Upotreba dubokog učenja za klasifikaciju tehničkih dokumenat

Kokanović, Karlo

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

Sveučilišni diplomski studij računarstva

Upotreba dubokog učenja za klasifikaciju tehničkih dokumenata

Diplomski rad

Karlo Kokanović

Osijek, 2025.

Sadržaj

1. UVOD

Tehnička dokumentacija skup je pisanih dokumenata koji opisuju primjenu, svrhu ili arhitekturu proizvoda i/ili usluge. Cilj je tehničke dokumentacije opisivanje proizvoda organizacije i pružanje detaljnih informacija o funkcionalnosti, dizajnu, upotrebi, održavanju i drugim aspektima nekog tehničkog proizvoda ili sustava. Dokumentacija služi kao vodič za inženjere, tehničare i korisnike te uključuje različite vrste informacija i formata, a pri tome daje na uvid korisne informacije o proizvodu što zainteresiranoj strani uvelike pomaže u učenju i analizi samoga dokumenta.

Opisne i složene informacije koje mogu biti zahtjevne za razumijevanje bez odgovarajućeg tehničkog znanja često se nalaze u tehničkoj dokumentaciji. Veliku količinu dokumentacije može biti teško organizirati i održavati što otežava pronalaženje specifičnih informacija. Pronalaženje relevantnih patenata također može biti izazovno zbog velikog broja dostupnih dokumenata i manjka potrebnih vještina za pretragu. Različite zemlje imaju različite patentne zakone i propise, kao i jezike kojima pišu dokumentaciju, što može rezultirati otežanim čitanjem krajnjeg korisnika. Međunarodna klasifikacija patenata (engl. International Patent Classification, IPC) međunarodno je dogovorena konvencija prihvaćena za rad s dokumentacijom. Klasifikaciju patenata Sjedinjenih Američkih Država (engl. United States Patent Classification, USPC) odredio je Ured za patente i žigove Sjedinjenih Američkih Država (engl. United States Patent and Trademark Office, USPTO). IPC sustav redovito se ažurira, na godišnjoj bazi, stoga je imperativ pratiti najnovije podjele i propise. Dobro organizirana klasifikacija tehničke dokumentacije omogućuje inženjerima lakši rad s dokumentima. Univerzalno priznata klasifikacija kao što je IPC pomaže u definiranju dobrih praksi za rukovanje tehničkom dokumentacijom. Jasna hijerarhija poboljšava učinkovitost pretraživanja sadržaja unutar dokumenata i automatizira proces analize. Poznavanjem i primjenom IPC klasifikacije inovatori se mogu lakše sporazumjeti i razmjenjivati vlastite ideje. Redovitim ažuriranjem IPC-a jasno se ističu trendovi kako bi svi sudionici i zainteresirani bili upoznati s najsuvremenijim tehnologijama i pristupima rješavanju nekog problema. Visoki standardi nužni su prilikom rada s dokumentacijom jer su javno dostupni, stoga moraju biti smisleni i interpretabilni svakom budućem čitatelju. Praćenjem trendova tvrtke mogu pravovremeno donositi odluke i proširiti strateške ciljeve.

Za patente u novonastalim područjima, IPC sustav može biti zastario u odnosu na definirane kategorije. Prilikom klasificiranja takvih patenata nužno je koristiti veći broj klasa i proširiti kategorije koristeći najnovije izdanje IPC-a. Patenti često obuhvaćaju više kategorija unutar IPC-a, zato je važno pokriti sva relevantna područja prilikom klasifikacije. Kako bi se precizno odredilo mjesto unutar definirane hijerarhije dokumenata, patenti se redovito pregledavaju i nadopunjuju. Različiti patentni uredi mogu različito primijeniti IPC sustav, zato postoje male varijacije u tumačenju ili dodatni kodovi koji se koriste na nacionalnoj razini. Patenti sadrže povjerljive i osjetljive podatke pa se moraju etički koristiti.

Klasifikacija tehničke dokumentacije nerijetko iziskuje tim stručnjaka dobro upoznatih sa specifičnim podjelama. Pogrešno tumačenje dovodi do netočne klasifikacije. Netočna klasifikacija rezultira pogrešnim razumijevanjem ideje, usporenom i lošom interpretacijom, a i smanjenjem potencijalnih budućih nadogradnji nekog sustava. Timovi stručnjaka usko su specijalizirani i njihov je zadatak održavati konzistentnost klasifikacije. Proces klasifikacije zamoran je i skup pa se sve više napora ulaže u automatizaciju procesa. Automatizacijom bi se drastično smanjili potrebni napori klasifikacije i interpretacije dokumenata, a cijeli proces bi se pojednostavio i ubrzao.

1.1. Zadatak

Inženjeri diljem svijeta imaju problema sa svrstavanjem tehničkih dokumenata u odgovarajuće kategorije. Ponajviše se taj problem odražava na područje elektrotehnike i elektronike te je stoga ključno imati razrađen pristup razvrstavanju i preciznome rukovanju s takvim dokumentima. Zbog složenosti tehničke terminologije u tim dokumentima, pronalaženje odgovarajućih stručnjaka za njihovu obradu predstavlja značajan izazov. Tehnička dokumentacija uključuje tekstualne i grafičke elemente, a to je razlikuje od drugih vrsta dokumenata. Proces rukovanja tehničkom dokumentacijom dugotrajan je i skup, stoga je u ovome radu predstavljeno algoritamsko rješenje temeljeno na strojnome učenju za precizno hijerarhijsko označavanje i klasifikaciju tehničkih dokumenata. Predstavljeni algoritam na temelju tekstualnog opisa specifikacija, zahtjeva i realizacije, tj. sažetka s obradom shematskog prikaza (dijagrama/crteža/skica) pravilno raspoređuje odgovarajuće dokumente u odjeljak s odgovarajućom klasom. U ovom radu za raspodjelu dokumenata odabran je odjeljak elektricitet koji pokriva područja elektrotehnike i elektronike.

U drugom poglavlju opisano je područje rada, odnosno radovi koji su inspirirali predloženo rješenje te ideje na kojima se vlastito rješenje temelji.

Opis problematike izrade rješenja dan je u trećem poglavlju u vidu obrade i pregleda samog dokumenta i njegovog sadržaja. Slikovni i tekstualni sadržaj obrađen je koristeći tehnologije koje su izlistane. Proces predobrade također je obrađen jer je neophodan kako bi model mogao pravilno obrađivati unesene podatke i kako bi ti isti podatci bili reprezentativni za svaki dokument i njegove konkretne značajke.

U četvrtom poglavlju provedena je evaluacija rješenja razmatranjem i ocjenjivanjem rezultata. Različiti parametri vrednovani su kako bi objektivno procijenili proces klasifikacije.

2. PREGLED PODRUČJA RADA

Razvijanjem novih pristupa strojnome učenju te povećanjem preciznosti algoritama i složenosti sustava koji ih koriste, razvijaju se i novi pristupi problematici klasifikacije tehničke dokumentacije. Različiti znanstvenici bave se problematikom klasifikacije tehničke dokumentacije te pojednostavljenjem procesa klasifikacije. Istraživači su se najprije fokusirali na analizu isključivo tekstualnog dijela dokumenta. U radu [1] autor detaljno uspoređuje sljedeće algoritme za klasifikaciju tehničkih dokumenata: stroj s potpornim vektorima (engl. Support Vector Machines, SVM), linearni XGBoost, XGBoost stablo, stablo odlučivanja i slučajna šuma. Za ulaz su korištena tri različita dijela tekstualnog sadržaja dokumenta: opis, sažetak i zahtjevi. Pokazano je da SVM daje najbolje rezultate koristeći samo opis. S druge strane, XGBoost tree pokazao se najboljim ako se analiziraju sažetak ili zahtjevi. Problem pojednostavljenoga pristupa premalena je baza podataka od samo 100 dokumenata (20 po klasi) što je nedovoljno reprezentativan uzorak. Također, odabrane klase birane su nasumično. Taj pristup nema mogućnost kombiniranja više tekstualnih značajki. U radu [2] autori su analizirali mnogo veću bazu podataka, s više od 5 milijuna dokumenata. Predložena je duboka neuronska mreža za klasifikaciju koja je zasnovana na određivanju semantike tekstualnog dijela dokumenta. Iako je uspješno poboljšana semantička reprezentacija teksta u patentu, rezultati su klasifikacije samo 49% za top predikcije. Takav ishod solidan je početak, no nedovoljno precizan da bi se uzimao u obzir. Nedavno je predložen složeni model klasifikacije dokumenata koji kombinira računalni vid i neuronske mreže za obradu prirodnog jezika [3]. Taj je model testiran na 16 različitih klasa te je ostvarena točnost od 94.2%. Učinkovitost tog modela uvelike je ovisna o fiksnom skupu podataka, odnosno fokus nije isključivo na tehničku dokumentaciju nego je općenit pa uključuje novine, pisma, e-poštu itd. Razvojem dubokih neuronskih mreža za klasifikaciju proširuje se i količina dostupnih istraživanja na tu temu, stoga u radu [4] autori predstavljaju rješenje koje koristi nekoliko različitih mreža kako bi se odredili optimalni uvjeti za njihovu implementaciju. Za odjeljak *Elektricitet* ostvarena je preciznost svega 78% na bazi podataka s više od 5 000 primjeraka. Takav rezultat nije dovoljno precizan jer predloženo rješenje, kao i prethodno spomenuti modeli, analizira isključivo koristeći tekstualnu vrstu zapisa unutar dokumenta. Implementacija multimodalne arhitekture dubokog učenja TechDoc [5] koristi tri vrste informacija: tekstove na prirodnom jeziku, opisne slike unutar dokumenata te sličnosti među dokumentima. TechDoc obrađuje mnoštvo primjera tehničke dokumentacije i pokriva sve odjeljke unutar IPC klasifikacije s bazom od ukupno 800 000 dokumenata. Napravljena je usporedba Techdoc rješenja s različitim kombinacijama te se pokazalo kako u slučaju spajanja svih triju vrsta informacija se dobivaju najbolji rezultati. Konačni su rezultati problematični jer postoje prevelika odstupanja među klasama unutar odjeljka elektriciteta, s točnošću manjom od 40% i većom od 80%. Takav je ishod problematičan jer ukazuje na manjak konzistentnosti modela za različite primjerke unutar istog odjeljka.

3. IZRADA VLASTITOG RJEŠENJA

Stvaranje prikladnog modela za klasifikaciju tehničke dokumentacije uključuje rješavanje nekoliko ključnih problema. S obzirom na kompleksnost samih dokumenata, važno ih je pravilno obraditi i izvući najrelevantnije informacije na osnovu kojih se želi klasificirati. U ovom poglavlju opisani su dijelovi od kojih se predloženi sustav sastoji. Prikazane su korištene tehnologije za realizaciju svih koraka izrade vlastitog rješenja te je ukratko opisana sama klasifikacija unutar dokumenata. U poglavlju je pokrivena problematika realizacije vlastitog rješenja koja uključuje procese kreiranja same baze podataka koja sadržava dokumente nad kojima se model uči, predobradu tekstualnog sadržaja koji se izvlači iz dokumenta i predobradu slikovnih značajki dokumenata. Potrebno je obraditi i tekstualni i slikovni dio dokumenta, spojiti te dijelove i koristiti za kasnije treniranja predloženoga modela strojnog učenja. Detaljno su opisani procesi kreiranja graf strukture i duboke neuronske mreže, te procesi treniranja, validacije i testiranja iste. Zadnje potpoglavlje pokazuje način na koji se pokreće realizirano rješenje.

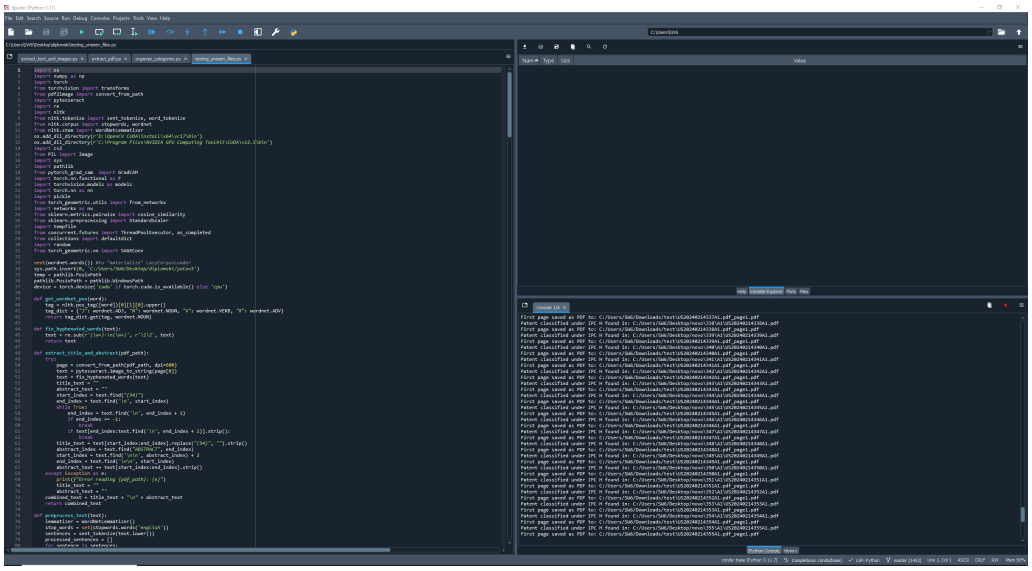
3.1. Korištena tehnologija

Za izradu i realizaciju svih modela, algoritama i koncepata korišten je programski jezik *Python*. *Python* je poznat po svojoj jednostavnosti i čitljivosti što ga čini dostupnim i početnicima i iskusnim programerima. Njegova jasna i koncizna sintaksa omogućuje brz razvoj i jednostavno održavanje koda. U sklopu programskog jezika *Python* dostupan je golem sustav biblioteka i paketa koji pokrivaju različite aspekte razvoja softvera, a posebice su korišteni aspekti poput strojnog učenja, obrade podataka, obrade prirodnog jezika i računalni vid. Ta opsežna podrška olakšava implementaciju složenih funkcionalnosti. *Python* ima veliku i aktivnu zajednicu što se očituje u detaljnoj dokumentaciji, brojnim tečajevima i mnoštvom alata i paketa za svaku razinu znanja. Podrška zajednice osigurava lako dostupnu pomoć, a sam proces razvoja neprestano se poboljšava. *Python* se može lako povezati s drugim jezicima i alatima, omogućujući bezbolnu integraciju s postojećim sustavima. Ta je mogućnost ključna u multidisciplinarnim projektima gdje se različite komponente mogu razvijati na različitim jezicima. *Python* je višeplatformski jezik, što znači da kôd napisan u *Pythonu* može raditi na različitim operativnim sustavima kao što su *Windows*, *macOS* i *Linux* uz minimalne izmjene. U radu je odabrana *Python* 3.11 verzija za razvoj.

Za razvoj *Python* programskog koda korišteno je znanstveno *Python* razvojno okruženje (engl. Scientific Python Development Environment, Spyder). *Spyder* je integrirano razvojno okruženje prilagođeno znanstvenim i inženjerskim aplikacijama. *Spyder* je posebno dizajniran za *Python* i pruža mnoštvo alata za analizu podataka, vizualizaciju i široki raspon drugih primjena pružajući interaktivnu konzolu za izvršavanje koda u stvarnom vremenu. Korištenjem *Spydera* korisnicima se omogućuje pregled i izmjena varijabli u trenutnom prostoru naziva. Ovo razvojno okruženje (slika 3.1) nudi napredne mogućnosti uređivanja s isticanjem sintakse, dovršavanjem koda i provjerom pogrešaka što uvelike olakšava proces razvoja rješenja.

Pip je standardni upravitelj paketa za *Python* programski jezik. Korisnicima je korištenjem *pipa* omogućeno jednostavno instaliranje i upravljanje dodatnim bibliotekama koje nisu uključene u standardnu *Python* biblioteku. *Pip* pruža jednostavno sučelje za instaliranje i deinstaliranje paketa iz indeksa *Python* paketa (engl. Python Package Index, PyPI). Upravo je *pip* korišten za instalaciju paketa neophodnih za rad.

Alternativa lokalnome radu pomoću različitih razvojnih okruženja je *Google Colab*. *Google Colab* besplatno je *Jupyter* okruženje temeljeno na oblaku koje pruža Google i dizajniran je za različite primjene. *Google Colab* radi u oblaku, eliminirajući potrebu za lokalnim postavljanjem okruženja i pružajući pristup puno značajnijim računalnim resursima, uključujući grafičke kar-



Slika 3.1: *Spyder razvojno okruženje*

tice najviših performansi. Prednost je rada u *Google Colabu* podržan rad u stvarnom vremenu dopuštajući većem broju korisnika da rade istovremeno na istom računalu. Početno stanje *Google Colab* bilježnice dolazi s mnogo unaprijed instaliranih biblioteka za različite primjene, a uvijek se mogu dodatno instalirati paketi koji su potrebni. Korisnici mogu spremati i učitavati datoteke izravno s Google diska olakšavajući upravljanje i dijeljenje podataka. Za generiranje nužnih težina za klasifikaciju slikovnih prikaza unutar dokumenta korišten je *Google Colab* kako bi se u punome smislu iskoristili dostupni besplatni resursi.

Za realizaciju projektnog rješenja korištene su sljedeće biblioteke [6]:

- *os* - Interakcija s operativnim sustavom, uključujući manipulaciju datotekama i direktorijima.
- *sys* - Omogućuje interakciju s *Python* interpreterom, pružajući pristup parametrima specifičnim za sustav.
- *pathlib* - Pruža objektno orijentirani pristup rukovanju i manipuliranju datotečnog sustava.
- *re* - Podrška za regularne izraze, omogućava podudaranje uzoraka i manipulaciju nizovima.
- *tempfile* - Generira privremene datoteke i direktorije korisne za privremeno pohranjivanje podataka tijekom obrade.
- *random* - Implementira generatore pseudoslučajnih brojeva za različite distribucije.
- *pickle* - Omogućuje spremanje i učitavanje *Python* objekata u i iz datoteka.
- *NumPy* - Biblioteka za numeričko računanje u *Pythonu*, pruža podršku za nizove, matrice i mnoge matematičke funkcije.
- *torch* - Glavna biblioteka za *PyTorch*, modul za strojno učenje koji pruža alate za duboko učenje i tenzorsko računanje.
- *torchvisiond* - Transformacija slike za računalni vid.

- *torch.nn* - Podmodul *PyTorcha* koji pruža slojeve neuronske mreže i pomoćne programe.
- *torch.nn.functional* - Funkcije za operacije neuronske mreže, uključujući funkcije aktivacije i funkcije gubitka.
- *torchtext.vocab* - Alati za izradu i rukovanje vokabularima iz tekstualnih podataka.
- *torch_geometric.utils* - Pomoćni programi za strojno učenje temeljeno na grafovima.
- *torch_geometric.nn* - Slojevi neuronske mreže i modeli za zadatke temeljene na grafovima.
- *OpenCV* - Alati za obradu slike i videa.
- *PIL* - Otvaranje, rukovanje i spremanje raznih formata slikovnih datoteka.
- *pdf2image* - Pretvara *.pdf* datoteke u popis slika.
- *PyPDF2* - Čitanje i rukovanje *.pdf* datotekama, uključujući dijeljenje, spajanje i izdvajanje teksta.
- *pytesseract* - Optičko prepoznavanje znakova za izdvajanje teksta iz slika s pomoću *tesseract* algoritma u *Pythonu*.
- *nltk* - Sveobuhvatna biblioteka za zadatke obrade prirodnog jezika.
- *nltk.tokenize* - Funkcije tokenizacije.
- *nltk.corpus* - Korpus i leksički izvori, uključujući *stopwords* i *wordnet*.
- *nltk.stem* - Stvaranje korijena i lematizacija riječi u njihove osnovne oblike.
- *NetworkX* - Stvaranje, manipuliranje i analiza složenih mreža ili grafikona.
- *sklearn.metrics* - Funkcije za procjenu izvedbe modela strojnog učenja.
- *sklearn.preprocessing* - Pretprocesiranje podataka, uključujući skaliranje i normalizaciju.
- *sklearn.model_selection* - Funkcije za dijeljenje podataka za treniranje i testiranje.
- *matplotlib.pyplot* - Crtanje statičnih/animiranih/interaktivnih vizualizacija u *Pythonu*.
- *seaborn* - Vizualizacija statističkih podataka temeljena na *matplotlibu*.
- *concurrent.futures* - Sučelje za asinkrono izvršavanje funkcija pomoću niti ili procesa.
- *collections* - Specijalizirani tipovi podataka.
- *shutil* - Operacije s datotekama kao što su kopiranje i uklanjanje datoteka.

Uz navedene biblioteke iskorištava se prednost izračuna jedinstvene arhitekture uređaja (engl. Compute Unified Device Architecture, CUDA). *CUDA* je platforma koju je razvila tvrtka Nvidia još 2007. godine. Platforma koristi resurse grafičke kartice kako bi ubrzala izvođenje i rad. CUDA je bazirana na C programskome jeziku i paralelizira proces na razini niti obavljajući konkretne operacije. Korištenjem CUDA-e omogućava se izmjenjivanje i raspoređivanje procesa i funkcija unutar programa na procesor, odnosno grafičku karticu, ovisno o potrebama. Kako bi se uspješno izvodila CUDA, nužno je instalirati CUDA duboke neuronske mreže (engl. CUDA Deep Neural Network, cuDNN) CUDA biblioteku za rad s neuronskim mrežama. Naravno, kako bi se mogao iskoristiti potencijal CUDA-e, nužno je imati Nvidia grafičku karticu.

Mnoge biblioteke i ugrađene funkcije podržavaju rad CUDA-e, naročito *OpenCV*. Ubrzavanje obrade podataka putem *OpenCV* biblioteke nosi veliku prednost. Kako bi se postigao željeni efekt, mora se preuzeti *OpenCV* direktno s Github javnog repozitorija te ugraditi direktno iz preuzetih direktorija koristeći *CMake*. Odabiru se parametri *ENABLE_FAST_MATH*, *OPENCV_DNN_CUDA*, *OPENCV_PYTHON3_VERSION* i *WITH_CUDA* prilikom ugradnje. Izgrađuje se konačni projekt koji generira *CMake* pa se taj projekt instalira. Kako bi biblioteke se pravilno učitale u *Python* okruženje, potrebno je dodati poveznice do direktorija u kojem se nalaze *.dll* datoteke CUDA-e i *OpenCV-a*. Nakon uspješnog instaliranja potrebnih paketa, nastavlja se dalje s realizacijom konačne ideje. Potrebno je napomenuti kako je u razvoju korištena *Nvidia Geforce RTX 3050 Dual OC* grafička kartica.

3.2. Klasifikacija dokumenata

IPC sustav klasifikacije uspostavljen je Strazburškim sporazumom iz 1971. godine. Taj sustav klasifikacije predstavlja hijerarhijski sustav jezično neovisnih simbola za klasifikaciju patenata, kao i korisnih modela prema različitim tehničkim područjima kojima pripadaju. Klasifikacijski sustav sadrži oko 70 000 unikatnih unosa, tj. klasifikacijskih simbola/kodova koji se pridjeljuju tehničkoj dokumentaciji [7]. Tim sustavom upravlja Svjetska organizacija za intelektualno vlasništvo (engl. World Intellectual Property Organization, WIPO). IPC je standardizirani sustav koji olakšava pronalaženje patentnih dokumenata dijeleći dostupnu tehnologiju u odjeljke, klase, potklase, grupe i podskupine. Njegova je struktura sljedeća [8]:

1. Odjeljci - Najviša razina hijerarhije, predstavljena jednim slovom (A-H). Postoji 8 odjeljaka od kojih svaki pokriva široko područje tehnologije:
    - A: Ljudske potrebe
    - B: Proizvodni postupci, transport
    - C: Kemija, metalurgija
    - D: Tekstil, papir
    - E: Građevinarstvo, rudarstvo
    - F: Strojarstvo, rasvjeta, grijanje, naoružanje, miniranje
    - G: Fizika
    - H: Elektricitet
2. Klase - Svaki odjeljak podijeljen je u klase predstavljene dvoznamenkastim brojem iza slova odjeljka (npr. A01, B32). Ova podjela raščlanjuje tehnološka područja unutar odjeljka.
3. Potklasa - Svaka klasa podijeljena je u potklase predstavljene velikim slovom iza broja klase (npr. A01B, B32B). Potklase pružaju specifičniju kategorizaciju unutar klase.
4. Grupe - Svaka potklasa podijeljena je u grupe koje su glavne grupe ili podskupine:
    - Glavne grupe - predstavljene jednoznamenkastim do troznamenkastim brojem iza kojeg slijedi kosa crta i nula (npr. A01B 1/00). Definiraju teme unutar potklase.
    - Podgrupe - predstavljene jednoznamenkastim do troznamenkastim brojem iza kojeg slijedi kosa crta i neki broj (npr. A01B 1/02). Pružaju specifične detalje za kategorizaciju unutar glavnih grupa.

Zadatak je ovoga rada klasifikacija odjeljka H, tj. elektriciteta s odgovarajućom klasom toga odjeljka. Odjeljak H sustava IPC pokriva široko područje električne energije. Taj odjeljak je podijeljen u različite klase, od kojih svaka predstavlja određenu domenu unutar polja električne energije. Detaljno objašnjenje klasa unutar odjeljka H:

1. H01 - Osnovne električne komponente
    - H01B - Kabeli, vodiči, izolatori
    - H01C - Otpornici
    - H01D - Magneti, induktori, transformatori, odabrani materijali na osnovu njihovih magnetnih svojstva
    - H01G - Kondenzatori, odabrani materijali na osnovu njihovih dielektričnih svojstava
    - H01H - Električni prekidači, releji, selektori, zaštitni uređaji za hitne slučajeve
    - H01J - Električne ispušne cijevi ili lampe
    - H01K - Električne žarulje sa žarnom niti
    - H01L - Poluvodički uređaji, električni uređaji u čvrstom stanju koji nisu predviđeni na drugom mjestu
    - H01M - Procesi ili sredstva
    - H01P - Valovodi, rezonatori, vodovi ili drugi uređaji tipa valovoda
    - H01Q - Antene
    - H01R - Električno vodljivi spojevi, spojni uređaji, kolektori struje, strukturalne asocijacije više međusobno izoliranih električnih veznih elemenata
    - H01S - Uređaji koji koriste proces pojačavanja svjetlosti stimuliranom emisijom zračenja, uređaji koji koriste stimuliranu emisiju elektromagnetskog zračenja u valnim područjima osim optičkih
    - H01T - Iskrišta, odvodnici prenapona korištenjem iskri, svjećice, corona uređaji, generirajući ioni koji se uvode u nezatvorene plinove
2. H02 - Proizvodnja, pretvorba ili distribucija električne energije
    - H02B - Ploče, trafostanice ili sklopni uređaji
    - H02G - Instalacija električnih kabela, vodova, kućišta ili vodiča
    - H02H - Uređenje zaštitnih krugova u slučaju nužde
    - H02J - Uređenje strujnih krugova, sustavi za opskrbu ili distribuciju električne energije
    - H02K - Dinamo - električni strojevi
    - H02M - Uređaji za pretvorbu između istosmjerne i izmjenične struje, između istosmjerne i istosmjerne struje ili između izmjenične i izmjenične struje
    - H02N - Električni strojevi koji nisu drugdje predviđeni
    - H02P - Upravljanje ili regulacija elektromotora, električnih generatora ili dinamoelektričnih pretvarača, upravljanje transformatorima, reaktorima ili prigušnim zavojnicama
    - H02S - Proizvodnja električne energije konverzijom infracrvenog zračenja, vidljive svjetlosti ili ultraljubičaste svjetlosti

3. H03 - Osnovni elektronički sklopovi
    - H03B - Generiranje oscilacija, izravno ili pretvorbom frekvencija elektroničkih oscilacija
    - H03C - Modulacija
    - H03D - Demodulacija ili prijenos modulacije s jednog nositelja na drugi
    - H03F - Pojačala
    - H03G - Kontrola pojačanja
    - H03H - Mreže impedancije, npr. rezonantni krugovi ili distribucijske mreže
    - H03J - Podešavanje i odabir rezonantnih kruga
    - H03K - Pulsna tehnika
    - H03L - Automatsko upravljanje, pokretanje, sinkronizacija ili stabilizacija generatora elektroničkih oscilacija ili impulsa
    - H03M - Kodiranje, dekodiranje ili konverzija koda
4. H04 - Tehnika električne komunikacije
    - H04B - Prijenos
    - H04H - Emitirana komunikacija
    - H04J - Multipleksna komunikacija
    - H04K - Tajna komunikacija, ometanje komunikacije
    - H04L - Prijenos digitalnih informacija
    - H04M - Telefonska komunikacija
    - H04N - Slikovna komunikacija, npr. televizija
    - H04Q - Odabir
    - H04R - Zvučnici, mikrofoni, gramofonski snimači ili slični akustični elektromehanički pretvornici, setovi za gluhe osobe, razglasni sustavi
    - H04S - Stereofonski sustavi
    - H04W - Bežične komunikacijske mreže
5. H05 - Električne tehnike koje nisu predviđene na drugom mjestu
    - H05B - Električno grijanje, električna rasvjeta koja nije predviđena na drugom mjestu
    - H05C - Električni krugovi ili uređaji posebno dizajnirani za upotrebu u visokonaponskim sustavima
    - H05F - Statički elektricitet, prirodni elektricitet
    - H05G - X-ray tehnologija, tehnologija gama zraka
    - H05H - Plazme, plazma tehnike
    - H05K - Tiskani krugovi, kućišta ili konstrukcijski detalji električnih aparata, proizvodnja sklopova električnih komponenti
6. H10 - Poluvodički uređaji i električni uređaji u čvrstom stanju koji nisu predviđeni na drugom mjestu

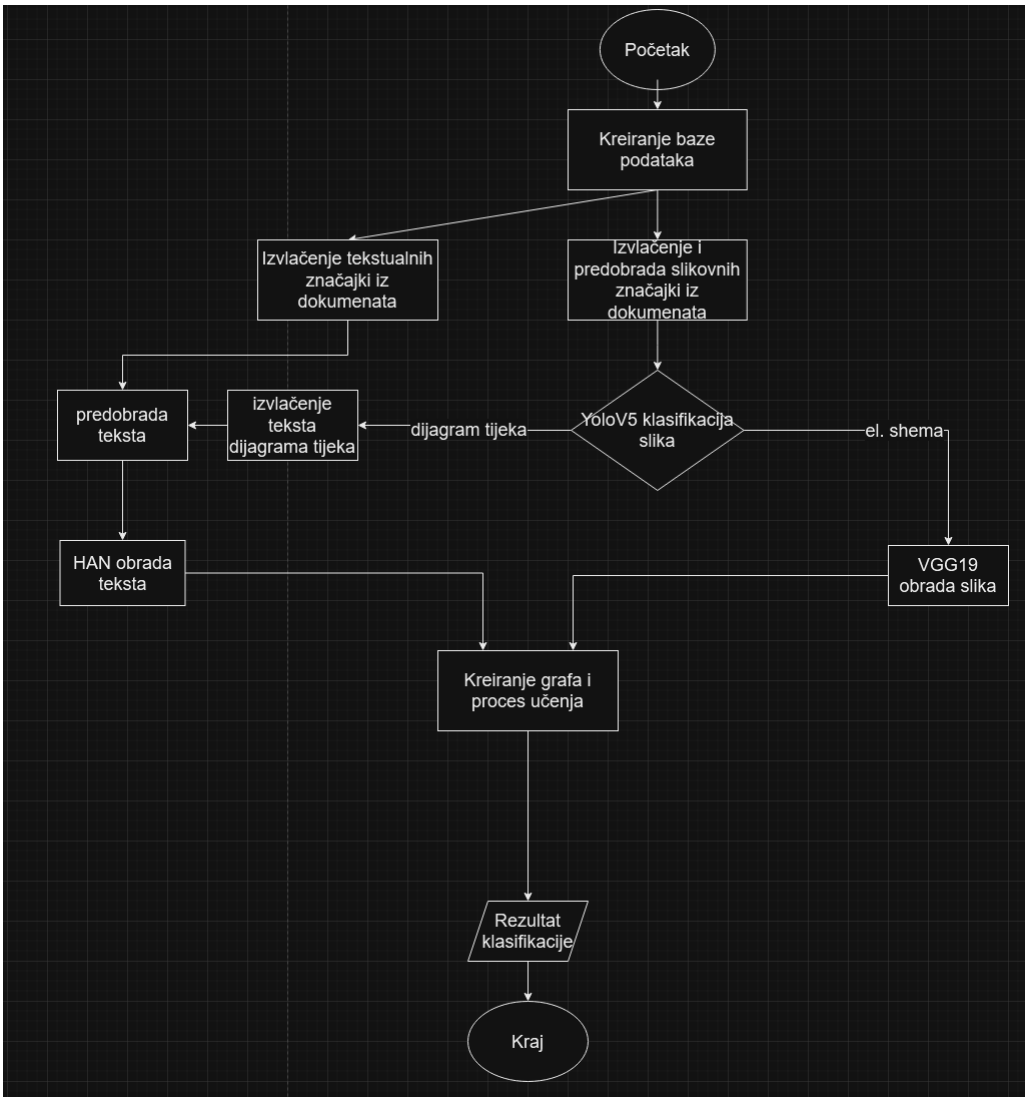
- H10B - Elektronički memorijski uređaji
- H10D - Anorganski električni poluvodički uređaji
- H10F - Anorganski poluvodički uređaji osjetljivi na infracrveno zračenje, svjetlost, elektromagnetsko zračenje kraće valne duljine ili korpuskularno zračenje
- H10H - Anorganski poluvodički uređaji koji emitiraju svjetlost i imaju potencijalne barijere
- H10K - Organsko električno čvrsto stanje
- H10N - Električni uređaji u čvrstom stanju koji nisu predviđeni na drugom mjestu

H01 klasa uključuje osnovne komponente koje se koriste u električnim sustavima, kao što su otpornici, kondenzatori, tranzistori, sklopke itd. H02 obuhvaća tehnologije koje uključuju strojeve poput generatora i transformatora, kao i sustave za distribuciju električne energije i zaštitu iste. H03 obuhvaća temeljne gradivne dijelove elektroničkih sklopova poput oscilatora, modulatora, pojačala, mreže impedancije. Te su komponente neophodne za rad raznih elektroničkih uređaja i sustava. H04 pokriva sve aspekte komunikacije, od osnovnog prijenosa i prijema signala do složenih sustava za digitalnu i složenu komunikaciju. U H04 klasu uključene su tehnologije za telefoniju, emitiranje raznih signala i televizija. H05 sadrži specijalizirana područja električne tehnologije koja nisu pokrivena u drugim razredima. To uključuje električno grijanje i rasvjetu, visokonaponske sustave, statički elektricitet i sl. Klasa H10 pokriva poluvodičke i električne uređaje u čvrstom stanju koji ne spadaju u druge specifične IPC kategorije, a pri tome uključuje napredne tehnologije poluvodiča, nove dizajne tranzistora i elektroničke komponente čvrstog stanja u nastajanju.

Ključno je poznavati materiju koja se obrađuje te iz koje se skupljaju relevantni primjerci za svaku od šest mogućih klasa.

3.3. Opis rješenja

Slika 3.2 prikazuje dijagram tijeka u kojem su ukratko opisani koraci realizacije projektnog rješenja. Prvi je korak kreirati bazu podataka koja sadrži dovoljno primjeraka tehničke dokumentacije odjeljka elektriciteta. Dokumenti moraju biti reprezentativni za svaku od šest klasa unutar odjeljka elektriciteta. Za svaki od dokumenata u formiranoj bazi izvučena je kombinacija značajki. Značajke su slikovne ili tekstualne, te ih se shodno tome mora na pravilan način obraditi. Ako su značajke tekstualne, predobrađuju se te se dobiju pohranjene riječi u korijenskom obliku. U slučaju da su značajke slikovne, moraju se prvo klasificirati. Sadržaj slika unutar dokumenta može biti elektronička shema ili dijagram tijeka, stoga je implementirana klasifikacija slika. Za slike dijagrama tijeka izvučen je pripadajući tekstualni sadržaj, koji se predobrađuje na isti način kao i ranije spomenuti tekstualni sadržaj dokumenta. Kombinirani tekst nužno je i obraditi tako da odgovara unosu koji očekuje konačan model. Sadržaj elektroničkih shema najprije se predobrađuje, a zatim dodatno obrađuje u tom obliku kako bi se spojio s obrađenim tekstualnim sadržajem. Takav spojeni prikaz se predaje graf strukturi, kako bi se generirao prikaz potreban za konačni model klasifikacije odjeljaka. Konačni se model trenira, te se provodi validacija i testiranje nad podatcima.

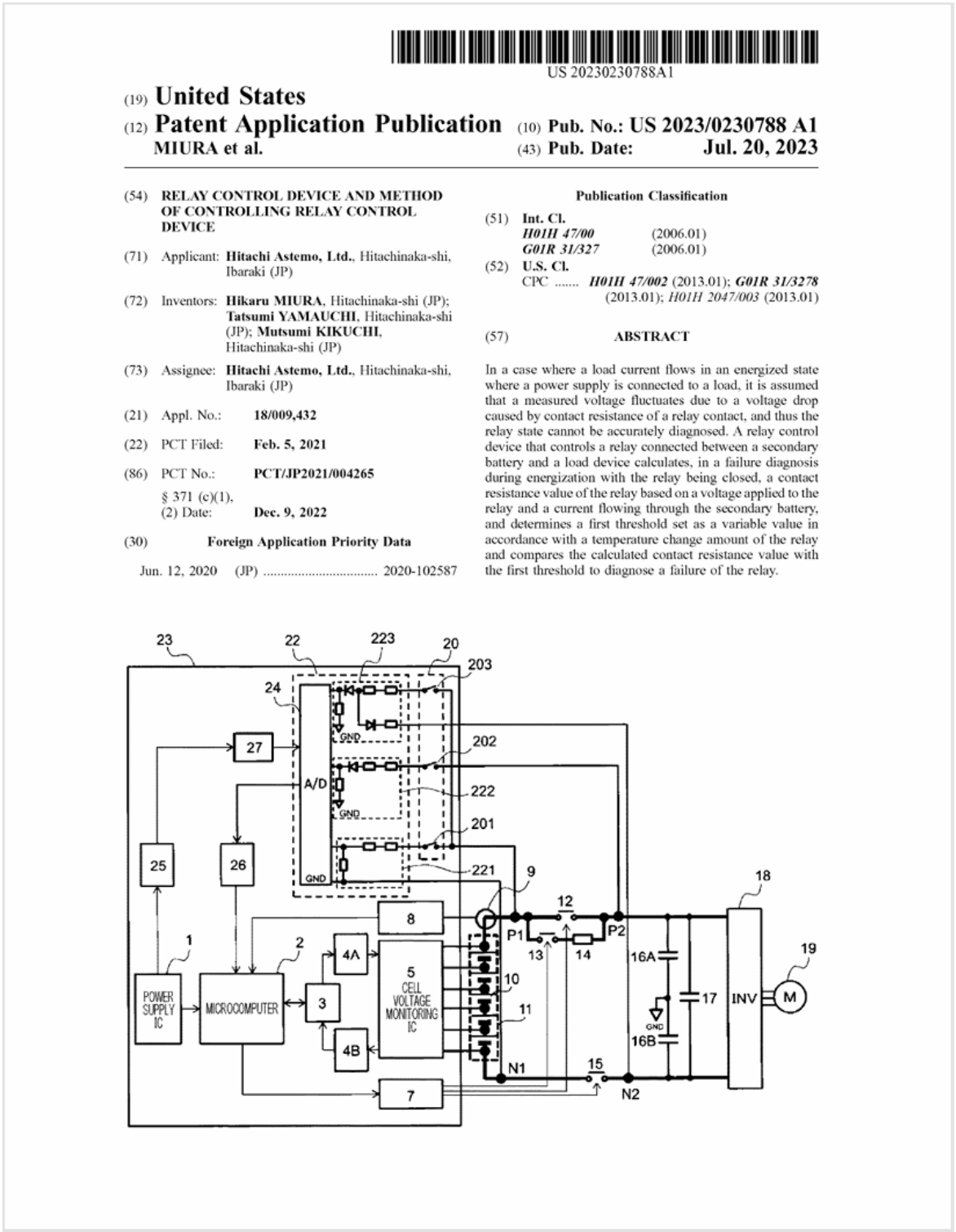


Slika 3.2: *Dijagram tijeka vlastitog rješenja*

3.4. Kreiranje baze podataka

Da bi se neki model mogao trenirati na podatcima, nužno ih je prvo prikupiti. S obzirom na kompleksnost pretraživanja tehničke dokumentacije putem Interneta, nužno je odabrati dobar izvor. Postoje različiti javno dostupni izvori, kao što su *Espacenet* [9], *Kaggle* [10] i *Patentscope* [11]. Problem s *Kaggleom* manjak je dostupnih baza koje sadrže potrebnu tehničku dokumentaciju odjeljka H, dok su *Patentscope* i *Espacenet* odlični izvori, no proces sakupljanja velike količine podataka potrebnih za treniranje modela bio bi vremenski složen. Stoga su odgovarajući dokumenti klasificirani putem IPC standarda, preuzeti s portala otvorenih podataka ranije spomenutog USPTO-a [12]. Ono što portal otvorenih podataka USPTO-a čini idealnim jest jednostavnost preuzimanja velikog broja različitih dokumenata i njihova razvrstanost po odjeljcima od A do H. Svaki je odjeljak pravilno raspoređen po klasama, od najniže do najviše vrijednosti, što omogućava bolji uvid i preglednost dostupnih dokumenata. Svaki dokument unutar *.tar* datoteke pravilno je klasificiran i u (uglavnom) visokoj kvaliteti. Arhive dokumenata (*.tar* datoteke) na popisu su izlistane po datumu, od najnovijeg unosa prema najstarijem, s time da se baza redovno popunjava svakih sedam dana. Postoje zapisi sve do ožujka 2001. godine, što je pregršt dostupnog materijala. U prosjeku se po *.tar* datoteci nalazi oko 6 000 unikatnih dokumenata.

Najprije se unutar strukture preuzete *.tar* datoteke filtriraju H odjeljci, pronalazeći datoteke koje pripadaju tom odjeljku. Datoteke su jasno definirane, što se vidi po primjeru (slika 3.3). Svaki dokument sadrži informacije kao što su naslov, datum predaje, različite oz-



US 20230230788A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2023/0230788 A1**

MIURA et al. (43) **Pub. Date: Jul. 20, 2023**

(54) **RELAY CONTROL DEVICE AND METHOD OF CONTROLLING RELAY CONTROL DEVICE**

(71) Applicant: **Hitachi Astemo, Ltd.**, Hitachinaka-shi, Ibaraki (JP)

(72) Inventors: **Hikaru MIURA**, Hitachinaka-shi (JP); **Tatsumi YAMAUCHI**, Hitachinaka-shi (JP); **Mutsumi KIKUCHI**, Hitachinaka-shi (JP)

(73) Assignee: **Hitachi Astemo, Ltd.**, Hitachinaka-shi, Ibaraki (JP)

(21) Appl. No.: **18/009,432**

(22) PCT Filed: **Feb. 5, 2021**

(86) PCT No.: **PCT/JP2021/004265**

§ 371 (c)(1),
(2) Date: **Dec. 9, 2022**

(30) **Foreign Application Priority Data**

Jun. 12, 2020 (JP) 2020-102587

Publication Classification

(51) **Int. Cl.**
H01H 47/00 (2006.01)
G01R 31/327 (2006.01)

(52) **U.S. Cl.**
CPC ***H01H 47/002*** (2013.01); ***G01R 31/3278*** (2013.01); *H01H 2047/003* (2013.01)

(57) **ABSTRACT**

In a case where a load current flows in an energized state where a power supply is connected to a load, it is assumed that a measured voltage fluctuates due to a voltage drop caused by contact resistance of a relay contact, and thus the relay state cannot be accurately diagnosed. A relay control device that controls a relay connected between a secondary battery and a load device calculates, in a failure diagnosis during energization with the relay being closed, a contact resistance value of the relay based on a voltage applied to the relay and a current flowing through the secondary battery, and determines a first threshold set as a variable value in accordance with a temperature change amount of the relay and compares the calculated contact resistance value with the first threshold to diagnose a failure of the relay.

Slika 3.3: *Primjerak tehničkog dokumenta odjeljka*

nake, specifikacije i upute za korištenje. Dokumenti nerijetko pokrivaju i više IPC klasifikacija, što je očekivano s obzirom na problem kojim se bave. Najvažnija je kategorija uvijek navedena prva, a nju se ujedno i predviđa. Najbitniji su elementi naslov, koji se svaki put nalazi uz oznaku (54), i sadržaj (engl. Abstract), koji se nalazi uz oznaku (57). To su tekstualne

informacije koje se predaju modelu za učenje. Također, svaki dokument ima ranije spomenutu shemu. Ona se uglavnom nalazi na prvoj stranici, neposredno ispod sadržaja. Navedeni dokumenti sadrže puno različitih skica, no ona na prvoj stranici uvijek je najviše relevantna te se zbog toga koristi za treniranje. Datoteke su poredane od H01 pa do H10, stoga je automatiziran proces izvlačenja prve stranice *.pdf* dokumenta tako da se otvaraju sve datoteke unutar zadanoga direktorija koje su tipa *.pdf* te se izreže samo početna stranica u slučaju da je dokument H odjeljka. Kako bi se znao pročitati tekst iz dokumenta, korištene su ranije opisane biblioteke *pdf2image*, *PyPDF2* i *pytesseract*. Najprije se isključivo prva stranica pretvori u sliku iz koje se izvlače relevantne informacije. Za pohranu teksta iz slike korišten je *tesseract* algoritam za optičko prepoznavanje znakova (engl. Optical character recognition, OCR), konkretno *pytesseract* kao implementacija u *Pythonu*. U usporedbi s nekim drugim OCR rješenjima, *tesseract* nema visoku brzinu izvođenja. No, ono što ga izdvaja u odnosu na druga rješenja jest njegova preciznost. Prilikom izvlačenja teksta, pri dovoljno visokoj rezoluciji, *tesseract* daje na uvid raspored teksta najsličniji onome u pravome dokumentu. Druge implementacije OCR algoritama izvlače kontekstualne informacije u redovima, što predstavlja problem jer se, primjerice naslov dokumenta nalazi u istome redu kao i oznaka "Publication Classification". Rezultat je toga da većina isprobanih OCR implementacija kao *EasyOCR*, *OCRmyPDF* i *PaddleOCR* nepravilno raspoređuje izvučen tekst i uvelike otežava izvlačenje najbitnijih informacija u dokumentu. *OCRmyPDF* također je baziran na *tesseractu* pa bi se naišlo na slična ograničenja kao i prilikom korištenja samog *tesseracta*. Naravno, postoje i neka kompatibilnija rješenja poput *Google Cloud Vision* i *Amazon Textract*, no ona zahtijevaju uvijek stabilnu Internet vezu, te se i naplaćuju. Stoga je, zbog preciznosti na uštrb brzine, korišten *tesseract*. Kako bi se provjerilo da poredak dokumenata u direktorijima odgovara rastućim vrijednostima od H01 do H10, uzima se podatak o odjeljku i klasi otvorenog dokumenta. Ako su svi uvjeti zadovoljeni, prva se stranica sprema u novi direktorij. Postupak se ponavlja sve dok se ne ostvari 1 000 primjeraka za svaku od klasa odjeljka H. Time se formira baza za rad, kao i priprema podataka za predobradu koja slijedi.

3.5. Izvlačenje tekstualnih značajki iz dokumenata

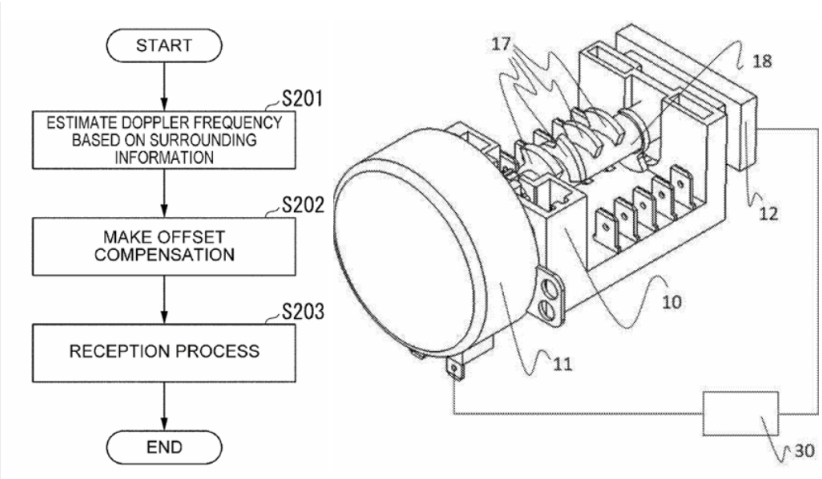
Prvi vektor koji predstavlja značajke korištene za proces treniranja modela klasifikacije jest vektor tekstualnih značajki. Taj se vektor dobije pronalaskom teksta uz oznake (54) i (57) u dokumentu. Po uzoru na filtriranje dokumenata za stvaranje baze podataka, ponovno se koristi *tesseract*. Najprije se dostupnu stranicu pretvori u sliku jer je to očekivan format od strane *tesseracta* te se izvlači tekst pozivanjem ugrađene funkcije *image_from_string()*. S obzirom na ograničenja dostupnog prostora prilikom pisanja sažetka, postoje slučajevi gdje se određene riječi ne mogu napisati u istome retku pa se dio slova koji stane napiše u trenutnome retku i znakom "-" dovrši ciljana riječ. Nužno je pravovremeno reagirati i spojiti znakove kako bi se formirala jasna riječ. Indeksiranjem se prati kompletan naslov patenta od oznake (54) pa do praznoga mjesta jer neki naslovi sadrže velik broj riječi pa se prostiru na više od jednoga reda. Sličan se postupak ponavlja za oznaku (57), tj. sažetak. Kompletan tekst do kraja se sprema, sve do pojavljivanja sheme. Dobiveni naslov i sažetak spajaju se u novu varijablu kako bi se dobio potreban tekst iz dokumenta.

3.6. Izvlačenje i predobrada slikovnih značajki

Nastavno na izvlačenje teksta nužno je izvući relevantne regije sheme. Prateći pravilni raspored svakog dokumenta, shematski se prikaz uvijek nalazi na dnu, točno gdje završava sažetak. Prva se stranica učitava u memoriju grafičke kartice kao slika. Dobivena se slika pretvara u sive

nijanse (engl. Grayscale) iz plavo-zeleno-crvenog prostora boja (engl. Blue-Green-Red, BGR). Taj je korak standard prilikom obrada slika. Slika u sivim tonovima zamućuje se Gaussovom metodom (engl. Gaussian blur) kako bi se smanjili prisutni šumovi i izgladila konačna slika radi lakšeg detektiranja rubova. Rezultantna se slika obrađuje *Cannyjevim* algoritmom otkrivanja rubova kako bi se jasno naglasile promatrane granice. Takvi se rubovi dodatno naglase dilatacijom, no bitno je napomenuti da *OpenCV* nema podršku za obavljanje dilatacije putem grafičke kartice, stoga se privremeno radi na procesoru. Slika se ponovno vrati u memoriju grafičke kartice te se traže konture. Mora se odrediti granični okvir (engl. Bounding box) koji bi uspješno obuhvatio sliku te se on ažurira sve dok nisu jasno određene granice koje pravilno pokrivaju shemu u njenoj pravoj veličini. Tako izrezana slika predstavlja zadovoljavajući format te se sprema za daljnju predobradu.

Modeli koji obrađuju slike su robusni pa zahtijevaju da slike budu određenoga formata. Stoga se dobivene slike moraju reformatirati pa se svakoj slici mijenja veličina na fiksnih 224×224 piksela. Taj je format standard u radu s neuronskim mrežama koje obrađuju slikovne formate jer je u prosjeku najčešći. Slika se pretvori u *PyTorch* objekt gdje se skalira raspon piksela od [0,255] do [0,1]. Slika se zatim mora normalizirati tako da se vrijednosti piksela promijene na način da imaju srednju vrijednost od 0,485, 0,456 i 0,406 i standardnu devijaciju od 0,229, 0,224 i 0,225 za crveni, zeleni i plavi kanal (engl. Red-Green-Blue, RGB). Te se vrijednosti temelje na skupu podataka *ImageNet* koji sadrži preko 14 milijuna ručno označenih slika na kojima se vježbaju mnogi modeli. Tako izrezana slika spremna je za konačnu predobradu, no nisu sve slike istog tipa. Razlikuju se slike koje su dijagrami tijeka i shematski prikazi. Sheme prikazuju nacrte, skice, planove kako funkcionira neki proizvod, dok dijagrami tijeka pravilno raspoređuju proces izgradnje/modeliranja nekog proizvoda, a cjeline se povezuju strelicama. S obzirom na ove razlike, drugačije se pristupa prilikom obrade nekog od tipova prikaza (slika 3.4). S lijeve se strane nalazi primjer dijagrama tijeka iz nasumičnog dokumenta, a s desne shematski prikaz stroja.

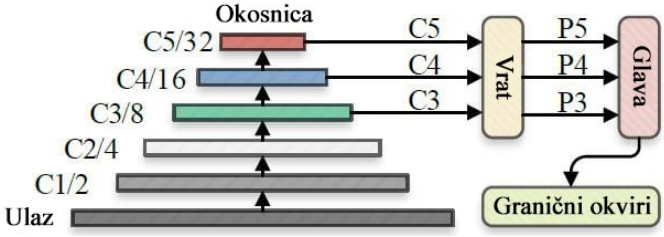


Slika 3.4: *Usporedba prikaza u različitim dokumentima*

3.6.1 YOLOv5 klasifikacija slike

Slike u tehničkim dokumentima mogu biti dijagrami tijeka ili elektroničke sheme. Svaku sliku unutar dokumenta potrebno je locirati i pravilno rasporediti u jednu od dvije moguće klase tako da se kasnije mogu pravilno obraditi slike koje se nalazi u pojedinom dokumentu. Za klasifikaciju slika unutar dokumenta korišten je model Pogledaj samo jednom (engl. You Only

Look Once, YOLO). YOLO je napredni model detekcije objekata, a prednost je YOLO modela brzina izvođenja kao i preciznost otkrivanja željenih predmeta [13]. Model dubokog učenja baziran na neuronskoj mreži YOLO predviđa granične okvire i vjerojatnosti klasa za objekte unutar svake ćelije mreže za postavljenu ulaznu sliku. Postoji nekoliko YOLO verzija koje je razvio *Ultralytics* tim jer se model konstantno nadograđuje i poboljšava. Verzija v5 s dovoljno dostupne dokumentacije odabrana je za klasifikaciju. Arhitektura YOLOv5 općenito se dijeli u tri glavne komponente: okosnica (engl. Backbone), vrat (engl. Neck) i glava (engl. Head) (slika 3.5). Okosnica YOLOv5 odgovorna je za izdvajanje značajki iz ulazne slike. Oznake C1/2, C2/4, C3/8, C4/16, C5/32 predstavljaju mape značajki različitih veličina od sitnih detalja do apstrakcija visoke razine gdje svaki sljedeći sloj izdvaja složenije i apstraktnije značajke iz slike. Značajke iz različitih slojeva okosnice su agregirane u vratu kako bi se poboljšala preciznost otkrivanja. Te mape značajki koristi glava za predviđanje graničnih okvira i klasa objekata. Značajke su spojene kroz vrat koristeći različite mape značajki C3, C4 i C5 kako bi se na kraju generirale konačne mape značajki P3, P4 i P5 za otkrivanje malih, srednjih i velikih objekata na slici. Tako agregirane značajke korištene su za predikciju slike.



Slika 3.5: *YOLOv5 arhitektura*

YOLO modeli za klasifikaciju mogu se razviti različitim pristupima. Dokumentacija i javni tečajevi za razvoj YOLO modela širom su dostupni, a posebice je to vidljivo na aplikaciji *Roboflow* [14]. *Roboflow* olakšava treniranje YOLOv5 modela putem web aplikacije tako da se klikom pokazivača mogu odabrati svi parametri nužni prilikom kompletnog procesa treniranja. Ovim se pristupom smanjuje potreba za pisanje programskoga kôda, a generirani model besplatno se šalje na korisnikovu adresu kada se završi proces online treninga. Na istoj web stranici se mogu pronaći i gotova rješenja jer se svaki model može javno podijeliti. Nije pronađen odgovarajući model, stoga ga je nužno ručno generirati. Proces generiranja modela zahtijeva bazu podataka nad kojom će model učiti. Uz same modele, na web stranici *Roboflow* su dostupne i javne baze podataka s raznim primjerima slika za treniranje modela. Pretražene su dostupne baze podataka kako bi se pronašla odgovarajuća baza podataka s obzirom na specifični problem. Odabrana baza podataka sadrži dovoljan broj dijagram tijekova koji predstavljaju prvu klasu, ali i dovoljan broj elektroničkih shema predstavljenih drugom klasom [15]. Baze podataka dolaze unaprijed raspoređene i ručno označene za trening, test i validacijski skup. 2 150 unikatnih unosa slika u formatu 640x640 piksela čini odabranu bazu podataka, gdje 1 131 primjeraka pripada klasi shematskog prikaza, a 1 019 dijagramu tijeka. Slike koje se nalaze u odabranoj bazi neophodno je predobraditi, a parametri predobrade moraju se pomno odabrati. Predobrada slika obavljena je na web stranici *Roboflow*. Parametri predobrade slika unutar baze su:

- auto-orient - Vodi se računa o načinu kako kamera pohranjuje piksele slika bez obzira je li kamera orijentirana u pejsažnom ili portretnom načinu. Odlučuje se treba li piksele prikazati onakve kakve jesu ili ih rotirati za 90 ili 180 stupnjeva prilikom prikaza slike.
- promjena veličine - Slike se rastežu na zahtijevani format 224x224.
- promjena tonova - Sivi tonovi.

- automatsko podešavanje kontrasta - Tehnika adaptivnog izjednačavanje histograma služi za poboljšanje kontrasta slika tako što dijeli sliku u različite blokove i izjednačava histogram za svaki odjeljak.

Izjednačavanje histograma izračunava nekoliko histograma, od kojih svaki odgovara određenom dijelu slike, pa ih koristi za preraspodjelu vrijednosti osvjetljenja slike. Za svaki se odjeljak obavlja izjednačavanje histograma tako što se najprije stvori lista sa svim mogućim vrijednostima intenziteta nekog piksela unutar slike sivih tonova, tj. vrijednostima od 0 do 255. Prolazeći kroz svaki piksel neke slike, u listi se inkrementira vrijednost intenziteta koja odgovara vrijednosti intenziteta piksela. Npr. ako prvi piksel ima vrijednost 100, to znači da se u listi inkrementira na indeksu 100. U trenutku kada je slika do kraja obrađena, metoda adaptivnog izjednačavanja histograma računa normalizirane kumulativne vrijednosti broja piksela za svaki spremljeni intenzitet. Metoda adaptivnog izjednačavanje histograma zahtijeva normalizirane vrijednosti kako bi se mogle mapirati početne vrijednosti intenziteta u nove.

Uz predobradu, obavljena je i augmentacija putem *Roboflow* web aplikacije, koristeći se sljedećim parametrima:

Okretanje	**Rezanje**	**Rotiranje**	**Mjenjanje nijansi**	**Šum**
vodoravno	0 - 20%	između $-15°$ i $+15°$	između $-15°$ i $+15°$	do 0.1% piksela

Tablica 3.1: *Prvi dio parametara augmentacije slika*

Mjenjanje zasićenosti	**Svjetlina**	**Izloženost**	**Zamućenje**
između -20% i $+20\%$	između -20% i $+20\%$	između -10% $+10\%$	do 3 piksela

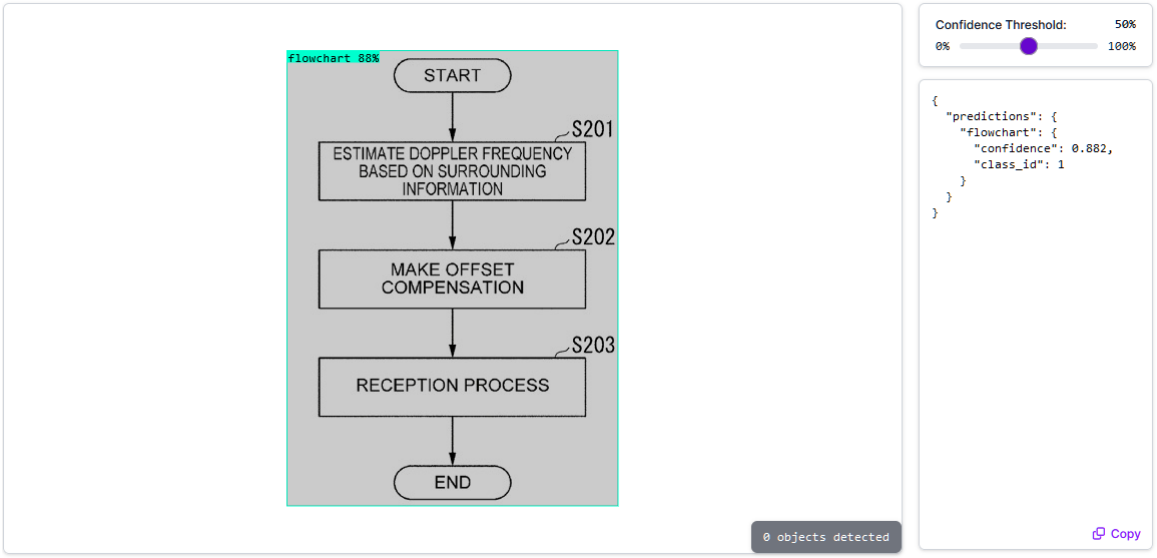
Tablica 3.2: *Drugi dio parametara augmentacije slika*

Za svaki unos iz trening skupa generirana su tri nova primjera koja će pokrivati navedene parametre augmentacije, ali je svaki primjer s drugačijim vrijednostima parametara augmentacije (tablica 3.1 i tablica 3.2). Takav pristup poboljšava preciznost prilikom klasifikacije slika lošije kvalitete, loše izrezanih slika, slika izvučenih iz loše skeniranog *.pdf* dokumenta i slično. Konačan skup daje ukupno 5711 slika. YOLOv5 model kreira se automatski te je dostupan za preuzimanje poveznicom dobivenom putem elektroničke pošte povezane s *Roboflow* profilom. Validacijska točnost iznosi 98.8%, a izračuna se kao omjer broja ispravnih oznaka u odnosu na ukupan broj oznaka. Dostupne su mnoge opcije za korištenje konačnoga modela YOLOv5 [16], no u kontekstu ovoga rada najvažnije su težine (engl. Weights) kako bi se model mogao učitati u lokalnome okruženju. Kako bi se iz novog modela dobile težine u *.pt* formatu, mora ih se ručno generirati.

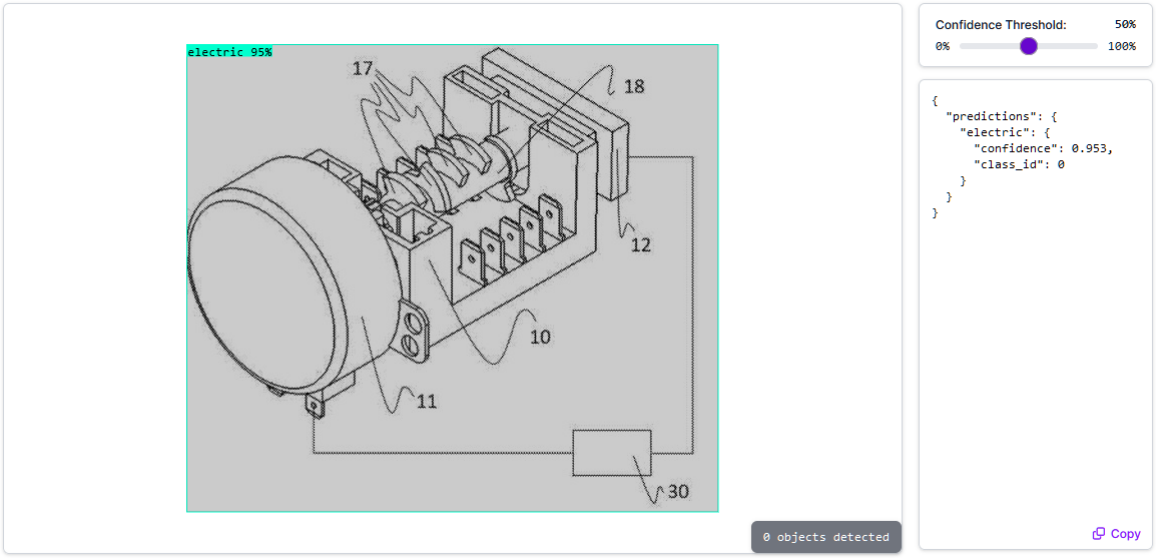
Na sreću, proces generiranja dostupan je javno u obliku kratkog vodiča [17]. *Google Colab* bilježnica se otvara i prate upute za generiranje težina. Nužni paketi najprije se instaliraju, zatim se trenira pristupajući bazi podataka putem API ključa, a rezultat, uključujući težine i kompletan programski kôd, sprema se lokalno. Datoteka težina nužna je za lokalno učitavanje modela.

Dobiveni prilagođeni model učitava se u lokalno okruženje gdje on svaku izrezanu sliku klasificira. Izlaz iz modela je vjerojatnost da je slika određene klase. Dobivena se vjerojatnost uspoređuje s pragom koji je definiran na 0.5. To znači da su sve vjerojatnosti veće od 0.5 uzete u obzir i slijedi pretpostavka da je slika neke klase. Na prethodna dva primjera klasa

može se prikazati kako se definira prag te kolika je vjerojatnost pripadnosti klasi dijagram tijeka (slika 3.6) ili elektroničke sheme (slika 3.7).

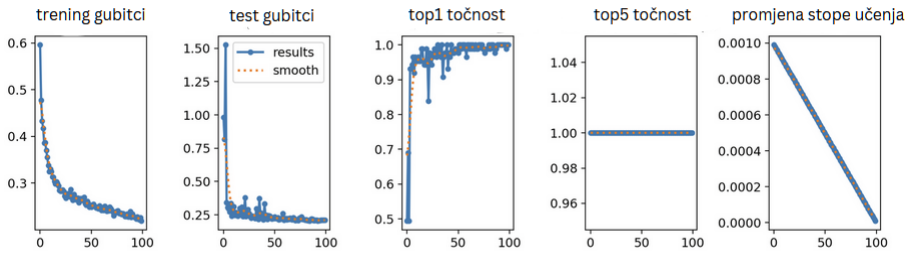


Slika 3.6: *Vjerojatnost pripadanja klasi dijagram tijeka*



Slika 3.7: *Vjerojatnost pripadanja klasi elektronička shema*

YOLO također prilikom generiranja modela sprema rezultate gubitaka kroz trening i testiranje, kao i top1 i top5 točnost s promjenom stope učenja kroz epohe (slika 3.8). Top1 točnost odnosi se na omjer koliko je puta model dao predikciju za klase, pri čemu je klasa s najvišim postotkom ujedno i ona koja se predviđa. Top5 primjenjuje isti postupak, ali za pet klasa. Međutim, s obzirom na to da model razlikuje samo dvije klase, metrika top5 točnosti u ovome slučaju je redundantna. Npr., ako model daje predikciju od 0.6 za elektroničku shemu, a 0.4 za dijagram tijeka, to znači da top1 točnost odgovara samo u slučaju da je primjer koji se promatra zapravo elektronička shema, inače je izlaz netočno. U slučaju da je definirano više od pet klasa, top5 metrika bila bi relevantna jer bi se ekvivalentno top1 metrici promatralo može li model u svojih pet najboljih predikcija dobro pretpostaviti klasu.



Slika 3.8: *Metrike YOLOv5 modela*

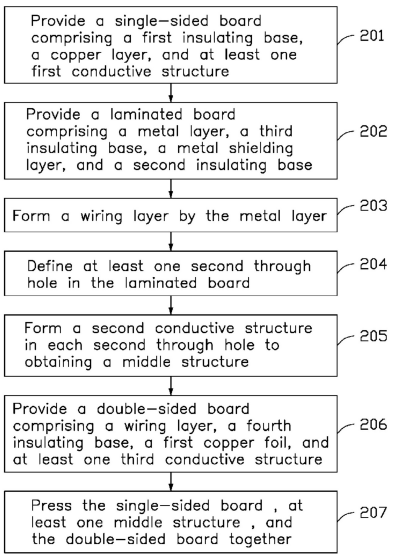
Model treniranjem kroz epohe ostvaruje dobre rezultate u vidu niskih trening i test gubitaka koji se pri kraju skoro pa izjednačavaju, što ukazuje na dobar fit parametara. Također, top1 točnost približava se 100% pri kraju izvođenja.

3.6.2 Izvlačenje teksta dijagrama tijeka

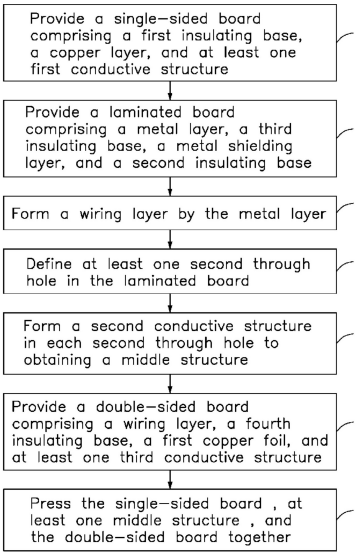
U nekoliko primjera dijagrama tijeka kao slikovnog prikaza unutar dokumentacije, prilikom inicijalnog rezanja sadržaja, može se dogoditi da se odreže dio teksta. Osnovni algoritam rezanja sadržaja više je prilagođen za rezanje shema, stoga se realizirala posebna komponenta koja, u slučaju klase dijagram tijeka, na poseban način dodatno obrađuje sliku (slika 3.9 i slika 3.10).

Iskorištava se potencijal biblioteke *OpenCV* s CUDA platformom za brže izvođenje procesa predobrade slike dijagrama tijeka. Dobivena se slika učitava u memoriju, a pri tome je slika u sivim tonovima. Na sliku u sivim tonovima primijenjen je adaptivni prag (engl. Adaptive thresholding). Slika se pretvara u binarni oblik gdje je pozadina bijele boje (255), a objekti dijagrama u crnoj boji (0) te se primjenjuje Gaussovo zamućenje kako bi se smanjio šum slike, ali i izgladila slika radi lakšeg detektiranja dijagrama. Takva se slika ponovno učitava u memoriju grafičke kartice te definira granični okvir koji obuhvaća dijagram. Primjenom morfološkog filtra spajaju se susjedna bijela područja i ispunjavaju male rupe, osiguravajući da su komponente dijagrama tijeka povezane. Preostali se šum uklanja filtriranjem obrisa s površinom manjom od 1000 piksela. Zatim se traži najveća kontura za koju se pretpostavlja da je dijagram tijeka. Granični okvir najveće konture izreže dijagram tako da se pokriju svi njegovi aspekti. Preciznim određivanjem dijagrama tijeka osigurava se izdvajanje pravilnog relevantnog dijela slike za daljnju obradu. Kao što se može primijetiti, glavna je stavka svakog dijagrama tijeka zapravo tekst koji ga čini. Stoga se korištenjem *tesseracta* iz svakog konačno obrađenog dijagrama tijeka izvlači tekst koji sadrži. Dakle, nema pridruživanja konkretnih vrijednosti vektoru slikovnih značajka nego se pridružuje nula ako je slika dijagram tijeka.

H05K 3/46	(2006.01)	third conductive structure; pressing the single-sided board,
H05K 3/00	(2006.01)	at least one middle structure, and the double-sided board in
H05K 1/02	(2006.01)	that sequence to form the circuit board.



Slika 3.9: *Primjer dijagrama tijeka prije dodatne obrade*



Slika 3.10: *Primjer dijagrama tijeka nakon dodatne obrade*

3.7. Predobrada teksta

Dobiveni tekst iz svakog dokumenta ne smije se ostaviti u originalnom obliku. Razlog tomu je što se svaka riječ može napisati u više različitih oblika, a da pri tome zadržava isto značenje. Primjerice, jabuka, jabučni, jabuke, jabučica su sve izvedeni oblici riječi istoga korijena koja označava vrstu voća. Korijen je riječi ključan jer on predstavlja korisnu informaciju. Svođenjem riječi na korijenski oblik poboljšava se preciznost klasifikacije. Upravo zato nužno je izvučeni tekst predobraditi. Korijen neke riječi se dobiva korištenjem funkcionalnosti *WordNetLemmatizera* iz biblioteke alata za prirodne jezike (engl. Natural Language Toolkit, NLTK). Dostupni se resursi iz NLTK preuzimaju jer su neophodni za ciljanu predobradu. Oni uključuju elemente kao što su:

- *punkt* - Dijeljenje teksta u riječi i rečenice.
- *stopwords* - Predstavlja popis uobičajenih riječi na engleskome jeziku koje se filtriraju jer

nemaju nikakvo kontekstualno značenje, prilozi/prijedlozi, upitne riječi i sl.

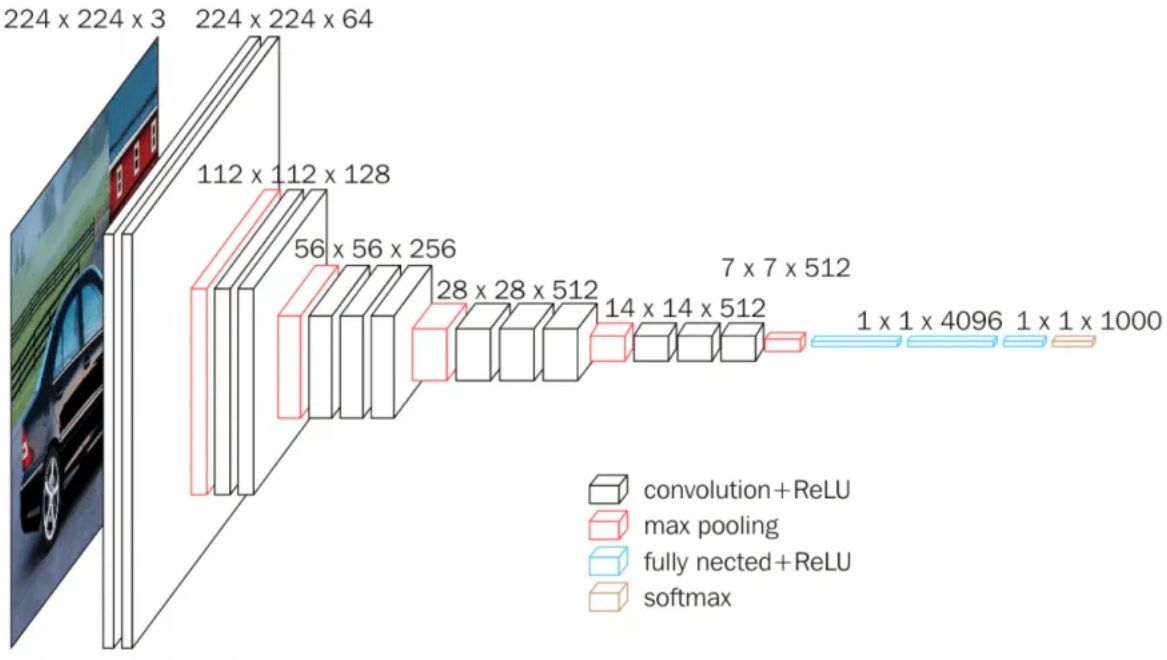
- *wordnet* - Baza koju koristi *WordNetLemmatizer* za lematizaciju, odnosno pretvaranje u korijenski oblik riječi.
- *omw-1.4* - Nema direktno pozivanje unutar funkcije, ali proširuje *wordnet* bazu dodatnim resursima.

Prvi je korak prilikom predobrade teksta pretvoriti kompletan tekst tako da svaka riječ bude pisana malim slovima. Razlika između velikih početnih slova i malih nije relevantna prilikom analize jer je važan isključivo sadržaj, tj. što pojedina riječ predstavlja. Sve se pojedinačne rečenice unutar teksta uzmu te se svaka rečenica dijeli na pojedinačne riječi. Svaka se riječ zatim pročisti od znakova koji nisu alfanumerički, kao što su znakovi interpunkcije. Takav je oblik riječi očekivan jer su samo slova i brojke korisne informacije. Zatim je potrebno izbaciti sve stop riječi koje služe samo za formuliranje misli, odnosno rečenica (npr. na engleskome jeziku riječi "the", "is", "a/an"...). U ovome izdanju riječi su spremne za proces lematizacije, tj. svođenje na njihov korijenski oblik. Ovaj postupak doprinosi normalizaciji i smanjivanju ukupnog broja riječi po dokumentu. Kako bi se poboljšala lematizacija, svakoj riječi određuje se vrsta (imenica, glagol, pridjev ili prilog), time definirajući kontekst riječi. Svakoj riječi pridružuje se odgovarajuća oznaka vrste (engl. Part Of Speech, POS). Ako se riječ ne pronađe na popisu definiranih vrsta, primjerice riječ "with" u engleskome jeziku smatra se prepozicijom, očekivano je ponašanje da se toj riječi pridruži oznaka imenice. Takav pojednostavljen pristup standard je prilikom rada s bibliotekama prirodnih jezika jer želimo sve preostale riječi uzeti u obzir. Proces lematizacije bez korištenja vrste riječi može potencijalno biti problematičan i dovesti do netočnih korijenskih oblika. Dobivene pretprocesirane riječi iz svakog dokumenta korištene su za formuliranje rječnika ili vokabulara. Svakoj riječi unutar rječnika postavlja se unikatan indeks pridružujući joj prirodni broj. Važno je inicijalizirati matricu ugradnje za vokabular, gdje je svaka riječ u vokabularu predstavljena vektorom fiksnih dimenzija (dimenzija ugradnje). Za potrebe inicijalizacije matrice ugradnje za vokabular koriste se unaprijed obučene ugradnje *GloVe*. Unaprijed obučene ugradnje (poput *Word2Vec*, *GloVe* ili *fastText*) često se koriste jer hvataju bogate semantičke odnose. U ovome radu korištena je verzija *GloVe 6b 50d* koja kombinira sve riječi dostupne na web stranici Wikipedia 2014. godine, zajedno s *Gigaword 5*. *Gigaword 5* sveobuhvatna je arhiva tekstualnih podataka s vijesti koje su prikupljane nekoliko godina. 50d označava da je svaka riječ predstavljena 50-dimenzionalnim vektorom. Ideja je stvoriti matricu ugradnje gdje svaki redak odgovara riječi u vokabularu i sadrži njezin vektor ugradnje.

Koristeći se ranije definiranim rječnikom, indeksiraju se sve riječi i rečenice koje su izvučene iz dokumenata u bazi. Proces započinje najprije rečenicu po rečenicu, uzimajući sve riječi koje čine tu rečenicu, ranije pretvorene u korijenski oblik pa im se pridružuje odgovarajući indeks iz rječnika. Tako se indeksiraju i same rečenice koje bi formirale dokumente. Nakon što su pravilno indeksirani elementi tekstualnog zapisa, dimenzije obrađenih dokumenata se definiraju tako da se odabire maksimalan broj rečenica po dokumentu i maksimalan broj riječi u rečenici. Odabrani su parametri fiksni i iznose maksimalno 25 rečenica u dokumentu, i maksimalno 15 riječi u rečenici. Inicijalno je isprobana varijanta u kojoj bi se maksimalna vrijednost određivala dinamički, odnosno tako da se kao maksimum uzme najveći broj rečenica u nekom dokumentu i najdulja rečenica prema broju riječi. No, ipak su se ove fiksne vrijednosti 25 i 15 pokazale dovoljno opširne za problematiku ovoga rada. Definiranjem maksimuma prolazi se kroz indeksirane rečenice i riječi kako bi se provela ova ograničenja. Ako neki dokument prelazi maksimalan broj rečenica skraćuje se do definiranog maksimuma, a preostale rečenice se neće uzimati u obzir. Identičan je postupak ako neka rečenica premašuje maksimalan broj riječi. Ako je broj rečenica ili riječi manji od definiranog, nadodaju se nule kako bi imali konzistentnu dimenziju za svaki dokument.

3.8. VGG19 obrada slika

VGG19 je arhitektura konvolucijske neuronske mreže (engl. Convolutional Neural Network, CNN) dizajnirana za klasifikaciju i prepoznavanje slika. Razvio ga je Visual Geometry Group (VGG), a model je predstavljen u radu [18]. Model VGG19 karakterizira 19 slojeva s težinama, uključujući konvolucijske slojeve, potpuno povezane slojeve i slojeve s maksimalnim udruživanjem. Duboki konvolucijski slojevi VGG19 mogu se koristiti za izdvajanje značajki iz prikaza. Pohranjene značajke kasnije se koriste u kombinaciji s tekstualnim dijelom. Ulaz u model je slika veličine 224x224 piksela, slično kao i kod YOLOv5 modela. Za svaku se sliku izračuna prosječna RGB vrijednost na skupu za treniranje pa se takva slika koristi kao ulaz u model. Primjenjuje se 3x3 filtar za izdvajanje značajki jer je to najmanja veličina za hvatanje svih smjerova na slici (lijevo/desno, gore/dolje, centar). Model karakteriziraju tri VGG potpuno povezana sloja (engl. Fully connected layer, fc), a preostalih 16 čine konvolucijski slojevi [19] (slika 3.11). Pri tome svaki konvolucijski sloj sadrži dvije do četiri konvolucijske operacije. Korištenjem tri filtra 3x3 umjesto jednog 7x7 dobiju se tri ReLu sloja umjesto jednog, što rezultira smanjenim brojem parametara i više nelinearnosti, jer se dobiju tri aktivacije. Model se treba inicijalizirati s težinama unaprijed obučenim na skupu podataka *ImageNet*. *ImageNet* težine trenirane su na velikom skupu podataka, što znači da će izdvajati opće značajke poput rubova, oblika i tekstura. Te osnovne značajke mogu biti korisne i za slike poput dijagrama tijeka i elektroničkih shema jer te slike i dalje sadrže linije, oblike i strukture koje model može prepoznati. Najčešće iteracije VGG arhitekture su VGG16 i VGG19, a razlike između njih su vidljive u broju konvolucijskih slojeva (tablica 3.3). Tablica 3.3 ukazuje kako su konvolucijski slojevi



Slika 3.11: *Arhitektura VGG19 modela*

veličine receptivnog polja tri, a VGG19 koristi tri više konvolucijska sloja, jedan s 256 i dva s 512 kanala. Setovi konvolucijskih slojeva uvijek su popraćeni maksimalnim skupljanjem preko značajki susjeda (engl. Maxpool) tako da se 3x3 filter pomiče preko svakog kanala i samim time sažimaju značajke. Nakon konvolucijskih slojeva nalaze se tri potpuno povezana sloja s 4096 neurona, odnosno 1000 u zadnjem, iza kojih slijedi *softmax* koji daje konačnu vjerojatnost. Cilj ove primjene je dobiti značajke aktivacije s jednog od dubljih slojeva mreže koje obuhvaćaju prikaze ulazne slike.

VGG16	**VGG19**
ulazna 224x224 slika	
conv3-64	conv3-64
conv3-64	conv3-64
maxpool	
conv3-128	conv3-128
conv3-128	conv3-128
maxpool	
conv3-256	conv3-256
conv3-256	conv3-256
conv3-256	conv3-256
	conv3-256
maxpool	
conv3-512	conv3-512
conv3-512	conv3-512
conv3-512	conv3-512
	conv3-512
maxpool	
conv3-512	conv3-512
conv3-512	conv3-512
conv3-512	conv3-512
	conv3-512
maxpool	
FC-4096	
FC-4096	
FC-1000	
softmax	

Tablica 3.3: *Usporedba VGG16 i VGG19 modela po slojevima*

Nužno je obraditi značajke izvučenih slika, a naročito slučajeve u kojima slike možda nedostaju ili ih nema, a zatim pripremiti značajke slika za spajanje sa značajkama iz HAN mreže. Taj je proces od presudne važnosti u kontekstu projekta koji uključuje kombiniranje tekstualnih značajki iz HAN modela i slikovnih značajki za multimodalni zadatak učenja. U skupovima podataka iz stvarnog svijeta mogu postojati slučajevi u kojima neki dokumenti

nemaju pridružene slike (sheme). U takvom slučaju nužna je zamjena nedostajućih značajki nultim vektorom. Postupkom zamjene značajki slike osigurava se da odsutnost slike ne prekida obradu, a zamjenom značajki koje nedostaju s nul vektorom prikladne veličine (1, 1000) osigurano je da svi unosi imaju istu dimenzionalnost. Veličina (1, 1000) odgovara očekivanoj veličini polja nakon obrade slike, s obzirom na to da je izlaz nakon predobrade slike 1000-dimenzionalni vektor koji predstavlja distribuciju vjerojatnosti preko 1000 klasa u skupu podataka *ImageNet*. Ova uniformnost ključna je za spajanje s drugim skupovima značajki, konkretno tekstualnim zapisima provučenim kroz HAN. Posljednji korak slaganja obrađenih značajki slike u jedno polje čini spajanje ovih značajki sa značajkama HAN-a. Taj objedinjeni prikaz značajki se koristi za zadatak klasifikacije, gdje su i tekstualne i slikovne informacije ključne. Modeli strojnog učenja često zahtijevaju da ulazi budu u određenim formatima, kao što su *NumPy* polja za određene biblioteke. Bitno je osigurati da su sve značajke slike u dosljednom formatu i dimenziji.

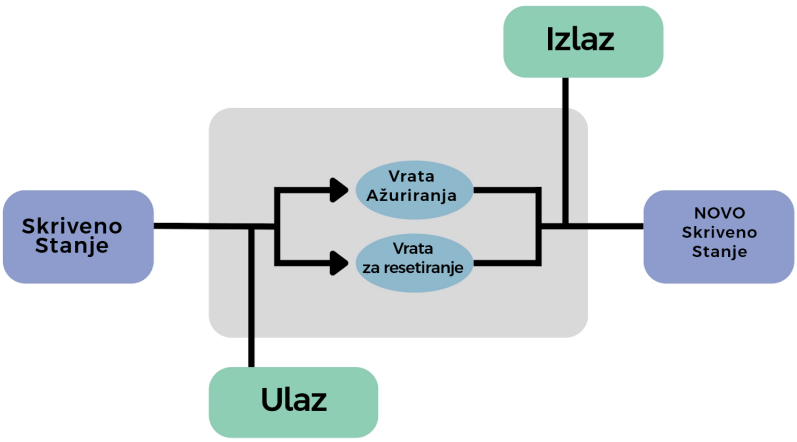
Važno je voditi računa i o dimenzionalnosti vektora slikovnih značajki. Značajke slike i značajke teksta moraju imaju kompatibilne dimenzije za ulančavanje i naknadnu obradu. HAN model proizvodi vektore značajki specifične dimenzionalnosti. Ako su vektori značajki slike duži od potrebnog, njihovo skraćivanje osigurava da odgovaraju očekivanoj dimenzionalnosti bez značajnih gubitaka. Ovo je pragmatičan pristup kada dodatne značajke možda neće značajno utjecati na performanse modela. U slučaju da su vektori obilježja slike kraći od potrebnog, njihovo popunjavanje nulama osigurava podudaranje dimenzionalnosti bez uvođenja dodatnih informacija. Time se održava cjelovitost izvornih značajki, a istovremeno omogućuje dosljednost dimenzija. Multimodalni modeli učenja koji obrađuju i tekstualne i vizualne informacije zahtijevaju da značajke unosa budu dosljednih dimenzija. Taj korak osigurava da se obje vrste značajki mogu pravilno integrirati i zajedno obrađivati, omogućujući modelu da učinkovito uči iz oba vektora. Postoje i alternative skraćivanju i popunjavanju, kao npr. analiza glavnih komponenti (engl. Principal Component Analysis, PCA). PCA je statistički postupak koji putem ortogonalne transformacije pretvara skup koreliranih varijabli u skup nekoreliranih varijabli. Ideja je smanjiti dimenzionalnost skupa podataka očuvanjem najvažnijih obrazaca ili odnosa između varijabli bez prethodnog znanja o ciljanim varijablama [20]. Za realizaciju korišten je jednostavniji pristup kao što je popunjavanje, odnosno skraćivanje.

3.9. Obrada teksta korištenjem HAN mreže

Obraditi tekstualni zapis tako da se izvuku relevantne informacije iz njega složen je problem. Postoje različiti modeli strojnoga učenja koji pomažu pri rješavanju problema. Tradicionalni pristup strojnog učenja koji se nameće je, primjerice SVM. No, SVM može imati poteškoća s vrlo velikim skupovima podataka. Nadalje, algoritam Nasumične šume metoda je koja može biti vrlo učinkovita, a posebice sa strukturiranim podacima i manjim skupovima podataka. No, kao i kod SVM-a, algoritam Nasumične šume problematičan je za rad s velikim skupom podataka, a i algoritam zahtjeva veliku količinu računalnih resursa prilikom izvođenja. Postoje i složeniji modeli i pristupi, kao što su prikazi dvosmjernog kodera iz transformatora (engl. Bidirectional Encoder Representations from Transformers, BERT), unaprijed obučeni model transformatora koji se može podesiti za specifične zadatke klasifikacije teksta. BERT dohvaća kontekst iz oba smjera, kao i Robusno optimizirani BERT pristup (engl. Robustly Optimized BERT Pretraining Approach, RoBERTa), koji je optimizirana verzija BERT-a, a pokazuje poboljšanu izvedbu u mnogim zadacima obrade prirodnih jezika (engl. Natural Language Processing, NLP).

Hijerarhijska mreža pažnje (engl. Hierarchical Attention Network, HAN) posebno je pogodna mreža za zadatke klasifikacije dokumenata i iz tog razloga odabrana je za obradu teksta. HAN obrađuje dobiveni tekst na razini riječi i na razini rečenica. Ključno je odrediti kontekst riječi unutar rečenica te definirati strukturu rečenica koje formiraju tekst koji se obrađuje.

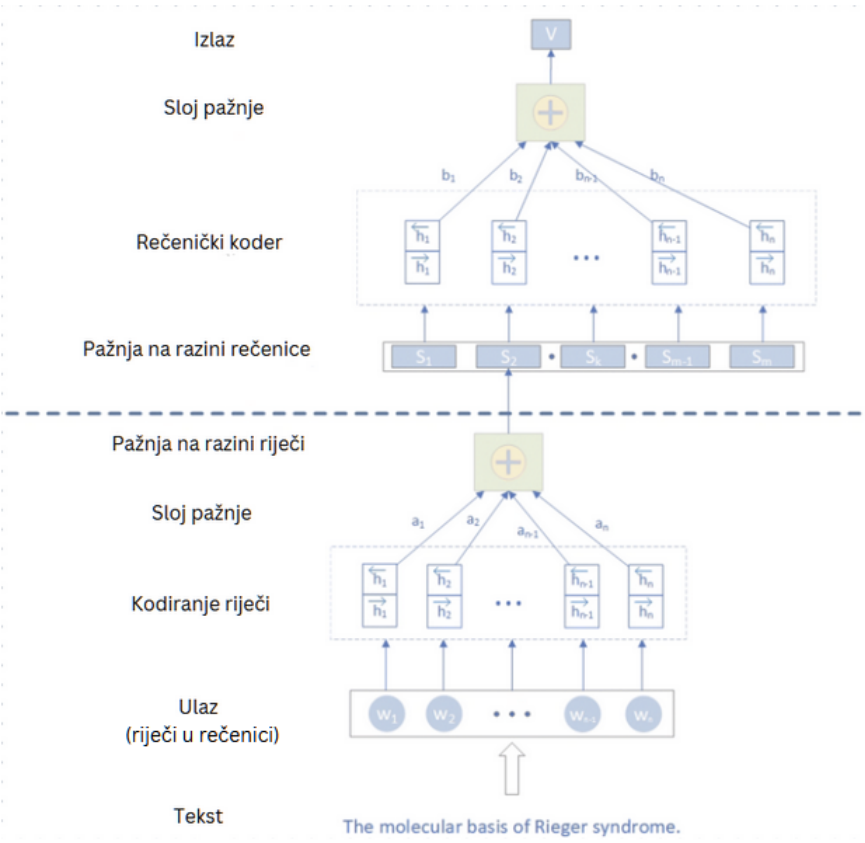
Različite neuronske mreže mogu se koristiti kako bi se obradile riječi i rečenice. U ovom projektu koristila se zatvorena ponavljajuća jedinica (engl. Gated Reccurent Unit, GRU) za kodiranje niza riječi. GRU je vrsta ponavljajuće neuronske mreže (engl. Recurrent Neural Network, RNN) koja se koristi za obradu sekvencijalnih podataka. Mehanizam pažnje (engl. Attention mechanism) određuje važnost svake riječi u oblikovanju rečenice te važnost svake riječi unutar rečenice. Taj mehanizam omogućuje modelu da se usredotoči na najrelevantnije riječi u rečenici i najrelevantnije rečenice u dokumentu poboljšavajući rezultate klasifikacije, osobito na dugim i složenim dokumentima. Za isticanje najvažnijih riječi u rečenici, ali i najvažnijih rečenica u kontekstu sažetka, koriste se težine pažnje (engl. Attention weights). Autori su u radu [21] pokazali da HAN model odlično klasificira primjere tehničke dokumentacije na temelju teksta.



Slika 3.12: *Arhitektura GRU*

Za kodiranje riječi koristi se ranije spomenuta neuronska mreža GRU. GRU koristi dvije vrste vrata: vrata za ažuriranje i vrata za resetiranje (slika 3.12). Ta vrata kontroliraju protok informacija i upravljaju stanjem memorije. U kontekstu HAN-a, ulaz u GRU su riječi koje formiraju neku rečenicu. Primjerice za rečenicu "Jabuka je slatka, a limun je kisel.", svaka se riječ ("jabuka", "je", "slatka"...) predstavlja kao vektor brojeva. Skriveno stanje pomaže GRU-u pamteći relevantne informacije iz prethodnih riječi. Dok GRU obrađuje svaku riječ u rečenici, skriveno se stanje ažurira kako bi se odražavalo kontekstu. Npr., nakon obrade riječi "jabuka", skriveno stanje sadržava kontekstualne informacije o toj riječi. Ažuriranje se obavlja pomoću vrata ažuriranja. Ta vrata kontroliraju koliko prethodnog skrivenog stanja treba prenijeti naprijed i koliko novog unosa treba koristiti za ažuriranje skrivenog stanja. Ako je trenutna riječ "slatka", vrata za ažuriranje određuju koliko bi kontekst "Jabuka je" trebao utjecati na novo skriveno stanje u odnosu na trenutnu riječ. Vrata za resetiranje kontroliraju koliko prethodnog skrivenog stanja treba zanemariti. Primjerice, rečenica može prijeći iz jednog kontekstualnog dijela ("Jabuka je slatka") u drugi ("a limun je kisel"), a vrata za resetiranje u tom slučaju pomažu smanjiti utjecaj prethodnog konteksta fokusirajući se na drugi dio rečenice. Novo skriveno stanje formira se kombiniranjem novih informacija sa prethodnim skrivenim stanjem. Drugim riječima, nakon obrade riječi "slatka", dobije se novo skriveno stanje sažimajući cijeli prethodni kontekst "Jabuka je slatka". Izlaz nakon obrade svake riječi u nekoj rečenici posljednje je skriveno stanje i ono postaje prikaz neke rečenice. Analogni postupak slijedi se i za kodiranje rečenica u dokumentu.

Promatrajući arhitekturu HAN modela (slika 3.13), jasno se uočavaju ranije navedene ključne komponente. Zadana je rečenica s riječima, a riječi su ugrađene u vektore preko matrice ugradnje. Sloj kodiranja riječi koristi dvosmjerni GRU za dobivanje konteksta riječi sažimanjem informacija iz oba smjera, unaprijed i unatrag. Kombiniraju se dva GRU sloja,



Slika 3.13: *Arhitektura HAN modela*

jedan koji obrađuje ugrađene riječi u smjeru unaprijed (od prvog elementa prema zadnjem) i drugi u smjeru unatrag (od zadnjeg elementa prema prvome), uzimajući kontekst s obje strane svake riječi. Izlaz je niz skrivenih stanja za svaku riječ. Slijedi sloj pažnje na razini riječi. On izračunava težine pažnje svake riječi unutar rečenice i izračunava ponderirani zbroj reprezentacija riječi kako bi se stvorila konačna reprezentacija rečenice. Ponderirani zbroj reprezentacija riječi u rečenici dobije se tako da se svaki rečenični vektor množi sa svojom odgovarajućom težinom pažnje, a zatim se svi težinski vektori zbrajaju. Linearni sloj unutar sloja pažnje na razini riječi projicira GRU izlaz u skalarni rezultat koji predstavlja važnost svake riječi. *Softmax* pretvara rezultate u težinu pozornosti, normalizirajući ih za sve riječi u rečenici. Zatim se množi svaka reprezentacija riječi odgovarajućom težinom pozornosti i zbraja kako bi se dobio jedan vektor koji predstavlja cijelu rečenicu. Pažnja na razini rečenice određuje važnost svake rečenice unutar dokumenta, slično pozornosti na razini riječi, ali na višoj razini hijerarhije. Linearni sloj unutar pažnje na razini rečenice projicira izlaz iz GRU-a za rečenice u skalarne rezultate, čime se ukazuje na važnost svake rečenice u kontekstu dokumenta. Kao i kod pažnje na razini riječi, izračunaju se težine pažnje i ponderirani zbroj reprezentacija rečenica kako bi se dobio prikaz jednog dokumenta. Ponderirani zbroj dobije se zbrajanjem umnožaka svakog vektora rečenica s odgovarajućom težinom pažnje. Rečenički koder obrađuje prikaze rečenica hvatajući odnose između rečenica unutar dokumenta. Slično principu rada sloja kodiranja riječi, dvosmjerni GRU obrađuje prikaze rečenica u oba smjera kako bi generirao skrivena stanja koja hvataju kontekst svake rečenice unutar dokumenta. Naposljetku, izlaz HAN modela preslikava konačni prikaz dokumenta u višedimenzionalni prostor. Praćeni prikaz dokumenta prolazi kroz potpuno povezani sloj koji daje konačni prikaz dokumenta. Željeni je izlaz tenzor gdje svaki red odgovara prikazu značajke dokumenta nakon prolaska kroz ranije opisanu hijerarhiju.

S obzirom na ograničenja resursa procesora *Ryzen 5 5600* koji se koristi, a i s ciljem

povećanja brzine izvođenja, uveden je koncept paralelizma u kodu. Naime, koriste se niti za istovremeno paralelno izvođenje procesuiranja svih dokumenata unutar nekog direktorija. Svaki direktorij predstavlja jednu od šest klasa H01-H10 i time su označeni dokumenti. U svakome direktoriju obavlja se ranije opisan proces kako bi se dobile potrebne značajke, ali pritom iskoristivši potencijal *ThreadPoolExecutor* klase. *ThreadPoolExecutor* omogućuje da se svakoj od niti pošalje zadatak koji se obavlja pozivajući funkciju koja definira upravo te zadatke. Svaka je nit indeksirana kako bi se pratilo stanje i koja je nit obavila koji zadatak. Konačni mapirani skup sadržava tri unikatne vrijednosti: vrijednost riječi, vrijednost dijagrama tijekova i vrijednost shematskih prikaza. Prvo se čeka na izvršenje svih niti pa se tek onda spreme rezultati. Pomoću indeksa ispravno se poredaju dokumenti prema strukturi direktorija, neovisno o tome kako i kada su obrađeni. Ova funkcionalnost ključna je prilikom rada s velikim brojem dokumenata.

Konačni vektor *node_features* dobije se povezivanjem vektora značajki slika i vektora tekstualnih značajki duž x osi. Rezultat je prikaz gdje svaki red kombinira obje značajke.

3.10. Kreiranje grafa i proces učenja

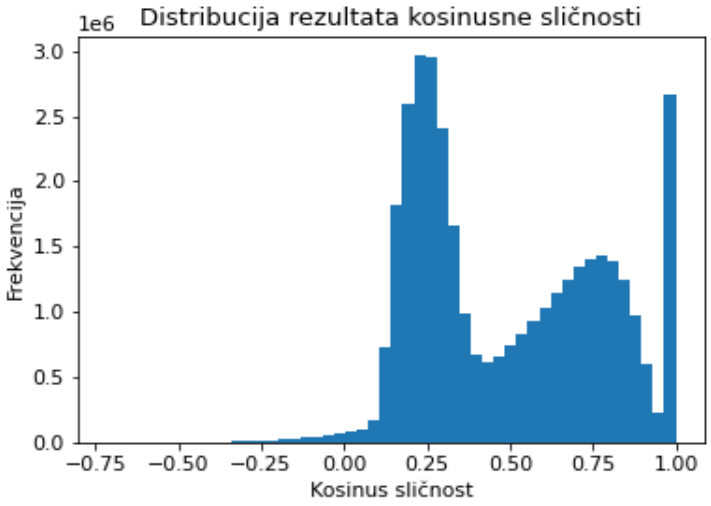
Konačni je korak izgraditi model koji može manipulirati prethodno obrađenim informacijama i iskoristiti ga za klasifikaciju dokumenata. Predloženo rješenje implementacija je grafičke neuronske mreže zvane GraphSAGE. GraphSAGE uči na graf strukturiranim podacima (engl. Graph-structured data), omogućujući predviđanje nevidljivih čvorova prikupljanjem poduzorkovanih lokalnih susjedstva.

Prije definiranja GraphSAGE mreže, kreirana je graf struktura na kojoj konačni model uči. Graf se kreira koristeći *NetworkX* biblioteku, a njemu se dodaju značajke čvorova i rubova na temelju praga sličnosti. Najprije se inicijalizira prazan graf koji se popunjava tako da se broj čvorova postavi na temelju duljine vektora značajki čvorova *node_features*. Svaki se čvor popunjava normaliziranim prethodno dobivenim vrijednostima vektora značajki čvorova. Rubovi između čvorova stvaraju se na temelju rezultata sličnosti kosinusa. Uz sličnost kosinusa veže se i pojam praga gdje prag određuje minimalnu vrijednost sličnosti potrebnu za povezivanje dva čvora s rubom. Algoritam prolazi kroz sve parove čvorova i provjerava odgovarajuću vrijednost u matrici sličnosti. Korištenjem funkcije kosinusa generira se matrica sličnosti, pri čemu svaki unos predstavlja sličnost između dva čvora. Ako sličnost između čvorova premašuje prag, između njih se dodaje rub. Kosinus sličnosti mjera je sličnosti između dva vektora, a računa se kao kosinus kuta između dva vektora, što rezultira vrijednošću između -1 i 1. Za vektore značajki koji nisu negativni, vrijednost se najčešće kreće od 0 do 1. Rezultat kosinusa 1 označava da su vektori identični, 0 da su ortogonalni (nema sličnosti), a vrijednost oko -1 znači da su vektori suprotni. Formula po kojoj se računa kosinus je [22]:

$$\cos(\theta) = \frac{\overrightarrow{A} \cdot \overrightarrow{B}}{\left\|\overrightarrow{A}\right\| \left\|\overrightarrow{B}\right\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}, \tag{3-1}$$

gdje su $\overrightarrow{A}$ i $\overrightarrow{B}$ proizvoljni vektori, A_i i B_i komponente vektora, a n dimenzija vektora. Na primjer, ako kut između dva vektora $\overrightarrow{A}$ i $\overrightarrow{B}$ iznosi 20 stupnjeva, onda bi kosinus sličnosti za te značajke iznosio $cos(20) = 0.93969262$, odnosno $\sim 93.7\%$. Promatrajući distribuciju sličnosti značajki u bazi (slika 3.14) bira se optimalna vrijednost praga - odabran je prag od 0.2, kako bi se uzelo u obzir dovoljno sličnih instanci iz prikazane distribucije.

Rubovi i značajke čvora iz grafa konvertiraju se u tenzor objekt zbog zahtjeva ulaza u GraphSAGE. Značajke čvorova izdvoje se iz grafa te pridružuju *data.x*, a indeksi rubova u

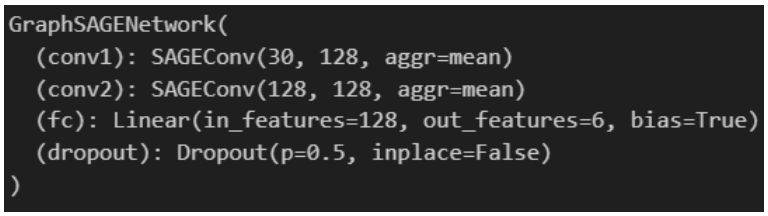


Slika 3.14: *Distribucija sličnosti značajki*

data.edge_index, što je standardno mjesto za ulazne podatke neke neuronske mreže. Zatim se iz popisa direktorija, gdje je svaki direktorij posebno indeksiran za odgovarajuću klasu pridružuju oznake svakoj putanji. Zbog jednostavnosti se koriste brojevi od nula do pet. Dakle, klasa H01 odgovara nuli, H01 jedinici, itd. Svaka *.pdf* datoteka koja se nalazi pod nekom oznakom se broji te se dobije ukupno 6000 oznaka od nula do pet za svaku klasu. Taj se rezultat spremi u obliku tenzora pod *data.y*, što je također standardna praksa prilikom procesa klasifikacije.

GraphSAGE uči funkciju koja generira ugradnje čvorova za svaki čvor na temelju njegovih značajki i značajki njegovih susjeda. Ova neuronska mreža koristi značajke čvora zajedno sa značajkama njegovih susjeda za kreiranje ugradnje čvorova koje obuhvaćaju lokalnu strukturu grafa. Za svaki čvor odabran je fiksan broj susjeda, a njihove značajke se agregiraju pomoću diferencijalne funkcije. Diferencijalne funkcije mogu biti različite, LSTM, srednja agregacija ili udruživanje. U ovome radu korištena je diferencijalna funkcija srednje agregacije (engl. Mean aggregation). Srednja agregacija prikuplja vektore značajki svih susjednih čvorova i izračunava prosjek tih vrijednosti. Konkretno, funkcija srednje agregacije zbraja odgovarajuće elemente svakog vektora značajki i podijeli s ukupnim brojem susjeda. Rezultantni vektor predstavlja agregirane informacije iz definiranog susjedstva. Taj se vektor spaja s vektorom obilježja svakog čvora, pa novi vektor prolazi kroz sloj neuronske mreže kako bi se dobilo ažurirano ugrađivanje. Slaganjem više GraphSAGE slojeva mreža može uhvatiti informacije iz sve većih susjedstava. Definirana mreža ima dva GraphSAGE sloja, prvi sloj agregira od izravnih susjeda, a drugi sloj agregira od njihovih susjeda. Svaki sloj uzima ulaznu dimenziju i daje na izlaz novi vektor obilježja s određenom skrivenom dimenzijom. Nakon agregacije, izlazne značajke prolaze kroz potpuno povezani sloj kako bi transformirale ugradnje čvorova u željenu izlaznu dimenziju. *Dropout* se primjenjuje nakon svakog GraphSAGE sloja kako bi se spriječilo prekomjerno prilagođavanje nasumičnim ispadanjem neurona tijekom treninga.

U konkretnoj realizaciji GrapSAGE modela (slika 3.15), ulazna obilježja značajki čvorova *data.x* prvo prolaze kroz prvi GraphSAGE konvolucijski sloj zajedno s rubnim indeksima *data.edge_index*, koji predstavljaju povezanost grafa. U prvom konvolucijskom sloju nakon dobivanja ažuriranog ugrađivanja, izlaz prolazi kroz funkciju aktivacije *ReLU* kako bi se uvela nelinearnost, nakon čega slijedi *Dropout* radi regularizacije. Izlaz prvog sloja prolazi još jedan krug agregacije susjedstva, nelinearne transformacije i *Dropouta*, generirajući nove ugradnje čvorova. Izlaz drugog sloja prolazi kroz potpuno povezani sloj, koji preslikava ugradnje čvorova u konačnu izlaznu dimenziju koja odgovara broju klasa u zadatku klasifikacije (šest).



Slika 3.15: *Arhitektura GraphSAGE modela*

Uz sam GraphSAGE model, važno je i definirati metode za rad s GraphSAGE modelom. Prva metoda uključuje proces treniranja, a ostvaruje se tako što se najprije model postavlja u *train* način rada označavajući modelu da koristi dropout i normalizaciju. Optimizatoru prilikom treniranja se naznači da koristi nulte gradijente jer inače se tijekom procesa treniranja gradijenti akumuliraju prema zadanim postavkama, tako da ovaj korak osigurava da se novi gradijenti ne dodaju starim. Ta metoda briše sve prethodno nakupljene gradijente iz parametara modela. Ranije definirani vektori *data.x, data.y, data.edge_index* se pomiču na memoriju grafičke kartice kako bi se maksimalno iskoristio potencijal CUDA biblioteke. Uz te parametre, pomiče se i trening maska koja je skup *boolean* vrijednosti (nula ili jedinica za predstavljanje točnog/netočnog), a koja određuje koji se čvorovi u grafu trebaju koristiti za obuku. Ulazni vektori x i *edge_index* predaju se definiranome modelu GraphSAGE kako bi ih obradio te spremio izlaz iz mreže. Kako bi se tijekom treninga procijenili gubitci, računa se kriterij funkcija koja uspoređuje omjer predviđenih čvorova (definiranih u trening maski) sa stvarnim čvorovima definiranim u y vektoru. Gradijenti gubitka računaju se s obzirom na parametre modela izvođenjem povratnog širenja (engl. Backpropagation). Metoda povratnog širenja uzima stopu pogreške širenja naprijed i prenosi taj gubitak unatrag kroz slojeve neuronske mreže radi podešavanja težina. Svi oni gradijenti koji prelaze maksimalnu normu od 1.0 se izrežu kako bi spriječili nestabilnost tijekom trening procesa. Koristeći se tim gradijentima, parametri modela ažuriraju se s pomoću definiranog algoritma optimizacije, a vrijednost gubitka se sprema.

Druga je metoda za rad s modelom metoda validacije, a koristi se kako bi procijenili ponašanje modela na validacijskom skupu podataka nakon treniranja modela. Validacija se koristi za praćenje koliko dobro model generalizira nevidljive podatke, osim toga proces validacije smanjuje vjerojatnost pojavljivanja prekomjernog prilagođavanja. Izračunom validacijske točnosti i gubitka može se odlučiti kada prekinuti trening ili prilagoditi hiperparametre. Validacijski gubitak računa se korištenjem unakrsne entropije (o tome više u sljedećem poglavlju) na čvorovima u validacijskom skupu, dok se validacijska točnost računa kao omjer točnih predviđanja i ukupnog broja validacijskih čvorova.

Uz validaciju i treniranje, definirana je i metoda testiranja koja procjenjuje izvedbu treniranog modela na testnom skupu podataka, izračunavajući različite metrike za procjenu njegove učinkovitosti (o samim metrikama više u sljedećem poglavlju). Ta funkcionalnost ključna je za ocjenjivanje generaliziranja modela na neviđenom skupu podataka, kao i za usporedbu performansi različitih hiperparametara. Prije pozivanja trening funkcije mora se definirati vrsta optimizatora koji se koristi. Optimizatori su algoritmi korišteni za smanjenje funkcije pogreške i povećanje učinkovitosti modela. Postoje različite vrste kao što su: gradijentni pad (engl. Gradient descent), stohastički gradijentni pad (engl. Stohastic gradient descent), mini serija gradijentnog pada (engl. Mini batch gradient descent) itd. Za potrebe izrade korišten je Adam (engl. Adaptive Moment Estimation) optimizator. Taj optimizator najčešće se koristi prilikom rada s velikom količinom podataka, osobito jer odlično upravlja memorijom. Adam optimizator koristi se za ubrzavanje algoritma spuštanja gradijenta uzimajući u obzir eksponencijalno

ponderirani prosjek gradijenata[23]:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)[\frac{\delta L}{\delta w_T}], \tag{3-2}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)[\frac{\delta L}{\delta w_T}]^2, \tag{3-3}$$

gdje su:

- m_t - Agregat gradijenata u trenutnom vremenu t (početno, $m_t = 0$).
- m_{t-1} - Zbroj gradijenata u vremenu t-1 (prethodni).
- w_t - Težine u trenutku t.
- w_{t+1} - Težine u trenutku t+1.
- δL - Derivacija funkcije gubitka.
- δw_t - Derivacija težina u trenutku t.
- β_1 i β_2 - Pomični prosjek (konstanta 0.9).

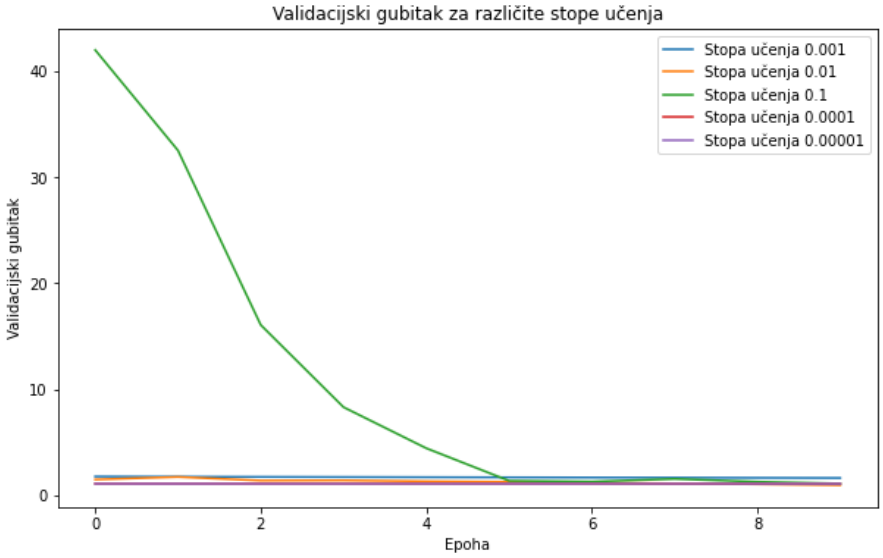
Adam optimizator izračunava eksponencijalno opadajući prosjek prošlih kvadratnih gradijenata t ega iskorištava prilikom skaliranja stope učenja za svaki parametar.

Uz optimizator, važno je definirati i stopu učenja. Stopa učenja regulira težine neuronske mreže s gradijentom gubitka. Kako bi odabrali prikladnu stopu učenja implementirana je metoda pretrage rešetke (engl. Grid search). Primjenom te metode se otkriva optimalna vrijednost stope učenja za proces treninga tako što se evaluira performanse treniranja koristeći se različitim vrijednostima stope učenja. Isprobane su sljedeće vrijednosti stopa učenja za metodu pretrage rešetke: $0.1, 0.01, 0.001, 0.0001, 0.00001$. U sklopu pretrage rešetke trenira se GraphSAGE model. Stoga, prije nego se započne proces treniranja, nužno je rasporediti podatke u skupove za trening, test i validaciju. Za potrebe algoritma pretrage rešetke koristi se jednostavniji pristup gdje se skup podataka podijeli u omjeru 70% podataka za trening set, 15% za validaciju i 15% za testiranje. Koristeći ugrađenu funkcionalnost iz *scikit-learn* paketa za podjelu skupa, prvo se dijele podatci na trening i privremeni skup. Zatim se taj privremeni skup dodatno dijeli na validacijske i testne skupove, osiguravajući da je svako dijeljenje stratificirano na temelju oznaka klase kako bi imali ravnomjeran omjer iz svih klasa u sva tri skupa. Model GraphSAGE inicijalizira se s ulaznom dimenzijom koja odgovara broju značajki čvorova, skrivenom dimenzijom 128 i izlaznom dimenzijom šest koja odgovara broju klasa. Dropout stopa postavlja se na 0.5 označavajući da 50% neurona ispada tijekom procesa treniranja. Vrijednost 0.5 je zadana ako se ne definira eksplicitno te se pokazala kao najbolja prilikom razvoja konačnog modela. Model se također inicijalizira u memoriju grafičke kartice jer se tamo već nalaze podatci koji se obrađuju. Za svaku definiranu stopu učenja redom se inicijalizira Adam optimizator, pa se trenira kroz deset epoha i izračuna trening gubitke, validacijske gubitke i validacijsku točnost. Za određivanje idealne stope učenja uspoređuju se isključivo validacijski gubitci. Na kraju desete epohe sprema se najmanja vrijednost validacijskog gubitka. Za svaku stopu učenja uspoređuju se validacijski gubitci te se za najbolju stopu učenja bira ona koja je u deset epoha treniranja ostvarila najmanji iznos validacijskog gubitka. U ovom radu najbolji iznos stope učenja iznosi 0.01.

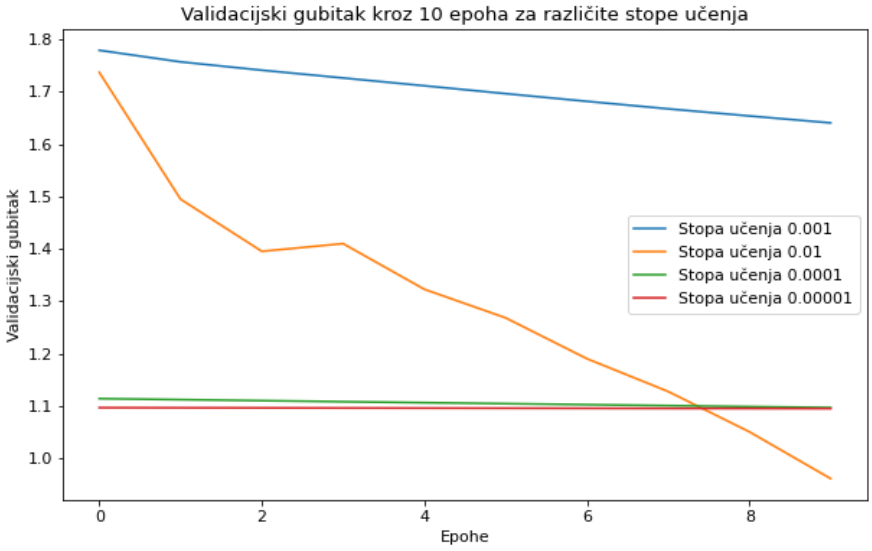
Dodana je i metoda ranoga zaustavljanja (engl. Early stopping). Metoda funkcionira tako što čeka na poboljšanje validacijskog gubitka, a u slučaju da se kroz odabrani broj epoha ne poboljša iznos, trening se prekida. Nužno je definirati delta vrijednost koja predstavlja

minimalan iznos koji se smatra poboljšanjem. Ako je promjena manja od delta vrijednosti taj se ishod ne smatra poboljšanjem. Broj se epoha tijekom kojih nije došlo do poboljšanja prati i ako taj broj pređe prag strpljenja trening se prekida. Odabran je prag vrijednosti pet, jer ukazuje na to da model prekomjerno prilagođava, ali neće prerano prekinuti izvođenje treninga. Rano zaustavljanje pomaže modelu spriječiti do prekomjerno prilagođavanje.

Slika 3.16 pokazuje kako za stopu učenja 0.1 validacijski gubitak ima drastičnu inicijalnu vrijednost a do kraja zadnje epohe znatno se promijeni što ukazuje na preveliku nestabilnost. slika 3.17, na kojoj je zbog lakše analize, izostavljena vrijednost 0.1, prikazuje kako su sve preostale vrijednosti (osim odabrane 0.01) skoro konstantne, što indicira sporo učenje kroz epohe i samim time prenisku vrijednost parametra stope učenja. Tim postupkom se dodatno potvrđuje odabir 0.01 kao najbolje vrijednosti stope učenja (preostali hiperparametri su navedeni u sljedećem poglavlju).



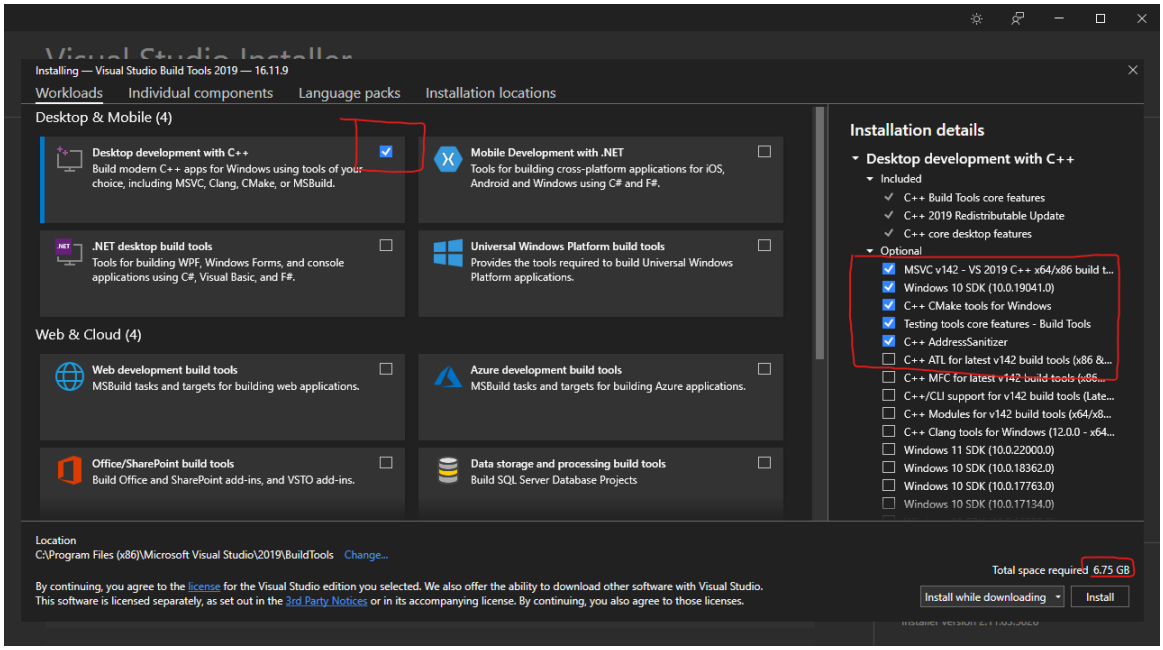
Slika 3.16: *Validacijski gubitci kroz epohe*



Slika 3.17: *Validacijski gubitci kroz epohe bez vrijednosti 0.1*

3.11. Pokretanje vlastitog rješenja

Kako bi se konačni projekt mogao lokalno pokrenuti, bitno je poduzeti nekoliko koraka. Najprije se klonira sadržaj javnog repozitorija na lokalno računalo koristeći naredbu *git clone* dodavajući poveznicu projekta. Kompletan programski kôd projekta dostupan je na poveznici [24]. Nakon što se klonira kompletan sadržaj, nužno je instalirati potrebne pakete korištene u kontekstu projekta. Ako je na operacijskome sustavu prvi puta instaliran *Python*, obavlja se prvo par predkoraka. Prvo se moraju instalirati potrebne biblioteke s *pip install prox, data, tight, common, dual*. Uz te biblioteke, nužno je imati i alate *Microsoft C++ Build Tools* s poveznice [25] jer se neke biblioteke instaliraju koristeći te alate. U instalacijskom prozoru potrebno je označiti prozor naziva *C++ build tools* i instalirati potrebne pakete (slika 3.18).



Slika 3.18: *Instalacija potrebnih C++ build tools paketa*

Posljednji je korak, u slučaju da korisnik nema Nvidia grafičku karticu, otvoriti *requirements.txt* datoteku prije pokretanja instalacije i obrisati dio teksta *+cu121* iz sljedeće linije: *torch==2.3.1+cu121*. Također, u slučaju da korisnik ima drugačiju verziju CUDA-e, treba verziju *+cu121* promijeniti u korištenu *+cu118* ili *+cu124*. Za rad potrebna je verzija *Python* programskog jezika koja nije novija od 3.11, a pritom nije starija od 3.8 verzije. Važno je i instalirati *tesseract* direktno sa službene stranice, uz *Python* inačicu koja se kasnije dodaje. *Tesseract* se preuzima s poveznice [26] pa zatim instalira prateći upute na ekranu. Lokaciju na koju je instaliran nužno je dodati u prozoru *Uređivanje varijabli okruženja sustava* unutar *Windows* operacijskog sustava. Prozor varijabli okruženja je odabran i dodaje se nova lokacija instaliranog *tesseracta* u sistemske varijable pod oznakom *Path*. Nakon toga, pozivajući naredbu *pip install -r requirements.txt* instaliraju se preostale biblioteke. Paketi su generirani koristeći se *pipreqs* bibliotekom. Biblioteka *OpenCV* ovim se putem instalira u izvornome obliku, a ako korisnik želi iskoristiti potencijal CUDA u suradnji s *OpenCV*, mora instalirati *OpenCV* s pomoću ranije opisane metode koristeći *CMake*. Detaljan vodič je dostupan na [27]. U kloniranom programskome kôdu otvara se datoteka *k_fold_edition.py* koja sadrži vlastito rješenje. Unutar datoteke postoji 17 linija u kojima se referencira putanja do direktorija gdje su spremljene sve datoteke koje se pozivaju u projektu i svaka počinje s *'C:/Users/SW6/Desktop/diplomski/'*. Važno je prilagoditi svaku instancu putanje novoj lokaciji gdje se nalazi klonirani projekt na novom

računalu. Ako se želi iskoristiti potencijal Nvidia grafičke kartice, u početnome dijelu, gdje su pozivane uvezene biblioteke, potrebno je izmijeniti dvije putanje za rad s CUDA bibliotekom na direktorije gdje je instalirana CUDA i cuDNN na lokalnome računalu. Ako korisnik nema Nvidia grafičku karticu, kompletan se proces izvodi u okviru procesora pa se ove dvije linije mogu obrisati. U slučaju da korisnik prvi puta preuzima resurse NLTK, mora zakomentirati i liniju koda broj 46 *next(wordnet.words())* jer se ne može pozvati objekt ako nisu dostupni resursi koji ga čine. Svakim sljedećim pokretanjem datoteke resursi su lokalno spremljeni, tako da se za tu liniju nakon preuzimanja NLTK resursa može maknuti komentar. Sama datoteka pokreće se pozivanjem *py k_fold_edition.py*, a prije prvog pokretanja preporučljivo je ponovno pokrenuti računalo kako bi se spremile sve promjene.

4. EVALUACIJA RJEŠENJA

Ranije je za potrebe implementacije metode pretrage rešetke korištena nasumična podjela skupa podataka po postotcima, gdje su prema omjerima 75% - 15% - 15% raspodijeljeni bili redom trening, validacijski, pa test skupovi. Kako bi se dodatno pripazilo na problem prekomjernog prilagođavanja, a i kako bi se imao bolji uvid u rad modela u različitim uvjetima za treniranje, koristi se metoda stratificirane unakrsne validacije k nabora (engl. Stratified k-fold cross validation). Standardna unakrsna validacija nasumično dijeli skup podataka u k nabora jednake veličine, dok stratificirana čuva izvornu distribuciju klasa u svakom naboru, osiguravajući očuvanje proporcija između klasa tako da svaki nabor ima približno isti postotak uzoraka iz svake klase kao izvorni skup podataka [28]. Model se trenira k puta, svaki put koristeći $k-1$ nabora za trening, a preostali nabor koristi za test. Odabrano je deset nabora za trening kroz 100 epoha, što znači da se u svakome naboru 90% podataka koristi za trening, a od toga 10% se pridružuje validacijskom setu, što zapravo znači 81% za trening, 9% za validaciju, a preostalih 10% preostaje za testiranje. Ovaj omjer raspodjele test/trening/validacijskih skupova je konzistentan kroz svih deset nabora, ali se svaki puta koriste drugačiji podatci za trening, validaciju i testiranje. Svi korišteni hiperparametri za realizaciju rješenja su:

Ulazna dimenzija	**Skrivena dimenzija**	**Izlazna dimenzija**	**Dropout**
broj čvorova	128	6	0.5

Tablica 4.1: *Hiperparametri GraphSAGE modela*

Stopa učenja	**Optimizator**	**Broj epoha**	**Prag ranog zaustavljanja**	**Broj nabora**
0.01	Adam	100	5	10

Tablica 4.2: *Preostali hiperparametri*

Kriterij za računanje gubitaka je križni gubitak entropije (engl. Cross entropy loss), koji je korišten i u ranijim primjerima treninga. Postoje dvije vrste unakrsnog gubitka entropije, a to su binarna i višerazredna. S obzirom na to da je problem klasifikacije definiran sa šest klasa, neophodno je koristiti višerazredni tip. Ideja je predvidjeti vjerojatnost pripadnosti uzorka određenoj klasi. Ako se zna da svaki uzorak može pripadati samo određenoj klasi, prava vjerojatnost bi bila 1 za određenu klasu i 0 za druge klase. Unakrsna entropija mjeri razliku između predviđene i stvarne vjerojatnosti [29]. Gubitak unakrsne entropije se računa s pomoću formule:

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}(y_{i,j}.log(p_{i,j}), \tag{4-4}$$

gdje je C broj klasa, $y_{i,j}$ prave oznake za klasu j na primjeru i, $p_{i,j}$ predviđena vjerojatnost klase j za primjer i, a N broj primjeraka. Za svaki od deset nabora model se trenira kroz 100 epoha i radi validacija usput spremajući sve relevantne parametre. Za svaki od nabora se iscrta matricu zabune za koju je posebno definirana jednostavna funkcija. Ovaj se proces ponavlja dok se ne prođe svih deset nabora. Promatraju se prvo matrice zabune za pojedinačne nabore. Matrica zabune je tablični prikaz kojim se vrednuju performanse nekoga modela, a sastoji se od četiri parametra:

- TP (engl. True positives) - ukupan broj predikcija koje je model točno pretpostavio (primjer model klasificira dokument kao H01, a on zapravo je H01 to je TP za H01 klasu),

- FP (engl. False positives) - ukupan broj predikcija gdje je model pretpostavio neku klasu, ali dokument ne pripada toj klasi (primjer model klasificira dokument kao pripadnika H01 klase, a on pripada H02 to je FP za H01 klasu),
- TN (engl. True negatives) - ukupan broj predikcija gdje je model pretpostavio da dokument ne pripada nekoj klasi, a on zapravo pripada toj klasi (primjer dokument je H02 klase, a klasificiran kao da nije H01, to je TN za H01 klasu),
- FN (engl. False negatives) - ukupan broj predikcija gdje model pretpostavlja da dokument pripada nekoj klasi, ali joj on ne pripada (primjer model klasificira dokument H01 klase kao H02 to je FN za H01 klasu).

Četiri ključne metrike za vrednovanje klasifikacije uz matricu zabune su uzete u obzir[30]:

- točnost (engl. Accuracy) - postotak točnih predikcija,
- preciznost (engl. Precision) - dio pravih pozitivnih predviđanja među svim pozitivnim predviđanjima,
- odziv (engl. Recall) - udio pravih pozitivnih predviđanja među svim stvarnim pozitivnim slučajevima,
- F1 rezultat - uravnotežuje odziv i preciznost, kažnjava ekstremno negativne vrijednosti.

Poznavajući parametre koji definiraju promatrane metrike, metrike se mogu izračunati na sljedeći način:

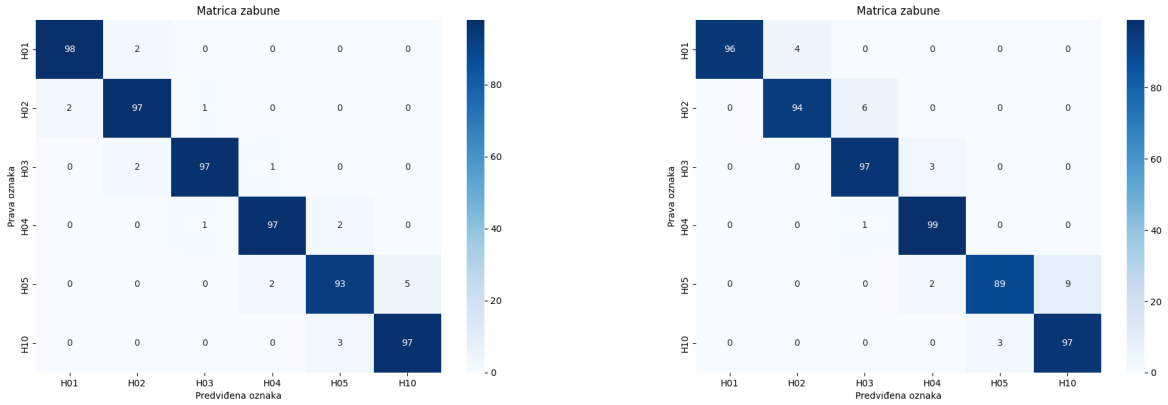
$$\text{točnost} = \frac{TP + TN}{TP + FN + TN + FP}, \tag{4-5}$$

$$preciznost = \frac{TP}{TP + FP}, \tag{4-6}$$

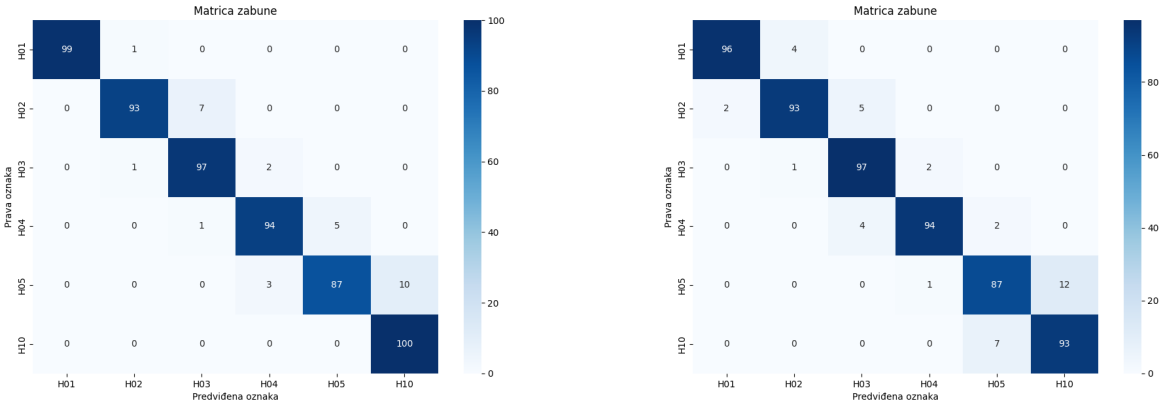
$$odziv = \frac{TP}{TP + FN}, \tag{4-7}$$

$$F1 = \frac{2 \times preciznost \times odziv}{preciznost \times odziv}. \tag{4-8}$$

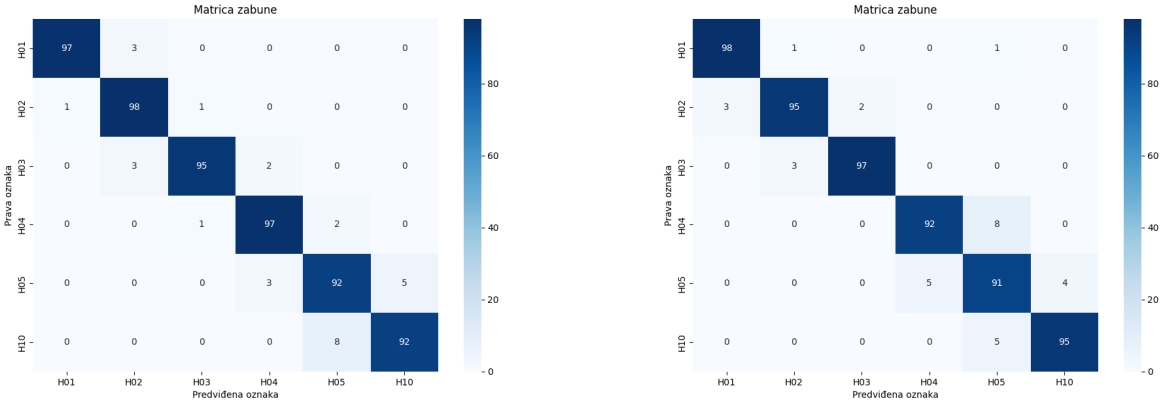
Analiziraju se sljedeće matrice zabune za 10 nabora (slika 4.1, slika 4.2, slika 4.3, slika 4.4, slika 4.5):



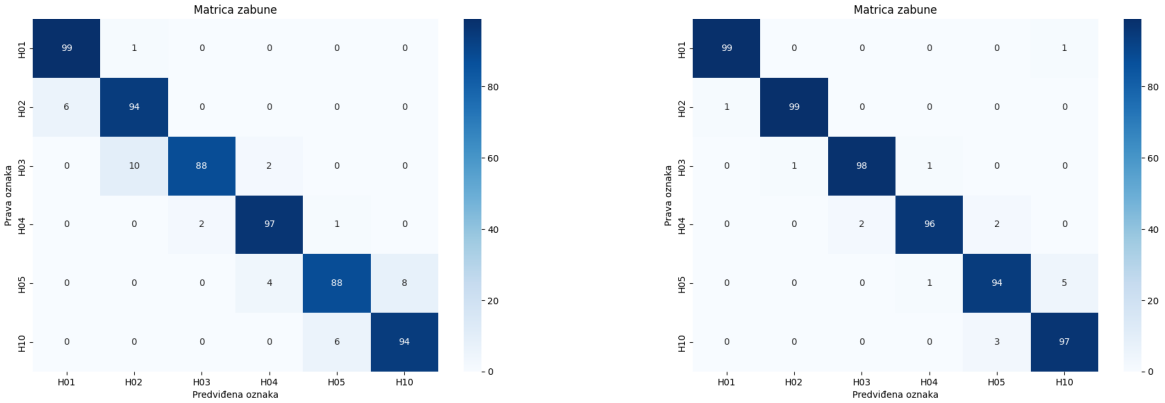
Slika 4.1: *Matrice zabune za prvi i drugi nabor*



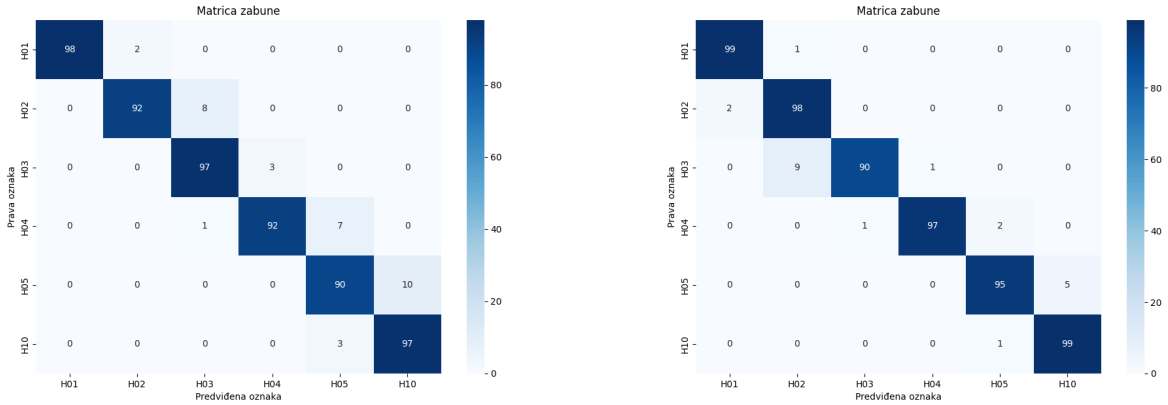
Slika 4.2: *Matrice zabune za treći i četvrti nabor*



Slika 4.3: *Matrice zabune za peti i šesti nabor*

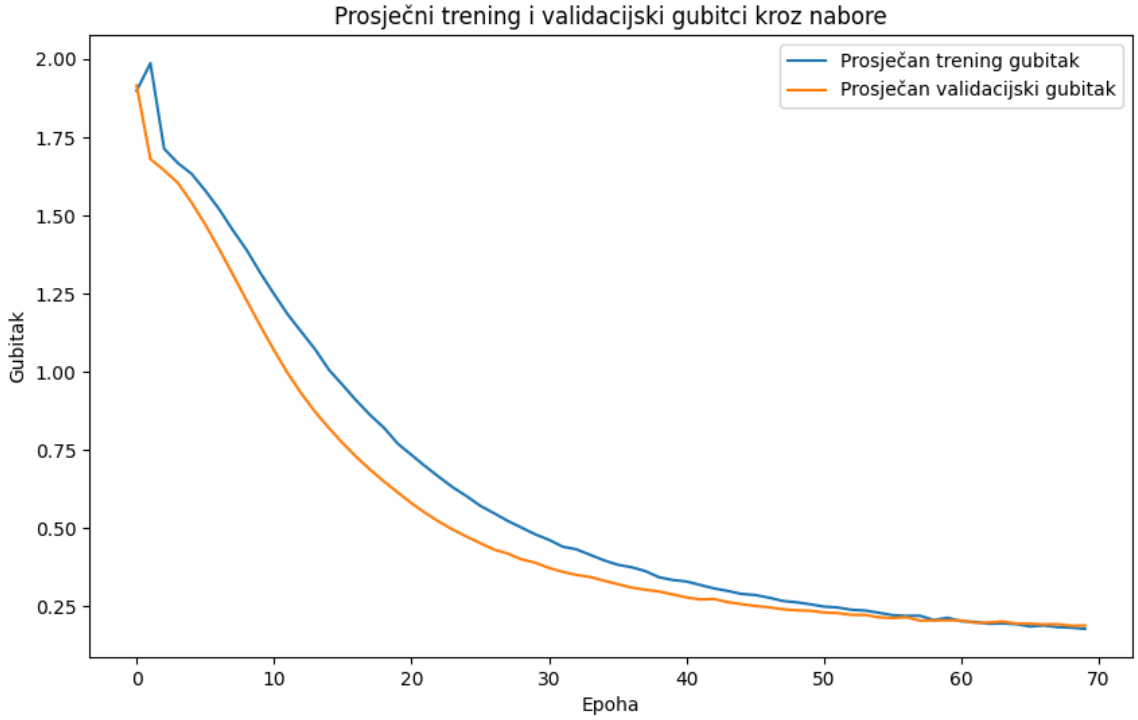


Slika 4.4: *Matrice zabune za sedmi i osmi nabor*

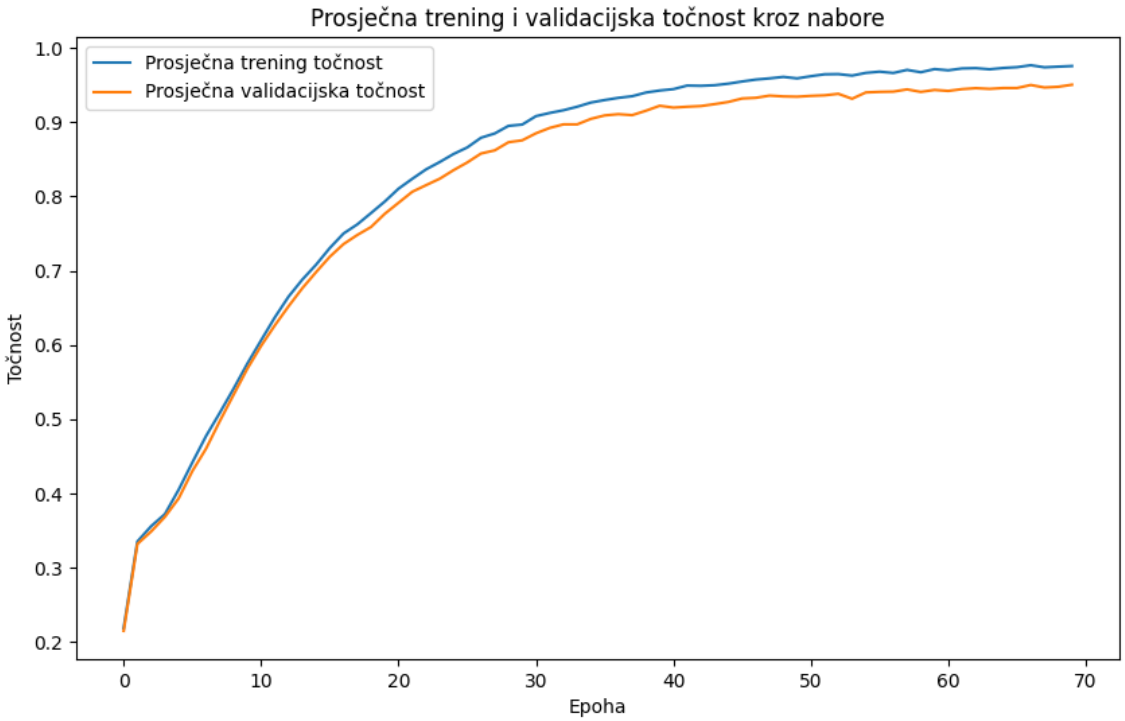


Slika 4.5: *Matrice zabune za deveti i deseti nabor*

Primjećuje se da model izvrsno klasificira na testnome skupu podataka u svakom od deset nabora, neovisno o raspodjeli podataka unutar skupa i neovisno o klasi. Ova se tvrdnja dodatno potvrđuje analizom sljedećih grafova i tablice. Prvo se promatraju rezultati prosječnih trening i validacijskih gubitaka kroz deset nabora. Prolazeći kroz epohe opadaju vrijednosti gubitaka i sve su bliže jedna drugoj što ukazuje da model u procesu treniranja dobro uči na podatcima i pravilno uočava uzorke, a da ne dolazi do prekomjernog prilagođavanja (slika 4.6). Validacijski gubitak je na početku manji od trening gubitka, a razlog tomu je što se *Dropout* primjenjuje tijekom obuke, ali ne i tijekom validacije/testiranja. Također, trening gubitak se kontinuirano računa tijekom cijele epohe, međutim, validacijske metrike se računaju tek nakon što se završi trenutna epoha treninga što znači da u prosjeku se trening gubici mjere pola epohe ranije nego validacijski [31]. Drugi graf (slika 4.7) ukazuje na konstantan rast prosječne validacijske točnosti kroz epohe implicirajući da su predikcije na neviđenom dijelu podataka sve preciznije. U slučaju da je došlo da stagnacije rasta ili pada validacijske točnosti to bi ukazivalo da se model prekomjerno prilagođava, a da je konstantno niska vrijednost validacijske točnosti moralo bi se dodatno poraditi i bolje odabrati hiperparametre. Model je treniran nad trening podatcima, pa se očekuje bolja izvedba trening točnosti što je i ostvareno. Posljednja tablica 4.3 predstavlja ranije spomenute metrike evaluacije modela na testnome skupu podataka, a to su točnost, preciznost, odziv i F1. Sve četiri metrike su u rasponu 0.9525 ± 0.0175. S obzirom na kompleksnost zadatka i baze s kojom se radi, kao i procesa predobrade ovo su odlični rezultati. Malo odstupanje od svega 1.75% u odnosu na polaznih 95.25% ukazuju na dobar balans te je model konzistentan i pouzdan a pri tome dobro generalizira neovisno o podatcima. Sve su metrike jako sličnih rezultata a to znači da konačni model dobro rukuje s različitim tipovima pogrešaka (FP i FN).



Slika 4.6: *Prosječni trening i validacijski gubitci*



Slika 4.7: *Prosječna validacijska točnost*

Broj nabora	Test točnost	Test preciznost	Test F1	Test odziv
1	0.965	0.965	0.9645	0.965
2	0.9533	0.9544	0.9532	0.9533
3	0.95	0.9512	0.95	0.95
4	0.9333	0.9341	0.9334	0.9333
5	0.9517	0.9521	0.9517	0.9517
6	0.9467	0.9474	0.9469	0.9467
7	0.9333	0.9343	0.933	0.9333
8	0.9717	0.9718	0.9717	0.9717
9	0.9433	0.9448	0.9435	0.9433
10	0.9633	0.9646	0.9633	0.9633

Tablica 4.3: *Rezultati metrika evaluacije modela na test skupu podataka*

5. ZAKLJUČAK

Tehnička je dokumentacija sačinjena od velikog broja značajki koje je važno pomno analizirati kako bi se dokumentacija mogla pravilno raspodijeliti unutar odgovarajuće kategorije. Kategorija je mnogo, a sam je proces kategorizacije skup i zahtjevan. Stoga je važno pravilno obraditi takve tipove dokumenata i pokušati automatizirati proces klasifikacije dokumenata. U ovom je radu predložen model koji može precizno klasificirati tehničku dokumentaciju. Kako bi se model realizirao, najprije je kreirana baza podataka koja sadrži 6 000 unikatnih primjeraka za H odjeljak IPC klasifikacije. Svaki je od dokumenata iz baze podataka predobrađen kako bi se mogle izvući najrelevantnije informacije korištene za klasifikaciju. Informacije korištene za klasifikaciju ključne su te uključuju tekstualne i slikovne značajke. Tekstualne su značajke naslov i sažetak svakog dokumenta, dok su slikovne elektronička shema ili dijagram tijeka, ovisno o tipu dokumenta. Nakon dobro definirane predobrade, započinje obrada dobivenih podataka. U procesu obrade podataka korišteni su različiti pomoćni modeli. Pomoćni model YOLOv5 korišten je za klasifikaciju slikovnih prikaza, odnosno za klasifikaciju električnih shema i dijagrama tijeka. Ako je slikovni prikaz električna shema, bit će obrađen od strane VGG19 modela, a ako je slikovni prikaz dijagram tijeka, iz njega će biti izvučen sav tekstualni sadržaj i spojen s ranije dobivenim sažetkom i naslovom dokumenta. HAN model korišten je za obradu konačnih tekstualnih značajki dokumenta. Model dubokog učenja GraphSAGE korišten je za klasifikaciju samih dokumenata, a nužno je odabrati optimalne hiperparametre što rezultira preciznom klasifikacijom IPC odjeljaka za dostupne dokumente. Ovaj projekt ukazao je da GraphSAGE uspješno klasificira tehničke dokumente zahvaljujući pažljivom upravljanju graf strukturama i dinamičkom ažuriranju tijekom zaključivanja, pri čemu je model postigao dosljedne i pouzdane rezultate. Promišljenim odabirom hiperparametara i treniranjem modela GraphSAGE pokazana je učinkovitost primjene dubokog učenja za klasifikaciju tehničke dokumentacije iskorištavanjem odnosa među dokumentima, čime su potvrđene ranije postavljene teze i ideje prezentirane u radovima slične tematike. U procesu evaluacije rješenja pokazano je da model, promatrajući više različitih metrika vrednovanja klasifikacije, konzistentno daje odlične i precizne rezultate.

LITERATURA

[1] Nala Yehe. *Automatic Patent Classification*. School of Engineering, Jönköping University, 2020.

[2] Ralf Krestel, Julian Risch. *Domain-Specific Word Embeddings for Patent Classification*. University of Potsdam, 2019.

[3] Subhayu Dutta, Subhrangshu Adhikary, Ashutosh Dhar Dwivedi. *VisFormers — Combining Vision and Transformers for Enhanced Complex Document Classification*. Machine Learning and Knowledge Extraction, 2024.

[4] Md Shajalal, Sebastian Denef, Md. Rezaul Karim, Alexander Boden, Gunnar Stevens. *UNVEILING BLACK-BOXES: EXPLAINABLE DEEP LEARNING MODELS FOR PATENT CLASSIFICATION*. Springer, 2023.

[5] Jie Hu, Christopher L. Magee, Jianxi Luo Shuo Jiang. *Deep Learning for Technical Document Classification*. IEEE Transactions on Engineering Management, 2019.

[6] Popis te deskripcija dostupnih python paketa. https://pypi.org/. Zadnji pristup: 28.7.2024.

[7] Definicija IPC. https://ec.europa.eu/eurostat/statistics-explained/index.phptitle=Glossary:International_patent_classification_(IPC). Zadnji pristup: 28.7.2024.

[8] Struktura IPC. https://www.wipo.int/classifications/ipc/en/. Zadnji pristup: 28.7.2024.

[9] Espacenet baza tehničke dokumentacije. https://worldwide.espacenet.com/. Zadnji pristup: 28.7.2024.

[10] Kaggle baza podataka za strojno učenje. https://www.kaggle.com/. Zadnji pristup: 28.7.2024.

[11] Patentscope pretraživanje tehničke dokumentacije. https://patentscope.wipo.int/search/en/search.jsf. Zadnji pristup: 28.7.2024.

[12] USPTO baza podataka tehničke dokumentacije. https://developer.uspto.gov/product/patent-application-multi-page-pdf-images. Zadnji pristup: 28.7.2024.

[13] YOLO v5 arhitektura. https://docs.ultralytics.com/models/yolov5/. Zadnji pristup: 28.7.2024.

[14] Roboflow aplikacija. https://roboflow.com/. Zadnji pristup: 28.7.2024.

[15] Baza podataka za klasifikaciju slika. https://universe.roboflow.com/carlota-gutirrez/circuits-bvn3j/. Zadnji pristup: 28.7.2024.

[16] YOLO v5 model za klasifikaciju slika. https://universe.roboflow.com/electronics-vzpdq/electronics-classification/dataset/3. Zadnji pristup: 28.7.2024.

[17] Paul Guerrie, Trevor Lynn - How to Train YOLOv5-Classification on a Custom Dataset. https://blog.roboflow.com/train-yolov5-classification-custom-data/. Zadnji pristup: 28.7.2024.

[18] Karen Simonyan, Andrew Zisserman. *Very deep convolutional networks for large-scale image recognition*. Visual Geometry Group, Department of Engineering Science, University of Oxford, 2015.

[19] Atulanand - VGGNet Complete Architecture. https://medium.com/codex/vggnet-complete-architecture-5c6fa801502b. Zadnji pristup: 28.7.2024.

[20] Principal component analysis(PCA). https://www.geeksforgeeks.org/principal-component-analysis-pca/. Zadnji pristup: 28.7.2024.

[21] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, Eduard Hovy. *Hierarchical Attention Networks for Document Classification*. Carnegie Mellon University, Microsoft Research, Redmond, 2016.

[22] Varun - Cosine similarity: How does it measure the similarity, Maths behind and usage in Python. https://towardsdatascience.com/cosine-similarity-how-does-it-measure-the-similarity-maths-behind-and-usage-in-python-50ad30aad7db. Zadnji pristup: 28.7.2024.

[23] What is Adam Optimizer? https://www.geeksforgeeks.org/adam-optimizer/. Zadnji pristup: 28.7.2024.

[24] Kompletno projektno rješenje diplomskog rada. https://github.com/kokanovick/diplomski. Zadnji pristup: 9.8.2024.

[25] Microsoft c++ build tools. https://visualstudio.microsoft.com/visual-cpp-build-tools/. Zadnji pristup: 9.8.2024.

[26] Tesseract OCR. https://digi.bib.uni-mannheim.de/tesseract/. Zadnji pristup: 9.8.2024.

[27] Instalacija OpenCV koristeći CUDA. https://github.com/chrismeunier/OpenCV-CUDA-installation. Zadnji pristup: 9.8.2024.

[28] Juan C Olamendy - A Comprehensive Guide to Stratified K-Fold Cross-validation for Unbalanced Data. https://medium.com/@juanc.olamendy/a-comprehensive-guide-to-stratified-k-fold-cross-validation-for-unbalanced-data-014691060f17. Zadnji pristup: 28.7.2024.

[29] What Is Cross-Entropy Loss Function? https://www.geeksforgeeks.org/what-is-cross-entropy-loss-function/. Zadnji pristup: 28.7.2024.

[30] Stephen M. Walker II - F-Score: What are Accuracy, Precision, Recall, and F1 Score? https://klu.ai/glossary/accuracy-precision-recall-f1. Zadnji pristup: 28.7.2024.

[31] Why is my val loss lower than train loss? https://pyimagesearch.com/2019/10/14/why-is-my-validation-loss-lower-than-my-training-loss/. Zadnji pristup: 30.8.2024.

SAŽETAK

U sklopu ovog rada razvijen je model duboke neuronske mreže za klasifikaciju tehničke dokumentacije. Tehnička dokumentacija, za čiju je klasifikaciju model razvijen, se odnosi na dokumente odjeljka elektricitet unutar IPC kategorizacije. Za potrebe treniranja najprije je prikupljena baza podataka od 6000 unikatnih dokumenata iz šest različitih klasa unutar elektriciteta (H01, H02, H03, H04, H05 i H10). Podatci unutar svakog dokumenta izvlačeni su putem *tesseract* algoritma, pri čemu su izdvojeni i tekstualni i slikovni elementi. Od tekstualnog dijela korišteni su naslov dokumenta i njegov sažetak, a za slikovni dio korištena je shema koja se nalazi na prvoj stranici dokumenta. Te su informacije predobrađene kako bi odgovarale formatu konačnog modela. Budući da se slike unutar dokumenata razlikuju, korišten je model YOLOV5 za klasifikaciju slika na elektroničke sheme ili dijagrame tijeka. Ako je neka slika elektronička shema, na nju je primijenjen VGG19 model za rukovanje i izvlačenje relevantnih značajki. Ako je pak slika dijagram tijeka, izvlači se tekstualni sadržaj tog dijagrama i predobrađuje. Za klasifikaciju dokumenata korišten je model graf neuronskih mreža GraphSAGE. Konačni model duboke neuronske mreže postigao je značajne rezultate prilikom procesa treniranja, validacije i testiranja. Pokazano je da model ima konzistentnu predikciju uz minimalna odstupanja metrika klasifikacije te da može pravilno rukovati značajkama unutar tehničke dokumentacije.

KLJUČNE RIJEČI

Python, duboko učenje, GraphSAGE, klasifikacija, tehnička dokumentacija

USING DEEP LEARNING FOR CLASSIFICATION OF TECHNICAL DOCUMENTS

ABSTRACT

As part of this work, a deep neural network model was developed for the classification of technical documentation. For classification of technical documentation, a model was developed that refers to documents of the electricity section within the IPC categorization. For training purposes, a database of 6000 unique documents from six different classes within electricity (H01, H02, H03, H04, H05 and H10) was first collected. Data within each document was extracted using the *tesseract* algorithm, where both textual and image elements were extracted. The document title and its summary were used from the textual part, and the diagram located on the first page of the document was used for the image part. This information is pre-processed to match the format of the final model. Since the images within the documents differ, the YOLOV5 model was used to classify the images into electronic diagrams or flowcharts. If an image is an electronic schematic, the VGG19 model is applied to it to handle and extract relevant features. If the image is a flowchart, the textual content of that flowchart is extracted and preprocessed. The GraphSAGE neural network graph model was used for document classification. The final deep neural network model achieved significant results during the training, validation and testing process. The model was shown to have consistent prediction with minimal deviations in classification metrics and to be able to correctly handle features within technical documentation.

KEYWORDS

Python, deep learning, GraphSAGE, classification, technical documentation

ŽIVOTOPIS

Karlo Kokanović rođen je 2. prosinca 1999. godine u Slavonskome Brodu. Živi i odrasta u Županji gdje pohađa Osnovnu školu Mate Lovraka u Županji. Nakon završetka osnovne škole 2014. godine upisuje Gimnaziju Županja, prirodoslovno-matematički smjer. Srednju školu završava 2018. godine te upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, preddiplomski sveučilišni smjer računarstvo. 2022. godine stječe akademski naziv sveučilišni prvostupnik inženjer računarstva. Iste godine upisuje diplomski sveučilišni studij računarstva, smjer Informacijske i podatkovne znanosti na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.

PRILOZI

- P.3.1. - Cjelokupan programski kôd sa svim popratnim datotekama korištenim za izradu vlastitog rješenja - https://github.com/kokanovick/diplomski