlo malo vjerojatno). Navedeni sustav pokazuje dobra svojstva za slike lica, gdje ispravno zaključuje o trenutno izraženom sentimentu osobe na njenom licu, što je pogodno za primjenu u analizi sentimenta i ispitivanju zadovoljstva korisnika. Sustav je jednostavan i fleksibilan te vrlo lako ugradiv u postojeće aplikacije, programe i sustave. Korištenje Vision API-ja je besplatno za prvih 1000 slika mjesečno, a za veći broj se naplaćuje.

3.2.2. DeepFace

DeepFace je projekt tvrtke Systemnegar Saina namijenjen integriranju prepoznavanja i analize lica u sustave. Koristi metode dubokog učenja te osigurava vrhunsku preciznost, efikasan odaziv u stvarnom vremenu pogodan za aplikacije u stvarnom svijetu te pouzdanost i visoku dostupnost. Nudi četiri usluge – detekcija lica i znamenitosti, analiza lica i svojstava, ispitivanje izraza lica i emocija te verifikacija, pretraga i ispitivanje sličnosti lica.

Detekcija lica je osnovna usluga te prvi korak pri daljnjoj analizi i obradi lica – DeepFace algoritmi detektiraju i lociraju lice na slici i uokviruju ga okvirima visoke preciznosti. Analiza lica i svojstava kao rezultat daje procjenu atributa lica kao što su dob, spol, položaj glave, stanje očiju, boju kože i slično. Verifikacija provjerava vjerojatnost da dva lica pripadaju istoj osobi, uz prikaz rezultata i pragova pouzdanosti kojima se procjenjuje sličnost. Analiza izraza lica i emocija prepoznaje pozitivne, negativne i neutralne sentimente na licima te klasificira emocije lica u radost, ljutnju, gađenje, tugu, strah i iznenađenje. Upravo se ta usluga analize emocija s lica osobe, u eventualnoj kombinaciji s ostalima, može koristiti za analizu sentimenta i procjenu zadovoljstva korisnika. Sustav se integrira u postojeću aplikaciju preko DeepFace aplikacijskog programskog sučelja. Korištenje istog se naplaćuje.

3.3. Karakteristike sustava i zahtjevi na sustav

Zadatak ovog rada, kako je opisano u poglavlju 1.1, podrazumijeva primjenu sustava s implementiranim strojnim učenjem u svrhu analize sentimenta odnosno prepoznavanja osjećanja (pozitivnog ili negativnog) na temelju sadržaja slike. Prema tome, zahtjevi na sustav su:

- Sustav treba uspješno implementirati određen oblik strojnog učenja odnosno koristiti model strojnog učenja za predviđanje.
- Primjena sustava je vizualna analiza sentimenta (sa slikovnog sadržaja).

- Ulazi sustava su slike (jedna po zahtjevu ili kontinuirana analiza slika – video).
- Izlazi sustava su informacije o sentimentu sa slike – je li on pozitivan, negativan ili neutralan uz eventualnu informaciju o razini pouzdanosti procjene.
- Razina pouzdanosti procjene treba biti dovoljno visoka odnosno zadovoljavajuća za područje primjene.
- Model strojnog učenja sustav gradi i trenira na adekvatnoj bazi podataka (slika). Različiti skupovi slika trebaju biti podržani.
- Sustav treba biti modularno izgrađen – funkcionalno razdijeljen na zasebne module.
- Sustav treba imati prilagodljivu ugradnju odnosno mogućnost fleksibilne implementacije odabranog postupka i modela strojnog učenja.
- Sučelja koje sustav treba imati su korisničko sučelje, sučelje za razmjenu podataka, kontejner koji omogućuje izvođenje primijenjenih postupaka i modela strojnog učenja, te rad s bazom podataka.

Karakteristike takvog sustava su jednostavnost, fleksibilnost i modularnost. Realizirano rješenje treba na najjednostavniji mogući način zadovoljiti zahtjeve uz modularnu strukturu i mogućnost prilagodljive ugradnje, a da istovremeno ispunjava svrhu odnosno daje rezultate zadovoljavajuće preciznosti. Budući da je cilj rada razviti fleksibilan sustav, analiza sentimenta će se provesti na konkretnom primjeru palac-gore/palac-dolje principa (engl. *thumbs-up/thumbs-down*), a sustav će se moći po potrebi skalirati i primijeniti i na druge oblike sentimenta (primjerice slike lica). Problematika primjene je usredotočena na određivanje poželjnih osobina odnosno zadovoljstva (palac gore) ili nepoželjnih osobina odnosno nezadovoljstva (palac dolje).

4. ALGORITAM STROJNOG UČENJA ZA ANALIZU SENTIMENTA

U skladu sa zahtjevima na sustav za prilagodljivu ugradnju i izvođenje postupaka i modela strojnog učenja, odabrani su odgovarajući algoritam i model strojnog učenja za analizu vizualnog sentimenta te se u ovom poglavlju opisuju značajke odabranog algoritma i modela, kao i princip rada i arhitektura sustava koji implementira takav model.

4.1. Algoritam i model strojnog učenja

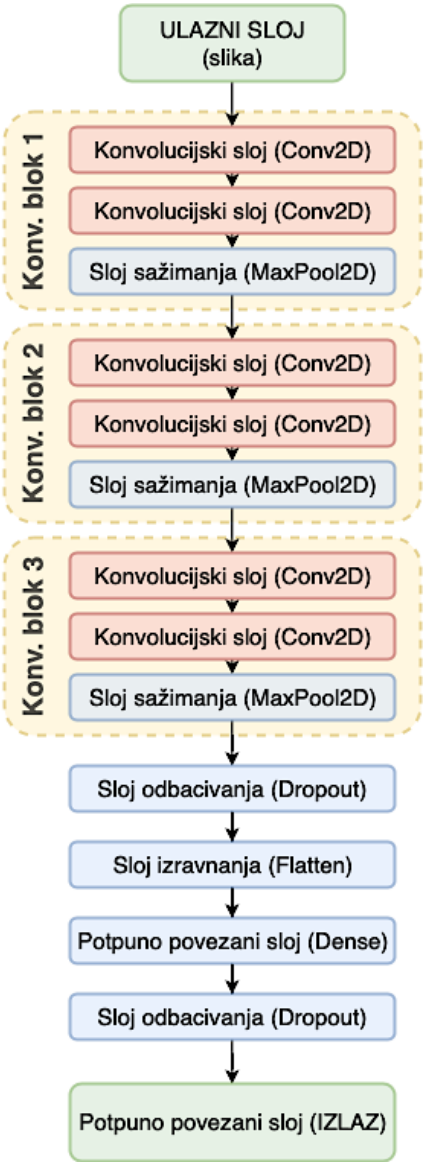
Budući da je primjena traženog sustava analiza vizualnog sentimenta odnosno analiza sentimenta sa slika (dvodimenzionalnih podataka), za algoritam strojnog učenja odabrana je **konvolucijska neuronska mreža** (CNN), opisana u poglavlju 2.4. Konvolucijske neuronske mreže, kao vrsta dubokih neuronskih mreža, općenito daju jako dobre rezultate pri analizi podataka sa slika te se često koriste za zadatke detekcije, identifikacije i prepoznavanja objekata. Odabrani algoritam, u svojoj strukturi mreže slojeva, koristi konvolucijske slojeve, aktivacijske slojeve, slojeve sažimanja, slojeve izravnanja (engl. *flatten*), slojeve odbacivanja (engl. *dropout*) te potpuno povezane slojeve (engl. *fully connected* ili *dense*). Struktura korištene mreže prikazana je na slici 4.1, a detaljnija struktura mreže (sa dimenzijama ulaza i izlaza za svaki sloj) priložena je u prilogu P.4.1.

Ulaz u mrežu predstavlja niz učitanih, prethodno obrađenih i pripremljenih podataka (slika sentimenta) iz baze. Slike su jednokanalne odnosno monokromatske (u sivim tonovima, tj. pikseli sadrže samo informaciju o intenzitetu svjetline) i veličine 224 × 224 piksela te se takve dostavljaju mreži u serijama definirane veličine (odabrana je veličina od 32 slike po koraku) i predstavljaju **ulazni sloj**. Idući sloj je **konvolucijski sloj** koji obavlja konvoluciju nad ulaznim dvodimenzionalnim podacima koje nadopunjuje nulama (engl. *zero padding*) kako bi za zadanu veličinu konvolucijske jezgre, dobiveni izlaz iz sloja bio jednakog oblika odnosno dimenzija kao i ulaz. Odnos između veličine jezgre (F) i dopunskih piksela (P), za jediničnu vrijednost pomaka (S) prikazan je izrazom (4 – 1).

$$P = \frac{F - 1}{2} \tag{4 – 1}$$

Odabrana je veličina konvolucijske jezgre od 5 × 5 piksela te jedinični pomak (engl. *stride*), što znači da je ulazna slika nadopunjena sa po dva okolna piksela nulte vrijednosti pa je izlaz iz konvolucijskog sloja jednakih dimenzija kao i ulaz (224 × 224). Time se održava

konstantnost dimenzija i poboljšava učinak, jer bi se u suprotnom, pri svakoj konvoluciji, dimenzije smanjivale za malu količinu te bi došlo do gubitka podataka.



Slika 4.1 *Struktura korištene konvolucijske neuronske mreže*

Prema [10], jedinični pomaci odnosno maleni pomaci jezgre općenito daju bolje rezultate u praksi, kao i manje veličine jezgri (3×3 ili 5×5) te su stoga korištene takve vrijednosti. Broj filtera odnosno neurona (čvorova) u konvolucijskom sloju odabran je u skladu s eksperimentalnim rezultatima, kompleksnosti i vremenu treniranja modela, broju dobivenih parametara te kompleksnosti baze podataka. Prvi konvolucijski sloj sadrži 16 filtera, a dalje se kroz mrežu taj broj razmjerno povećava. Uloga konvolucijskog sloja je da konvolucijom svakog od filtera (neuron sa svojim težinama) s ulaznim skupom podataka stvori niz dvodimenzionalnih mapa značajki (koji se može shvatiti kao trodimenzionalni volumen podataka) koje predstavljaju

aktivacije za različite pojave na slici (primjerice horizontalni ili vertikalni rubovi, kutovi, oblici, teksture i slično).

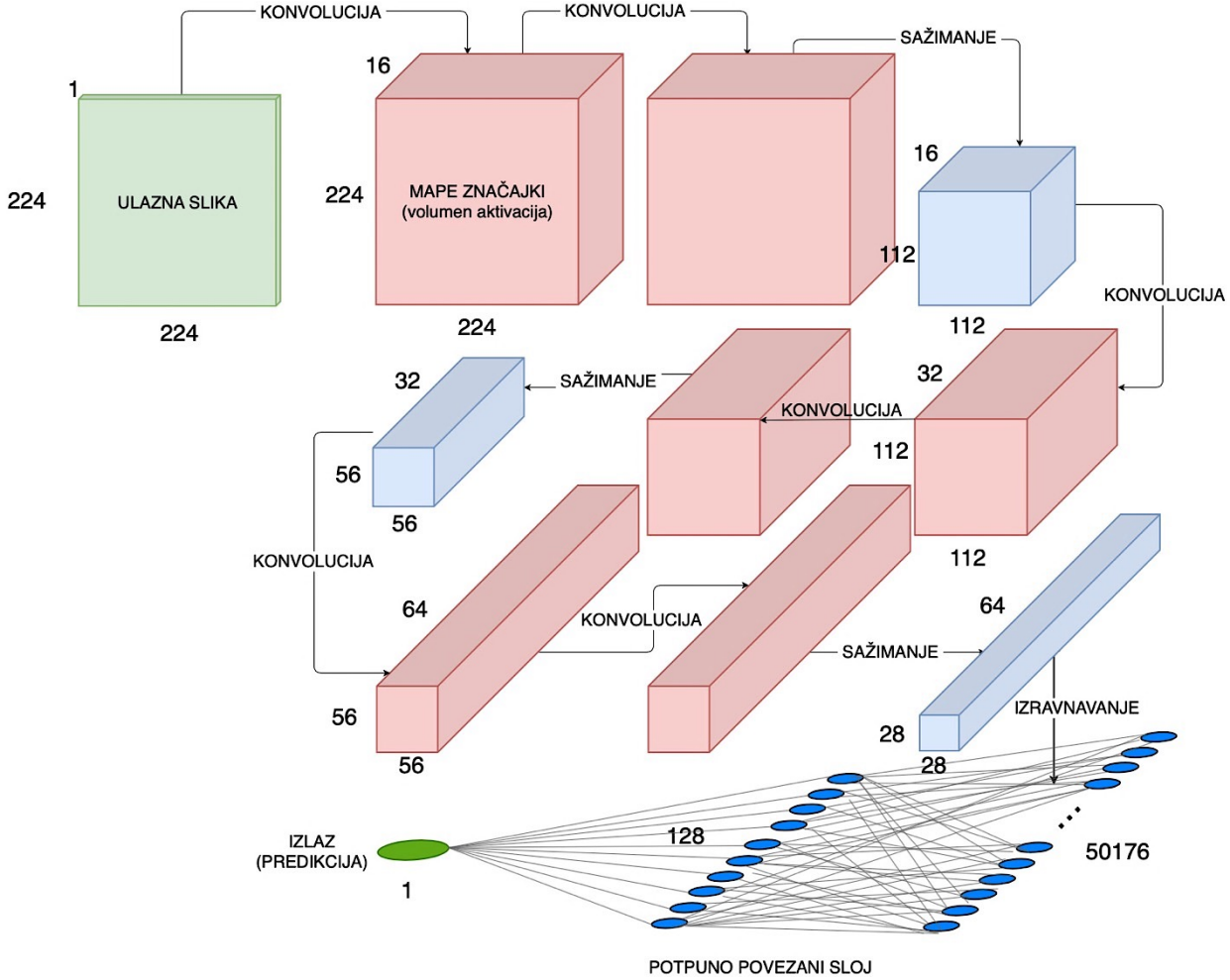
Također, u sklopu svakog od konvolucijskih slojeva, koristi se **aktivacijski sloj**, odnosno, filterima (neuronima) se pridružuje određena nelinearna aktivacijska funkcija. U ovom slučaju radi se o ispravljenoj linearnoj aktivacijskoj funkciji (ReLU) definirana izrazom (4 – 2) koja propušta vrijednosti veće od nule.

$$f(x) = \max(0, x) \tag{4 – 2}$$

Nakon prvog konvolucijskog sloja dolazi još jedan konvolucijski sloj iste konfiguracije (daje izlaz istih dimenzija te koristi identičan aktivacijski sloj) te nakon njega sloj sažimanja što skupa predstavlja jedan konvolucijski blok. Kako je navedeno u [10], prednost korištenja uzastopnih konvolucijskih slojeva (stogova), pri čemu se sloj sažimanja ne postavlja odmah nakon svakog konvolucijskog sloja, je mogućnost dobivanja mnogo kompleksnijih značajki ulaza prije same operacije sažimanja koja odstranjuje informacije. Za **sloj sažimanja** odabrano je sažimanje maksimalnom vrijednosti, veličina prozora od 2×2 piksela te pomak od 2 piksela što kao rezultat daje 4 puta manji volumen aktivacija kao izlaz (75% ulaznih aktivacija se odbacuje). Funkcija ovog sloja je, dakle, progresivno smanjivanje prostorne veličine značajki, čime se smanjuje potrebni broj parametara u mreži, njena kompleksnost te kontrolira prenaučenost modela.

Uzorak početnog konvolucijskog bloka ponavlja se kroz mrežu s povećanim brojem neurona odnosno filtera s čime raste i broj parametara u mreži, ali je mreža sposobna naučiti složenije uzorke podataka. Ukupno su korištena tri takva konvolucijska bloka – prvi sa 16 filtera, drugi sa 32 filtera te posljednji (treći) sa 64 filtera. Nakon konvolucijskih blokova, slijedi **sloj odbacivanja** kojim se nasumično zanemaruje određen udio neurona pri treniranju. Time se sprječava prevelika međusobna povezanost neurona te prenaučenost modela. Korišten je sloj odbacivanja s udjelom od 0.2, što znači da se zanemaruje 20% neurona. Nadalje, koristi se **sloj izravnanja** čija je uloga da pretvori nizove dvodimenzionalnih značajki (trodimenzionalne volumene aktivacija) u jednodimenzionalan niz značajki kako bi isti dalje mogli koristiti potpuno povezani slojevi. Nakon ovog sloja, dolazi **potpuno povezani sloj** koji koristi ReLU aktivacijsku funkciju sa 128 neurona (svaki od neurona povezan je sa svim aktivacijama prethodnog sloja) te koji na osnovu prethodno dobivenih značajki, primjenjujući težine, pokušava učiti i klasificirati značajke. Slijedi još jedan sloj odbacivanja iste stope odbacivanja za reguliranje prenaučenosti te, konačno, **izlazni sloj** koji je zapravo potpuno povezani sloj sa samo jednim neuronom, budući da se radi o binarnom problemu klasifikacije (pozitivni ili negativni sentiment). Taj izlazni sloj

koristi sigmoidnu aktivacijsku funkciju karakterističnog oblika slova „S“. Vrijednost izlaza se može interpretirati kao predikcija oznake slike odnosno vjerojatnost da ulazna slika pripada jednoj ili drugoj klasi (pozitivnom ili negativnom sentimentu). Prikaz ulaza i izlaza pojedinih slojeva, odnosno transformacija mapi značajki prikazan je na slici 4.2, gdje su naznačene dimenzije ulaza i izlaza pojedinih slojeva te operacije slojeva.



Slika 4.2 *Ulazi i izlazi za pojedine slojeve mreže*

Kao rezultat dobivena konvolucijska neuronska mreža, sastoji se iz ukupno 6 konvolucijskih slojeva (i aktivacijskih slojeva), tri sloja sažimanja, dva sloja odbacivanja, dva potpuno povezana sloja te jednog sloja izravnanja. Aktivacijski slojevi, slojevi sažimanja, odbacivanja i izravnanja ne sadrže **parametre** (težine i parametar pogreške koji se uče) – njih sadrže konvolucijski te potpuno povezani slojevi. Broj parametara svakog od konvolucijskih slojeva ovisi o dimenzijama korištene konvolucijske jezgre odnosno filtera ($F \times F$), broju ulaznih mapa značajki („dubina“ ulaznog volumena aktivacija, L), broju izlaznih mapa značajki (K) koji

odgovara broju filtera sloja te dodatno parametar pogreške (engl. *bias*) za svaku od mapi značajki. Izraz (4 – 3) definira broj parametara.

$$N = (F \cdot F \cdot L + 1) \cdot K \tag{4 – 3}$$

Tako, primjerice, prvi konvolucijski sloj sa 16 filtera dimenzija 5×5 i jednom mapom značajki (koja je u ovom slučaju ulazna slika) sadrži 416 parametara. Broj parametara potpuno povezanog sloja, kako je prikazano izrazom (4 – 4), ovisit će o broju ulaznih neurona (n), broju izlaznih neurona (m) te dodatni parametar pogreške za svaki od izlaznih neurona.

$$N = (n + 1) \cdot m \tag{4 – 4}$$

Prvi potpuno povezani sloj, s 50.176 ($28 \times 28 \times 64$) ulaznih neurona i 128 izlaznih neurona, imat će, dakle, 6.422.656 parametara, što predstavlja većinu parametara u mreži. Tablica 4.3 prikazuje popis parametara po slojevima mreže, kao i ukupan broj parametara mreže koji se treniraju.

Tablica 4.1 *Popis izlaza i parametara slojeva mreže*

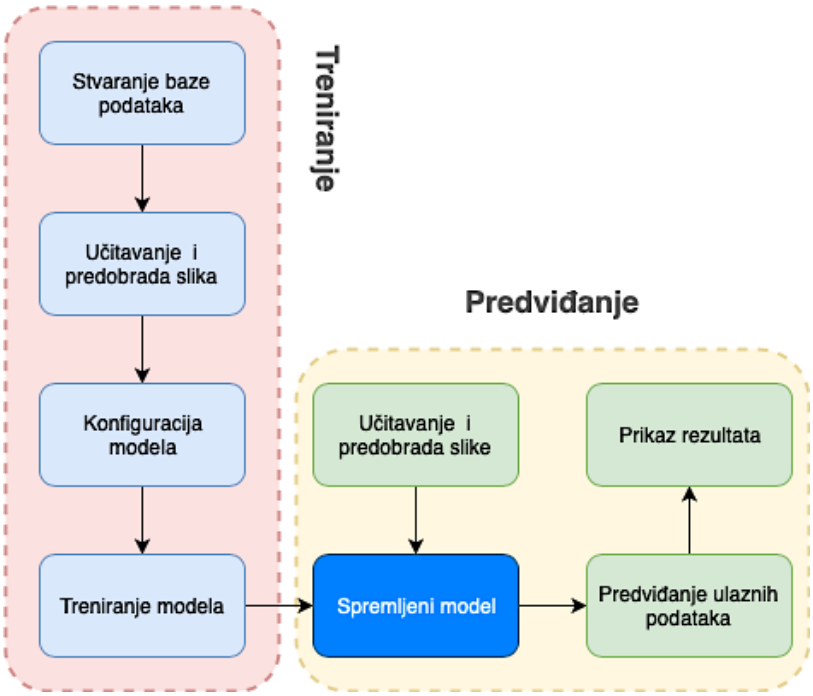
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 224, 224, 16)      416
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 224, 224, 16)      6416
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 112, 112, 16)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 112, 112, 32)      12832
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 112, 112, 32)      25632
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 56, 56, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 56, 56, 64)        51264
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 56, 56, 64)        102464
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 28, 28, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 28, 28, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 50176)             0
_________________________________________________________________
dense (Dense)                (None, 128)               6422656
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
=================================================================
Total params: 6,621,809
Trainable params: 6,621,809
Non-trainable params: 0
_________________________________________________________________
```

Treniranje ovakve mreže, u svojoj osnovi, podrazumijeva određivanje i postavljanje vrijednosti svakog od parametara mreže (svakog potpuno povezanog sloja i svakog filtra za svaki konvolucijski sloj), tako da određena ulazna slika, kada se „propusti“ kroz slojeve mreže, aktivira točno određene neurone zadnjih slojeva, odnosno, pobudi ili ne pobudi posljednji, izlazni neuron koji izvršava binarnu klasifikaciju.

Algoritam kao metodu minimizacije pogreške koristi **binarnu unakrsnu entropiju** (engl. *binary cross-entropy*) – za svaku ulaznu sliku određuje se izlaz (predikcija) mreže te se uspoređuje sa očekivanim izlazom (popratna oznaka slike). Budući da izlaz (kao i oznaka ulazne slike) predstavlja vjerojatnost da ulazna slika pripada jednoj klasi (odnosno da ne pripada drugoj), moguće je odrediti entropiju na osnovu tih vjerojatnosti. Dobivena vrijednost entropije predstavlja mjeru odstupanja predikcije od stvarne vrijednosti – što je predviđanje preciznije, to je entropija sve manja i obratno. Na osnovu vrijednosti entropije, mreža podešava svoje parametre kako bi ona bila što manja. To se postiže metodom optimizacije, a odabrani algoritam kao optimizator koristi procjenu adaptivnog momenta, takozvani **Adam optimizator** (engl. *Adaptive Moment Estimation*) koji kontrolira brzinu učenja te općenito iskazuje dobra svojstva kao optimizator u mnogim slučajevima. Kao metoda **regularizacije**, koriste se prethodno spomenuti jednostavni slojevi odbacivanja za nasumično „isključivanje“ neurona mreže. Postoji i dosta eksternih čimbenika koji utječu na učinak algoritma i svojstva dobivenog modela, kao što su izgled baze podataka, postupci predobrade slika, podešavanje hiperparametara modela, način generiranja ulaznih slika te način treniranja i slični čimbenici koji će biti objašnjeni po pojedinim fazama u idućim potpoglavljima.

4.2. Princip rada sustava

Kako i nalažu zahtjevi na sustav iz prethodnog poglavlja te sam zadatak diplomskog rada, princip rada sustava za analizu sentimenta zasniva se na treniranju modela na osnovu određenog skupa podataka o sentimentima te korištenju tog istog modela za predviđanje odnosno analizu sentimenta iz novih ulaznih podataka. Dvije su, dakle, glavne faze rada sustava – **treniranje** modela strojnog učenja te **predviđanje** odnosno primjena modela za odabranu detekciju i analizu sentimenta (palac-gore/palac-dolje) kojom se ulazna slika klasificira kao pozitivan sentiment (palac-gore) ili negativan sentiment (palac-dolje). Slika 4.3 prikazuje osnovni tijek i princip rada realiziranog sustava.



Slika 4.3 *Princip rada sustava*

Fazi predviđanja nužno prethodi faza treniranja koja kao rezultat daje istrenirani model potreban za postupke predviđanja. U nastavku su pobliže objašnjeni pojedini postupci za svaku od faza u sustavu.

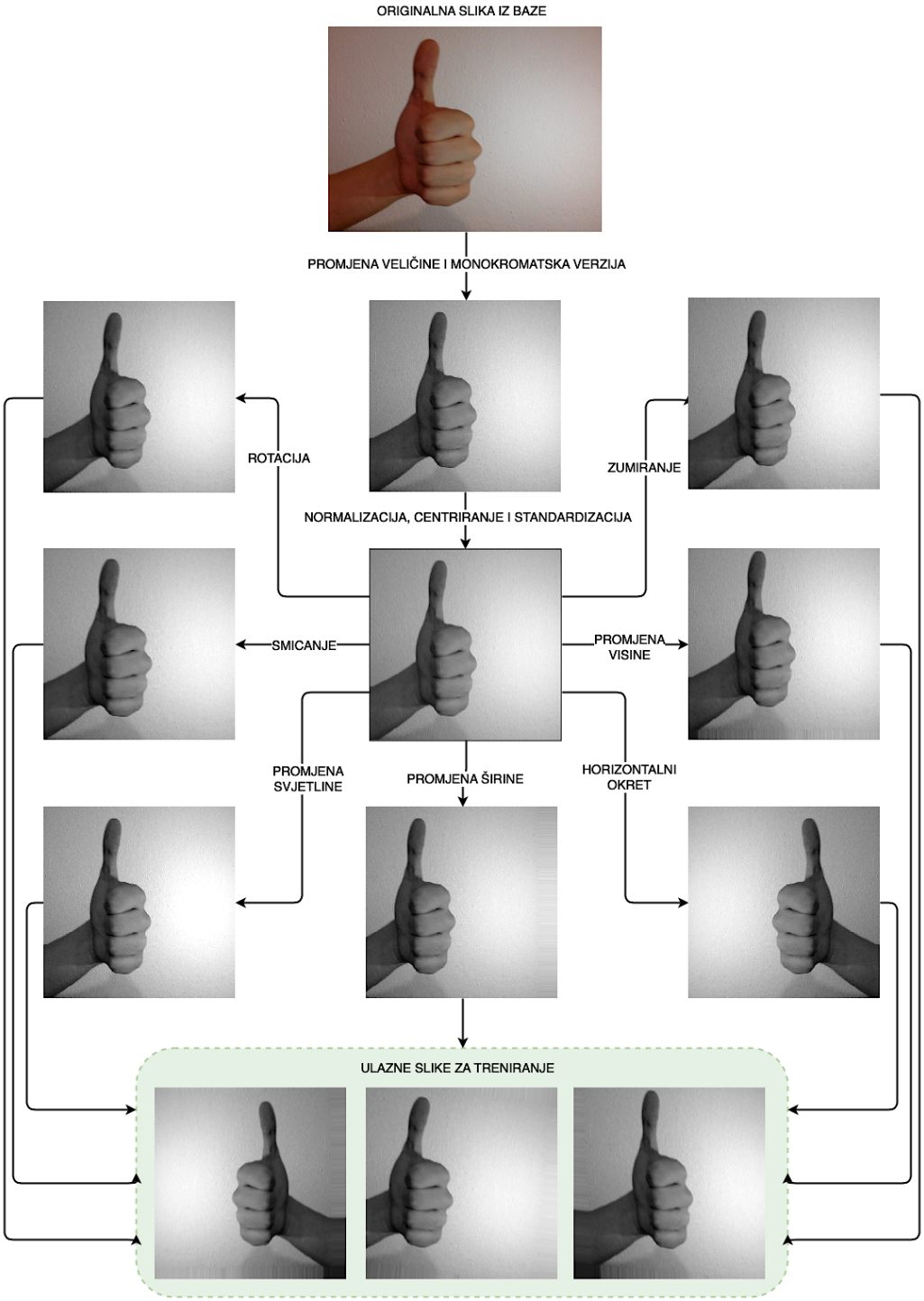
4.2.1. Treniranje

Kako je i prikazano na slici 4.3, prvi korak koji prethodi ostalim postupcima jest upravo **stvaranje baze podataka**, odnosno, prikupljanje, segmentiranje i oblikovanje adekvatne baze slika pozitivnog i negativnog sentimenta. Koristi se skup podataka od 3000 slika: 1500 slika pozitivnog sentimenta (palac gore) i 1500 slika negativnog sentimenta (palac dolje) na osnovu kojih se modeli treniraju. Slike su razdijeljene na skup za treniranje i skup za validaciju. Omjer veličine skupa za treniranje i validacijskog skupa je 4:1 u korist skupa za treniranje, odnosno, na skup za treniranje otpada 80% slika (2400 slika), a 20% slika (600) pripada skupu za validaciju, što je jedan od uobičajenih načina podjele skupa podataka.

Iz strukturirane baze podataka, sustav učitava slika sentimenta odnosno priprema ulazne podatke. Za tu se svrhu koriste određene klase i funkcije Keras okruženja koje će biti objašnjene u idućim poglavljima. **Učitavanje slika** se obavlja na način da se slike iz baze, koje su u boji i dimenzija 640 × 480 piksela, prebacuju u monokromatsku verziju (engl. *grayscale*) dimenzija 224 × 224 piksela. Razlog nekorištenja slika u boji kao podatkovnih ulaza leži u činjenici da za ovu primjenu boja ne igra ključnu ulogu – bitniji je oblik (ruka, šaka i palac) za koji je dovoljna i

monokromatska verzija slike. Svaki od piksela nosi informaciju samo o intenzitetu (količini svjetline), u rasponu od 1 do 255. Tako svaka učitana slika predstavlja dvodimenzionalni niz (224 × 224) cjelobrojnih brojeva. Prilikom učitavanja slika, prema direktorijima baze podataka u kojima se nalaze, slikama se dodjeljuju pripadne oznake – u ovom slučaju odabrana je oznaka „0“ za palac gore, a oznaka „1“ za palac dolje. Slike skupa s pridruženim oznakama čine uređene parove koji se kao takvi i dostavljaju modelu (ulazna slika i oznaka njene klase).

Slike, međutim, ponajprije prolaze kroz dodatni korak **predobrade** (engl. *preprocessing*) slika koji cjelobrojne vrijednosti prebacuje u decimalne vrijednosti te ih normalizira odnosno skalira u raspon od 0.0 do 1.0. To se postiže množenjem svakog od elementa s faktorom 1/255 (dijeljenje s maksimalnom mogućom vrijednosti). Nakon normalizacije, slikovni podaci se centriraju, odnosno srednja vrijednost slike (prosječna vrijednost svakog piksela) se postavlja na nulu. To se ostvaruje računanjem srednje vrijednosti slike te oduzimanjem te vrijednosti od vrijednosti svakog od piksela. Time se slikovni podaci smještaju u raspon od -1.0 do 1.0 te im je raspodjela vrijednosti centrirana na nulu. Iduća obrada je standardizacija slike koja omogućuje standardnu Gaussovu raspodjelu vrijednosti koja ima nultu srednju vrijednost te jediničnu standardnu devijaciju. To se postiže računanjem standardne devijacije vrijednosti skupa te dijeljenjem svakog od elementa sa standardnom devijacijom, čime su podaci otprilike smješteni u raspon od -3.0 do 3.0. Navedenim se postupcima postiže smanjenje validacijske pogreške pri treniranju te se oni primjenjuju za sve ulazne slike (treniranja i validacije). Uz to, kako bi se poboljšala generalizacijska svojstva modela, dodatno se, samo za slike iz skupa treniranja, primjenjuju nasumične transformacije ulaznih slika – rotiranje (do 5°), pomjeranje po širini (do 10% širine), pomjeranje po visini (do 5% visine), smicanje (do 10°), zumiranje (od 90% do 110%), promjena svjetline slike (od 90% do 110%) te horizontalno okretanje. Slika 4.4 prikazuje izmjene kroz koje slika prolazi pri učitavanju i predobradi slike. Na slici se daju uočiti izmjene dobivene svakim od postupaka i transformacija te su kao primjer prikazane tri slike nasumične kombinacije transformacija kao ulazne slike za treniranje modela. Navedenim transformacijama se ostvaruje veća raznolikost ulaznih podataka za treniranje pa model, učeći na raznovrsnijem skupu, iskazuje poboljšana generalizacijska svojstva.



Slika 4.4 *Postupci učitavanja i predobrade slike*

Idući postupak je **konfiguriranje modela** prema odabranom algoritmu strojnog učenja. Iako konačno rješenje odnosno sustav koristi algoritam i model strojnog učenja konfiguriran sa svojstvima opisanim u potpoglavlju 4.1, sustav je namijenjen implementaciji modela različitih konfiguracija koje se podešavaju u ovom koraku. Svojstva koja se podešavaju su primjerice ukupan broj konvolucijskih slojeva, ukupan broj potpuno povezanih slojeva, raspored

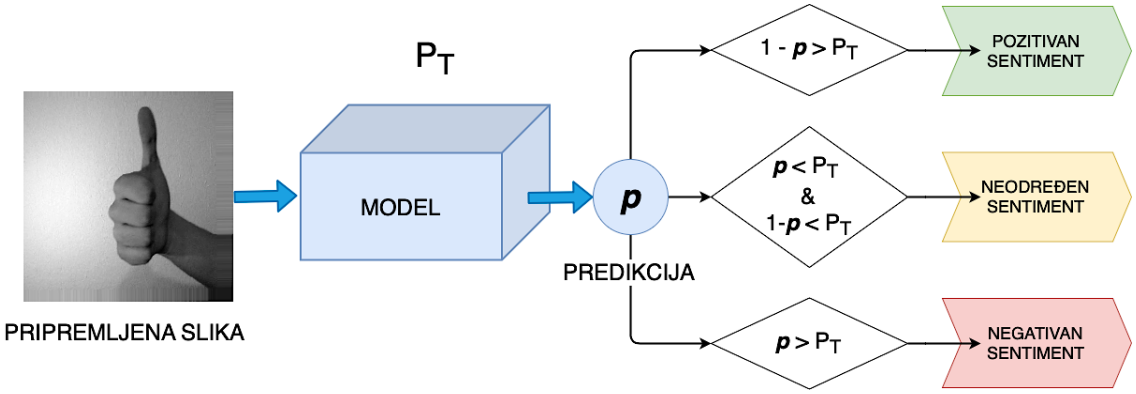
konvolucijskih slojeva i slojeva sažimanja, veličina filtera, broj neurona po slojevima, broj epoha treniranja, veličina serije, broj i raspored slojeva odbacivanja te iznosi stopa odbacivanja i slično. Obično je korišteno 50 ili 60 epoha za treniranje modela ili 10 epoha za potrebe usporedbi modela po određenim kriterijima. Korištene su veličine serije od 16, 32 i 64 slike, ali je najčešća upravo veličina serije od 32 slike. Veličine slike su obično 224 × 224, 164 × 164 ili 128 × 128 piksela. Veličine jezgre su obično ili 3 × 3 ili 5 × 5, a broj neurona 16, 32 ili 64 u prvom sloju s razmjernim višekratnim povećanjem prema krajnjim slojevima. Prema odabranim svojstvima, rasporedu slojeva i hiperparametrima, slojevi neuronske mreže se nižu jedan na drugi te se model konfigurira prema zadanoj funkciji minimizacije pogreške treniranja, optimizatoru i mjerama učinkovitosti (preciznost).

Konfigurirani model je spreman za postupak **treniranja**. Za zadani broj epoha, sustav na osnovu predobrađenih slika učitanih iz baze trenira model, odnosno, podešava i mijenja parametre modela te bilježi performanse modela (pogreška i preciznost pri treniranju i validaciji) u svakom koraku. Nakon zadanog broja epoha, treniranje modela je gotovo i isti se **sprema** u HDF5 formatu (engl. *Hierarchical Data Format*) koji omogućuje pohranjivanje i jednostavno upravljanje ogromnim količinama numeričkih podataka. Time je završena faza treniranja. Za dobivanje svakog idućeg, novog modela, ponavljaju se koraci učitavanja slika, predobrade, konfiguriranja, treniranja i spremanja modela.

4.2.2. Predviđanje

Na osnovu dobivenog i spremljenog modela iz faze treniranja, sustav može obavljati predviđanje koristeći nove ulazne slike. Nova ulazna slika prolazi jednak proces **učitavanja i predobrade slike** kao i u fazi treniranja, s izuzetkom da transformacije, koje se primjenjuju na slike za treniranje, ovdje nisu primijenjene. Dakle, slika se opet prebacuje u monokromatsku verziju, dimenzija jednakih dimenzijama ulaznog sloja modela (224 × 224 piksela za konačno rješenje) te se obavlja normalizacija, centriranje i standardizacija slike. Dobivena slika spremna je za postupak **predviđanja**. Sustav učitava odabrani model iz baze spremljenih modela, dostavlja dobivenu ulaznu sliku modelu te model na osnovu ulaznih podataka i vlastitih parametara daje izlaznu vrijednost koja predstavlja vjerojatnost da ulazna slika pripada odnosno ne pripada jednom od sentimenata. Koristeći dobivenu izlaznu vrijednost modela, sustav određuje o kojoj se klasi sentimenta radi – pozitivnom (palac gore), negativnom (palac dolje) ili neodređenom sentimentu (ni jedno ni drugo). Sustav definira neodređeni sentiment kao nedovoljno pouzdanu procjenu sentimenta odnosno klase sa ulazne slike – ako je vjerojatnost da ulazna slika pripada jednom ili

drugom sentimentu, dobivena kao izlaz predviđanja modela, manja od određene granične vrijednosti (vrijednosti praga), procjena se označava kao neodređen sentiment. Postupak dobivanja predikcije te konačno klasificiranje ulaza prikazano je na slici 4.5.



Slika 4.5 *Dobivanje predikcije i klasificiranje ulaza*

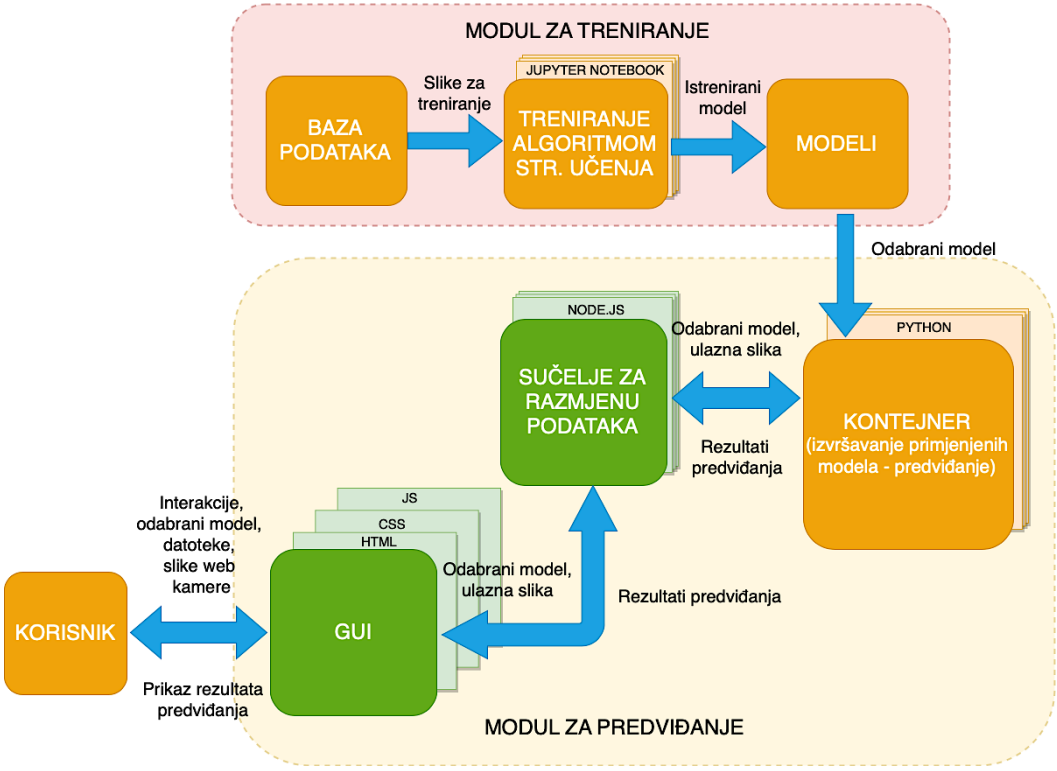
U konkretnom slučaju traženog sustava, dobivena izlazna vjerojatnost, koja poprima vrijednosti od 0 do 1, može se protumačiti i kao vjerojatnost da ulaz pripada negativnom sentimentu, budući da je za oznaku klase negativnog sentimenta izabrana vrijednost 0 (jednako je moguće korištenje i obratne anotacije). Tako izlaz 1.0 znači da je model stopostotno siguran da je ulaz negativan sentiment, odnosno, izlaz 0.0 značit će stopostotnu vjerojatnost da izlaz nije negativni sentiment, tj. da se radi o pozitivnom sentimentu. Dakle, vjerojatnost *p* predstavlja vjerojatnost da ulazna slika prikazuje negativan sentiment, a vjerojatnost *1-p* vjerojatnost da ulazna slika prikazuje pozitivan sentiment. Sa slike 4.5 se da uočiti da će sustav, ako su obje navedene vjerojatnosti manje od prethodno definirane vrijednosti praga, ulaz okarakterizirati kao neodređen sentiment. Vrijednost praga ovisit će o iskazanim performansama modela te se podešava u skladu s tim. Za odabrani model konačnog rješenja, sustava koristi vrijednost praga od 0.95.

Dobivene rezultate predviđanja (izlaznu vjerojatnost i procijenjenu klasu ulaza) sustav šalje grafičkom sučelju za interakciju s korisnikom te se **prikazuju rezultati** predviđanja. Korisničko sučelje, kao i preostale komponente sustava, opisane su u idućem potpoglavlju.

4.3. Arhitektura sustava

Sustav je realiziran iz nekoliko glavnih funkcionalnih komponenti koje se mogu grupirati u dva glavna modula – modul za treniranje i modul za predviđanje. Komponente modula za treniranje zadužene su za postupke rada s bazom podataka i treniranja modela, a komponente modula za predviđanje za postupke predviđanja opisane u prethodnom potpoglavlju (4.2.2).

Arhitektura takvog sustava za prilagodljivu ugradnju i izvođenje postupaka i modela strojnog učenja prikazana je na slici 4.6.



Slika 4.6 *Arhitektura sustava za prilagodljivu ugradnju i izvođenje postupaka i modela strojnog učenja*

Arhitektura sustava se, dakle, zasniva na radu komponenti i sučelja dvaju glavnih modula koji su opisani u nastavku.

4.3.1. Modul za treniranje

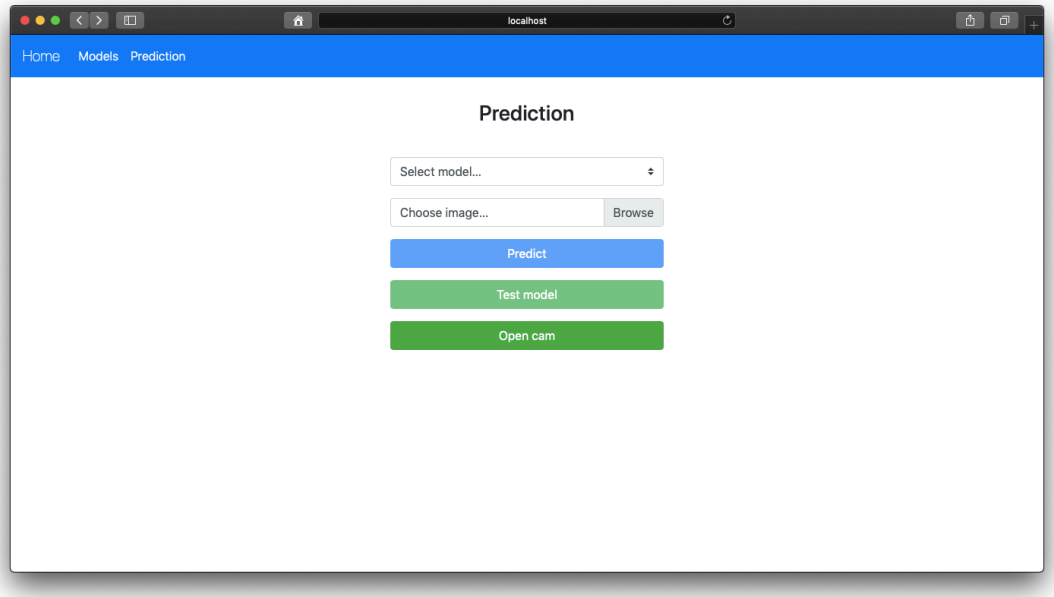
Modul za treniranje obuhvaća komponente sustava koje se u pozadini (na poslužiteljskoj strani) obavljaju prethodno samom izvođenju traženog modela. To podrazumijeva komponente **baze podataka** odnosno slika za treniranje (i validaciju), **komponentu za dobivanje modela** odnosno treniranje određenim algoritmom strojnog učenja te skup već **istreniranih modela**. Komponenta za treniranje modela učitava i obrađuje podatke iz baze (ulazi komponente) te na osnovu zadanog algoritma, konfiguracije modela i hiperparametara stvara i trenira novi model za predviđanje sentimenta (izlaz komponente). Po obavljenom treniranju, istrenirani se model sprema za daljnje korištenje. Podaci koji se razmjenjuju su ulazne originalne slike (iz baze u komponentu za treniranje) te istrenirani model (iz komponente za treniranje u skup modela odnosno direktorij za istrenirane modele). Baza podataka je izrađena kao uređena lokalna baza slika sentimenta strukturirana prema potrebama komponente za treniranje. Komponenta za treniranje obuhvaća

potrebne Jupyter Notebook (okruženje objašnjeno u petom poglavlju) datoteke, a skup modela skup datoteka s numeričkim podacima u HDF5 formatu. Prilagodljivost ugradnje postupka i modela strojnog učenja, kao svojstvo sustava, ogleda se u mogućnosti korištenja različitih algoritama, primjena odnosno analiza različitih sentimenata. Primjerice, ako se sustavom želi analizirati sentiment sa slika lica umjesto slika palca gore-dolje, potrebno je samo u bazu na jednako strukturiran način ubaciti adekvatan broj slika takvih sentimenata te će sustav moći istrenirati model za tu primjenu, koji se može na isti način koristiti za daljnje predviđanje.

4.3.2. Modul za predviđanje

Modul za predviđanje podrazumijeva komponente i sučelja koja omogućavaju dohvaćanje, obradu i razmjenu podataka o sentimentu koje unosi korisnik te izvođenje primijenjenih modela za računanje predikcije sentimenta na osnovu tih podataka. To su, dakle, korisničko sučelje odnosno grafičko korisničko sučelje (engl. *Graphic User Interface*, GUI), sučelje za razmjenu podataka te kontejner za izvršavanje primijenjenih modela.

Korisničko sučelje predstavlja osnovnu sponu između sustava i korisnika, koji interakcijom sa sučeljem unosi potrebne podatke za predviđanje – odabir željenog modela, ulazni sentiment za analizu (datoteka sa slikom sentimenta, slika sentimenta sa web kamere ili tok slika sentimenta sa web kamere odnosno video prijenos slika u stvarnom vremenu). Grafičko sučelje je izrađeno kao *frontend* dio web aplikacije koristeći HTML, CSS i JavaScript tehnologije te je korisniku dostupno kroz bilo koji web preglednik. Slika 4.7 prikazuje glavni početni izgled korisničkog sučelja.



***Slika 4.7** Izgled korisničkog sučelja*

Osnovnim interaktivnim elementima, korisničko sučelje dohvaća korisnikove podatke potrebne za predviđanje te ih razmjenjuje s kontejnerom za predviđanje preko sučelja za razmjenu podataka kojem iste šalje u obliku zahtjeva na poslužitelj. Kada dobije povratne podatke o predikciji (klasa sentimenta i vjerojatnost), te rezultate prikazuje korisniku na razumljiv način.

Sučelje za razmjenu podataka služi kao poveznica između korisničkog sučelja i kontejnera za izvršavanje modela (predviđanje) preko koje navedeni komuniciraju odnosno razmjenjuju podatke. Osmišljeno je kao aplikacijsko programsko sučelje (engl. *Application Programming Interface*, API) na poslužiteljskoj strani web aplikacije izrađene u Node.js tehnologiji. Ulazne podatke od korisničkog sučelja dobiva kao HTTP zahtjeve (na GET ili POST rute poslužitelja) iz kojih preuzima informaciju o traženom modelu te ulazne slike sentimenta (prenesenu datoteku ili sliku uslikanu web kamerom). Te podatke u odgovarajućem formatu kao izlaz prosljeđuje kontejneru za predviđanje preko toka podataka između glavnog procesa poslužitelja i procesa kontejnera. Sučelje za razmjenu podataka se u tom kontekstu ponaša kao pretplatnik ili osluškivač (engl. *listener*) izlaznih podataka iz procesa kontejnera (rezultati predviđanja) te kada iste i zaprimi prosljeđuje ih korisničkom sučelju kao odgovor poslužitelja.

Kontejner za izvođenje primijenjenih modela pokreće se kao Python proces od strane web aplikacije (poslužitelja) te preko ulaznog toka podataka, preko sučelja za razmjenu podataka, prima ponajprije informacije o potrebnom modelu. Ulaz u kontejner predstavlja i traženi model koji kontejner dohvaća iz skupa istreniranih modela iz modula za treniranje. Nakon što je učitao model, kontejner o tome obavještava prethodna sučelja putem izlaznog toka podataka te od sučelja za razmjenu podataka zaprima ulaznu sliku sentimenta u odgovarajućem formatu. Obavljaju se potrebni procesi predobrade slike te takva se slika dostavlja učitanom modelu koji radi predviđanje, odnosno, daje predikciju. Na osnovu izlazne vjerojatnosti, određuje se oznaka sentimenta te preko izlaznog toka podataka šalje sučelju za razmjenu podataka preko kojeg se konačno dostavljaju i prikazuju na grafičkom sučelju.

5. KORIŠTENI ALATI I TEHNOLOGIJE

U ovom poglavlju, dat je opis korištenih alata, tehnologija, programskih jezika, biblioteka i okruženja potrebnih za konkretnu realizaciju traženog sustava.

5.1. Python

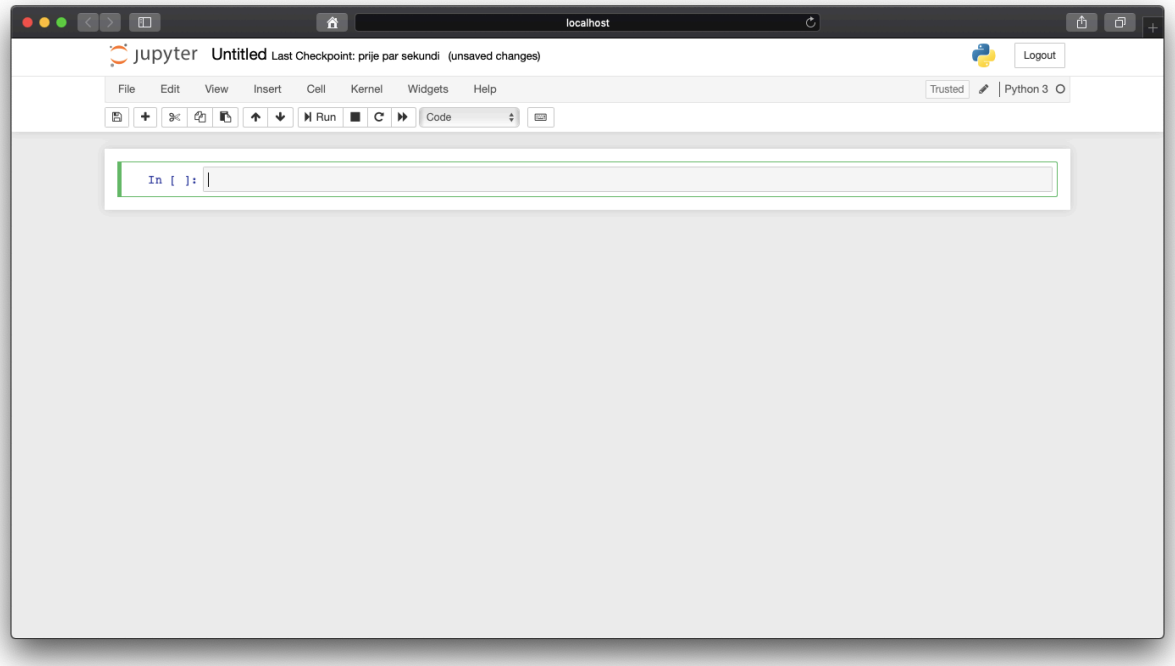
Python je jasan i snažan objektno-orijentiran programski jezik te, kao takav, usporediv s Perl, Ruby ili Java programskim jezicima. Koristi jednostavnu i elegantnu sintaksu, jednostavan je za korištenje te dolazi sa širokom standardnom bibliotekom koja podržava mnoge uobičajene zadatke poput povezivanja na web poslužitelje, rada s datotekama, numeričkim podacima i slično. Podržava i interaktivni način rada za jednostavno testiranje kraćih dijelova koda. Vrlo se lako proširuje dodavanjem novih modula te se može koristiti na raznim operacijskim sustavima. Za realizaciju dijelova traženog sustava korištena je verzija Python 3.7.

Prema [11], neke od značajki Pythona kao programskog jezika su šarolikost osnovnih tipova podataka, ali i naprednih programskih svojstava (primjerice generatora i listi), objektno-orijentirano programiranje klasama i višestrukim nasljeđivanjem, mogućnost grupiranja koda u module i pakete, jasno upravljanje pogreškama i iznimkama te automatsko upravljanje memorijom koja se ne mora ručno alocirati i dealocirati. Također, jedna od prednosti Pythona je velik broj postojećih modula i biblioteka koje se vrlo lako mogu uključiti u projekt i omogućiti funkcionalnosti već postojećih rješenja. Uz to, postoje i različita okruženja za rad u Python jeziku, od kojih je jedno i opisano u nastavku.

5.1.1. Jupyter Notebook

Jupyter Notebook je okruženje za prošireni rad u Python programskom jeziku koji omogućuje razvoj, dokumentiranje i izvođenje koda, kao i komuniciranje rezultata u web aplikaciji. Kombinira dvije komponente: web aplikaciju (alat temeljen na pregledniku za interaktivno upravljanje datotekama koje koriste popratne tekstove, matematičke i računske operacije i sadržajno bogati medijski izlaz) te Notebook dokumente (reprezentacije cjelokupnog sadržaja vidljivog u web aplikaciji). Podržava uređivanje i izvršavanje koda u pregledniku, automatsko označavanje sintakse, uvlake, prikaz rezultata računanja koristeći razne načine prikaza (HTML, LaTeX, PNG, SVG i ostali formati) te uređivanje koristeći označni jezik. Iako se koristi

u pregledniku, Jupyter Notebook je sasvim siguran jer se pokreće na vlastitom računalu koje postaje poslužitelj. Izgled sučelja za rad u Jupyter Notebooku prikazan je na slici 5.2.



Slika 5.1 *Izgled rada u Jupyter Notebook dokumentu*

Jupyter Notebook dokumenti (ili skraćeno engl. *notebook*) sastoje se iz više ćelije koje sadrže dijelove koda. Svaka od ćelije može se zasebno izvršiti, a s ostalim ćelijama unutar istog *notebooka* dijele memoriju odnosno prostor varijabli i metoda.

5.1.2. Korištene biblioteke

Za potrebe realizacije sustava korištene su razne biblioteke odnosno Python moduli i paketi koji omogućuju dodatne funkcionalnosti. Korišteni su TensorFlow, odnosno Keras (za strojno učenje), Numpy (za rad s nizovima i numeričkim operacijama), OpenCV (za računalni vid), Matplotlib i PyDot (za prikaz nizove odnosno slika), H5py (za upotrebu HDF5 formata) i drugi. Neki od modula dolaze već instalirani s Python jezikom (primjerice Numpy) dok je ostale potrebno instalirati u postojeće Python okruženje te uvesti u odgovarajući program odnosno *notebook*.

OpenCV (engl. *Open source Computer Vision*) biblioteka je programskih funkcija, originalno razvijena od strane Intela te namijenjena prvenstveno računalnom vidu u stvarnom vremenu. OpenCV podržava okruženja dubokog učenja kao što su TensorFlow, Torch ili PyTorch i Caffe. Prema [12], OpenCV posjeduje modularnu strukturu, što znači da paket uključuje nekoliko dijeljenih ili statičkih biblioteka. Jedna je od najkorištenijih biblioteka za računalni vid te

programskih bibilioteka uopće, a primjene pronalazi u prepoznavanju lica, robotici, detekciji objekata, prepoznavanju gesti ruku, praćenje pokreta i slično. Bibilioteka je napisana u C++ programskom jeziku, kao i njeno primarno sučelje, a od ostalih sučelja podržava i Python, Java i MATLAB sučelja te podržava i rad na Windows, Linux, Android i MacOS operacijskim sustavima. U sustavu je biblioteka korištena pretežno za jednostavan rad i operacije sa slikama.

5.2. TensorFlow (Keras) okruženje

TensorFlow je besplatna softverska biblioteka otvorenog koda za protok podataka i diferencijabilno programiranje za široki spektar zadataka. Koristi se za primjene strojnog učenja kao što su primjerice neuronske mreže, a između ostalog koristi je i Google u svojim istraživanjima i proizvodima te mnoge druge tvrtke. Prema [13], predstavlja *end-to-end* platformu za jednostavno strojno učenje te olakšava izradu modela strojnog učenja. Omogućuje izradu i treniranje modela koristeći Keras API visoke razine. Također, omogućava postavljanje modela na poslužitelje, krajnje uređaje ili na Internet te podržava razne dodatne biblioteke i gotove modele. Nudi u potpunosti stabilne API-jeve za Python i C programski jezik na svim platformama te C++, Java, JavaScript, Go i Swift bez unazadne API kompatibilnosti. Rad u TensorFlow okruženju i s TensorFlow API-jevima zasniva se na višedimenzionalnim nizovima podataka odnosno tenzorima. U TensorFlow okruženju često se koristi TensorBoard alat za mjerenje i vizualizaciju tijeka rada strojnog učenja, prateći određene mjere, predočavanjem grafova i slično.

Keras je Python biblioteka za duboko učenje odnosno aplikacijsko programsko sučelje (API) visoke razine za neuronske mreže, pisano u Pythonu i izgrađeno nad TensorFlowom. Razvijen je s fokusom na omogućavanje brzog eksperimentiranja te se koristi za jednostavan i brz razvoj prototipa, izgradnju konvolucijskih i povratnih neuronskih mreža te njihovih kombinacija te za neprimjetno izvođenje na procesorskim i grafičkim jedinicama. Kao što je navedeno u [14], karakteriziraju ga principi razumljivosti, modularnosti, jednostavnog proširivanja i rada s Pythonom.

5.2.1. Keras modeli

U Kerasu se modeli grade slaganjem slojeva (engl. *layers*) pa model obično predstavlja graf slojeva. Najčešći tip modela je sekvencijalni model (engl. *Sequential*) koji je predstavlja niz (stog) slojeva. Programski kod 5.1, kako je navedeno u [15], opisuje primjer izgradnje sekvencijalnog modela potpuno povezane mreže (višeslojnog perceptrona).

```
model = tf.keras.Sequential()
# Adds a densely-connected layer with 64 units to the model:
model.add(layers.Dense(64, activation='relu'))
# Add another:
model.add(layers.Dense(64, activation='relu'))
# Add a softmax layer with 10 output units:
model.add(layers.Dense(10, activation='softmax'))
```

Programski kod 5.1 *Izrada i nizanje slojeva Keras sekvencijalnog modela* [15]

Dakle, jednostavnim pozivanjem metode *add()* na objekt modela, dodaje se novi sloj u mrežu, predajući pri tome potrebne argumente za pojedini sloj (primjerice broj neurona ili aktivacijsku funkciju). Neki od često korištenih klasa za slojeve neuronskih mreža su: *Dense* (potpuno povezani sloj), *Conv2D* (konvolucijski sloj), *MaxPooling2D* (sloj sažimanja), *Dropout* (sloj odbacivanja neurona), *Flatten* (sloj izravnanja), *Activation* (aktivacijski sloj), *ZeroPadding2D* (sloj nadopune nulama) i slični slojevi.

Nakon što su dodani svi slojevi modela, model je potrebno konfigurirati odnosno sastaviti (engl. *compile*) s odgovarajućom funkcijom minimiziranja gubitka (pogreške), optimizatorom i mjerama performansi modela, što se postiže *compile()* metodom. Takav, konfiguriran model sada je moguće trenirati korištenjem metode *fit()* i sličnih izvedenih metoda koje Keras nudi (npr. *fit_generator*), navodeći pri tome potrebne argumente (niz podataka za treniranje, niz oznaka za podatke za treniranje, veličina serije, broj epoha, funkcije poziva, validacijske podatke i slično). Nakon što je model treniran, metodom *predict()* moguće je dobiti predikciju odnosno procjenu klase ili vjerojatnosti, predajući ulazne podatke kao argument. Također, Keras nudi i funkcionalnost jednostavnog testiranja odnosno evaluacije svojstava modela nad testnim skupom podataka metodom *evaluate()*. Jednom istreniran model, moguće je spremiti u odgovarajućem formatu te ga poslije učitati iz spremljene datoteke sa svim njegovim težinama.

5.2.2. ImageDataGenerator

Također, jedna od korisnih klasa koja olakšava rad s Keras modelima i cjelokupni proces njihovog treniranja, je i ImageDataGenerator klasa koja omogućava predobradu slike generiranjem serija tenzora slikovnih podataka sa sposobnošću povećanja podataka (engl. *data augmentation*) u stvarnom vremenu. Moguće je primijeniti normalizaciju, centriranje i standardizaciju podataka (po slici ili cijelom skupu slika), rotiranje, zumiranje, smicanje slika, kao i promjene svjetline, širine, visine, horizontalnog i vertikalnog okretanja ili pak primijeniti vlastitu prilagođenu funkciju predobrade. Iz objekta ImageDataGenerator klase, metodama toka podataka stvaraju se generatori podataka koji iterativnim postupcima nad originalnim podacima daju nove,

izmijenjene podatke. To su metode *flow()*, *flow_from_dataframe()* ili *flow_from_directory()*. U konačnici, ti se generatori predaju metodi treniranja modela te se iteracijama koraka generiraju potrebne serije podataka zadane veličine nad kojima model uči. Prednost korištenja povećanja podataka odnosno ImageDataGenerator klase, osim jednostavne predobrade i transformacija, je i povećanje veličine i raznovrsnosti skupa podataka što za posljedicu ima poboljšanje generalizacijskih svojstava.

5.3. JavaScript

JavaScript je skriptni programski jezik visoke razine koji zadovoljava ESCMAScript sepcifikacije. Konstruiran je da bude sličan Java programskom jeziku, ali nema objektnu orijentiranost Java jezika već se ona temelji na prototipu. Predstavlja jednu od središnjih tehnologija World Wide Weba jer omogućuje izradu interaktivnih web stranica i aplikacija. Podržava funkcijske, imperativne te programske stilove upravljane događajima. Iako je prvenstveno zamišljen i implementiran na klijentskoj strani preglednika, JavaScript se sve više koristi i u mnogim drugim tipovima softvera, uključujući i područje web poslužitelja i baza podataka kao JavaScript pogoni (engl. *engine*). Tako je primjerice Node.js razvijen kao JavaScript okruženje nad Chromeovim V8 JavaScript pogonom.

5.3.1. Node.js

Node.js je *run-time* JavaScript okruženje otvorenog koda koje omogućuje izvođenje JavaScript koda izvan samog preglednika na raznim platformama. To često podrazumijeva pisanje JavaScript skripti na strani poslužitelja za dinamičko generiranje sadržaja web stranica. Time se ujedinjuje razvoj web aplikacija jedinstvenim programskim jezikom i za klijentsku i za poslužiteljsku stranu. Node.js koristi arhitekturu upravljanu događajima s mogućnošću asinkronih ulaza i izlaza (što omogućava razne primjene u stvarnom vremenu). Dizajniran je za razvoj skalabilnih mrežnih aplikacija asinkronim programiranjem koje se zasniva na funkcijama poziva (engl. *callback*) – funkcije koje se izvršavaju nakon što se odvije određeni događaj. Radi bez upotrebe klasičnih niti (engl. *threads*) pa ne može doći do blokiranja procesa. Također, kako je objašnjeno u [16], različitim funkcijama podržava pokretanje potprocesa (engl. *child process*) koji mogu izvršavati vlastite operacije neovisno o glavnom Node.js procesu, ali s istim mogu komunicirati.

5.3.2. Express

Express je brzo, fleksibilno i minimalističko web okruženje za Node.js koji omogućava brz razvoj web aplikacija robusnih značajki. Također je besplatan i otvorenog koda te se često koristi kao *backend* komponenta uz MongoDB sustav upravljanja bazama podataka i AngularJS okruženje za *frontend*. Koristi karakterističnu strukturu projekta koja podrazumijeva glavnu aplikacijsku datoteku („app.js“), Node.js module (engl. *node modules*) koje projekt koristi, rute (engl. *routes*), poglede (engl. *views*), javne datoteke za skripte (engl. *scripts*), stilove (engl. *stylesheets*) i slike (engl. *images*) te konfiguracijske datoteke. Express, Node.js i JavaScript korišteni su za jednostavnu i brzu izradu web aplikacije kao implementacije modula za predviđanje koja podržava grafičko korisničko sučelje (*frontend*), sučelje za razmjenu podataka (kao Express Node.js *backend*) i izvršavanje procesa kontejnera kao potprocesa glavnog Node.js procesa. Pri tome je korišten Visual Studio Code programski paket za uređivanje, otklanjanje pogreški i izvršavanje koda. U idućem poglavlju opisan je način implementacije sustava – njegovih komponenti i sučelja i glavnih funkcionalnosti.

6. IMPLEMENTACIJA SUSTAVA

Korištenjem opisanih tehnologija, okruženja i alata iz prethodnog poglavlja, realiziran je sustav za prilagodljivu ugradnju i izvođenje postupaka i modela strojnog učenja, implementirajući sva potrebna sučelja i komponente od kojih se sastoji. Realizirani sustav sastoji se iz nekoliko glavnih direktorija – „Dataset“ (baza podataka), „notebooks“ (Jupyter Notebooks datoteke), „models“ (istrenirani modeli) te „mlApp“ (web aplikacija za predviđanje). Kompletni kodovi odnosno programske datoteke dane su u prilogu. Opis postupaka implementacije sustava razdijeljen je na implementiranje modula za treniranje te implementiranje modula za predviđanje.

6.1. Implementacija treniranja modela

Modul za treniranje sadrži komponente baze podataka, treniranja modela algoritmom strojnog učenja (komponenta implementacije postupka strojnog učenja) i skupa istreniranih modela. U ovom potpoglavlju, objašnjeni su koraci implementiranja i programskog ostvarivanja navedenih cjelina i funkcionalnosti.

6.1.1. Izrada baze podataka

Kao što je navedeno u 4.2, prvi postupak pri realizaciji sustava, koji prethodi ostalim koracima, je stvaranje baze podataka. Zbog nemogućnosti pronalaska adekvatne već postojeće baze pozitivnog i negativnog sentimenta u vidu mirnih slika isključivo palca gore i palca dolje, odabran je pristup izrade vlastite baze slika. Programski su i sustavno prikupljene spomenute slike uz pomoć web kamere te spremljene u lokalnu bazu u JPEG formatu. Korišten je DataCollecting notebook. Ponajprije se pokreće ugrađena kamera sustava uz pomoć openCV biblioteke VideoCapture funkcijom, kao što je prikazano programskim kodom 6.1.

```
def collect_images():
    try:
        cap = cv2.VideoCapture(0)
    except:
        return
```

Programski kod 6.1 *Pokretanje web kamere*

Zatim se dohvaća trenutni okvir (engl. *frame*) videa odnosno slika, što se da vidjeti na programskom kodu 6.2.

```
try:
        while(True):
            # Capture frame-by-frame
            ret, frame = cap.read()
```

Programski kod 6.2 *Dohvaćanje trenutnog okvira (slike)*

Slika se zapisuje u bazu (odgovarajući direktorij) te se proces ponavalja svakih 500 milisekundi, kao što je prikazano programskim kodom 6.3.

```
cv2.imwrite("../test/test/none/Thumbs Up/thumbs_up{}.jpg"
                        .format(int(time.time())), frame
                        )
            time.sleep(0.5)
```

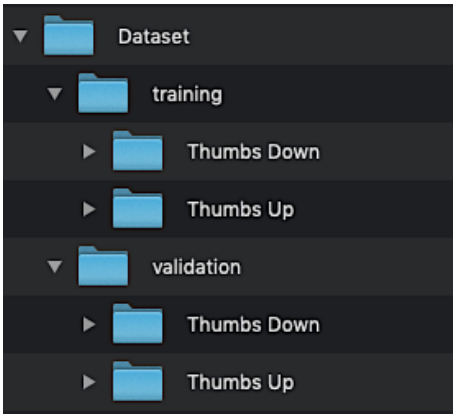
Programski kod 6.3 *Zapisivanje slike u direktorij baze*

Prvotno je prikupljeno ukupno 1000 slika (500 slika palca gore i 500 slika palca dolje) dimenzija 640 × 480 piksela te je naposljetku taj skup proširen sa dodatnih 2000 slika istih dimenzija (1000 slika palca gore i 1000 slika palca dolje) kako bi se poboljšala raznovrsnost podataka, kao i generalizacijska svojstva modela. Slike su stvarane s različitim osvjetljenjima, različitim položajem ruke i šake te različitim pozadinama. Ukupno su korištena četiri različita tipa pozadina pa se sve slike baze mogu ugrubo razdijeliti na ta četiri tipa prikazanih na slici 6.1.

Slika 6.1 *Četiri tipa pozadina slika sentimenta*

Prikupljene slike (njih ukupno 3000) su zatim sortirane po pripadnim klasama sentimenta te su zatim nasumično razdijeljene na skup za treniranje (*training*) i skup za validaciju (*validation*). Realizirana baza podataka lokalna je baza slika čija je gotova struktura prikazana slikom 6.2. Kao što je prethodno konstatirano, za negativni i pozitivni sentiment korištene su slike palca-dolje

odnosno palca-gore te direktoriji u svakom od skupova (treniranje i validacija) nose naziv klasa sentimenta (*Thumbs Up* i *Thumbs Down* u ovom slučaju). U slučaju potrebe korištenja druge vrste sentimenta, direktoriji će biti u skladu preimenovani s pripadnim novim slikama.



Slika 6.2 *Struktura baze podataka*

6.1.2. Učitavanje i predobrada podataka

Postupci učitavanja i predobrade podataka obavljaju se u ModelTraining notebooku, koji je ujedno i glavna datoteka modula za treniranje te predstavlja implementaciju komponente treniranja algoritmom strojnog učenja. Učitavanje slika iz baze obavlja se Kerasovim generatorom dobivenim na osnovu ImageDataGenerator objekta. Kako je prikazano programskim kodom 6.4, ponajprije se kreiraju objekti navedene klase i to jedan za trenirajući (train_datagen) te jedan za validacijski skup (validation_datagen), definirajući operacije skaliranja u predobradi (redom normalizacija, centriranje i standardizaciju) te nasumične transformacijske operacije predobrade (redom rotacija, promjena po širini i visini, smicanje, zumiranje, promjena svjetline i horiznontalno okretanje) koje se primjenjuju samo za trenirajući skup slika.

```
train_datagen = ImageDataGenerator(
    rescale = 1./255,samplewise_center=True,samplewise_std_normalization=True,
    rotation_range=5,
    width_shift_range=0.1,
    height_shift_range=0.05,
    shear_range=10.0,
    zoom_range=0.1,
    brightness_range=[0.9, 1.1],
    horizontal_flip=True)

validation_datagen = ImageDataGenerator(
    rescale = 1./255,
    samplewise_center=True,
    samplewise_std_normalization=True)
```

Programski kod 6.4 *Stvaranje ImageDataGenerator objekata za skupove treniranja i validacije*

Na dobivenim objektima se poziva metoda *flow_from_directory()* kojoj se predaju direktoriji odgovarajućih skupova, zadana potrebna veličina slika, broj kanala slike (monokromatska verzija), nazivi klasa, veličina serije, način klasifikacije (binarni) te uključuje opcija nasumičnog ispremetanja slika. Time se dobivaju odgovarajući generatori ulaznih slika – nad skupom za treniranje (train_generator) te validacijskim skupom (validation_generator), što je i prikazano programskim kodom 6.5.

```
train_generator = train_datagen.flow_from_directory(
                                                    '../Dataset/training',
                                                     target_size=IMG_DIMENS,
                                                     color_mode='grayscale',
                                                     classes=CATEGORIES,
                                                     batch_size=32,
                                                     class_mode='binary',
                                                     shuffle=True)

validation_generator = validation_datagen.flow_from_directory(
                                                    '../Dataset/validation',
                                                     target_size=IMG_DIMENS,
                                                     color_mode='grayscale',
                                                     classes=CATEGORIES,
                                                     batch_size=32,
                                                     class_mode='binary',
                                                     shuffle=True)
```

Programski kod 6.5 *Stvaranje generatora slika*

Dobiveni generatori su spremni i mogu se dalje koristiti u postupku treniranja.

6.1.3. Konfiguriranje modela i treniranje

Ključni notebook **ModelTraining** za postupke konfiguriranje i treniranja modela ponajprije definira broj konvolucijskih slojeva, broj potpuno povezanih slojeva, početnu veličinu konvolucijskog sloja (broj filtara prvog konvolucijskog sloja) te broj epoha. Na osnovu toga se formatira naziv modela te se zatim pomoćnom klasom TrainValTensorBoard, koja nasljeđuje osnovnu klasu TensorBoard, definira funkcija poziva kojom će se bilježiti (engl. *log*) podaci o pogreški i preciznosti za treniranje i validaciju za svaki od koraka epohe. Spomenuta izvedenena klasa omogućuje praćenje pogreški za treniranje i validaciju na jednom zajedničkom dijagramu, kao i preciznosti. Stvara se sekvencijalni model kao novi objekt klase Sequential te se počinju dodavati i nizati slojevi (programski kod 6.6).

```
#input layer
#224 x 224
model.add(Conv2D(
    layer_size, (5, 5),
    padding='same',
    activation='relu',
    input_shape=(IMG_DIMENS[0],IMG_DIMENS[1],1)
    )
)
model.add(Conv2D(layer_size, (5, 5), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

#112 x 112
model.add(Conv2D(layer_size*2, (5, 5), padding='same', activation='relu'))
model.add(Conv2D(layer_size*2, (5, 5), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

#56 x 56
model.add(Conv2D(layer_size*4, (5, 5), padding='same', activation='relu'))
model.add(Conv2D(layer_size*4, (5, 5), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.2))
model.add(Flatten())

model.add(Dense(layer_size*8, activation='relu'))
model.add(Dropout(0.2))

#output layer
model.add(Dense(1, activation='sigmoid'))
```

Programski kod 6.6 *Nizanje slojeva*

U ovom slučaju dodano je 6 konvolucijskih slojeva (Conv2D), tri sloja sažimanja (MaxPooling2D), dva sloja odbacivanja (Dropout), dva potpuno povezana sloja (Dense) te jedan sloj izravnanja (Flatten). Za svaki sloj definirana je aktivacijska funkcija („relu“ označava ReLU aktivacijsku funkciju). Slojevima sažimanja se definira veličina prozora (2,2) uz zadani pomak od 2, a konvolucijskim slojevima se definira veličina filtera kao argumenta te način nadopune volumena ulaznih aktivacija odnosno mapa značajki, gdje „padding='same'“ znači da će sloj koristiti takvu nadopunu da će izlaz sloja biti jednakog oblika kao i ulaz u sloj (u ovom slučaju to će biti po 2 okolne vrijednosti). Kad su nadodani svi slojevi, model se može sastaviti *compile* metodom (programski kod 6.7).

```
model.compile(loss='binary_crossentropy',
  optimizer='adam',
  metrics=['accuracy'])
```

Programski kod 6.7 *Sastavljanje i konfiguracija modela*

U argumentima metode, specificira se binarna unakrsna entropija kao funkcija minimizacije gubitka, optimizator Adam te za mjeru učinkovitosti se postavlja preciznost (engl. *accuracy*). Zatim se može konačno pristupiti treniranju modela, koje je prikazano programskim kodom 6.8).

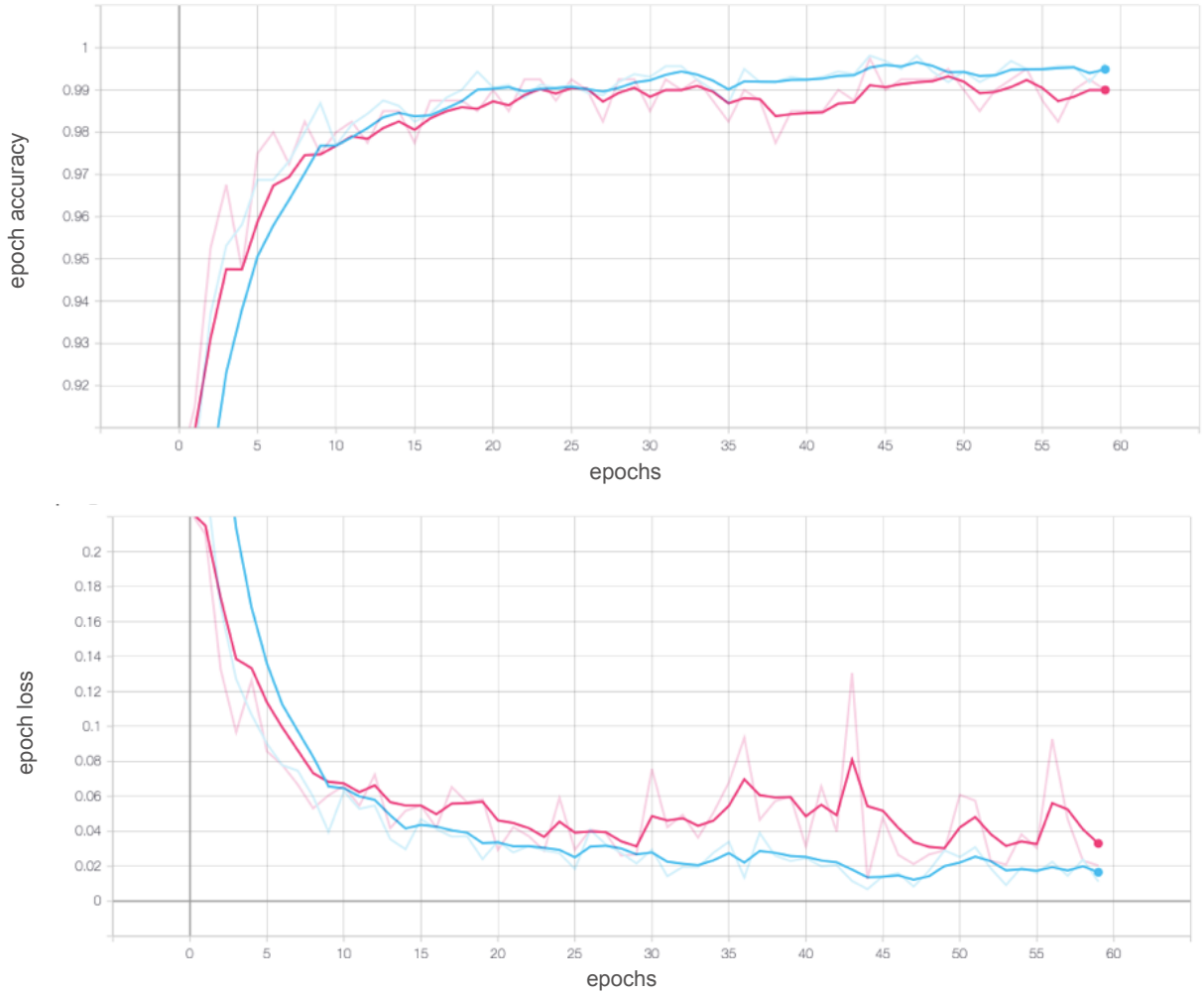
```
history = model.fit_generator(
    train_generator,
    steps_per_epoch = 100,
    callbacks=[trainValTensorboard],
    validation_data = validation_generator,
    validation_steps = 25,
    epochs = epochs_no)

model.save("../models/" + NAME + ".h5")
```

Programski kod 6.8 *Treniranje modela*

Poziva se metoda *fit_generator* za treniranje modela pripremljenim generatorima ulaznih slika. Predaju se trenirajući i validacijski generatori, postavlja broj koraka po epohi (ukupan broj serija slika zadane veličine u jednoj epohi) za treniranje i validaciju, dodjeljuje broj epoha treniranja te predaje prethodno stvorena funkcija poziva za praćenje mjera modela. Za odabranu veličinu serije od 32 slike i odabrani broj koraka po epohi iznosa 100, model će u svakoj epohi biti treniran na 32×100 = 3.200 slika. Nakon toga slijedi validacija na 25×32 = 800 slika čime se završava jedna epoha treniranja, bilježe se podaci o performansama i postupak se ponavlja definirani broj puta (broj epoha).

Podaci o pogreškama i preciznostima mogu se pratiti u TensorBoard sučelju zahvaljujući predanoj funkciji poziva pri treniranju. Primjer dobivenog dijagrama pogreške i preciznosti po epohama za treniranje i validaciju jednog od modela, prikazan je na slici 6.3. Crvenom bojom označene su karakteristike validacije, a plavom treniranja. S porastom epoha pogreške se smanjuju, a preciznosti rastu (ponajprije naglo, a zatim sve sporije) te počinju poprimati ustaljene vrijednosti. Pri tome je validacijska pogreška obično nešto veća od pogreške treniranja (generalizacijski raskorak), a validacijska preciznost nešto niža, što se da uočiti i sa dijagrama.

Nakon završetka postupka treniranja, model se lokalno sprema u HDF5 formatu koristeći *save()* metodu modela, gdje se kao naziv upotrebljava prethodno konstruirano ime uz dodatak „.h5“ ekstenzije. Skup istreniranih modela modula za treniranje je zapravo skup HDF5 numeričkih datoteka iz kojih se na osnovu naziva mogu ponovno rekonstruirati objekti modela.



Slika 6.3 *Karakteristike treniranog modela (TensorBoard) – preciznost (gore) i pogreška (dolje) za treniranje (plava krivulja) i validaciju (dolje)*

6.2. Implementacija predviđanja

Modul za predviđanje realiziran je kao Node.js web aplikacija („mlApp“) s klijentskim i poslužiteljskim dijelom izrađena u Express okruženju. Glavna struktura projekta odnosno aplikacije generira se Express instalacijom projekta te uključuje osnovne rute, aplikacijsku datoteku, osnovne poglede, module i javne datoteke. Klijentski dio aplikacije (javne datoteke i pogledi) odnose se na grafičko korisničko sučelje, a poslužiteljski (rute, aplikacijska klasa i pozadinske Python skripte) na sučelje za razmjenu podataka te kontejner za izvršavanje primijenjenih modela za predviđanja.

6.2.1. Grafičko korisničko sučelje

Grafičko korisničko sučelje realizirano je kao *frontend* dio Node.js web aplikacije. Korištene su HTML, CSS, JavaScript, JQuery i Ajax tehnologije te Jade pogon obrazaca (engl.

template engine) za jednostavniju izradu u suradnji s poslužiteljskom stranom koji koristi pojednostavljenu sintaksu koju prevodi u HTML i CSS. Grafičko sučelje za predviđanje uključuje poglede aplikacije (s naglaskom na „prediction/indeks.jade“ datoteku pogleda predviđanja), gotove uključene Boostrap skripte i stilove te vlastite prilagođene stilove („style.css“ datoteka) i skripte („script.js“ datoteka).

Središnji dio grafičkog sučelja čini datoteka pogleda za predviđanje („prediction/indeks.jade“) u koju se uključuju spomenuti vlastiti i gotovi stilovi i skripte za definiranje izgleda i funkcionalnosti elemenata sučelja. Izgled sučelja je podijeljen na tri cjeline (stupca): prvi dio za prikaz ulaza (ulazne slike za predviđanje sentimenta), središnji dio kao forma za unos podataka (naziva modela, slikovne datoteke) i odabir interakcija te posljednji dio za prikaz rezultata predviđanja sentimenta. Programski kod 6.9 prikazuje kod za formu za unos podataka.

```
form#predictionForm(name="predictionForm" method="POST"
                                action="/prediction/upload"
enctype='multipart/form-data')
                select.browser-default.custom-select.mt-
4#modelSelector(name="selectedModel", required)
                    option(value="", disabled, selected) Select model&hellip;
                    if models != null
                        - each model in models
                            option(value="#{model}")
#{model.toString().replace('.h5', '')}
                .custom-file.mx-auto.d-block.mt-3
                    input.custom-file-input#photoInput(type='file', name='photo',
accept='image/*')
                    label.custom-file-label.text-truncate#photoInputLabel(
                        for='photoInput'
                        ) Choose image&hellip;
                    input#camImageData(type='text', name='camImageData', hidden)
                button.btn.btn-primary.d-block.w-100.mt-3.mx-auto#uploadButton(
                    type='submit', disabled='true'
                    ) Predict
                input.submit#validateInputSubmit(type="submit", hidden)
```

Programski kod 6.9 *Forma za unos podataka*

Najprije su dodani elementi forme za odabir modela (element „selection“), element za unos slikovne datoteke („input“ tipa „file“), skriveni element za slanje slike uslikane web kamerom putem zajedničke forme te glavni gumb za pokretanje predviđanja. U elementu za **odabir modela**, popis modela (naziva svih modela iz skupa istreniranih modela) se dinamički generira pri kreiranju same stranice pogleda, gdje se podaci o modelima (nazivi) dobivaju od sučelja za razmjenu podataka („GET“ ruta) te prosljeđuju korisničkom sučelju (pogledu) kao varijabla pogleda (popis modela *models*). Pri promjeni odabrane stavke, pokreće se Ajax poziv definiran u skripti „script.js“

(programski kod 6.10) kojim se poslužitelju šalje naziv odabranog modela, na osnovu kojeg se u kontejneru za predviđanje učitava model iz skupa istreniranih modela u proces. Poziv se šalje POST metodom na „prediction/loadmodel“ rutu poslužitelja, koji, po obavljenom procesu učitavanja željenog modela, šalje nazad potvrdnu poruku, bez ponovnog učitavanja stranice.

```
// load the selected model (ajax post)
  $("#modelSelector").change(function(){
    var selectedModel = $(this).val();
    $("#testModelButton").prop('disabled', true);
    $.post(
      window.location.origin + '/prediction/loadmodel',
      {
        selectedModel : selectedModel
      },
      function (message, status){
        console.log(message);
        $("#testModelButton").prop('disabled', false);
      }
    );
  });
```

Programski kod 6.10 *Slanje zahtjeva za odabir modela*

Klikom na element za unos slikovne datoteke pokreće se dijalog operacijskog sustava za odabir datoteke, pri čemu se prikazuju samo datoteke u nekom od formata za slike. Nakon odabira, naziv datoteke se prikazuje u elementu.

Klikom na gumb „Open Cam“ pokreće se zahtjev za pristup ugrađenoj **web kameri** u pregledniku i korisniku se prikazuje obavijest o tome želi li dopustiti korištenje web kamere. Ako odobri zahtjev, pokreće se web kamera i u prvom dijelu sučelja za prikaz ulaza se u elementu „video“ prikazuje tok okvira s web kamere (engl. *cam stream*) u stvarnom vremenu. U suprotnom, aplikaciji nije odobren pristup kameri i nema promjena prikaza. Pokretanjem kamere, korisniku se nudi opcija snimanja slike (trenutnog okvira s web kamere) klikom na gumb „Snap photo“ kojim se pokreće izvršavanje funkcije *snapPhoto* prikazane programskim kodom 6.11.

```
function snapPhoto(){
  var canvas = document.getElementById('canvas');
  var context = canvas.getContext('2d');
  var video = document.getElementById('video');
  canvas.style.display = "block";
  context.drawImage(video, 0, 0, 320, 240);
  stopVideoStream();
  $("#photoInput").val('');
  $("#photoInputLabel").html('Choose image&hellip;');
  $("#uploadButton").prop('disabled', false);
}
```

Programski kod 6.11 *Snimanje slike s web kamere*

Iz elementa „video“ dohvaća se trenutni okvir (slika) te se prikazuje na elementu „canvas“. Tok okvira s web kamere se zaustavlja i rade se odgovarajuća ažuriranja prikaza.

Klikom na gumb „Predict“ pokreće se zahtjev za **predviđanje sentimenta** koristeći odabrani model (koji je prethodno učitan prema zahtjevu poslanom poslužitelju) i učitanu sliku (slika s web kamere ili učitana datoteka). Programski kod 6.12 prikazuje slanje podataka forme na poslužitelj. Presreće se zadano ponašanje forme te se zahtjev za predviđanje oblikuje kao Ajax POST poziv na „prediction/upload“ rutu.

```
// ajax prediction form submit
$("#predictionForm").submit(function(e){
  e.preventDefault();
  $("#uploadButton").prop('disabled', true);
  var file = $("#photoInput").val();
  if(!file){
    var src = canvas.toDataURL("image/png");
    var imageData = src.toString().replace('','');
    $("#camImageData").val(imageData);
  }
  $(this).ajaxSubmit({
    type:"POST",
    url: "/prediction/upload",
    contentType: 'application/json',
    success: function(response){
      showResults(response.category, response.probability, response.imageData);
    }
  });
  return false;
});
```

Programski kod 6.12 *Slanje zahtjeva za predviđanje sentimenta sa slike*

U slučaju da je slika učitana kao datoteka, ista se šalje kao dio zahtjeva na poslužitelj, a ako nije, slika snimljena web kamerom i prikazana na elementu „canvas“ se prevodi u tekstualni zapis po *base64* kodiranju te sprema u vrijednost skrivenog elementa forme za slanje slike pa se ta vrijednost šalje poslužitelju kao dio zahtjeva. Definira se i funkcija poziva koja se izvršava po uspješno obavljenom zahtjevu i primitku odgovora od poslužitelja (*showResults*) kojom se dobiveni rezultati predviđanja (procijenjena klasa sentimenta, vjerojatnost predikcije i ulazna slika za predviđanje) prikazuju na sučelju (naziv procijenjenog sentimenta te vjerojatnost da slika pripada pozitivnom te negativnom sentimentu uz grafičko predočavanje vjerojatnosti).

Uz to, podržana je i funkcionalnost predviđanja sentimenta u stvarnom vremenu s ugrađene web kamere klikom na gumb „Detect real-time“ kojim se pokreće funkcija *stream* za obavljanje istog (programski kod 6.13). Sustav kontinuirano uzima okvire (slike) s web kamere, pokreće podnošenje forme te šalje podatke poslužitelju na predikciju i prikazuje rezultate. Predviđanje u

stvarnom vremenu realizirano je kao predviđanje na osnovu jedne snimljene slike web kamerom, ali uzastopnim i kontinuiranim slanjem zahtjeva.

```
function stream(){
  if($("#modelSelector").val()){
    var canvas = document.getElementById('canvas');
    var context = canvas.getContext('2d');
    var video = document.getElementById('video');
    $("#photoInput").val('');
    $("#photoInputLabel").html('Choose image&hellip;');
    $("#realtimeButton").attr("onclick","stopVideoStream()").html("Stop real-
time");
    context.drawImage(video, 0, 0, 320, 240);
    var src = canvas.toDataURL("image/png");
    var imageData = src.toString().replace('','');
    $("#camImageData").val(imageData);
  }
  $("#validateInputSubmit").click();
}
```

Programski kod 6.13 *Predviđanje u stvarnom vremenu*

Po dobivenom odgovoru od poslužitelja, u funkciji poziva *showResults*, kako je i prikazano programskim kodom 6.14, provjerava se je li još uvijek otvoren tok podataka web kamere te, ako jest, ponovno se poziva funkcija za slanje zahtjeva za predviđanje u stvarnom vremenu (*stream*).

```
/* real-time detection (Stream) mode - prediction disabled,
  input image not shown (video instead)*/
  if(video.srcObject!=null){
    stream();
  }
```

Programski kod 6.14 *Nastavljanje predviđanja u stvarnom vremenu*

Kroz grafičko korisničko sučelje moguće je poslati i zahtjev za testiranje modela odnosno njegovu evaluaciju. Nakon što je model odabran te je stigla potvrda o obavljenom učitavanju istog u kontejnerski proces, moguće je klikom na gumb „Test model“ poslati Ajax GET zahtjev poslužitelju na rutu „prediction/trainmodel“ te se na poslužitelju, u pozadini kroz kontejnerski proces, pokreće evaluacija modela na zadanom testnom skupu podataka iz modula za treniranje. Nakon što je završeno, rezultati testiranja (pogreška i preciznost) se vraćaju korisničkom sučelju te se, kroz funkciju poziva navedenu u zahtjevu, prikazuju (programski kod 6.15).

```
// ajax get call to test model
$("#testModelButton").on('click', function(){
  $("#testModelButton").prop('disabled', true);
  $.get(
    window.location.origin + '/prediction/testmodel',
    function (response){
      $("#testModelButton").prop('disabled', false);
      displayTestResults(response.loss, response.accuracy);
    }
  );
});
```

Programski kod 6.15 *Zahtjev za testiranje odabranog modela*

6.2.2. Razmjena podataka

Sve navedene funkcionalnosti i zahtjevi korisnika napravljeni na korisničkom sučelju se zaprimaju i obrađuju na poslužitelju web aplikacije u sučelju za razmjenu podataka (datoteka „prediction.js“) koje predstavlja API s definiranim operacijama po rutama. Pri samom pokretanju aplikacije, koristeći *spawn* metodu Node.js *child_process* modula, pokretanjem Python skripte „prediction.py“, stvara se kontejnerski potproces „pyPrediction“ koji će obavljati izvršavanje primijenjenih modela (programski kod 6.16). Stvorenom potprocesu se dodjeljuju funkcije osluškivanja za događaje pogreški, zatvaranja te pojave izlaznih podataka.

```
const spawn = require('child_process').spawn;

var pyPrediction = spawn('python3', ["-u","./ml/prediction.py"]);
setListeners();

pyPrediction.stdout.on('data', (data) => {
  console.log(data.toString());
});
console.log("Started python process: " + pyPrediction.pid);
```

Programski kod 6.16 *Stvaranje kontejnerskog procesa*

Sve rute sučelja definirane su nad *router* objektom Express okruženja. Prvotnim **učitavanjem stranice za predviđanje** upravlja se koristeći GET rutu „prediction/index“, kao što je prikazano na programskom kodu 6.17. Poziva se pomoćna funkcija *getModels()* kojom se učitavaju nazivi svih istreniranih modela (datoteka u HDF5 formatu) u modulu za treniranje. Poslužitelj stvara i vraća pogled za predviđanje te mu predaje podatke o modelima kao varijablu.

```
router
  .get('/', function (req, res, next) {
    var models = getModels();
    res.render('prediction/index', { title: 'ML App - Prediction', models });
  })
```

Programski kod 6.17 *Upravljanje GET zahtjevima na "predicton/" rutu*

Zahtjevima za **učitavanje odabranog modela** upravlja se POST rutom „prediction/loadmodel“, gdje se iz tijela HTTP zahtjeva dohvaća naziv odabranog modela (programski kod 6.18). Tada se kontejnerskom procesu preko njegovog toka ulaznih podataka („stdin“) šalje poruka o nadolazećem pisanju odnosno slanju podataka („MODEL_MESSAGE_CODE“) koja služi kao indikator kako bi kontejnerski proces znao da slijede podaci o nazivu modela, čime je omogućeno slanje raznovrsnih poruka kontejnerskom procesu. Zatim se šalje dohvaćeni naziv odabranog modela te se dodjeljuje jednokratni osluškivač pojave podataka na toku izlaznih podataka („stdout“) kontejnerskog potprocesa. Kada se podaci pojave, ako se radi o kodu za potvrdu o učitanom modelu, poslužitelj šalje potvrdnu poruku klijentu (korisničkom sučelju).

```
  .post('/loadmodel', (req, res) => {
    var selectedModel = req.body.selectedModel.toString() + "\n";

    // sending singal for receiving model name data
    pyPrediction.stdin.write(MODEL_MESSAGE_CODE);
    pyPrediction.stdin.write(selectedModel);

    pyPrediction.stdout.once('data', (data) => {
      if(data == MODEL_LOADED_CODE){
        res.status(200).send("Model loaded");
      }
    });
  })
```

Programski kod 6.18 *Upravljanje POST zahtjevima na "prediction/loadmodel" rutu*

Zahtjevima za **predviđanje** upravlja se POST rutom „prediction/upload“ koja koristi dodatni međuprogram (engl. *middleware*) za učitavanje slikovne datoteke iz forme po zahtjevu. Korišten je modul Multer (modul za učitavanje podataka i datoteka pri HTTP zahtjevima) koji je prethodno inicijaliziran pri pokretanju aplikacije (programski kod 6.19), pri čemu je postavljena i maksimalna dopuštena veličina slikovne datoteke (3 MB).

```
var upload = multer({ storage: storage, limits: { fileSize: MAX_IMAGE_SIZE} });
```

Programski kod 6.19 *Definiranje Multer objekta*

Pri definiranju rute za predviđanje, predaje se inicijalizirani Multer objekt („upload“) s naznakom da se radi o jednoj datoteci („*single*“) i nazivom elementa forme za unos iste („photo“), što prikazuje programski kod 6.20.

```
.post('/upload', upload.single('photo'), (req, res) => {
    if (req.file) {
      var imageData = req.file.buffer.toString('base64');
    }
    else if(req.body.camImageData){
      var imageData = req.body.camImageData;
    }
    else res.redirect('/prediction');

    // sending singal for receiving image data
    pyPrediction.stdin.write(IMAGE_MESSAGE_CODE);
    // sending input data (image as base64 string) to python process for prediction
    // send encoded string length (max 7 chars based on max size of 3MB)
    pyPrediction.stdin.write(imageData.length.toString().padStart(7,'0'));
    // send encoded string
    pyPrediction.stdin.write(imageData);
    // output data handling
    pyPrediction.stdout.on('data', dataCallback = (data) => {
      if(data != MODEL_LOADED_CODE){
        var prediction_data = JSON.parse(data.toString().replace(/'/g,'"'));
        var category = prediction_data["category"];
        var probability = prediction_data["probability"];
        res.status(200).send({imageData, probability, category });
        pyPrediction.stdout.removeListener('data', dataCallback);
      }
    });
  })
```

Programski kod 6.20 *Upravljanje POST zahtjevima na "prediction/upload" rutu*

Definira se objekt slikovnih podataka („imageData“) ovisno o tome je li ulazna slika učitana datoteka ili se radi o slici snimljenoj web kamerom te kodiranom kao *base64* tekst, kako bi u konačnici objekt sadržavao slikovne podatke zapisane kao *base64* niz znakova. Kontejnerskom procesu se šalje poruka za naznaku o pisanju podataka o slici („IMAGE_MESSAGE_CODE“) te se zatim šalje informacija o duljini *base64* zapisa slike koji će se poslati kako bi potproces pravilno učitao sve znakove zapisa. Konačno šalje se i sam zapis slikovnih podataka te se dodjeljuje stalni osluškivač („dataCallback“) pojave podataka na toku izlaznih podataka potprocesa. Kada se podaci pojave, ako se radi o ispravnim podacima, raščlanjuju se (engl. *parse*) prema JSON formatu u objekt podataka predviđanja iz kojeg se dohvaćaju naziv procijenjene klase sentimenta („category“) te vjerojatnost procjene

(„probability“). Klijentskoj strani se ti podaci šalju (skupa sa objektom slikovnih podataka za prikaz ulazne slike na korisničkom sučelju) i uklanja se dodijeljeni osluškivač.

Zahtjevima za **testiranje modela** upravlja se GET rutom „predition/testmodel“, gdje se po zaprimljenom zahtjevu, kontejnerskom procesu šalje poruka „TEST_MESSAGE_CODE“ za naznaku o pokretanju testiranja učitanog modela. Zatim se dodaje jednokratni osluškivač na izlazni tok podataka potprocesa – kada se pojave podaci, raščlanjuju se prema JSON formatu u objekt podataka testiranja iz kojeg se dohvaćaju vrijednosti pogreške i preciznosti (programski kod 6.21). Isti se zatim šalju klijentskoj strani za prikaz na korisničkom sučelju.

```
.get('/testmodel', (req, res) => {
    // sending singal for testing model
    pyPrediction.stdin.write(TEST_MESSAGE_CODE);
    console.log("Model test started");

    pyPrediction.stdout.once('data', (data) => {
      var evaluation_data = JSON.parse(data.toString().replace(/'/g,'"'));
      var loss = evaluation_data['loss'];
      var accuracy = evaluation_data['accuracy'];
      res.status(200).send({loss, accuracy });
    });
  });
```

Programski kod 6.21 *Upravljanje GET zahtjevima na "prediction/testmodel" rutu*

6.2.3. Kontejner za izvršavanje primijenjenih modela

Središnja komponenta modula za predviđanje jest upravo kontejner za izvršavanje primijenjenih modela koji je realiziran kao pozadinski Python proces (skripta „prediction.py“) poslužitelja web aplikacije koji se pokreće pri samom pokretanju aplikacije. Uključuje module potrebne za rad s modelima, slikama i nizovima podataka. Proces je osmišljen kao program u petlji koji kontinuirano osluškuje podatke na ulaznom toku te po protokolu u skladu s vrstom podataka izvodi odgovarajuće operacije, što se da uočiti programskim kodom 6.22. Zadano stanje procesa je očekivanje poruke o tipu podataka definirane duljine na ulaznom podatkovnom toku („sys.stdin“). Kada proces zaprimi poruku, ovisno o njenoj vrsti, provodi odgovarajuće naredbe te se, nakon što su potrebne naredbe izvršene, ponovno vraća u stanje osluškivanja ulazne poruke o tipu podataka.

```
while True:
    inputMessage = sys.stdin.read(3)
    if inputMessage == IMAGE_MESSAGE_CODE:
        inputSize = sys.stdin.read(7)
        inputData = sys.stdin.read(int(inputSize))
        image = receive_image_data(inputData)
        do_prediction(image)
    elif inputMessage == MODEL_MESSAGE_CODE:
        modelName = sys.stdin.readline()
        if 'model' in globals():
            tf.keras.backend.clear_session()
            del model
        model = load_model(modelName)
    elif inputMessage == TEST_MESSAGE_CODE:
        evaluate_model()
```

Programski kod 6.22 *Glavna petlja kontejnerskog procesa za izvršavanje primijenjenih modela*

Ako je ulazni podatak naziv modela (pročitana poruka o tipu podataka je „MODEL_MESSAGE_CODE“), proces ulazi u granu za tu vrstu ulaza, čita i dohvaća naziv modela, uklanja eventualna prethodna memorijska zauzeća modela (s ciljem smanjenja gomilanja podataka i memorijske potrošnje procesa) te poziva *load_model* funkciju za **učitavanje modela** kojoj predaje naziv kao argument. Funkcija je prikazana programskim kodom 6.23, gdje se primjećuje da, na osnovu predanog naziva modela, učitava model iz skupa istreniranih modela uz pomoć ugrađene metode Keras modula *load_model*. Na temelju zadane putanje do modela, učitava se model i sprema se u varijablu koja se vraća procesu na daljnje korištenje. Uz to se u izlazni tok podataka ispisuje i poruka za potvrdu o učitanom modelu (ugrađena funkcija *print*).

```
def load_model(modelName):
    loaded = tf.keras.models.load_model(os.path.join("../models/",
modelName.rstrip('\n')))
    print(MODEL_LOADED_CODE)
    return loaded
```

Programski kod 6.23 *Funkcija za učitavanje modela (load_model)*

Ako su ulazni podaci slikovni podaci (poruka o tipu podataka je „IMAGE_MESSAGE_CODE“), proces ulazi u granu za tu vrstu ulaza, čita ponajprije podatak o veličini znakovnog niza kojim je slika kodirana te zatim čita točno toliku količinu znakova sa ulaza koje sprema u varijablu ulaznih podataka („inputData“). Koristeći pomoćnu funkciju *receive_image_data* kojoj predaje dobiveni niz ulaznih podataka, taj se niz znakova pretvara u sliku (dvodimenzionalni niz brojeva). Programski kod 6.24 prikazuje definiciju *receive_image_data* funkcije koja predani niz znakova (kodiran *base64* kodiranjem) dekodira u međuspremnik podataka (engl. *data buffer*), stvara jednodimenzionalni niz binarnih podataka iz

dobivenog međuspremnika podataka funkcijom Numpy modula *frombuffer* te konačno funkcijom OpenCV modula *imdecode* čita dobiveni niz i pretvara ga u originalni dvodimenzionalni niz brojeva (sliku) koji vraća nazad procesu.

```
def receive_image_data(data):
    imgdata = base64.b64decode(str(data))
    array = np.frombuffer(imgdata, dtype=np.uint8)
    img = cv2.imdecode(array, cv2.IMREAD_COLOR)
    return img
```

Programski kod 6.24 *Funkcija za primanje slikovnih podataka (receive_image_data)*

Proces zatim obavlja **predviđanje** koristeći funkciju *do_prediction* kojoj predaje dobiveni dvodimenzionalni niz brojeva (sliku). Kako pokazuje programski kod 6.25, funkcija, na osnovu predanog joj slikovnog niza, obavlja predviđanje vjerojatnosti ugrađenom metodom Keras modela *predict_proba* kojoj predaje pripremljenu i predobrađenu ulaznu sliku.

```
def do_prediction(image):
    prediction_proba = model.predict_proba([prepare(image)])
    if prediction_proba[0][0] > UNDEFINED_TRESH or prediction_proba[0][0] < 1-
UNDEFINED_TRESH:
        prediction = model.predict_classes([prepare(image)])
        category = CATEGORIES[int(prediction[0])]
    else:
        category = "UNDEFINED"
    data = {
        "category": category,
        "probability": prediction_proba[0][0]
    }
    print(data)
```

Programski kod 6.25 *Funkcija za predviđanje (do_prediction)*

Slikovni niz se prethodno obrađuje pomoćnom funkcijom *prepare* koja je prikazana programskim kodom 6.26. Predani slikovni niz (slika) se ponajprije prebacuje u monokromatsku verziju funkcijom OpenCV modula *cvtColor* te se zatim funkcijom *resize* postavlja na odgovarajuće dimenzije koje model traži (224 × 224 za konačno odabrano rješenje). Niz se zatim prebacuje u decimalne brojeve te se redom obavlja normalizacija, centriranje i standardizacija slike.

```
def prepare(input_image):
    img_array = cv2.cvtColor(input_image, cv2.COLOR_BGR2GRAY)
    new_array = cv2.resize(img_array, model.layers[0].input_shape[1:3])  # resize
image to match model's expected sizing
    new_array = new_array.astype('float32')
    new_array = new_array/255.0
    new_array = new_array - new_array.mean()
    new_array = new_array/new_array.std()
    return new_array.reshape(-1, model.layers[0].input_shape[1],
model.layers[0].input_shape[2], 1)  # return the image with shaping that TF wants.
```

Programski kod 6.26 *Funkcija za pripremu i predobradu slike (prepare)*

Naposljetku, funkcija vraća sliku u obliku niza s četiri dimenzije koju traže TensorFlow modeli. Model na osnovu predobrađene slike predane u odgovarajućem obliku računa predikciju te klasificira sentiment prema postupku opisanom u potpoglavlju 4.2.2. Ako je izlazna vjerojatnost da slika prikazuje pozitivan ili negativan sentiment veća od vrijednosti praga, procjenjuje se o kojoj se od dvije klase sentimenta radi (pozitivnom ili negativnom) ugrađenom metodom modela *predict_classes*. U suprotnom sentiment se označava kao neodređen („UNDEFINED“). Konstruira se objekt s podacima o procijenjenoj klasi i izlaznoj vjerojatnosti te šalje na izlazni tok podataka.

Konačno, ako je zaprimljena poruka o **testiranju** („TEST_MESSAGE_CODE“), proces ulazi u granu za testiranje i poziva pomoćnu funkciju *evaluate_model* čija je definicija prikazana programskim kodom 6.27. Stvara se objekt klase ImageDataGenerator s postupcima predobrade opisanim u potpoglavlju 4.2.1 (bez transformacija slika). Iz njega se funkcijom *flow_from_directory* stvara generator slika koji se predaje ugrađenoj metodi modela *evaluate_generator* za izvršavanje automatskog testiranja i evaluacije modela. Rezultati (pogreška i preciznost) se pohranjuju u objekt „evaluation“ i konstruira se objekt s podacima o pogreški i preciznosti te šalje na izlazni tok podataka.

```
def evaluate_model():
    datagen = ImageDataGenerator(
        rescale = 1./255,
        samplewise_center=True,
        samplewise_std_normalization=True)
    sys.stdout = open(os.devnull, "w")
    test_generator = datagen.flow_from_directory('../test,
        target_size=model.layers[0].input_shape[1:3],
        color_mode='grayscale',
        classes=CATEGORIES,
        batch_size=32,
        class_mode='binary',
        shuffle=True)
    sys.stdout = sys.__stdout__
    evaluation = model.evaluate_generator(test_generator)
    data = {
        "loss": evaluation[0],
        "accuracy": evaluation[1]
    }
    print(data)
```

Programski kod 6.27 *Funkcija za testiranje modela (evaluate_model)*

Sve podatke poslane na izlazni tok podataka zaprima sučelje za razmjenu podataka u svojim dodijeljenim osluškivačima događaja pojave podataka te po postupcima opisanim u potpoglavlju 6.2.2, te podatke dostavlja korisničkom sučelju.

7. TESTIRANJE I ANALIZA SUSTAVA

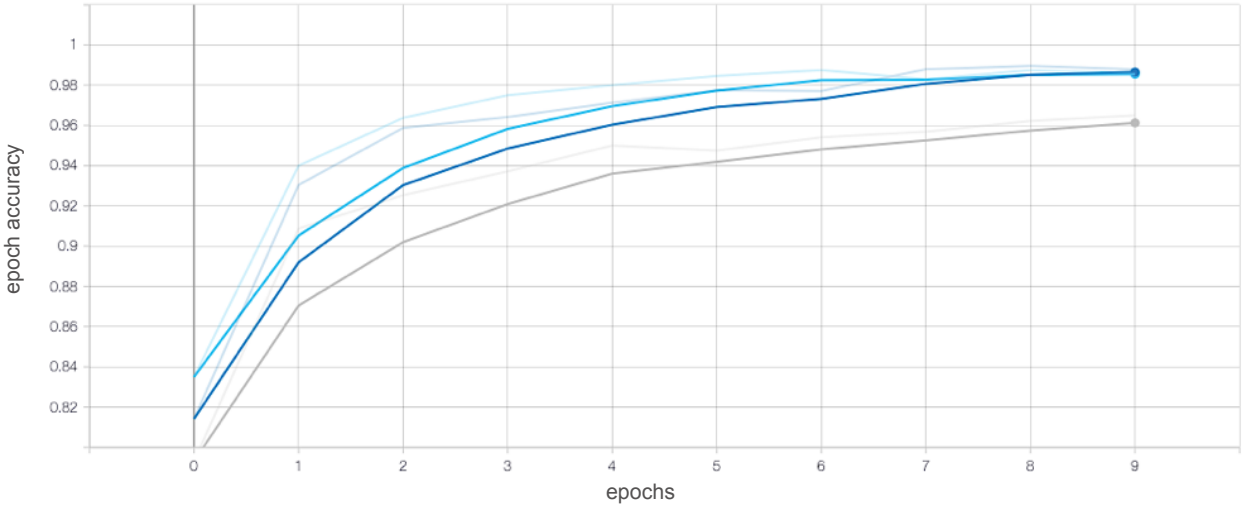
Konačna inačica realiziranog sustava rezultat je testiranja rada dobivenog sustava, kao i samog algoritma i modela strojnog učenja za analizu sentimenta. Desetine modela strojnog učenja su dobivene različitim konfiguracijama algoritma za treniranje te je, na osnovu rezultata i usporedbe rezultata testiranja pojedinih algoritama i modela, odabran konačni algoritam i korišteni model strojnog učenja. Obavljeno je i testiranje rada sustava s naglaskom na ispitivanje korisničkog sučelja (integracijski test) i ispitivanje funkcionalnosti sustava, uzimajući u obzir definiranje vrijednosti praga za oznaku neodređenog sentimenta. U konačnici su rezultati ispitivanja analizirani te je pružen osvrt na cjelokupan sustav.

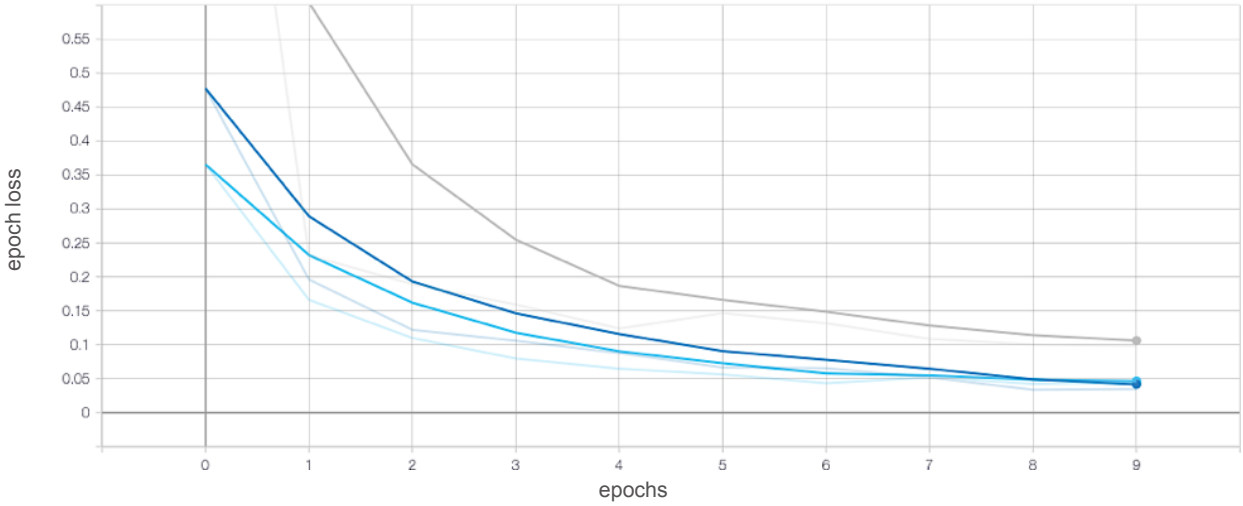
7.1. Testiranje modela i algoritma strojnog učenja

Iako je kao algoritam strojnog učenja odabrana konvolucijska neuronska mreža, istu je moguće značajno mijenjati podešavajući različite rasporede i broj pojedinih slojeva, broj i veličinu

filtera konvolucijskih slojeva, broj neurona potpuno povezanih slojeva, stopu odbacivanja slojeva odbacivanja, postupke predobrade slike i transformacije ulaznih slika, broj epoha treniranja, veličina serije, broj serija po epohi, omjer validacijskog i trenirajućeg skupa, količina i vrsta slika sentimenta u bazi, dimenzije ulaznih slika i slično. Zahvaljujući alatu TensorBoard, uz pomoć prilagođene funkcije poziva, moguće je pratiti karakteristike pojedinih treniranih modela (pogreška i preciznost treniranja i validacije) te na osnovu toga zaključivati o svojstvima algoritma.

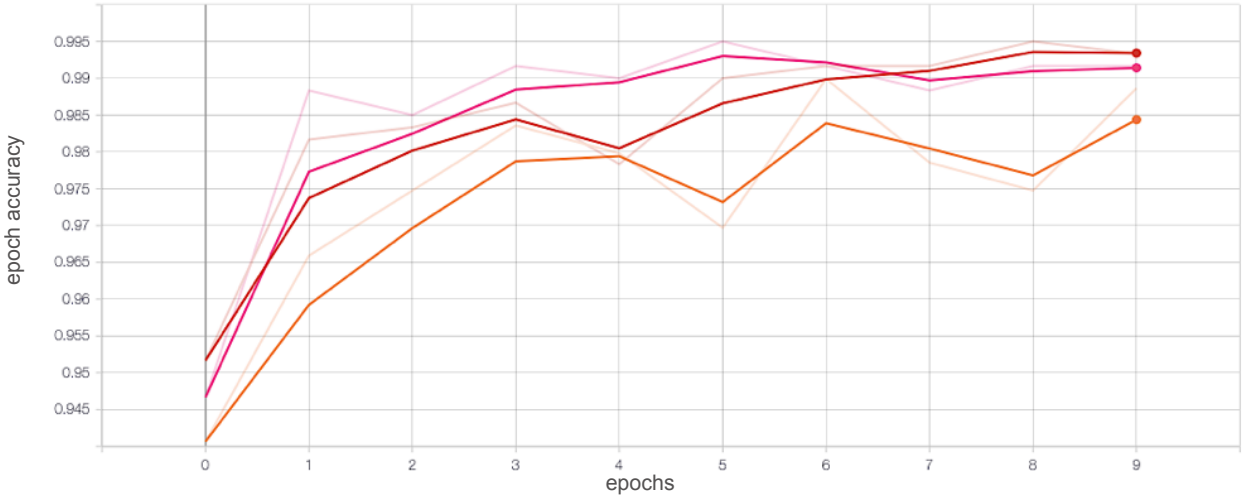
Ponajprije je ispitan utjecaj **broja konvolucijskih slojeva** na performanse (pogreška i preciznost) modela. Za kraći broj epoha (10), istrenirani su modeli s 1, 2 i 3 konvolucijska sloja (svaki popraćen slojem sažimanja). Slika 7.1 prikazuje dobivene karakteristike odnosno dijagrame pogreške i preciznosti treniranja navedenih modela. Iz dijagrama preciznosti treniranja po epohama (engl. *epoch training accuracy*) navedenih modela, da se uočiti kako najbolja svojstva iskazuje karakteristika modela s tri konvolucijska sloja (najveća razina preciznosti), što potvrđuje i dijagrama pogreške treniranja po epohama (engl. *epoch training loss*), gdje model s tri konvolucijska sloja (svijetloplava krivulja) ponovno iskazuje najbolja svojstva (najmanja razina pogreške). Da se zaključiti kako porastom broja konvolucijskih slojeva mreže, raste i preciznost dobivenog modela, a njegova pogreška treniranja se smanjuje (postaje sposobniji bolje naučiti skup nad kojim se trenira).

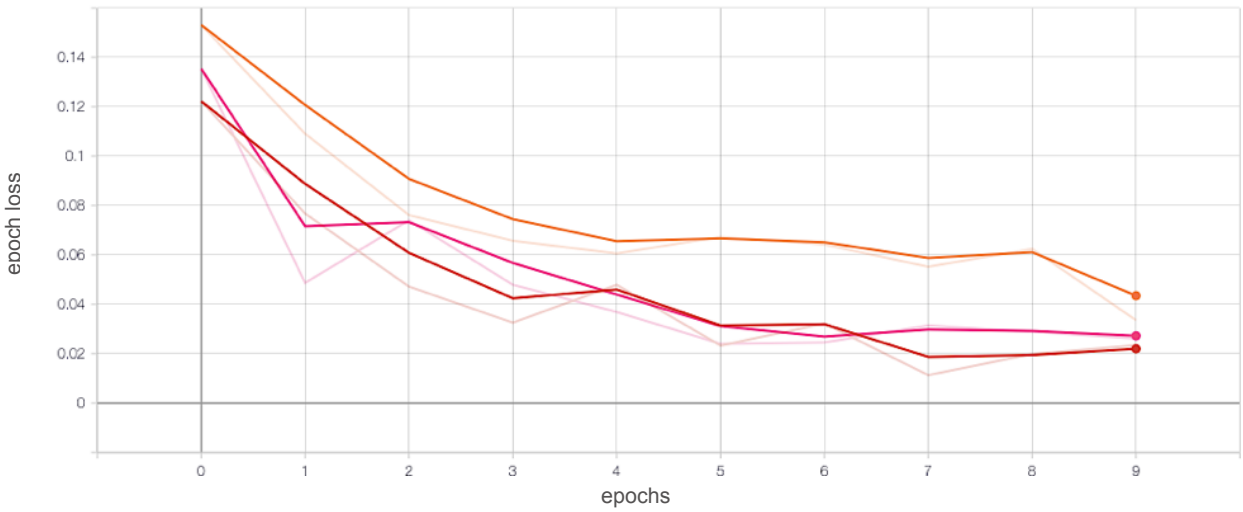




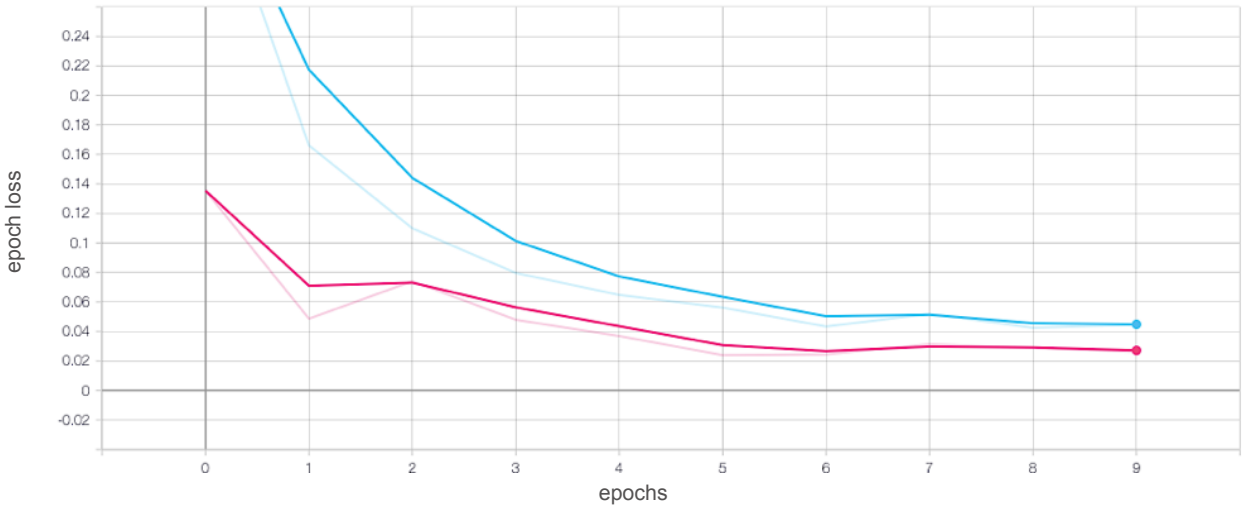
Slika 7.1 *Dijagrami preciznosti (gore) i pogreške treniranja (dolje) za konvolucijsku mrežu s 1 (siva krivulja), 2 (tamnoplava krivulja) i 3 (svijetloplava krivulja) konvolucijska sloja*

Promotrimo li karakteristike validacije modela – dijagram preciznosti validacije po epohama (engl. *epoch validation accuracy*) i dijagram pogreške validacije po epohama (engl. *epoch validation loss*) – uočit ćemo slične rezultate. Model s najviše konvolucijskih slojeva pokazuje najnižu pogrešku validacije te najveću preciznost (Slika 7.2). Iako je korišten relativno mali broj filtera po konvolucijskom sloju, upotrebom većeg broj konvolucijskih slojeva povećava se sposobnost modela da detektira i nauči kompleksnije oblike, teksture i slično. Međutim, za vrednovanje modela, nije dovoljno promatrati izdvojene karakteristike treniranja ili validacije. Kako je navedeno u potpoglavlju 2.2, o svojstvima modela (generaliziranje, podnaučenost, prenaučenost i slično) ispravnije se zaključuje promatrajući istovremeni odnos pogreške treniranja i pogreške validacije. Dijagram pogreške treniranja i validacije prikazan je na slici 7.3 te se na njemu može uočiti da, iako se obje pogreške snižavaju s epohama, validacijska je pogreška niža od pogreške treniranja. To upućuje na poteškoće algoritma strojnog učenja u optimizaciji.





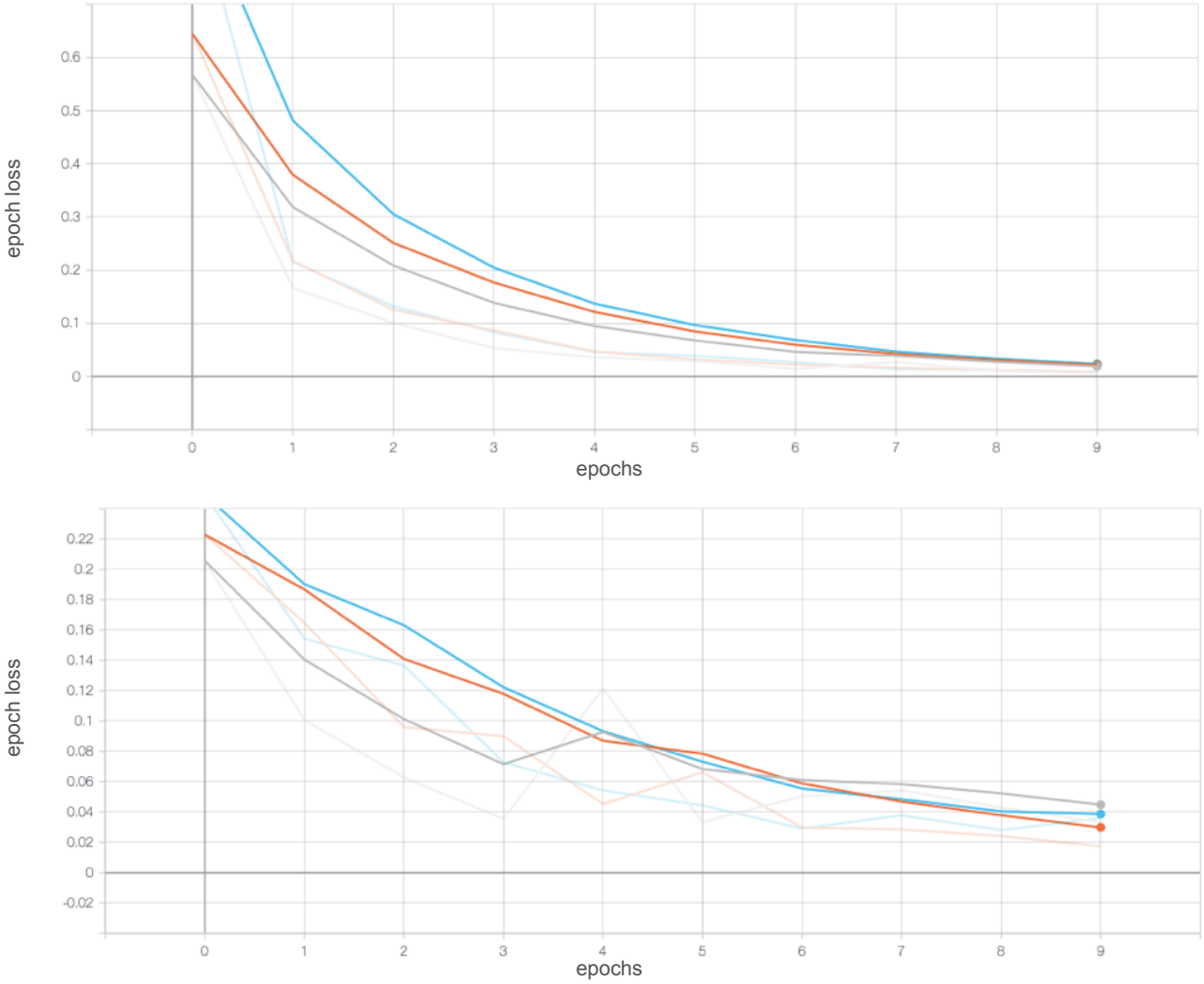
Slika 7.2 *Dijagrami preciznosti (gore) i pogreške validacije (dolje) za konvolucijsku mrežu s 1 (narančasta krivulja), 2 (crvena krivulja) i 3 (ružičasta krivulja) konvolucijska sloja*



Slika 7.3 *Dijagrami pogreške treniranja (plava krivulja) i validacije (ružičasta krivulja) za konvolucijsku mrežu s 3 konvolucijska sloja*

Model, iako iskazuje niske vrijednosti validacijske pogreške, nije u potpunosti savladao skup podataka nad kojim se trenira, budući da je treniran za malen broj epoha, pa je za očekivati da i dobivena validacijska pogreška neće zaista značiti dobra generalizacijska svojstva na novim, neviđenim slikama sentimenta (što je i bio slučaj pri testiranju funkcionalnosti sustava s ovakvim, jednostavnim modelima). Takvi modeli su pri testiranju funkcionalnosti na novim slikama obično davali ekstremne procjene sentimenta – izlazne vjerojatnosti su ili jako bliske 1 ili jako bliske 0, s jako rijetkim ili nepostojećim slučajevima neke međuvrijednosti (primjerice 0.6). Stoga se pristupilo kasnijem povećanju kapaciteta mreže (veći broj neurona odnosno slojeva), restrukturiranje mreže (promjena rasporeda slojeva) te povećanja broja epoha treniranja.

Ispitivanje utjecaja **broja neurona** na performanse jednostavnijih modela nije dalo pretjerano velika odstupanja ni u jednom od slučajeva. Testirani su algoritmi za 32, 64 i 128 neurona (filtera) u svakom od slojeva, pri čemu se broj i raspored slojeva nije mijenjao. Slika 7.4 prikazuje dobivene dijagrame pogreške treniranja i validacije za navedene brojeve neurona.

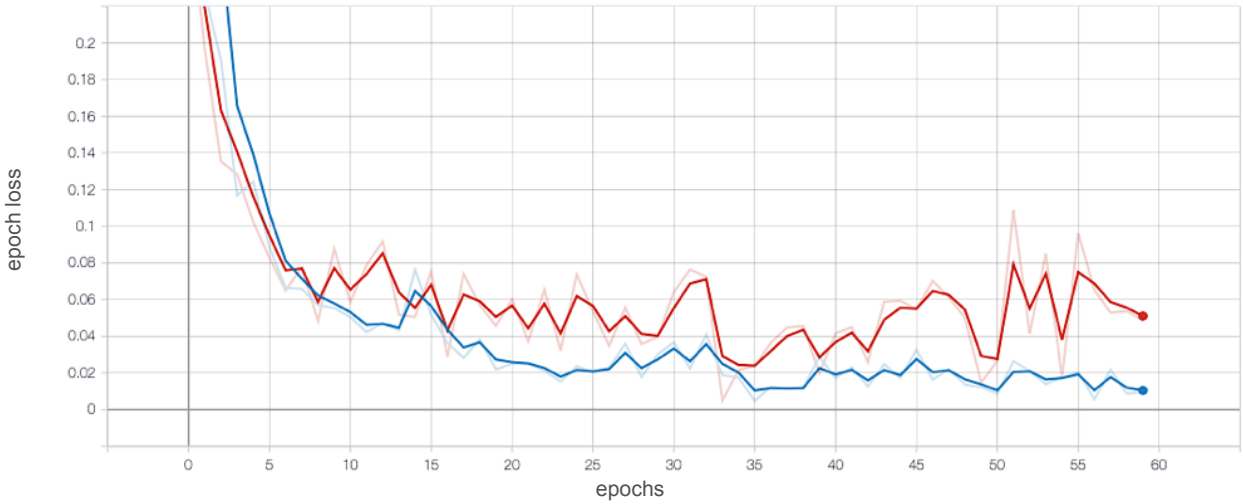


Slika 7.4 *Dijagrami pogreške treniranja (gore) i validacije (dolje) za mrežu slojeva s 32 (narančasta krivulja), 64 (plava krivulja) te 128 neurona (siva krivulja)*

Budući da porast broja neurona po slojevima nije davao značajan doprinos performansama modela, odabrana je najjednostavnija postavka jer zahtijeva najmanje procesorskih kapaciteta pri treniranju te je u skladu sa zahtjevom na sustav o najjednostavnijem mogućem rješenju koje daje zadovoljavajuće rezultate. U početku su to bila 32 neurona, a kasnije je upotrijebljen razmjerno rastući broj neurona po konvolucijskim blokovima (16 filtera slojevi prvog bloka, 32 drugog itd.), što je uočeno kao česta praksa u konfiguriranju konvolucijskih neuronskih mreža (primjerice u [10] i [17]). Slično, kao i utjecaj broja neurona, broj potpuno povezanih slojeva nije značajnije utjecao na preciznosti i pogreške modela, a s obzirom da porastom broj potpuno povezanih slojeva

značajno raste broj parametara te time i kompleksnost mreže, odabran je jedan potpuno povezani sloj kao zadovoljavajuće rješenje.

Utjecaj hiperparametara algoritma nešto je značajniji. Tako se povećanjem **broja epoha** treniranja, značajno snižavaju pogreške treniranja i validacije, ali postoji opasnost od prenaučenosti modela. Kao što je prethodno navedeno, početna treniranja modela za malen broj epoha (10) rezultirale su manjom sposobnošću optimizacije zbog preuranjenog završetka treniranja. S druge strane, prevelikim brojem epoha treniranja model postaje previše vezan za ulazne podatke treniranja i smanjuje mu se sposobnost generaliziranja, što se da uočiti tipičnim porastom validacijske pogreške dok se pogreška treniranja smanjuje ili ostaje na vrlo niskim vrijednostima (Slika 7.5).



Slika 7.5 *Početak pojave prenaučenosti modela s većim brojem epoha treniranja*

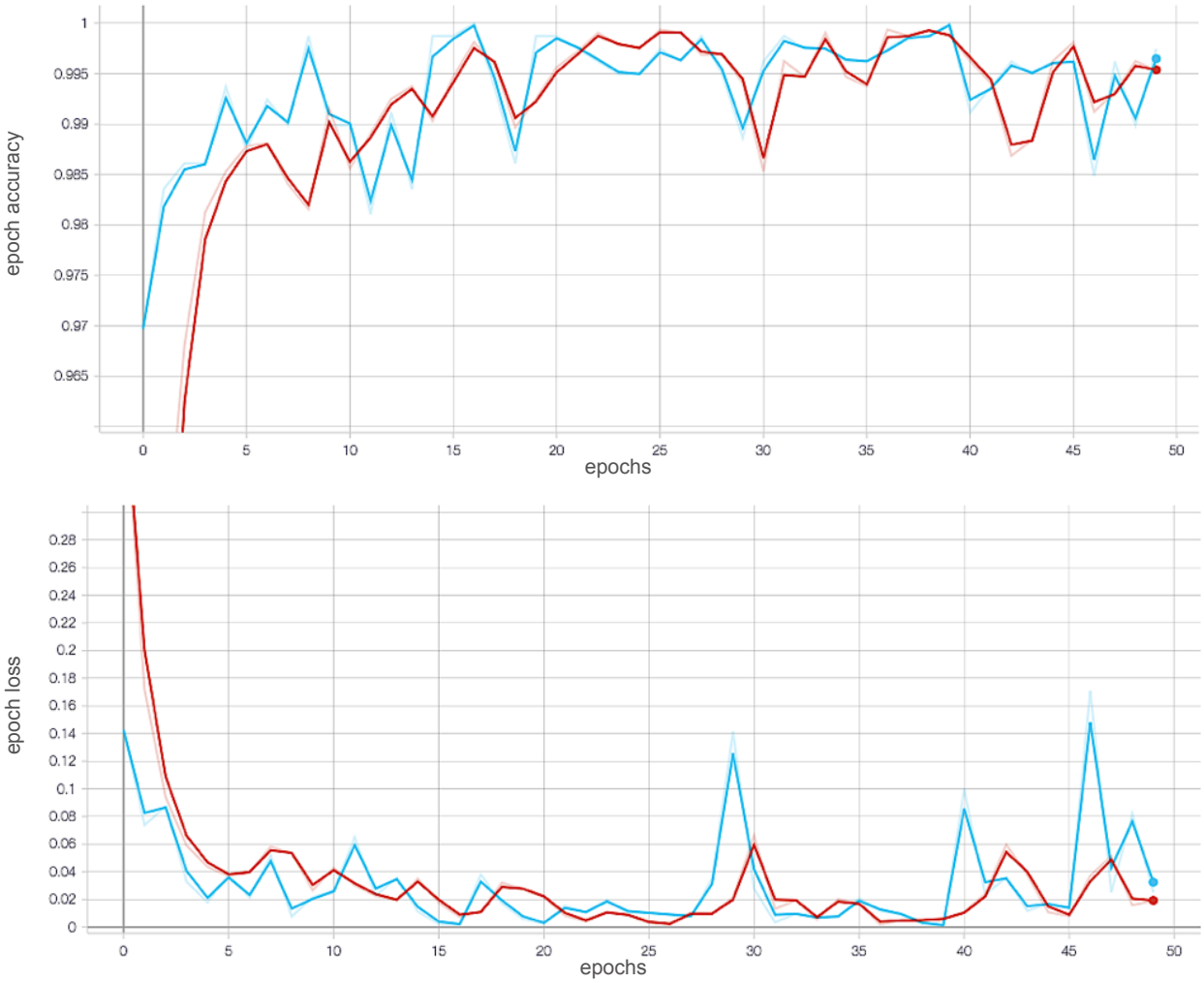
Na osnovu opisanih ponašanja modela, za broj epoha treniranja su odabirane vrijednosti u intervalu od 10 do 60 epoha, a za konačno rješenje primjerice korišteno je 50 epoha treniranja.

Još jedan bitan hiperparametar algoritma koji utječe na performanse modela je i **veličina serije** odnosno broj ulaznih slika po koraku nad kojima se model trenira. Veličina serije se po svojim mogućim vrijednostima kreće od krajnosti jedinične veličine do krajnosti veličine cijelog skupa podataka. Prema [18], kada se koristi jedinična veličina (model dohvaća sliku po sliku i uči na osnovu svake) radi se o stohastičkom gradijentnom padu (engl. *stohastic gradient descent*, SGD) koji računa parametre modela i provodi ažuriranje za svaki primjer treniranja iz skupa podataka (svaku označenu sliku), čime se unosi dosta šuma (osciliranja vrijednosti) pri minimiziranju funkcije pogreške i ostvaruje sporija konvergencija, ali je postupak procesorski manje zahtjevan. U slučaju korištenja veličine cijelog skupa podataka za veličinu serije, radi se o gradijentnom padu skupa (engl. *batch gradient descent*) koji računa gradijent za cijeli skup

podataka odnosno ažuriranje provodi tek nakon cijelog skupa podataka, što daje stabilnije učenje i konvergenciju do minimuma funkcije pogreške, ali je postupak procesorski zahtjevniji jer su potrebne velike količine radne memorije za pohranu cijelog skupa podataka prilikom računanja. Kao kompromis između te dvije krajnosti, obično se koristi metoda gradijentnog pada malog skupa (engl. *mini-batch gradient descent*) koja ažuriranja provodi za svaki mali skup (veličinu serije) podataka. Time se smanjuju kolebanja pri ažuriranju parametara i postiže se stabilnija konvergencija, a sam postupak je procesorski manje zahtjevan, ovisno o odabranoj veličini serije. Kako je navedeno u [19], najčešće korištene veličine serije su višekratnici broja 2 koji odgovaraju zahtjevima grafičke ili procesorske jedinice, kao što su primjerice 32, 64, 128, 256 i slično. Također, navedeno je kako je veličina serije od 32 označene slike (ulazna podatka) jako dobra početna zadana veličina u velikom broju slučaja pa je ta vrijednost i korištena za konačno rješenje.

Od **veličina filtera** isprobane su dimenzije 3×3 i 5×5, pri čemu su karakteristike modela s 3×3 filterima iskazivale velik generalizacijski raskorak (prosječno visoke vrijednosti validacijske pogreške), dok je se izborom filtera dimenzija 5×5 taj raskorak smanjivao (smanjila se vrijednost validacijske pogreške) te su u konačnici odabrani filteri konvolucijskih slojeva dimenzija 5×5.

Također, regularizacija modela odnosno poboljšavanje njegovih generalizacijskih svojstava postignuto je i uvođenjem predobrade slike (uvećanje podataka), povećanjem ukupnog broja podataka u validacijskom skupu (proširivanjem baze podataka) te eventualnim povećanjem broja koraka odnosno serija po epohi. U konačnici je algoritmom opisanim u poglavlju 4.1 dobiven model čiji su dijagrami preciznosti i pogreške treniranja i validacije prikazani na slici 7.6. Iako su postojali modeli s relativno nižim vrijednostima pogreške treniranja i validacije, izabrani model je pokazao najbolja svojstva pri testiranju funkcionalnosti sustava za nove, neviđene slike sentimenta.



Slika 7.6 *Dijagrami preciznosti (gore) i pogreške (dolje) treniranja (crvena krivulja) i validacije (plava krivulja)*

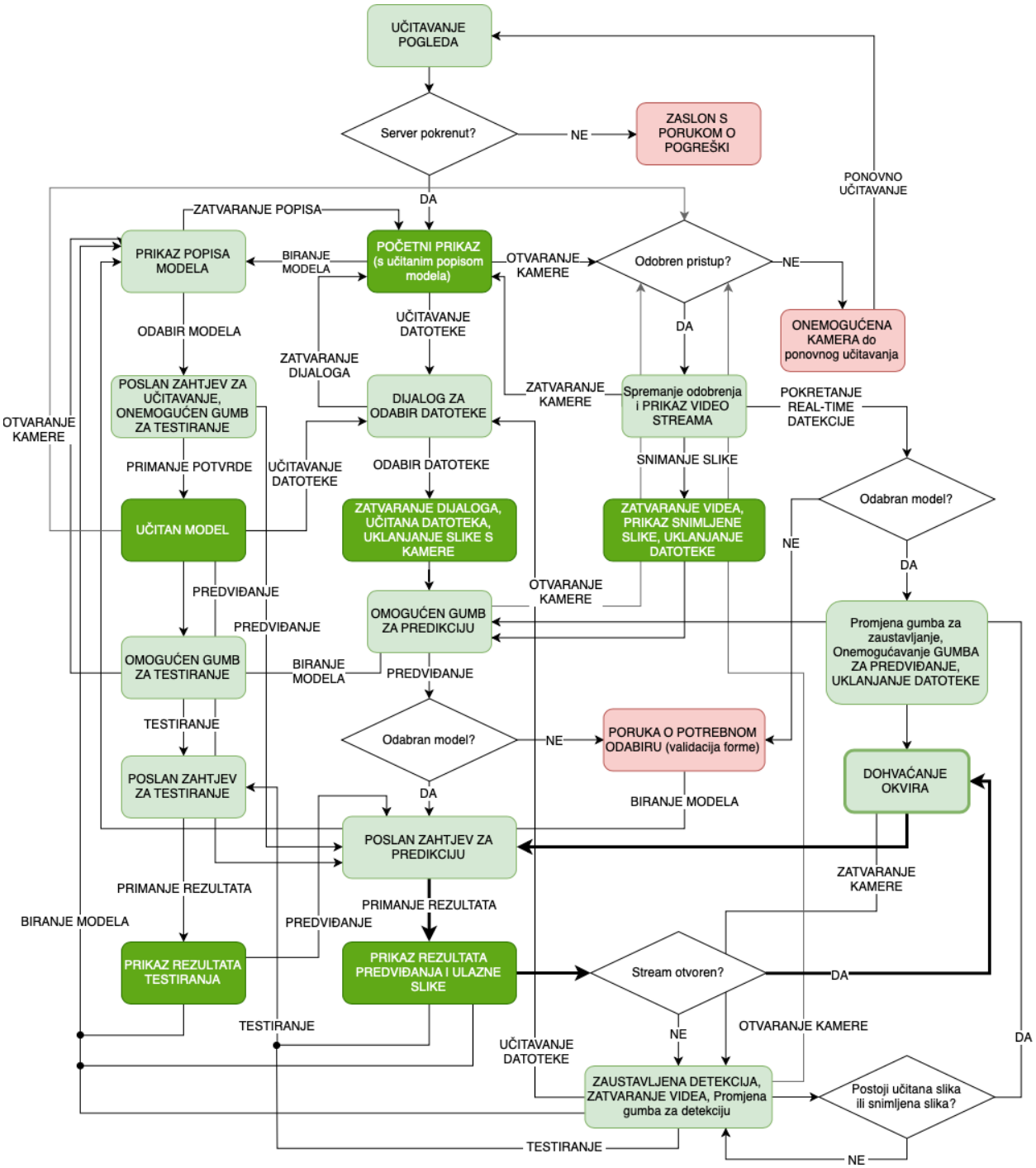
Općenito, osim dobivenih podnaučenih i prenaučenih modela, razlike performansi treniranih modela su većinom tek u nijansama pa je izbor modela, zbog velikog broj kombinacija postavki algoritma koje mogu blago unaprijediti ili unazaditi model, rezultat i intuitivne procjene u odnosu na njegov učinak pri radu u realiziranom sustavu.

7.2. Testiranje rada sustava

Rad sustava testiran je ručnim testiranjem korisničkog sučelja (integracijski test), koje predstavlja ulaznu točku za primanje, obradu i prikaz podataka u sustavu, te ispitivanjem funkcionalnosti sustava u vidu odnosa očekivanih i stvarnih dobivenih izlaza sustava.

7.2.1. Integracijski test

Glavni dio korisničkog sučelja jest upravo pogled za predviđanje, dostupan na ruti „prediction/“. S obzirom na omogućene funkcionalnosti i elemente prikaza, nekoliko je mogućih scenarija odnosno slučajeva korištenja (engl. *use cas*e). Svaki od scenarija podrazumijeva smisleni skup nekoliko interakcija korisnika s grafičkim sučeljem koje daju smislen rezultat. S obzirom da se radi o web aplikaciji, ispitivanje scenarija obavljeno je u pregledniku Safari. Slika 7.7 prikazuje tok glavnih mogućih interakcija s korisničkim sučeljem dobiven integracijskim testiranjem.



***Slika 7.7** Tok interakcija s korisničkim sučeljem*

Tok interakcija prikazuje moguća stanja (primjerice „Početni prikaz“ ili „Učitan model“), akcije (primjerice „Biranje modela“ ili „Predviđanje“) te uvjete (primjerice „Odabran model?“ ili „Stream otvoren“). Različitim akcijama, sa ili bez ispunjenja preduvjeta, moguće je doći u različita stanja sučelja.

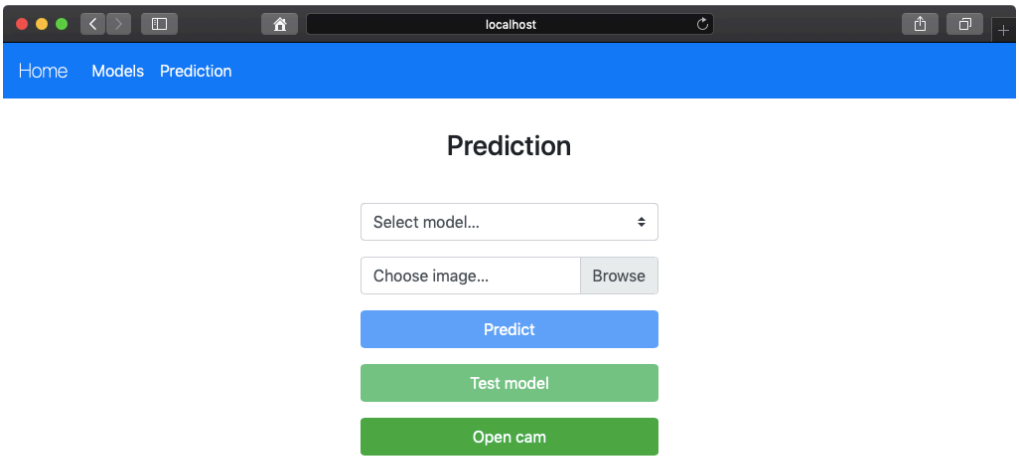
Ponajprije je razmotreno **učitavanje** samog pogleda za predviđanje odnosno početnog prikaza kojem se pristupa klikom na opciju „Prediction“ glavne izborničke trake aplikacije ili ručnim odlaskom na rutu „/prediction/“. Ako je poslužitelj odnosno web aplikacija pokrenuta, pogled se uspješno prikazuje, a u protivnom se ispisuje poruka da se preglednik ne može spojiti s poslužiteljem (Slika 7.8).

Slika 7.8 *Izgled sučelja pri zahtjevu za prikaz aplikacije bez pokrenutog poslužitelja*

Kada se poslužitelj pokrene, istovremeno se pokreće i pozadinski kontejnerski proces za izvršavanje primijenjenih modela te se po zatraženom pristupu pogledu za predviđanje od strane klijenta, isti dostavlja i učitava s prikazanim početnim elementima sučelja, potrebnim skriptama i stilovima za postizanje željenog izgleda i funkcionalnosti elemenata. Pri samom učitavanju početnog pogleda za predviđanje, ujedno se iz skupa istreniranih modula dohvaća popis naziva modela te se dostavlja skupa s pogledom u element „select“. Ako popisa nema ili se dogodi pogreška njegovom učitavanju, sučelje neometano nastavlja s radom, ali će element „select“ biti bez opcija za odabir.

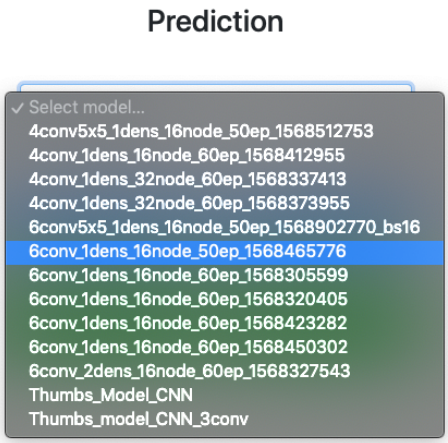
Korisniku se pogled za predviđanje prikazuje s već unaprijed onemogućenim gumbima za predviđanje i testiranje (Slika 7.9), jer pri prvom učitavanju nije odabran nijedan model s popisa. Klikom na iste se, stoga, ne izvršava nikakva radnja. Funkcionalnost testiranja modela omogućit

će se po odabiru jednog od modela i primitku potvrde o učitavanju istog u kontejnerskom procesu, dok će predviđanje biti omogućeno nakon učitane slike (datoteke ili slike s web kamere).



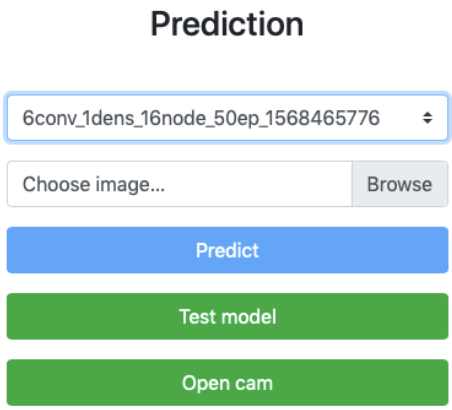
Slika 7.9 *Početni prikaz za predviđanje (onemogućeni gumbi za predviđanje i testiranje)*

Klikom na element odabira modela, korisniku se prikazuje popis modela (Slika 7.10).



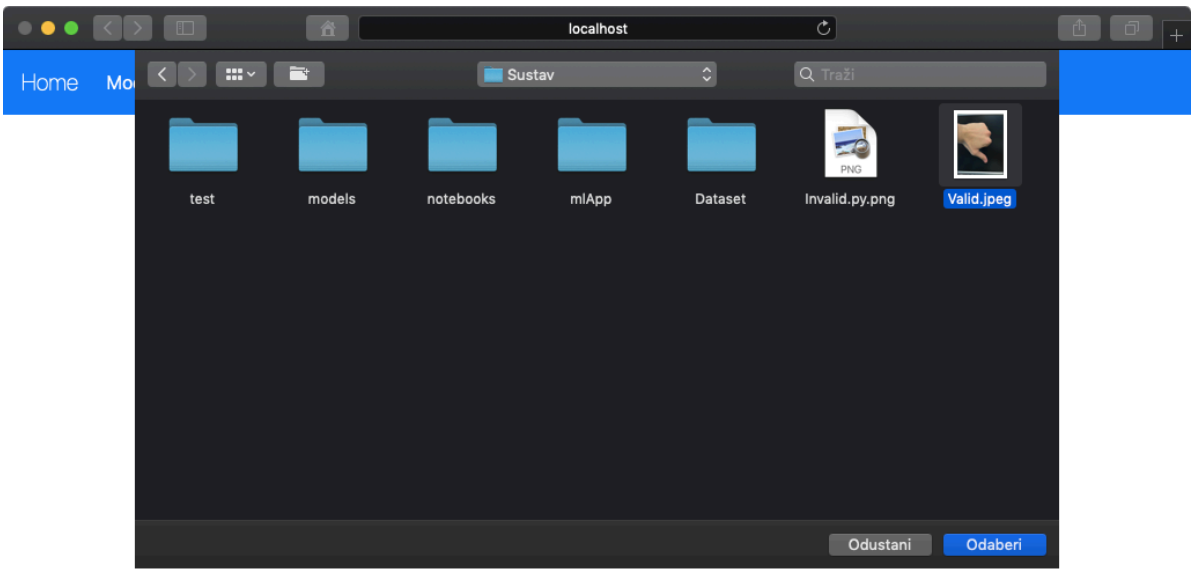
Slika 7.10 *Prikaz popisa modela*

Kada korisnik odabere jedan od modela, šalje se zahtjev za učitavanje tog modela. Kada stigne povratna poruka o učitanom modelu gumb za testiranje postaje omogućen (Slika 7.11). U slučaju da korisnik ponovno odabere neki od modela prije nego prethodni bude učitan, poslat će se još jedan zahtjev poslužitelju koji će redom učitati prvi model, poslati poruku, učitati drugi model te poslati poruku. Rezultat je uvijek učitan zadnji model. Ako korisnik i dalje nije odabrao model, može koristiti funkcionalnosti odabira datoteke, otvaranja i zatvaranja kamere te daljnjeg snimanja slike web kamerom.



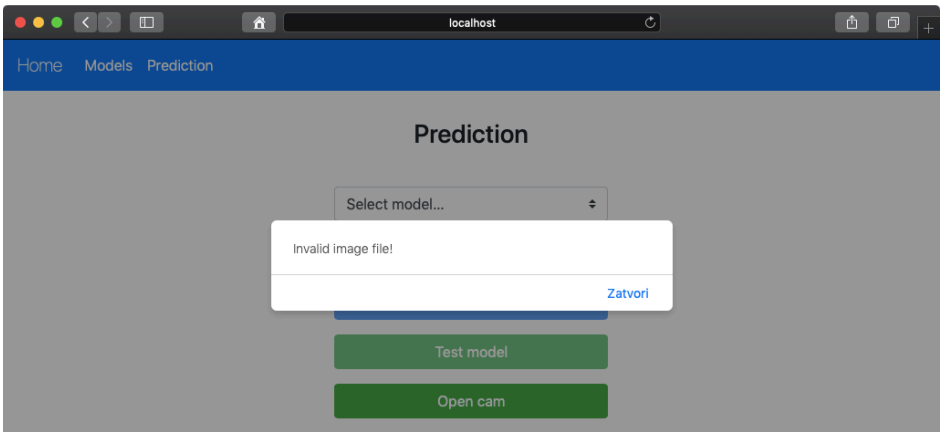
Slika 7.11 *Omogućen gumb za testiranje*

Klikom na element za **učitavanje datoteke** se pokreće dijalog za odabir datoteke s računala (Slika 7.12). Element je postavljen tako da prihvaća samo slikovne formate datoteka pa se ostale neće ni prikazati korisniku. Međutim, kako je prikazano na slici 7.12, ipak će se prikazati i datoteke koje nisu nužno slikovne, ali im je dodijeljena neka od ekstenzija za slikovne formate („Invalid.py.png“ je zapravo preimenovana Python skripta koju sustav sada interpretira kao sliku).



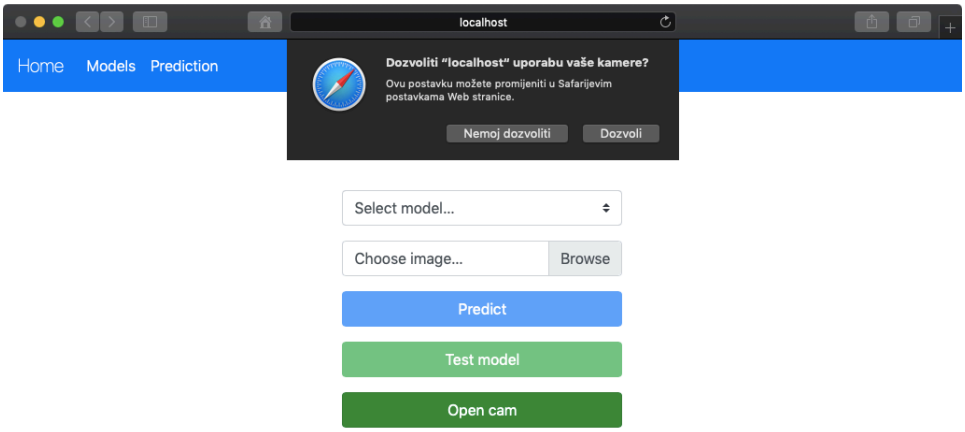
Slika 7.12 *Dijalog za učitavanje slika*

Navedeni problem je riješen na klijentskoj strani („script.js“ datoteka), gdje se pri odabiru datoteke konstruira *Image* objekt te se provjerava njegova ispravnost. Ako odabrana datoteka nije ispravna slikovna datoteka, unos se poništava i korisniku se ispisuje poruka (Slika 7.13). Kada učita ispravnu datoteku, gumb za predviđanje postaje omogućen.



Slika 7.13 *Poruka o neispravnoj slikovnoj datoteci*

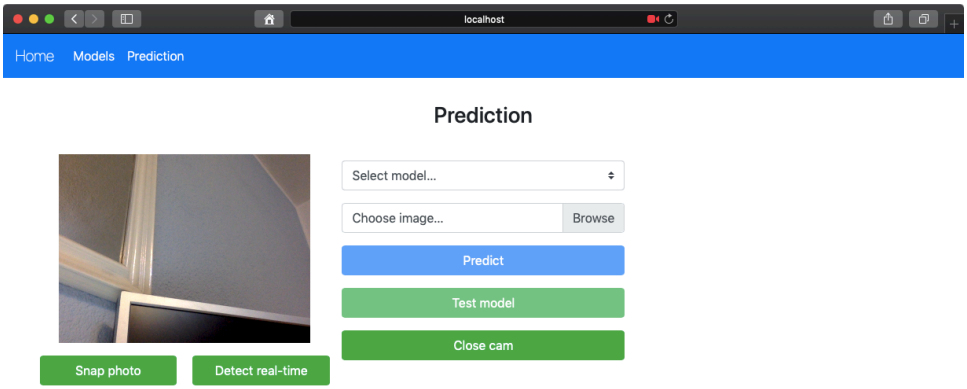
Klikom na gumb za **otvaranje kamere** pojavljuje se dijalog za odobravanje prava pristupa kameri aplikaciji (Slika 7.14). Ako korisnik odbije zahtjev, aplikacija ne može koristiti navedenu funkcionalnost sve dok se stranica ne osvježi kada se može zatražiti novi zahtjev klikom na isti gumb.



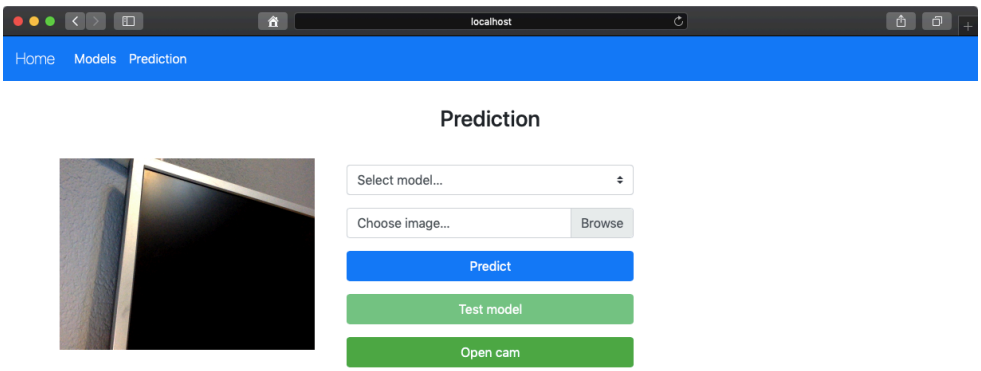
Slika 7.14 *Zahtjev za pristup web kameri*

Ako korisnik odobri pristup, odobrenje se sprema za tu sesiju, pokreće se tok video podataka i prikazuje na elementu „video“ (Slika 7.15). Korisnik dobiva dvije nove opcije – snimanja slike web kamerom („Snap photo“ gumb) te predviđanja sentimenta u stvarnom vremenu s videa kamere („Detect real-time“ gumb). Opcijom snimanja slike, uhvatit će se trenutni okvir toka video podataka s web kamere te će se spremiti na elementu „canvas“ i prikazati. Uz to, tok video podataka će biti prekinut, zauzeti resursi oslobođeni i kamera više neće biti aktivna. Ako korisnik učita ispravnu sliku kao datoteku ili istu snimi web kamerom, gumb za predviđanje postaje omogućen (Slika 7.16). Pri tome, opcija slikovne datoteke i slike s web kamere su međusobno isključive – pri potvrdi odabira datoteke, uklanja se postojeća slika web kamere s

elementa „canvas“, a klikom na gumb „Snap photo“ uklanja se učitana datoteka iz elementa „input“, ako postoji. Time je osigurano predviđanje na osnovu jedinstvenog ulaza, bez davanja prioriteta.

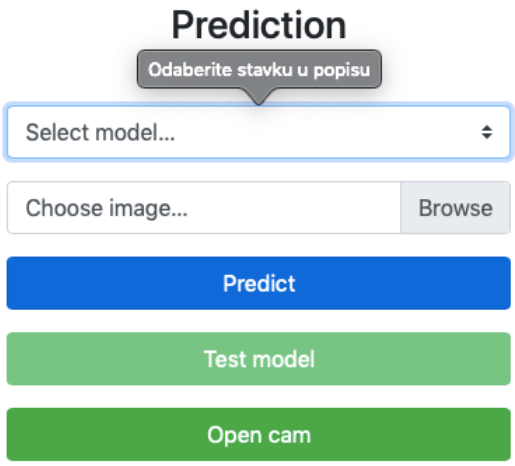


Slika 7.15 *Prikaz video streama kamere*



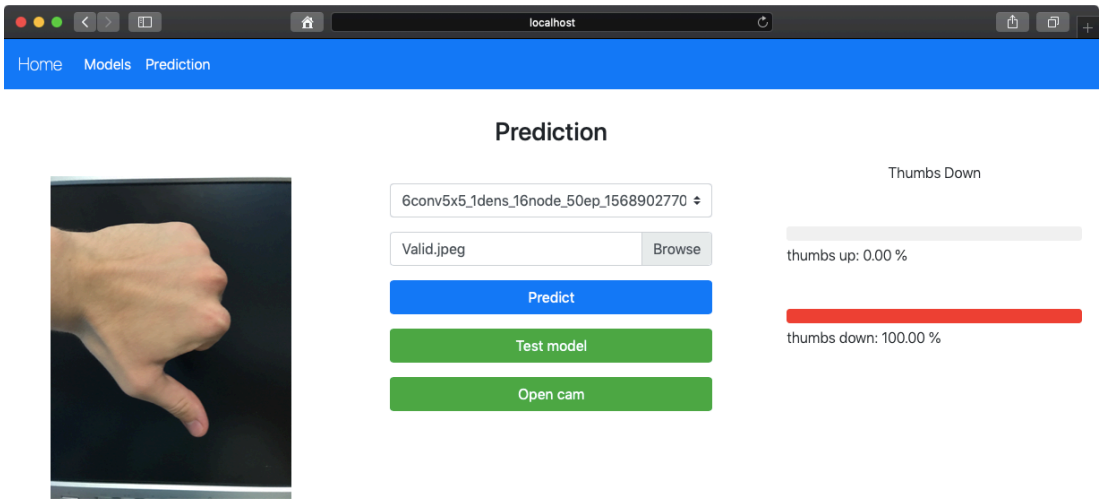
Slika 7.16 *Snimljena slika s kamere i omogućen gumb za predviđanje*

Klikom na gumb za **predviđanje** „Predict“, pokrenut će se proces validacije forme kako bi se izvršio zahtjev za predviđanje. Ako korisnik nije odabrao model (isti ne mora biti učitan), pojavit će se poruka o potrebnom odabiru modela prikazana na slici 7.17. Kada korisnik odabere model, prvo se šalje zahtjev za učitavanje modela, a zatim je moguće je poslati i zahtjev za predviđanje koji ne mora nužno čekati potvrdu o učitanom modelu. Naime, sustav će redom odgovoriti na zahtjeve, a budući da na poslužitelju postoji definirani protokol upravljanja različitim vrstama poruka (poglavlje 6.2.2), neće doći do pogreške. Ako je korisnik kliknuo na gumb za predviđanje prije nego je model učitan, odgovor poslužitelja s rezultatima predviđanja bit će vremenski odgođen za vrijeme učitavanja modela u kontejnerskom procesu. Ako je model već učitan, odgovor poslužitelja s rezultatima predviđanja stiže bez velikog čekanja (od 100 do 200 milisekundi prosječno po zahtjevu).



Slika 7.17 *Poruka o validacijskoj pogreški forme - model nije odabran*

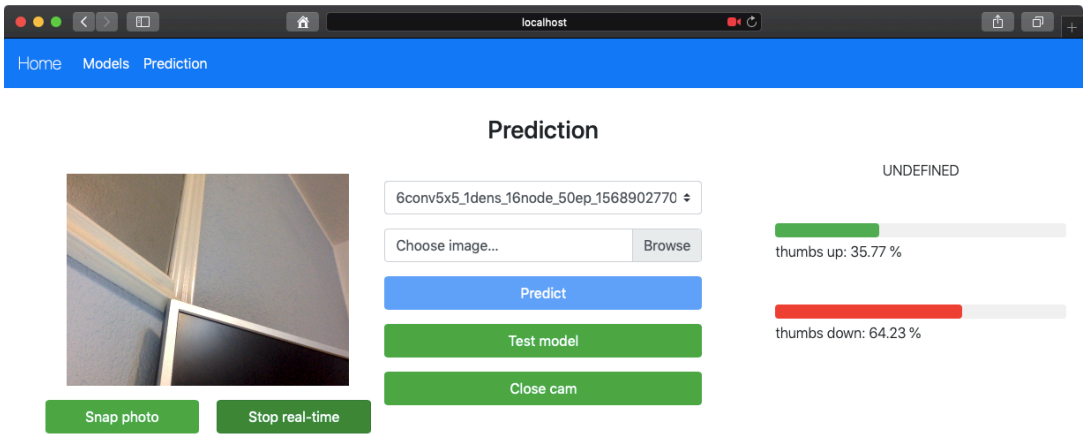
Rezultati se prikazuju u elementima za prikaz vjerojatnosti pozitivnog i negativnog sentimenta u postotcima te u elementu za prikaz procijenjene klase sentimenta. Uz to, prikazuje se i ulazna slika u lijevom stupcu prikaza, kako je predočeno a slici 7.18.



Slika 7.18 *Izgled prikaza rezultata predviđanja*

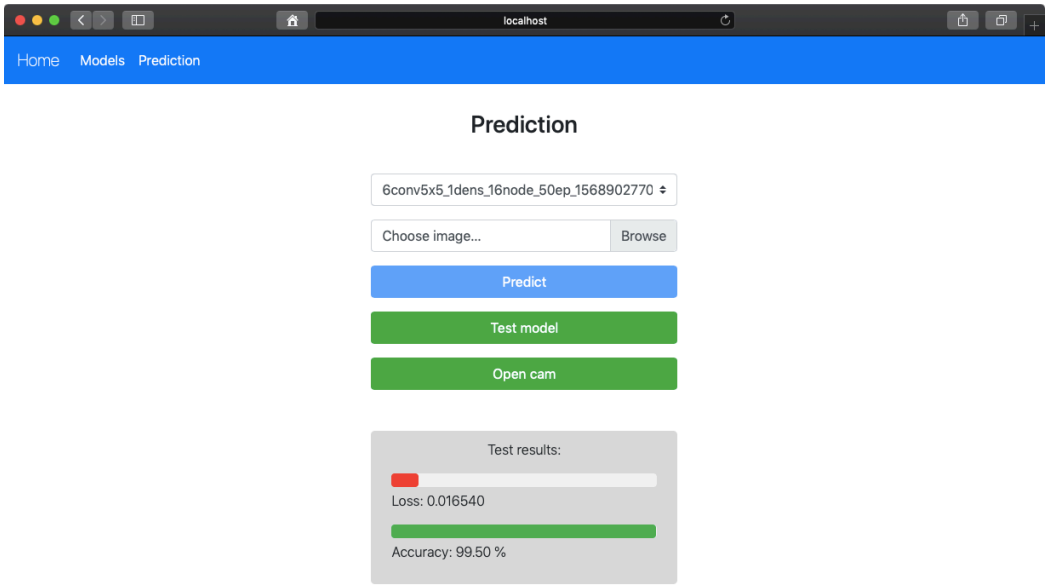
Isto se postiže i opcijom **detekcije sentimenta u stvarnom vremenu** – pri čemu se klikom na gumb ponajprije provjerava je li odabran model (validacija forme). Ako nije, ispisuje se ista poruka o validacijskoj pogreški kao na slici 7.17. Ako je model odabran, pokreće se detekcija u stvarnom vremenu (Slika 7.19) u vidu iterativnog ponavljanja zahtjeva za predviđanje: najprije se šalje jedan zahtjev za trenutni okvir te kada stigne odgovor od poslužitelja i rezultati budu prikazani, ako je tok video podataka s web kamere još uvijek otvoren, šalje se novi zahtjev za novi trenutni okvir. Postupak se ponavlja sve dok se tok video podataka s web kamere ne zatvori klikom

na gumb „Stop real-time“ ili „Close cam“ te sučelje završava u stanju kao u slučaju običnog prikaza rezultata (Slika 7.18).



Slika 7.19 *Izgled sučelja za detekciju sentimenta u stvarnom vremenu*

Konačno, klikom na gumb „Test model“ nakon što je model potvrdno učitan, šalje se zahtjev za **testiranje modela** i gumb se onemogućuje dok ne primi odgovor poslužitelja s rezultatima. Kada odgovor stigne rezultati se prikazuju kao na slici 7.20.



Slika 7.20 *Izgled sučelja za prikaz rezultata testiranja*

Prosječna **brzina odziva** sustava za zahtjeve predviđanja je u normalnim uvjetima prosječno od 100 do 200 milisekundi. Vrijeme odziva je kraće ako se radi o memorijski manjim ulaznim slikama, naročito slikama s web kamere. Kod predviđanja sa slika učitanih kao datoteke, vrijeme odziva je nešto dulje zbog rada s datotekom na klijentu i poslužitelju. Uz to, zbog

programske logike korištene u kontejnerskom procesu za određivanje klasa sentimenta (poglavlje 6.2.3), slike pozitivnog ili negativnog sentimenta prosječno traže i dulje vrijeme odziva od slika s neodređenim sentimentom, budući da se za slike, za koje je vrijednost predikcije ispod praga vjerojatnosti, ne određuje o kojoj se točno od binarnih klasa sentimenta radi. Slika 7.21 prikazuje odgovore poslužitelja za veću učitanu slikovnu datoteku, zatim sliku snimljenu web kamerom te na kraju učitanu manju sliku s neodređenim sentimentom.

```
POST /prediction/upload 200 197.468 ms - 689066
{'category': 'Thumbs Down', 'probability': 0.99999976}
```

```
POST /prediction/upload 200 137.922 ms - 221159
{'category': 'Thumbs Up', 'probability': 9.515569e-05}
```

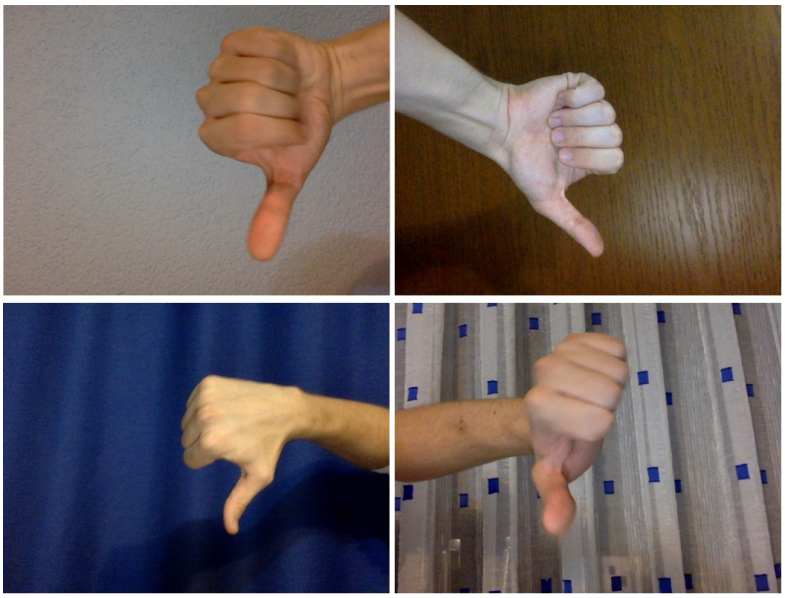
```
POST /prediction/upload 200 97.738 ms - 220860
{'category': 'UNDEFINED', 'probability': 0.37984407}
```

Slika 7.21 *Vremena odziva poslužitelja za različite zahtjeve predviđanja*

Također, vrijeme odziva za učitavanje modela ovisit će o veličini samog modela, dok za testiranje modela veličina modela ne igra toliko značajnu ulogu.

7.2.2. Funkcionalni test

Testiranje funkcionalnosti ovog sustava podrazumijeva provjeru procjene sentimenata sa zadanog testnog skupa ulaznih slika. Usporedit će se procijenjene klase sentimenata i njihove vjerojatnosti dobivene predviđanjem sa stvarnim sentimentima na slikama. Testni skup slika sentimenta podrazumijeva strukturiran skup slika odvojen od originalne baze podataka (slika) za treniranje i validaciju. Sastoji se od ukupno 300 slika sentimenta i to po 100 slika za svaki od moguće klase sentimenta – pozitivni (palac-gore), negativni (palac-dolje) i neodređeni (ni jedno ni drugo). Slike su prikupljene na isti način kao i slike baze podataka (poglavlje 6.1.1.) te sadrže pretežno 4 različite pozadine (jednostavna svijetla, tamna i neutralna te pozadina složenije teksture). Na slici 7.22 prikazani su primjeri slika (za negativan sentiment). Slike neodređenog sentimenta podrazumijevaju slike same pozadine, slike ruke i šake bez naznake sentimenta (palac-gore ili palac-dolje) te mutne slike gesti ili ruke.



Slika 7.22 *Primjer testnih slika (četiri vrste pozadine)*

Testirane su funkcionalnosti odnosno rezultati predviđanja na izrađenom testnom skupu za različite vrijednosti praga izlazne vjerojatnosti. Pri tome je korišten istrenirani model strojnog učenja opisan u poglavlju 4.1. Primjer rezultata testiranja, za veću vrijednost praga (0.95) prikazan je tablicom 7.1, a tablicom 7.2 primjer rezultata testiranja s korištenjem manje vrijednosti praga (0.6). Iz obje tablice se da uočiti kako je predviđanje pozitivnog i negativnog sentimenta daleko točnije od predviđanja neodređenog sentimenta, jer je model treniran upravo na skupovima pozitivnog i negativnog sentimenta. Neodređeni sentiment u tom kontekstu ne predstavlja pravu klasu sentimenta, već je to vjerojatnost da model nije dovoljno siguran da se radi o jednom od dva binarna sentimenta (pozitivni ili negativni).

Tablica 7.1 *Rezultati testiranja za vrijednost praga 0.95*

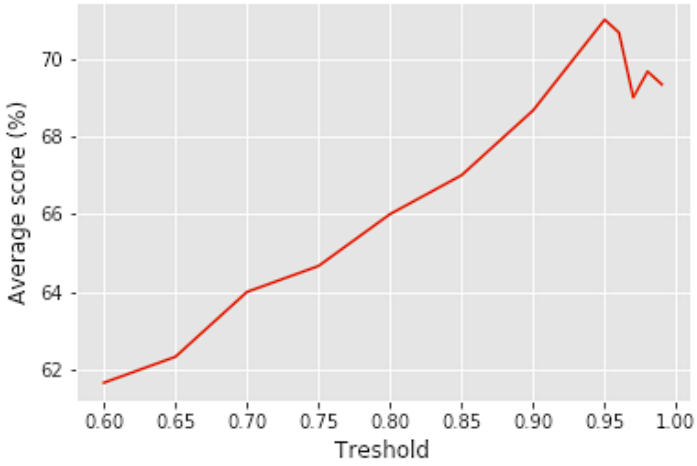
Procijenjeni sentiment / Ulazni sentiment	Pozitivan	Negativan	Neodređen
Pozitivan	87	4	9
Negativan	12	82	6
Neodređen	29	27	44

Tablica 7.2 *Rezultati testiranja za vrijednost praga 0.65*

Procijenjeni sentiment / Ulazni sentiment	Pozitivan	Negativan	Neodređen
Pozitivan	93	6	1
Negativan	13	85	2
Neodređen	45	46	9

Iz prikazanih rezultata da se zaključiti kako smanjivanjem vrijednosti praga raste broj točnih predviđanja pozitivnog i negativnog sentimenta jer se vrijednosti predviđene s manjom razinom pouzdanosti ne odbacuju klasificiranjem kao neodređen sentiment, a broj točnih predviđanja neodređenog sentimenta se smanjuje jer je smanjen i interval vjerojatnosti za koji se slike klasificiraju kao neodređen sentiment. Za veće vrijednosti praga smanjuje se broj točno predviđenih pozitivnih i negativnih sentimenata jer veća vrijednost praga traži i veću potrebnu pouzdanost odnosno vjerojatnost procjene, dok broj točno predviđenih neodređenih sentimenata raste jer je širi i interval vjerojatnosti u kojem se slike sentimenta klasificiraju kao takve.

Testiranjem funkcionalnosti sustava za različite vrijednosti praga vjerojatnosti, dobiven je graf ovisnosti prosječne točnosti o razini praga prikazan na slici 7.23. **Prosječna točnost sustava** podrazumijeva prosječni broj točno predviđenih sentimenata za pozitivne, negativne i neodređene ulazne sentimente. Te su vrijednosti na prethodnim tablicama (Tablica 7.1 i Tablica 7.2) naglašene i nalaze se na glavnoj dijagonali.



Slika 7.23 *Graf ovisnosti prosječne točnosti o vrijednosti praga vjerojatnosti*

Iz grafa ovisnosti prosječne točnosti sustava o vrijednosti praga vjerojatnosti se da uočiti kako s porastom vrijednosti praga raste i ukupna prosječna točnost. Međutim, uočena je točka maksimuma (za vrijednost praga 0.95), gdje se pri daljnjem povećavanju vrijednosti praga točnost sustava smanjuje ili neznatno mijenja. Stoga je, kao konačna vrijednost praga vjerojatnosti sustava, odabrana vrijednost 0.95, pri kojoj ukupna točnost sustava iznosi 71%.

7.3. Osvrt na cjelokupan sustav

Realizirani sustav pokazuje osnovne zadovoljavajuće performanse glede korisničkog sučelja i funkcionalnosti odnosno rezultata predviđanja, što potvrđuje i ukupna dobivena točnost sustava. Međutim sam model i algoritam strojnog učenja, pa tako i cjelokupan sustava, posjeduju određena ograničenja koja utječu na performanse sustava.

Jedan od bitnih čimbenika koji utječe na rad i rezultate predviđanja sustava je **baza podataka** odnosno slika sentimenta. Korištena je količinski relativno mala baza podataka s manjom raznolikošću u usporedbi s naprednijim modelima strojnog učenja koji koriste baze podataka od desetke tisuća slika (podataka). Uz to, bitan faktor za procjenu sentimenta i izvlačenje značajki sa slike je i **pozadina slike**. U slučaju kompleksne pozadine (različite teksture, rubovi, boje, objekti i osvjetljenja) sustav pretežito daje lošije ili netočne rezultate, dok za jednostavne pozadine (primjerice jednobojan zid) daje jako dobre rezultate. Zbog nedovoljno velike raznolikosti baze podataka, kao i njene same veličine, dobiveni modeli su obično imali problem s generalizacijom odnosno predviđanjem novih slika sentimenta.

Nastavno na problem utjecaja baze slika, nadovezuje se i problem **ograničenosti procesorske** moći za realizaciju sustava odnosno nemogućnost treniranja kompleksnijih modela nad većim bazama podataka što bi najvjerojatnije rezultiralo većom točnošću sustava. Stoga su i postavke modela i algoritma odabrane prema raspoloživim mogućnostima (primjerice kapacitet modela odnosno neuronske mreže, veličina serije i slično).

Odabrane tehnologije, okruženja i alati ne utječu značajno na same performanse sustava te je njihov izbor ovisio o njihovim značajkama. Tako je primjerice Keras okruženje odabrano zbog svoje jednostavnosti i veće razine apstrakcije, tj. ne zahtijeva puno koda niske razine (engl. *low level code*) za razliku od PyTorcha koji također predstavlja okruženje za strojno učenje, ali niže razine.

Sustav, iako uspješno realiziran po traženim zahtjevima, moguće je dodatno unaprijediti određenim **proširenjima i poboljšanjima**. Tako bi primjerice alternativan pristup prijenosnog

učenja (engl. *transfer learning*) za izgradnju modela na osnovi već gotovih, složenijih modela, istreniranih na velikoj količini podataka, zasigurno dao rezultate veće točnosti. Uz to, u funkciji poziva za praćenje pogreški i preciznosti po koracima pri treniranju, moguće je definirati funkcionalnost ranijeg zaustavljanja (engl. *early stopping*) treniranja ako se uoči pojava prenaučenosti modela. Također, moguće je koristiti dodatne razne metode optimizacije i regularizacije algoritma strojnog učenja za dobivanje preciznijih modela s boljim generalizacijskim svojstvima.

8. ZAKLJUČAK

Strojno učenje, kako je prikazano u ovome radu, moguće je implementirati u sustave za različite primjene. Izvodeći postupke i modele strojnog učenja, sustavi omogućuju upravljanje podacima te učenje i predviđanje iz istih, na osnovu čega se mogu izvući korisne informacije i predvidjeti ponašanja.

Ovim diplomskim radom realiziran je jedan takav sustav, s fokusom na primjenu za vizualnu analizu sentimenta. Kako nalaže i sam zadatak rada, osmišljen je, programski izveden i ispitan sustav za prilagodljivu ugradnju i izvođenje postupaka i modela strojnog učenja kojeg karakteriziraju jednostavnost, modularnost, prilagodljivost te relativno zadovoljavajuća razina točnosti. Realizirani sustav, kako je i opisano u radu, koristi neke od standardnih postupaka analize i strojnog učenja sa slikovnih podataka, tehnologije izrade namijenjene za brzo, jednostavno i učinkovito implementiranje algoritama i modela strojnog učenja u sustav te prikladne metodologije razvoja poslovnih programskih rješenja. Proučavanjem i izvođenjem gotovog sustava, stječe se uvid o utjecaju različitih parametara i postavki algoritma i modela strojnog učenja, izgleda i opsega baze podataka za treniranje te raspoložive procesorske moći na konačne performanse sustava. Svaki od navedenih elemenata moguće je dodatno unaprijediti kako bi se performanse sustava dodatno poboljšale.

Iako je sustav, za potrebe diplomskog rada, usredotočen na binarnu vizualnu analizu sentimenta u obliku palac-gore ili palac-dolje, sustav je moguće primijeniti za analizu bilo kojeg binarnog sentimenta sa pripadajućom bazom slika i to u različitim područjima – primjerice kod kupaca pri kupovini (ili korisnika općenito) kao način ocjenjivanja zadovoljstva uslugom.

LITERATURA

[1] J. Hurwitz, D., Kirsch, *Machine Learning For Dummies*, IBM Limited Edition, 2018.

[2] B. D. Bašić, J. Šnajder, Strojno učenje: *Uvod u strojno učenje*, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, 2016., dostupno na: https://www.fer.unizg.hr/_download/repository/SU-2016-01-Uvod.pdf [28. lipnja 2019.]

[3] E. Alpaydin: *Introduction to Machine Learning*, MIT Press, 2009.

[4] Goodfellow, Y., Bengio, A., Courville, *Deep Learning*, MIT Press, 2016.

[5] J. Brownlee, Better Deep Learning: *A Gentle Introduction to Learning Curves for Diagnosing Machine Learning Model Performance*, Machine Learning Mastery, 2019., dostupno na: https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/ [28. lipnja 2019.]

[6] F. Novoselnik, *Klasifikacija slika metodama dubokog učenja*, Diplomski rad, Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, Osijek, 2017.

[7] J. Krapac, S. Šegvić, *Konvolucijski modeli*, dostupno na : http://www.zemris.fer.hr/~ssegvic/du/du2convnet.pdf [28. lipnja 2019.]

[8] B. Liu, *Sentiment Analysis and Opinion Mining*, Morgan & Claypool Publishers, 2012.

[9] E. Cambria, D. Das, S. Bandyopadhyay, i A. Feraco, Ur., *A Practical Guide to Sentiment Analysis*, Springer International Publishing, 2017.

[10] Karpathy, CS231n Convolutional Neural Networks for Visual Recognition: *Convolutional Neural Networks (CNNs / ConvNets)*, 2019., dostupno na: http://cs231n.github.io/convolutional-networks/ [11. rujna 2019.]

[11] Beginners Guide/Overview, wiki.python.org, dostupno na: https://wiki.python.org/moin/BeginnersGuide/Overview [13. rujna 2019.]

[12] Open Source Computer Vision Documentation, dostupno na: https://docs.opencv.org/4.1.0/ [13. rujna 2019.]

[13] TensorFlow, dostupno na: https://www.tensorflow.org/about [13. rujna 2019.]

[14] Keras Documentation, dostupno na: https://keras.io [13. rujna 2019.]

[15] TensorFlow Core Guide: Keras, dostupno na: https://www.tensorflow.org/guide/keras [13. rujna 2019.]

[16] Node.js Documentation: Child Process, dostupno na: https://nodejs.org/api/child_process.html [14. rujna 2019.]

[17] C. Shorten, *Image Classification Keras Tutorial: Kaggle Dog Breed Challenge*, Towards Dana Science, 2018., dostupno na: https://towardsdatascience.com/image-classification-python-keras-tutorial-kaggle-challenge-45a6332a58b8 [17. rujna 2019.]

[18] T. Krambeger, B. Nožica, I. Dodig, D. Cafuta, *Pregled Tehnologija u Neuronskim Mrežama*, Tehničko Veleučilište u Zagrebu, 2019.

[19] J. Brownlee, Deep Learning: *A Gentle Introduction to Mini-Batch Gradient Descent and How to Configure Batch Size*, Deep Learning, Machine Learning Mastery, 2019., dostupno na: https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/ [17.09.2019]

SAŽETAK

Tema ovog diplomskog rada je razvoj sustava za prilagodljivu ugradnju i izvođenje postupaka i modela strojnog učenja za primjenu u području vizualne analize sentimenta, koji je potrebno i programski ostvariti. Opisani su zahtjevi na takav sustav te analizirani prikladni postupci i modeli strojnog učenja koji omogućuju detekciju i analizu sentimenta sa slike. Proučeno je i obrađeno područje umjetnih neuronskih mreža s naglaskom na konvolucijske neuronske mreže. Pobliže je opisano područje primjene odnosno analize sentimenta, uz prikaz već postojećih sustava za implementiranje strojnog učenja u tu svrhu. Odabran je odgovarajući algoritam te je objašnjen njegov princip rada i parametri. Zatim su opisane i razine te funkcionalne komponente arhitekture sustava te je objašnjena njihova međusobna komunikacija. Koristeći adekvatna okruženja, alate i tehnologije (TensorFlow, Keras, Python, Node.js i drugi), koji su također pobliže objašnjeni u radu, sustav je i programski razvijen. Ponajprije je izrađena baza podataka slika, na osnovu koje je odabranim algoritmom treniran i dobiven model za predviđanje sentimenta sa slike. Zatim je programski ostvaren modul za predviđanje kao web aplikacija koja omogućuje upotrebu sustava krajnjem korisniku. Konačno, sustav je testiran po pitanju integracije korisničkog sučelja i korištenja te funkcionalnosti sustava, uz analizu dobivenih rezultata i osvrt na cjelokupan sustav.

Ključne riječi: Analiza sentimenta, Konvolucijske neuronske mreže, Python, Razvoj sustava, Strojno učenje

ABSTRACT

Development of system for flexible implementation and execution of machine learning algorithms and models

The topic of this paper is the development of system for flexible implementation and execution of machine learning algorithms and models for application in the field of visual sentiment analysis as well as the realization of the very system programmatically. System requirements have been described and suitable macine learning algorithms and models that enable image sentiment analysis have been analyzed. The field of artificial neural networks has been studied and processed with emphasis on convolutional neural networks. The field of application, i.e. sentiment analysis has been elaborated in more detail, while presenting the already existing systems for machine learning implementation for this very application. An appropriate algorithm has been selected and its operating principle and parameters explained. Then, the levels and functional components of the system architecture have been described while explaining ways of their communication. Using the suitable environments, tools and technologies (TensorFlow, Keras, Python, Node.js and others), that had also been explained in the paper, the system has been programmatically developed. First, an image database had been created, based on which the selected algorithm has trained and obtained the model for predicting sentiment from an image. Then, the prediction module has been implemented as a web application that enables the system to be used by the end user. Finally, the system has been tested for the integration of the user interface and use, as well as for the functionalities of the system, with a provided analysis of the results obtained and a review of the overall system.

Keywords: Sentiment analysis, Convolutional neural networks, Python, System development, Machine learning

ŽIVOTOPIS

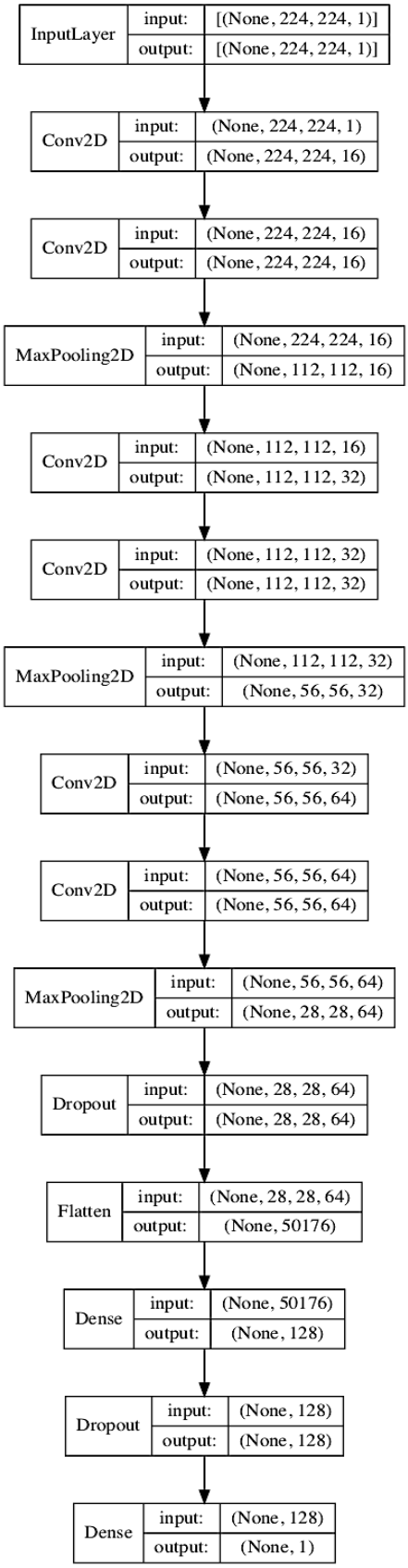
Vedran Ivić, student je druge godine diplomskog sveučilišnog studija elektrotehnike na Fakultetu Elektrotehnike, Računarstva i Informacijskih Tehnologija u Osijeku. Rođen je 09.10.1995. u Münchenu (Njemačka), a 2002. započinje svoje osnovnoškolsko obrazovanje u Odžaku (Bosna i Hercegovina) koje završava kao učenik generacije s istaknutim uspjesima na natjecanjima iz matematike i fizike. Godine 2010. upisuje opću gimnaziju u Srednjoj školi Pere Zečevića u Odžaku, gdje uz sudjelovanje i ostvarene uspjehe na natjecanjima te izvanškolskim aktivnostima ponovno postaje učenik generacije. U Osijeku, 2014. godine upisuje preddiplomski studij elektrotehnike na tadašnjem Elektrotehničkom Fakultetu, na kojem se odlučuje za smjer Komunikacije i informatika. Kao student FERIT-a, držao je demonstrativnu nastavu iz više kolegija te sudjelovao na više međunarodnih natjecanja studenata elektrotehnike (Elektrijada), gdje je 2017. osvojio osobno i ekipno prvo mjesto iz Engleskog jezika. Od 2016. do 2018. član je međunarodne organizacije mladih AIESEC, gdje, kroz rad u različitim timovima, organizira različite projekte, konferencije i međunarodne razmjene s naglaskom na društveno koristan rad i razvoj liderstva te postaje i potpredsjednik Lokalnog Odbora AIESEC Osijek. Kroz fakultetsko obrazovanje iz godine u godinu ostvaruje izvrstan uspjeh, a 2017. dobiva i dekanovu nagradu (za uspješnost u studiranju) kao najbolji student na svome smjeru. Preddiplomski studij elektrotehnike završava 2017. godine te stječe titulu sveučilišnog prvostupnika inženjera elektrotenike na temu završnog rada "C# web aplikacija za generiranje testova" izrađenog u suradnji s tvrtkom Mono, a nakon toga upisuje diplomski sveučilišni studij elektrotehnike, smjer Komunikacije i informatika, izborni bloka Mrežne tehnologije. Godine 2018. odrađuje stručnu praksu u tvrtki Plava Tvornica kao Android razvojni programer, gdje stječe znanja o naprednim konceptima i metodama razvoja Android mobilnih aplikacija te o radu u timu i tvrtki općenito. U akademskoj 2018./2019. godini, dobitnik je dekanove nagrade za izniman uspjeh u studiranju i ostvarene izvannastavne aktivnosti kojima je pridonio ugledu Fakulteta.

Vedran Ivić

PRILOZI

- Na CD-u:
  - **Prilog 1**: Elektronička verzija rada – dokument u *.pdf* formatu
  - **Prilog 2**: Elektronička verzija rada – dokument u *.docx* formatu
  - **Prilog 3**: Komprimirana datoteka realiziranog sustava (s potrebnim programskim datotekama i web aplikacijom te bazom slika i modelima)

- **Prilog 4**: Struktura korištene konvolucijske neuronske mreže:



P.4.1. *Struktura korištene konvolucijske neuronske mreže*