Razvoj aplikacije za stvaranje formalnog računalnog zapisa električnih shema uz razvoj računalnog algoritma za detekciju električnih elemenata i njihovih karakteristika na slikama shema električnih krugova

Lazar, Lorena

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:168:453676

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 593

RAZVOJ APLIKACIJE ZA STVARANJE FORMALNOG RAČUNALNOG ZAPISA ELEKTRIČNIH SHEMA UZ RAZVOJ RAČUNALNOG ALGORITMA ZA DETEKCIJU ELEKTRIČNIH ELEMENATA I NJIHOVIH KARAKTERISTIKA NA SLIKAMA SHEMA ELEKTRIČNIH KRUGOVA

Lorena Lazar

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 593

RAZVOJ APLIKACIJE ZA STVARANJE FORMALNOG RAČUNALNOG ZAPISA ELEKTRIČNIH SHEMA UZ RAZVOJ RAČUNALNOG ALGORITMA ZA DETEKCIJU ELEKTRIČNIH ELEMENATA I NJIHOVIH KARAKTERISTIKA NA SLIKAMA SHEMA ELEKTRIČNIH KRUGOVA

Lorena Lazar

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 593

Pristupnica:	**Lorena Lazar (0036525112)**
Studij:	Računarstvo
Profil:	Znanost o podacima
Mentorica:	izv. prof. dr. sc. Mihaela Vranić

Zadatak: **Razvoj aplikacije za stvaranje formalnog računalnog zapisa električnih shema uz razvoj računalnog algoritma za detekciju električnih elemenata i njihovih karakteristika na slikama shema električnih krugova**

Opis zadatka:

U elektrotehnici električne sheme daju prikaz načina povezivanja i međusobnog odnosa elemenata strujnih krugova. U shematskim prikazima važnu ulogu imaju grafički simboli. Oni predstavljaju standardni oblik određenog elementa kojeg možemo detektirati na slici. Uz detekciju elemenata strujnih krugova, važne su i njihove karakteristike koje su često dane na slici električne sheme. Računalna analiza slika je proces koji iz slika izvlači neke bitne informacije u svrhu njihovog razumijevanja. Pri tome koristi različite metode umjetne inteligencije, statistike i strojnog učenja. Vaš je zadatak razviti računalni algoritam za detekciju električnih elemenata i njihovih karakteristika na slikama shema električnih krugova odnosno mreža. Na osnovu toga potrebno je formirati formalni zapis sheme koji je pogodan za izvođenje daljnjih simulacija i testiranje zadanih karakteristika mreže. Algoritam je potrebno razviti na primjerima shema koje se koriste u zadacima na kolegiju Osnove elektrotehnike prijediplomskih studija FER-a. Razvijeni algoritam potrebno je uklopiti u aplikaciju koja će omogućiti ručne primjene određenih detektiranih elemenata i njihovih karakteristika, a u konačnici će davati formalni zapis koji dalje može služiti za izvođenje simulacija, provjere zadataka ili generiranje više validnih inačica zadataka.

Rok za predaju rada: 28. lipnja 2024.

Zahvaljujem mentorici izv. prof. dr. sc. Mihaeli Vranić na pomoći i usmjeravanju tijekom izrade diplomskog rada. Zahvaljujem doc. dr. sc. Franu Škopljanac-Mačina na prikupljenim slikama shema i pomoći u području elektrotehnike.

Posebno zahvaljujem roditeljima Josipu i Biserki, sestri Luciji, bratu Matiji, ostatku obitelji i prijateljima na podršci tijekom cjelokupnog školovanja.

Sadržaj

Uvod

Električne sheme grafički prikazuju kako su električni elementi međusobno povezani u strujnom krugu, odnosno mreži. Svaki električni element prikazuje se jedinstvenim grafičkim simbolom. Analizom strujnih krugova mogu se izračunati različite karakteristike mreže. Strujne krugove moguće je tekstualno zapisati u format koji je prikladan za izvođenje različitih simulacija ili testiranje zadanih karakteristika mreže. Trenutačno niti jedan dostupan alat za izvođenje simulacija ne može kao ulaz primati sliku sheme i na temelju nje raditi željene simulacije. Simulacije je moguće izvoditi ako se shema programski nacrta u zadanom alatu ili ako je poznat formalan zapis strujnog kruga.

Cilj ovog rada je razviti računalni algoritam koristeći različite metode umjetne inteligencije, statistike i strojnog učenja kako bi se sa slike električne sheme prikupile sve potrebne informacije te bi se na temelju toga napravio formalan zapis koji se može koristiti za simuliranje ili testiranje zadanih karakteristika mreže. Motivacija proizlazi iz potrebe za automatizacijom procesa analize električnih shema što bi značajno ubrzalo i olakšalo rad inženjerima i studentima. Za izradu ovog rada, algoritam je razvijen koristeći primjere shema koje se koriste na kolegiju Osnove elektrotehnike prijediplomskih studija Fakulteta elektrotehnike i računarstva. Postupak je podijeljen u nekoliko koraka, a razvijeni algoritam je integriran u aplikaciju koja omogućuje automatizirano generiranje formalnog zapisa električnih strujnih krugova.

Rad je organiziran na sljedeći način: u prvom poglavlju dan je kratak uvod u područje istraživanja, prikazana su dosadašnja postignuća i ključne teorijske informacije bitne za daljnje razumijevanje rada. U drugom poglavlju nalazi se metodologija istraživanja, koja pruža sveobuhvatan pregled potrebnih koraka i objašnjava pristupe korištene tijekom studije. Treće poglavlje uvodi čitatelja u područje elektrotehnike te je opisan postupak stvaranja formalnog zapisa sheme, što je krajnji cilj rada. Četvrto poglavlje započinje provedbom programskog rješenja. Detaljno je opisan razvoj modela za detekciju električnih elemenata i njihovih karakteristika korištenjem algoritama YOLOv5 i YOLOv8. Također, opisan je razvoj modela za detektiranje karakteristika mreže te postupak obrade slike za prepoznavanje povezanosti elemenata u strujnom krugu te konačna izrada formalnog zapisa - netlista. U petom poglavlju opisan je razvoj aplikacije za programsku platformu iOS te

povezivanje sa serverom na kojemu se izvršavaju API pozivi. Navedene su funkcionalnosti aplikacije i prikazan je njezin rad na primjeru odabrane sheme. U šestom poglavlju prikazana je mogućnost simuliranja strujnog kruga iz dobivene netlist mreže. Sedmo poglavlje sadrži diskusiju i usporedbu s postojećim radovima. Na kraju rada, dan je zaključak.

1. Pregled postojećih pristupa

1.1. Umjetna inteligencija i strojno učenje

Umjetna inteligencija je opsežno područje koje je posljednjih godina našlo široku primjenu u raznim sektorima. Zbog svojih mogućnosti stekla je veliku popularnost i postala jedno od najvažnijih područja suvremenih istraživanja. U radu [11] umjetnu inteligenciju nazivaju sustavom koji se ponaša ili razmišlja kao čovjek.

Primjena umjetne inteligencije je veoma široka. Umjetnu inteligenciju možemo zamisliti kao dobrog igrača šaha koji nas može pobijediti u nekoliko poteza, kao prijatelja koji nas dovoljno poznaje da nam može preporučiti pjesmu, film ili knjigu, kao bankara koji može procijeniti hoćemo li vraćati kredit po dogovoru, kao taksista koji poznaje sva prometna pravila i može nas sigurno odvesti do cilja ili kao osobnog asistenta koji može odgovarati na naša pitanja i pružiti traženi savjet. Vidljivo je da se umjetna inteligencija koristi masovno u raznim aspektima svakodnevnog života.

Strojno učenje je dio umjetne inteligencije koje se temelji na učenju iz odabranog podatkovnog skupa kako bi nakon učenja moglo donositi željene odluke. Podaci su glavni temelj jer ishod odluke modela ovisi o njihovoj kvaliteti, kvantiteti i raznolikosti. Poželjno je da podaci budu što raznovrsniji i sveobuhvatniji. Strojno učenje se dijeli na tri vrste učenja: **nadzirano učenje** (engl. *supervised learning*), **nenadzirano učenje** (engl. *unsupervised learning*) i **pojačano učenje** (engl. *reinforcement learning*) [11]. Kod nadziranog učenja poznati su ulaz i izlaz modela, a cilj je da model nauči predvidjeti izlaz na temelju ulaznih podataka. Primjeri nadziranog učenja su regresija i klasifikacija. Regresija je metoda koja se koristi kada je izlazna varijabla numerička, dok se klasifikacija koristi kada je izlazna varijabla kategorička. Nenadzirano učenje nema označene izlaze, ali se podaci grupiraju u smislene cjeline na temelju sličnosti ili karakteristika. Učenje pojačanjem temelji se na učenju kroz kazne i nagrade. **Ovaj rad koristi nadzirano učenje** jer su ulaz i izlaz poznati. Ulaz je slika električne sheme, a izlaz su električni elementi i mjesta na kojima se ti elementi nalaze. Primjena strojnog učenja na probleme stvarnog svijeta zahtijeva identifikaciju potrebnih podataka i znanje o domeni. To često uključuje suradnju sa stručnjacima iz određenih područja. Oni mogu biti uključeni u predobradu podataka koja uključuje čišćenje, transformaciju i provjeru valjanosti podataka. Glavni parametri za procjenu uspjeha

algoritama učenja su točnost predviđanja, F1 – vrijednost (engl. *F1 – score*), odziv (engl. *recall*) i preciznost (engl. *precision*) [16].

Duboko učenje je područje unutar strojnog učenja koje se bavi rješavanjem složenih problema s pomoću višeslojnih neuronskih mreža. Konvolucijske neuronske mreže (engl. *Convolutional Neural Networks, CNN*) su najreprezentativnije neuronske mreže u području dubokog učenja. CNN koristi višestruke slojeve za obradu podataka s pomoću konvolucijskih struktura kako bi se izdvojile značajke. Prednost CNN-a u usporedbi s potpuno povezanom mrežom je ta da svaki neuron nije spojen sa svakim drugim neuronom, što rezultira manjim brojem parametara, bržim učenjem i konvergencijom. Također, koristi se sloj sažimanja (engl. *pooling layer*) u kojem se lokalne informacije sažimaju smanjujući količinu podataka, a da su pritom očuvanje korisne informacije [12].

Područje istraživanja koje se bavi detekcijom objekata na slici naziva se **računalni vid**. Računalni vid doživio je izuzetan rast tijekom posljednjeg desetljeća zahvaljujući internetu koji omogućuje široku dostupnost vizualnih informacija i zbog razvoja alata za njihovu obradu. U radu [16] navodi se da je glavni cilj računalnog vida omogućiti računalima sposobnosti slične ljudskima, što uključuje percepciju okoline, razumijevanje vizualnih podataka i učenje iz iskustva za poboljšanje budućih performansi. Računalni vid evoluirao je od prepoznavanja jednostavnih uzoraka i obrade slika do naprednih modela i aplikacija koje razumiju složene slike. Unatoč značajnom napretku još uvijek nije postignuto da računalo interpretira slike kao čovjek. Na primjer, ljudi s lakoćom mogu prepoznati ljudske emocije na temelju izraza lica, dok je računalu za to potreban znatan napor i često je sklono pogreškama.

Detekcija objekata predstavlja temeljnu aktivnost u području računalnog vida. Obuhvaća pronalazak lokacije određenog objekta (lokalizacija) i njegovo svrstavanje u jednu od klasa (klasifikacija) [13]. **Klasifikacija** slika je proces u kojem naučen model utvrđuje koja klasa se nalazi na slici, to jest s kojom vjerojatnošću se svaka od mogućih klasa pojavljuje na slici. Klasa s najvećom vjerojatnošću jest odabrana klasa predikcije.

1.2. Pregled postojećih istraživanja vezanih uz stvaranje formalnog računalnog zapisa električnih shema

Većina istraživanja sa sličnom tematikom uključuju samo klasifikaciju električnih elemenata, dok samo mali broj istraživanja uključuje metode koje opisuju njihovu

međusobnu povezanost i položaj unutar sheme. Izazovi detekcije uključuju razlike u oblicima i veličinama shema i elemenata. Za ručno crtane sheme izazove predstavljaju kvaliteta papira i olovke, boja i urednost crtača, to jest crtačev crtački stil [3]. Ti problemi se često rješavaju digitalnom obradom slike.

A. Rad *A two-stage CNN-based hand-drawn electrical and electronic circuit component recognition system* [4]

U radu je razvijen model temeljen na konvolucijskim neuronskim mrežama te je postignuta točnost od 97,33 %. Podatkovni skup sastoji se od 20 klasa te je prikupljeno 150 slika za svaku klasu. Slike su transformirane u dimenzije 64 x 64 piksela. Crtači su bili studenti, istraživači i članovi fakulteta. Korištena je klasifikacija koja je podijeljena u dvije faze. U prvoj fazi se elementi grupiranju na temelju sličnosti u izgledu, na primjer oblik elementa. Elementi su podijeljeni u četiri grupe. Tijekom druge faze se svakom elementu dodjeljuje odgovarajuća klasa. Ovim pristupom klasifikacije u dvije faze je postignuta veća točnost, nego samo klasifikacijom bez prethodnog grupiranja.

B. Rad *Circuit recognition using netlist* [17]

Još jedan način za određivanje sadržaja slike je promatranje oznaka pored električnih elemenata i korištenje modela za optičko prepoznavanje znakova (engl. *Optical character recognition,* OCR). Na primjer, ako se na slici detektira oznaka slova R što je oznaka za otpornik (engl. *resistor*), element pored te oznake klasificira se kao otpornik. Također, mogu se promatrati mjerne jedinice pored elemenata pa ako se uoči slovo H što je mjerna jedinica za Henry, element vjerojatno pripada klasi zavojnice. Prednost ovog pristupa je razvijeno područje prepoznavanja teksta. Euklidska udaljenost korištena je za dodjeljivanje slova pripadajućim elementima. Gotovo svi elementi su točno klasificirani. Jedini problem predstavljao je element uzemljenja koji općenito nema nikakvu oznaku, no to je riješeno dodavanjem slova G (engl. *grounding*) uz element. Slike korištene u radu mogu prepoznati samo RLC (engl. *Resistor, Inductor, Capacitor*) krugove s istosmjernim napajanjem. Sheme su crtane na urednom bijelom papiru. Analiza povezanost komponenti ostvarena je praćenjem susjednih piksela tako da su susjedni pikseli označeni isto oznakom. Stvaranje formalnog zapisa sheme postiže točnost od 80 %.

C. Rad *Hand-Drawn Electrical Circuit Recognition using Object Detection and Node Recognition* [13]

U radu je razvijen algoritam koji prepoznaje ručno crtane elemente strujnog kruga i stvara formalan zapis. Podatkovni skup se sastoji od šest mogućih klasa i sadrži 154 slike koje je crtalo pet različitih osoba. Za detekciju električnih elemenata korišteni su algoritmi YOLOv3, YOLOv5 i SSD300. Pretpostavka je da su slike shema prikazane bez ikakvih oznaka komponenti ili mjernih jedinica. Detektiranje elemenata provedeno je bez prethodne predobrade slika, no za detekciju krajnjih točaka elemenata i čvorova slike su pretvorene u binarne. Evaluacija je provedena koristeći mjeru mAP (engl. *mean Average Precision*), koja je opisana u trećem poglavlju ovoga rada. Svi algoritmi postigli su visoku točnost, no YOLOv5 se istaknuo kao najbolji s postignutom točnošću od 99,1 % i gotovo dvostruko većom brzinom u odnosu na YOLOv3. Za detekciju vodova prvotno se uklanjaju detektirani elementi, a nakon toga se primjenjuje Houghova transformacija. Houghova transformacija koristi se za pronalazak vertikalnih i horizontalnih linija na slici. Linije koje imaju nagib između 45 i 135 stupnjeva detektira kao vertikalne, dok one s nagibom manjim od 45 i većim od 135 stupnjeva detektira kao horizontalne linije. Sjecišta ovih linija predstavljaju čvorove.

D. Rad *Circuit Component Detection in Offline Handdrawn Electrical/Electronic Circuit Diagram* [3]

U radu prvo sliku pretvaraju u sivu sliku, a zatim u binarnu sliku. Također, provedeno je uklanjanje šuma. Razvijen model ne klasificira elemente, nego samo traži njihove lokacije na temelju mrlja koje su ostale nakon uklanjanja vodova algoritmom RLSA. Točnost lokalizacije je 91,28 % na podatkovnom skupu od 60 slika shema s ukupno 172 komponente razvrstane u 14 različitih klasa

E. Rad *Machine recognition of hand-drawn circuit diagrams* [5]

U radu je provedena klasifikacija s pomoću kombinacije invarijantnih momenata i skalarne distribucije piksela. Za praćenje povezanosti se koristi stog za pohranu piksela te se tako detektiraju linije. Podatkovni skup sadrži slike skeniranih shema koje je napravila jedna osoba. Slike su bile prethodno obrađene tako da su pretvorene u binarne te je uklonjen šum tehnikom sol papar. Postignuta je točnost prepoznavanja čvorova od 92 % i točnost prepoznavanja komponenti od 86 %.

F. Rad *Handwritten electric circuit diagram recognition: An approach based on finite state machine* [21]

Model u radu razvijen je s pomoću algoritma stroja potpornih vektora (engl. *Support Vector Machine*, SVM). Prikupljeno je 1000 ručno crtanih shema koje su ručno izrezane iz shema i

razvrstane u devet klasa. Nije stvoren formalan zapis sheme, nego se prepoznaje vrsta sheme (npr. serijski, paralelni ili RLC krug). Detekcija komponenti ima točnost od 99 %, a točnost prepoznavanja vrste sheme iznosi 85 %.

G. Rad *Automated Netlist generation from offline hand-drawn circuit diagrams* [22]

Glavni fokus rada je razumijevanje električnih krugova, to jest ne samo lokalizacija i klasifikacija, nego i prepoznavanje oznaka elemenata i njihova međusobna povezanost. Izrada je podijeljena u sljedeće korake: detekcije objekta, identificiranje čvorova, grupiranje tekstualnih oznaka, grupiranje tekstualnih oznaka s električnim elementima i generiranje formalnog zapisa sheme. Podatkovni skup sastoji se od preko 1000 slika shema. Na slikama je moguće identificirati 23 različite klase od kojih su 5 njih grafički simboli za električne elemente, a preostalo su mjerne jedinice i brojevi. Na slike nisu primijenjene metode za obradu slike kako se ne bi izgubile informacije. Dodjeljivanje tekstualnih oznaka elementu provedeno je s pomoću Euklidske udaljenosti i provjerom mjerne jedinice. Na testnom skupu je ostvarena točnost od 87 %, a točnost stvaranja formalnog zapisa je 82 %.

H. Rad *Generation of Netlist from a Hand drawn Circuit through Image Processing and Machine Learning* [23]

U radu se koristi HOG (engl. *Histogram of Oriented Gradients*) za izdvajanje značajki te se zatim koristi algoritam stroja potpornih vektora za klasifikaciju elemenata. Početno je na slikama uklonjen šum s pomoću Gaussovog zamućivanja. Slike su pretvorene u binarne te je provedena dilatacija, to jest podebljavanje linija. Linije su zatim svedene na debljinu jednog piksela procesom skeletizacije. Rezultat rada je .ASC datoteka za izvođenje simulacija u alatu LTspice. Točnost nije navedena.

I. Rad *Segmentation and recognition of electronic components in hand-drawn circuit diagrams* [26]

Ovaj rad također koristi HOG za ekstrakciju značajki, a stroj potpornih vektora za klasifikaciju. Slike su binarizirane i uklonjen je šum. Slike za klasifikaciju su izrezane iz slika shema i podijeljene u 10 klasa. U skupu za učenje se nalazi 20 slika za svaku komponentu, a u skupu za testiranje 15 slika za svaku komponentu. Točnost segmentacije testirana je na cijelom skupu i iznosi 87.7 %, a točnost klasifikacije iznosi 92 %.

J. Rad *KNN based hand drawn electrical circuit recognition* [24]

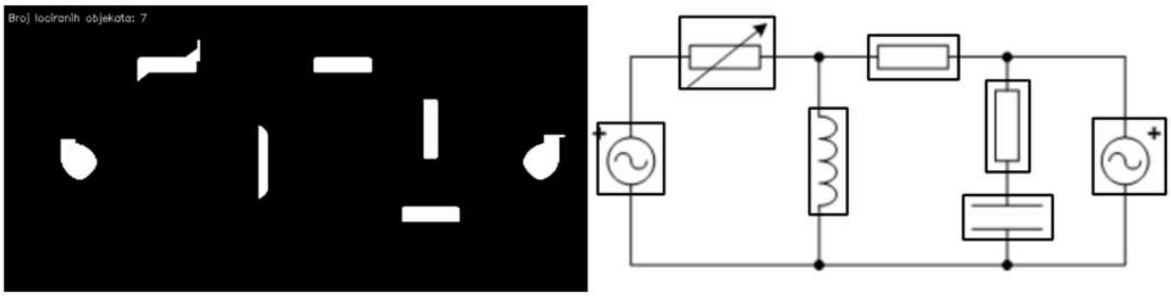
Algoritam k-najbližih susjeda (engl. *K-Nearest Neighbour*) korišten je u ovom radu. Proveden je postupak predobrade slike binarizacijom i uklanjanjem šuma. Prikupljeno je 100 slika koje su crtane na praznom papiru i skenirane. Elemente je moguće klasificirati u deset različitih klasa. Točnost na skupu za učenje iznosi 90 %.

K. Rad *Hand drawn optical circuit recognition. Procedia Computer Science* [25]

Prvi korak u radu je ekstrakcija značajki na temelju oblika, a drugi korak je klasifikacija s pomoću umjetne neuronske mreže (engl. *Artificial Neural Network*, ANN) kroz algoritam povratne propagacije. Podatkovni skup sadrži 31 klasu od koje su 10 brojeva, 5 slova, 9 mjerne jedinice i 7 simbola. Za učenje svaka klasa ima 55 slika, a za testiranje 20. F1-vrijednost iznosi 83,64 %.

L. Rad: *Računalna detekcija elemenata strujnog kruga na električnim shemama* [27]

U sklopu završnog rada s pomoću raznih metoda obrade slike pronađene su lokacije elemenata na slikama električnih shema, a uz pomoć algoritma stroj potpornih vektora izgrađen je model koji klasificira pronađene elemente. Postignuta je točnost klasifikacije od 82 %. Detekcija je provedena u dva zasebna koraka te se ovaj pristup neće koristiti za izradu ovog rada. Slika električne sheme se obrađuje tako da se početno detektiraju područja gdje se nalaze elementi s pomoću dilatacije i erozije. Na slici se na kraju nalaze bijele mrlje koje označuju mjesta gdje se elementi nalaze što je vidljivo na slici (Slika 1.1) s lijeve strane. S desne strane na slici (Slika 1.1) označena su mjesta koja se izrezuju iz slike te se šalju modelu na klasifikaciju, to jest predviđanje. Zatim se na tim mjestima uz svaki element ispisuje klasa, to jest naziv elementa koji mu je model dodijelio. Klasifikacija je ostvarena za 5 elemenata: otpornik, zavojnica, kondenzator te izvor izmjeničnog i istosmjernog napona.



Slika 1.1 Lijeva slika prikazuje shemu nad kojom su napravljeni postupci erozije i dilatacije, a desna slika prikazuje locirane objekte pomoću lijeve slike

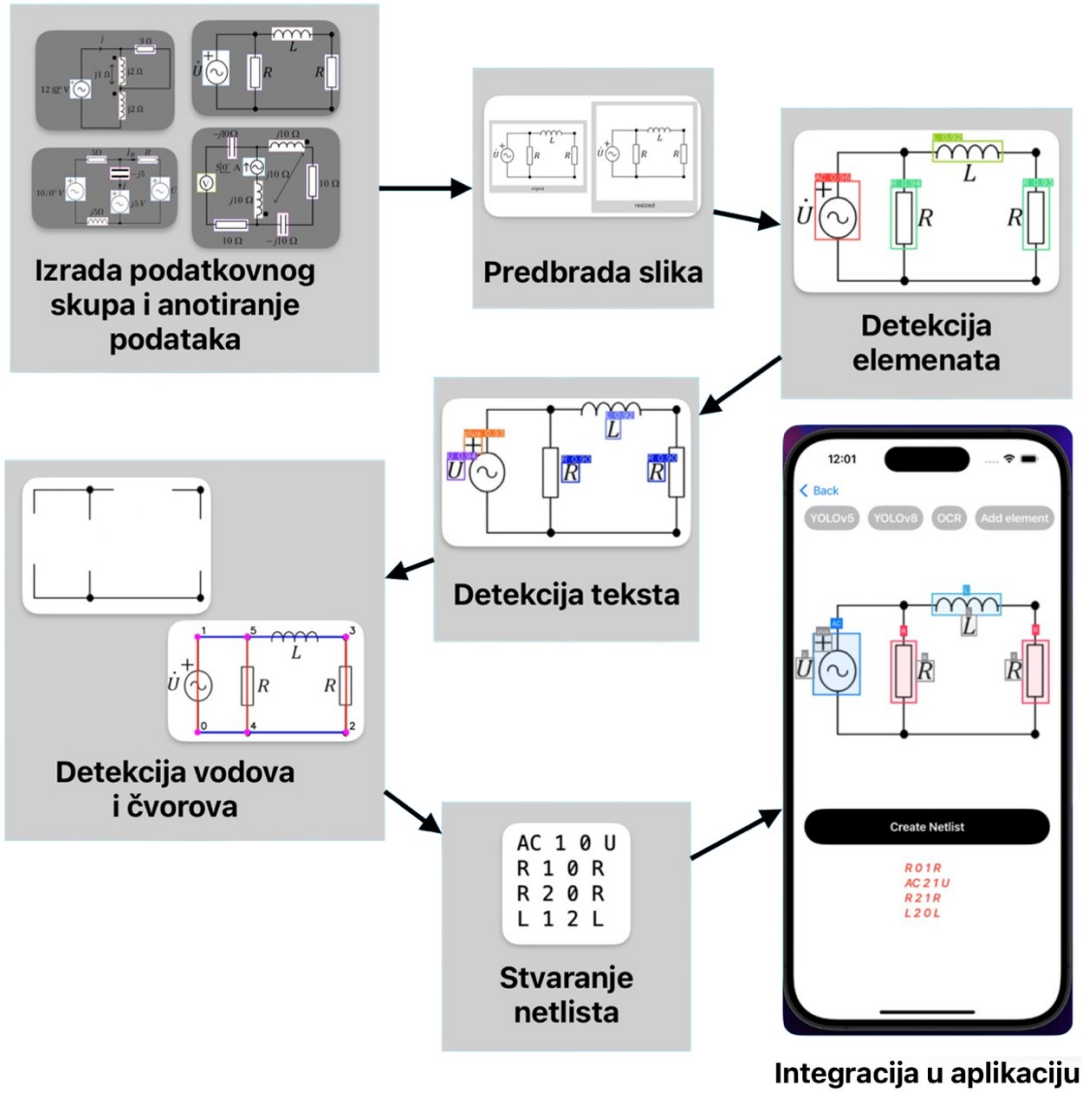
2. Metodologija rada

Kako bi se ostvarili postavljeni ciljevi, istraživanje i sam rad podijeljeni su u nekoliko cjelina:

- razvoj računalnih modela
    - izrada podatkovnog skupa i anotiranje podataka
    - predobrada podataka
    - učenje modela za detekciju elemenata i karakteristika
    - validacija razvijenih modela
    - detekcija vodova i čvorova
    - razvoj algoritma za stvaranje netlista
- razvoj aplikacije
    - izrada korisničkog sučelja
    - razvoj i povezivanje s poslužiteljem
- simuliranje u programskom alatu LTspice

U sklopu izrade diplomskog rada napravljen je vlastiti podatkovni skup slika shema strujnih krugova prikupljenih s kolegija Osnove Elektrotehnike na FER-u. Napravljen je postupak anotiranja koji obuhvaća označavanje lokacije elementa i dodjeljivanje pripadajućoj klasi. Slike su pretvorene u sive slike i sve su svedene na iste veličine. Za detekciju objekata korišteni su postojeći algoritmi YOLOv5 i YOLOv8 iz obitelji YOLO algoritama. Razvijeni modeli koriste se za detekciju električnih elemenata i karakteristika strujnog kruga. Detektirane karakteristike se grupiraju ako se nalaze dovoljno blizu te se time iz pojedinačnih slova ili brojeva dobivaju smislene cjeline. Svakom elementu se dodjeljuju najbliža cjelina. Blizina se računa s pomoću Euklidske udaljenosti. Locirana područja uklonjena su iz slike sheme kako bi bili prikazani samo vodovi. Nakon toga Houghovom transformacijom, po uzoru na rad [13], detektirani su vodovi. Čvorovi se računaju kao sjecišta linija ili kao mjesta između dva elementa na istom vodu. Na temelju toga formiran je formalni zapis sheme koji se naziva netlist. Razvijeni algoritam integriran je u aplikaciju u kojoj je moguće ručno promijeniti detektirane elemenata i njihove karakteristike. Moguće je dodati, urediti ili obrisati oznaku na slici sheme. Korisnikove promijenjene se uzimaju u obzir tijekom stvaranja netlista. Konačni rezultat je formalni zapis koji dalje može služiti za izvođenje simulacija, testiranje zadanih karakteristika mreže, provjeru zadataka ili

generiranje više validnih inačica zadataka. Postupak izrade podijeljen u korake zorno je prikazan na slici (Slika 2.1). Na slici su u gornjem lijevom kutu vidljive neke od shema korištene u ovom radu. Također, vidljivo je kako je slika obrađena u drugom koraku. Treći i četvrti korak prikazuju uspješnost detekcije elemenata i karakteristika na odabranoj shemi. U idućem koraku je prikazan izgled sheme nakon uklanjanja detektiranih objekata, a na shemi pored su označene vodoravne linije (plava boja), vertikalne linije (crvena boja) i čvorovi (ružičasta boja). Prikazan je dobiveni netlist za odabranu shemu i na kraju je prikazana aplikacija u kojoj se klikom na gumb „*Create Netlist*“ izvršavaju svi navedeni koraci.



Slika 2.1 Prikaz potrebnih koraka za izradu formalnog zapisa – netlista

U radu je dodatno testirana mogućnost simuliranja shema na temelju dobivenih netlista u alatu LTSpice.

3. Električne sheme i njihovi zapisi

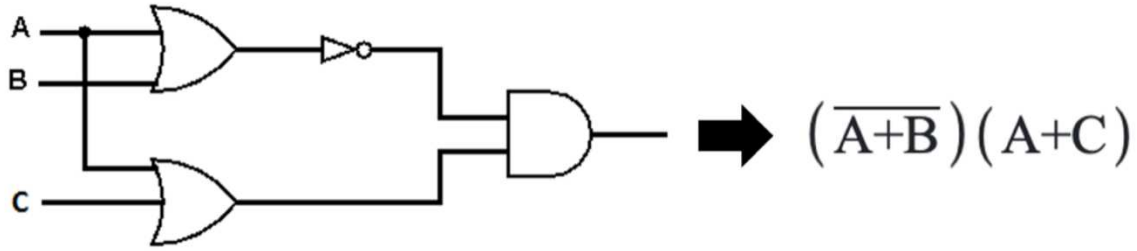
Električne sheme značajan su dio za područje elektrotehnike jer na jednostavan način prikazuju kako su električni elementi povezani u električnom strujnom krugu te koji je njihov međusoban odnos. Električni strujni krug se koristi kako bi se generirala, prenosila i koristila električna energija. Strujni krugovi se najčešće prikazuju električnim shemama gdje je svaki električni element prikazan međunarodno prihvaćenim jedinstvenim grafičkim simbolom. Često se uz grafički simbol elemenata strujnog kruga nalazi karakteristična oznaka elementa. Postoje brojni online alati koji omogućuju crtanje strujnih krugova. Neki od njih su CircuitMaker, *Computer-aided design* (CAD) i *CircuitStudio* [3].

U ovom poglavlju opisan je **alternativan način prikaza električnog strujnog kruga u digitalnom obliku**, to jest strojno čitljivom jeziku. Osnovni zakoni elektrotehnike koje je bitno razumjeti su Kirchhoffovi zakoni i Ohmov zakon. Prema Ohmovom zakonu struja je jednaka omjeru napona i otpora. Prvi Kirchhoffov zakon glasi: „*Zbroj struja koje ulaze u neki čvor je jednak zbroju struja koje izlaze iz tog čvora*.“ Zbog navedenog je zbroj struja u svakom čvoru nula. Struje koje ulaze u čvor (preme dogovoru) su pozitivnog predznaka, a struje koje izlaze iz čvora su negativnog predznaka. Drugi Kirchhoffov zakon glasi: „*U svakoj zatvorenoj petlji je algebarski zbroj napona izvora jednak algebarskom zbroju napona na otpornicima.*“ Odabire se smjer obilaženja zatvorene petlje te ako ulazimo u element kod minusa, napon elementa će imati pozitivan predznak. Ako ulazimo u element kod plusa, napon elementa će imati negativan predznak [19]. Razlikujemo strujne krugove istosmjerne i izmjenične električne struje.

3.1. Netlist – lista mreže

Jednostavan primjer pretvaranja slikovnog prikaza u tekstualni zapis su logički sklopovi. Na slici (Slika 3.1) s lijeve strane prikazan je logički sklop s tri ulaza, dvije "ILI" operacije, jednom negacijom i jednom "I" operacijom, a s desne strane je prikazan njegov tekstualni zapis. Ulazi se označuju slovima, "ILI" operacije znakom zbrajanja, operacije "I" znakom množenja, a negacije vodoravnom crtom iznad operacije ili slova koje se negira. Ovakav način pohranjivanja logičkih sklopova zauzima manje memorije u usporedbi s pohranjivanjem cijele slike. Također, računalno je lakše obraditi tekstualni zapis i izračunati

izlaz na temelju njega. Električnu shemu je puno teže tekstualno prikazati zbog njezine veće složenosti.

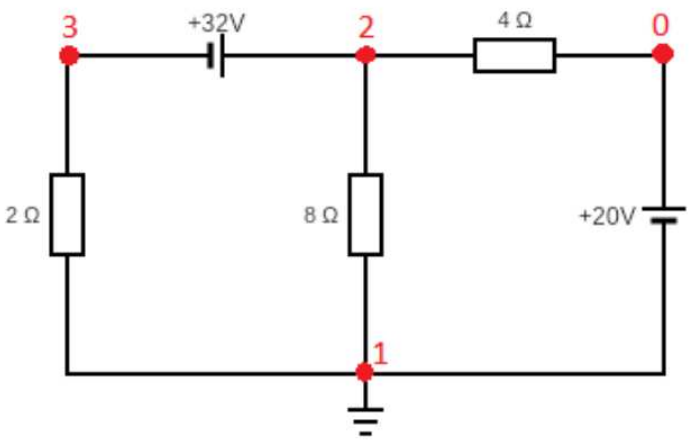


Slika 3.1 Pretvaranje shematskog logičkog sklopa u tekstualni zapis

Netlist ili lista mreže je tekstualni zapis koji opisuje jedinstvenu električnu shemu. Mnogi alati, poput Spice (engl. *Simulation Program with Integrated Circuit Emphasis),* omogućuju korisnicima unos električnih krugova kao netlist ili kao shematski prikaz. Ovi alati služe za simulaciju električnih krugova.

Netlist se sastoji od četiri stupaca, a broj redaka ovisi o broju elemenata u strujnom krugu. Prvi stupac označava element strujnog kruga. Na primjer, ako se na slici sheme nalazi otpornik, u prvi bi se stupac upisalo slovo R jer je R jedinstvena oznaka za otpornik. Radi lakše notacije, komponentama se dodjeljuje i nasumičan redni broj. Time bi u prvi stupac, koji označuje otpornik, bilo zapisano R1. Drugi i treći stupac označuju čvorove između kojih se nalazi element. Brojevi čvorova su nasumično odabrani počevši od nule. Zadnji, četvrti stupac označava iznos elemenata zapisanih u osnovnoj mjeri te zbog toga nije potrebno napisati mjernu jedinicu.

U nastavku je prikazan detaljan postupak stvaranja jednostavne netlist mreže iz strujnog kruga električne sheme koja se sastoji od tri otpornika, dva izvora istosmjernog napona i uzemljenja. Za generiranje sheme i netlista korišten je se javno dostupan alat [1] u kojemu je stvorena električna shema prikazana na slici (Slika 3.2). Elementi su povezani kako je prikazano te su vrijednosti elemenata: otpornici jačine 2 Ω, 4 Ω i 8 Ω te izvori napona jačine 20 V i 32 V.



Slika 3.2 Električna shema napravljena u alatu [1]

Početno je na električnoj shemi potrebno odabrati točke koje će predstavljati mjesta između kojih se nalazi pojedini element. Na slici (Slika 3.2) su odabrane četiri točke koje su označene crvenom bojom te su kraj njih napisani nasumično odabrani brojevi. Iz slike je vidljivo da se između točke 0 i 1 nalazi izvor istosmjernog napona s iznosom od 20 V. To je zapisano u prvom retku na slici (Slika 3.3) gdje je prikazan zapis netlista prikazane sheme sa slike (Slika 3.2). Ako dva ili više elemenata imaju istu oznaku točke u drugom ili trećem stupcu, to znači da su ti elementi susjedi u električnoj shemi. Na primjer, otpornik R4 nalazi se između točke 2 i 0 te je to susjedni element izvoru napona V1 jer oba sadrže točku 0. Bitno je napomenuti da se uzemljenje ne nalazi između dvije točke, nego točno na jednoj što je vidljivo u trećem retku. Konačan tekstualni zapis odabrane sheme prikazan je slikom (Slika 3.3).

```
V1 0 1 20
V2 2 3 32
V3 1   0
R4 2 0 4
R5 2 1 8
R6 3 1 2
```

Slika 3.3 Prikaz netlista za shemu na slici (Slika 3.2)

Bitno je napomenuti da ovo nije jedino rješenje, no ponovnom konstrukcijom električne sheme iz zapisa (Slika 3.3) dobit će se isti električni strujni krug. Usmjerenje izvora napona može se odrediti poretkom točaka u netlist zapisu. Prva točka (drugi stupac) je strana s koje se nalazi pozitivna strana naponskog izvora, a druga točka (treći stupac) je strana s koje se nalazi negativna strana naponskog izvora.

4. Razvoj računalnih modela

Kako bi se uspješno izgradio računalni model za detekciju električnih elemenata, razvoj je podijeljen u nekoliko koraka. Prvotno je potrebno prikupiti reprezentativne podatke koji će služiti za treniranje, validaciju i testiranje modela. Podaci u ovom radu su slike shema električnih krugova. Sheme moraju biti što raznovrsnije da bi model mogao prepoznati traženo i na novim neviđenim shemama. Također, potrebno je pripaziti da model ne bude previše prilagođen, to jest, da ne bude prenaučen (engl. *overfitting*) što bi značilo da veoma dobro detektira na poznatim shemama, dok na neviđenim shemama postiže loše rezultate.

Idući korak je anotacija elemenata na slikama shema. Za anotaciju je korišten alat Roboflow koji nudi jednostavno označavanje elementa na slici i dodjeljivanje pripadne klase. Prikupljene fotografije potrebno je obraditi tako da sve budu istih veličina. Nakon toga fotografije se dijele u skup za učenje, skup za validaciju i skup za testiranje. Skup za učenje služi modelu da na temelju tih podataka uči informacije. Skup za validaciju pomaže modelu da ispravi nedostatke i radi korekcije modela dok skup za testiranje služi za testiranje izgrađenog modela na neviđenim podacima. Obrada podataka i izrada podatkovnog skupa opisani su u poglavlju 4.1.

Sljedeći korak je implementacija modela za detekciju. Korištena su dva poznata algoritma: YOLOv5 i YOLOv8 čiji se detaljniji opis nalazi u poglavlju 4.2. Dobiveni rezultati prikazani su u pogljavlju 4.3. Idući korak je prepoznavanje teksta na slikama shema što je opisano u poglavlju 4.4. Nakon što se sve navedeno prepozna na slici izrađuje se netlist. Netlist je tekstualni zapis električne sheme. Za izradu netlista potrebno je detektirati vodove i čvorove. Detekcija vodova se radi s pomoću Houghove transformacije. Čvorovi su mjesta gdje se sijeku horizontalne i vertikalne linije ili mjesta između elemenata na istoj liniji. S pomoću Euklidske udaljenosti se svakom detektiranom elementu dodjeljuje najbliži detektiran tekst. Detaljan opis izrade netlista nalazi se u poglavlju 4.5.

4.1. Predobrada podataka i podatkovni skup

Slike shema su prikupljene s predmeta Osnove elektrotehnike na Fakultetu elektrotehnike i računarstva. Prikupljeno je 140 slika shema. Sve slike su ručno anotirane u alatu Roboflow te je ukupno 895 anotiranih elementa koji su raspoređeni u 9 klasa. Svaka slika u prosjeku ima 6.4 anotirana elementa. Klase koje model mora naučiti su: otpornik, zavojnica,

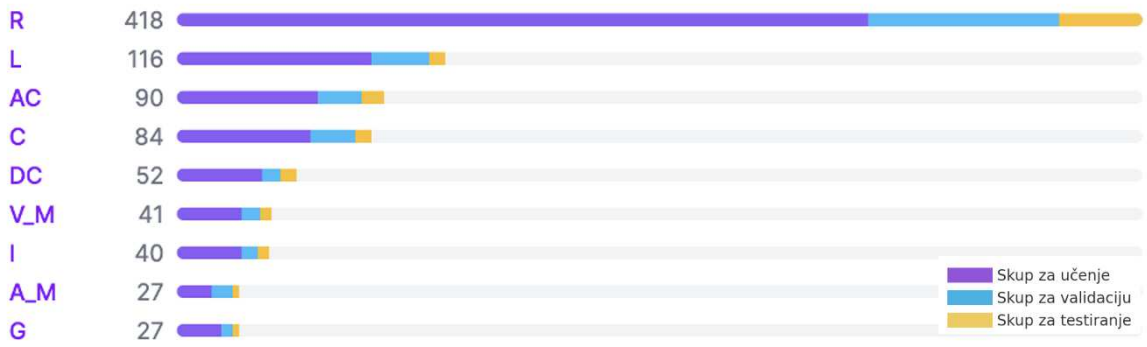
kondenzator, uzemljenje, ampermetar, voltmetar, izvor struje, izvor istosmjernog napona i izvor izmjeničnog napona. Omjer klasa nije ravnomjeran. Najzatupljenija klasa je klasa otpornika što je logično jer se otpornici nalaze gotovo u svakoj shemi, neovisno je li riječ o izmjeničnom ili istosmjernom strujnom krugu.

U tablici (Tablica 4.1) se nalazi broj zastupljenih elemenata na slikama shema. Također, prikazan je broj elemenata po klasama u skupu za učenje, skupu za validaciju i skupu za testiranje. Skup za učenje čini 70 % slika iz ukupnog podatkovnog skupa, skup za validaciju 20 % od ukupnog, a skup za testiranje 10 % od ukupnog. To znači da je 98 slika korišteno za treniranje, 28 za validiranje i 14 za testiranje. U tablici (Tablica 4.1) su elementi kojih ima dovoljno označeni zelenom bojom, a elementi kojih je nedovoljno zatupljeno crvenom bojom. Očekivano je da će model lakše naučiti elemente za koje ima više primjera te će ih bolje generalizirati.

Tablica 4.1 Broj anotiranih elemenata raspoređen po skupovima

Klasa	Cijeli podatkovni skup	Skup za učenje	Skup za validaciju	Skup za testiranje
Otpornik	418	299	83	36
Zavojnica	116	84	25	7
Izvor izmjeničnog napona	90	61	19	10
Kondenzator	84	58	19	7
Izvor istosmjernog napona	52	37	8	7
Voltmetar	41	28	8	5
Izvor struje	40	28	7	5
Uzemljenje	27	15	9	3
Ampermetar	27	19	5	3
Ukupno	895	629	183	83

Na slici (Slika 4.1) grafički je prikazan omjer klasa ukupno te po skupovima. Skup za učenje označen je ljubičastom bojom, skup za validaciju plavom bojom, a skupa za testiranje narančastom bojom. Iz slike se lako može uočiti spomenuti nerazmjer između klasa.

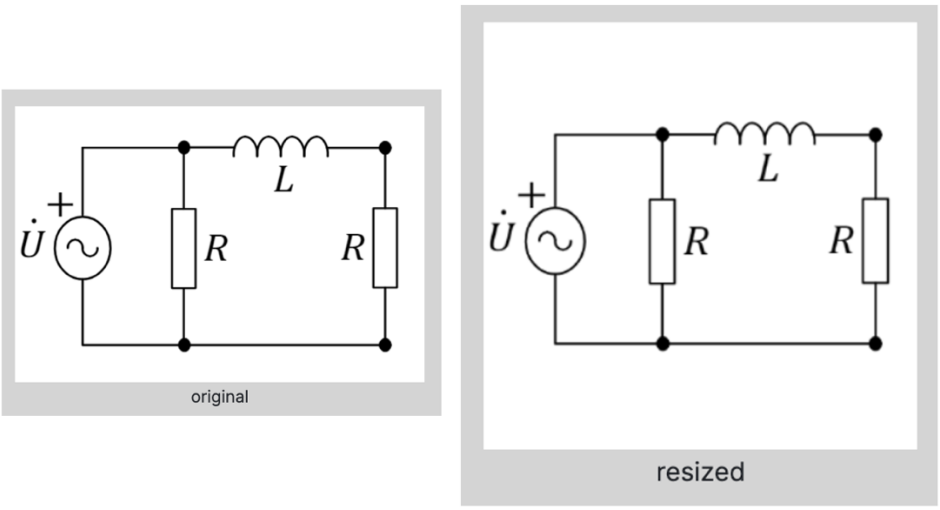


Slika 4.1 Grafički prikaz klasa po skupovima

Medijan veličine slika je 751 x 372 piksela. Za treniranje modela sve slike trebaju biti iste veličine. Odabrana veličina slika je 640 x 640 piksela. Slike su pretvorene u navedene dimenzije tako da je njihov originalan omjer sačuvan te je dodana bijela pozadina. To je napravljeno kako bi se očuvao izvorni izgled elemenata. Da su slike pretvorene u navedene dimenzije bez dodavanja bijele pozadine, neki elementi bi se suzili ili proširili. To bi moglo dovesti do neprepoznavanja nekih elemenata ili otežanog učenja.

Sve slike su konvertirane u crno-bijele kako bi treniranje bilo jednostavnije jer sive slike zauzimaju manje memorije i zahtijevaju manje vremena za treniranje. Kod RGB (engl. *Red Green Blue*) slika svaki piksel ima tri vrijednosti, po jednu vrijednost za svaku boju. Sive slike koriste samo jedan sloj za prikaz svjetline piksela, to jest za prikaz nijanse sive boje. Dovoljna je jedna vrijednost po pikselu.

Primjer sheme kojoj je promijenjena veličina i koja je pretvorena u crno-bijelu shemu prikazan je na slici (Slika 4.2).



Slika 4.2 Prikaz originalne slike sheme te slike nakon obrade veličine i boje

Podatkovni skup može se dodatno povećati primjenom augmentacije. To nije rađeno jer augmentacija na slikama električnih shema može iskriviti ili promijeniti ključne karakteristike elemenata što bi otežalo njihovo pravilno prepoznavanje i detekciju.

Slike anotirane u Roboflow alatu mogu se preuzeti u zip arhivi koja sadrži slike podijeljene u skupove za učenje, validaciju i testiranje. Anotacije elemenata moguće je preuzeti u XML, JSON, TXT ili CSV formatu. YOLO algoritmi koji se koriste spremaju lokacije objekata i dodijeljene klase u TXT formatu. Primjer tog TXT zapisa za shemu sa slike (Slika 4.2) dan je na slici (Slika 4.3).

```
0 0.159375 0.4765625 0.1734375 0.228125
7 0.4078125 0.5046875 0.0953125 0.2328125
7 0.903125 0.5046875 0.0890625 0.23125
6 0.65 0.25625 0.2609375 0.0828125
```

Slika 4.3 Izgled TXT dokumenta za zapis anotiranih elemenata

Svaki element zapisan je u jednom retku te su vrijednosti razdvojene jednim razmakom. Vrijednosti su redom ID klase, centar objekta na x osi, centar objekta na y osi, širina i visina anotiranog elementa. Vrijednosti su normalizirane te su u intervalu od nula do jedan. Nula predstavlja klasu izvora naizmjeničnog napona, sedam je ID od otpornika, a šest ID od zavojnice. Na slici (Slika 4.4) prikazane su vrijednosti koje se spremaju u TXT datoteku. Označen je centar za oba objekta s x i y vrijednostima te su označene visina i širina. Vrijednosti su normalizirane kao što je spomenuto ranije [15].

Slika 4.4 Prikaz slike na kojoj su anotirani objekti i prikazane su vrijednosti koje se šalju algoritmu za učenje: centar na x osi, centar na y osi, širina i visina [15]

4.2. Implementacija modela

Za izgradnju modela korištena su dva različita algoritma YOLOv5 i YOLOv8 iz obitelji YOLO algoritama. Naziv YOLO dolazi od engleske skraćenice „*You Only Look Once*“ što bi u prijevodu značilo „*Pogledaj samo jednom*“. Glavna ideja YOLO algoritama je da u jednom koraku prepoznaju što se nalazi na slici, odnosno da provedu cijeli proces detekcije objekata za razliku od nekih prethodnih modela koji su zahtijevali višestruke prolaze kroz sliku. Time se postiže veća brzina detekcije zbog čega su ovi algoritmi primjenjivi u stvarnom vremenu [10].

Originalan YOLO algoritam napisao je Joseph Redmon 2016. godine i od tada je razvijeno mnogo verzija tog algoritma. YOLO je prva neuronska mreža koja je istovremeno mogla identificirati položaj objekta na slici i dodijeliti tom objektu pripadajuću klasu. Do tada su svi modeli radili u dva koraka: prvo su identificirali položaj objekata, a zatim su u idućem koraku klasificirali te objekte. Primjer takvog modela je Faster R-CNN. Kao što je spomenuto, YOLO algoritam oba koraka izvršava u jednom prolazu zbog čega se često koristi za detekciju u stvarnom vremenu dok istovremeno postiže visoku točnost [9].

4.2.1. YOLOv5

YOLOv5 je peta verzija algoritma za detekciju objekata iz obitelji YOLO. Glavni autor algoritma je Glenn Jocher. Razvijen je od strane tima u Ultralyticsu čija je biblioteka također korištena pri treniranju modela. Ta biblioteka koristi biblioteku PyTorch koja nudi različite usluge za rješavanje problema iz područja dubokog učenja. Nudi podršku za korištenje grafičkih procesora (GPU) za brže treniranje modela. PyTorch je poznat po svojoj jednostavnosti korištenja.

YOLOv5 se sastoji od tri glavna dijela: kralježnica (engl. *backbone*), glava (engl. *head*) i vrat (engl. *neck*). Kralježnica ili osnovna mreža je glavni dio mreže te se tamo izvršava većina računanja. U njoj se agregiraju i oblikuju značajke slike. U YOLOv5 mreži koristi se struktura CSPDarknet-53 koja koristi CSP (engl. *Cross Stage Partial*) povezivanje kako bi se poboljšala učinkovitost i brzina obrade podataka. Vrat ili srednji sloj povezuje kralježnicu i glavu te se sastoji od niza slojeva koji kombiniraju značajke i šalju se glavi na predviđanje. Koristi SPPF i New CSP-PAN strukture. Glava ili izlazni sloj je zadnji sloj mreže i zadužena je za stvaranje izlaza modela. Izlaz je okvir i klasa kojoj objekt pripada [9].

Dva glavna faktora u procesu treniranja su tehnika povećanja podataka i izračun gubitka. Povećanje podataka proširuje originalni skup podataka različitim varijacijama što pomaže modelu u učenju na podacima koji nisu uključeni u izvorni skup. Izračun gubitka kombinira više funkcija gubitaka kako bi se maksimizirala srednja prosječna preciznost (mAP). Srednja prosječna preciznost je najčešća metrika za evaluaciju modela za detekciju [9].

YOLOv5 koristi različite tehnike povećavanja (engl. *Data Augmentation Techniques*) kako bi izbjegao prenaučenost [6]. Neke tehnika povečavanja iz [6] su:

- Mozaik augmentacija (engl. *Mosaic Augmentation*) - kombinacijom četiri nasumično odabrane slike nastaje nova slika koja pomaže modelu da nauči raditi s različitim skalama i translatiranim objektima
- Kopiraj-zalijepi augmentacija (engl. *Copy-Paste Augmentation*) - nasumično odabrani dijelovi slike se kopiraju i lijepe na nasumične dijelove druge nasumično odabrane slike, stvarajući tako novu sliku
- Nasumične afine transformacije (engl. *Random Affine Transformations*) – ove transformacije uključuju rotacije, skaliranja i translacije slika
- HSV augmentacija – ova tehnika uključuje nasumične promjene nijanse (engl. *Hue*), zasićenosti (engl. *Saturation*) i vrijednosti (engl. *Value*) slike

Za uspješno treniranje modela koriste se različite strategije za poboljšanje performansi. Na primjer, tijekom učenja se mijenjaju veličine slika kako bi model naučio detektirati objekte različitih veličina. Koriste se Warmup i Cosine LR Scheduler za prilagodbu stope učenja. Warmup postepeno povećava stopu učenja na početku učenja kako bi se model brže prilagodio problemu, a Cosine LR Scheduler koristi se kasnije kako bi model konvergirao prema optimumu. Model koristi strategiju Exponential Moving Average (EMA) za stabilizaciju treniranja i generalizaciju tako da se koristi prosjek parametara iz prethodnih koraka. Konačan gubitak YOLOv5 algoritma se prema [6] računa se kao kombinacija 3 gubitka:

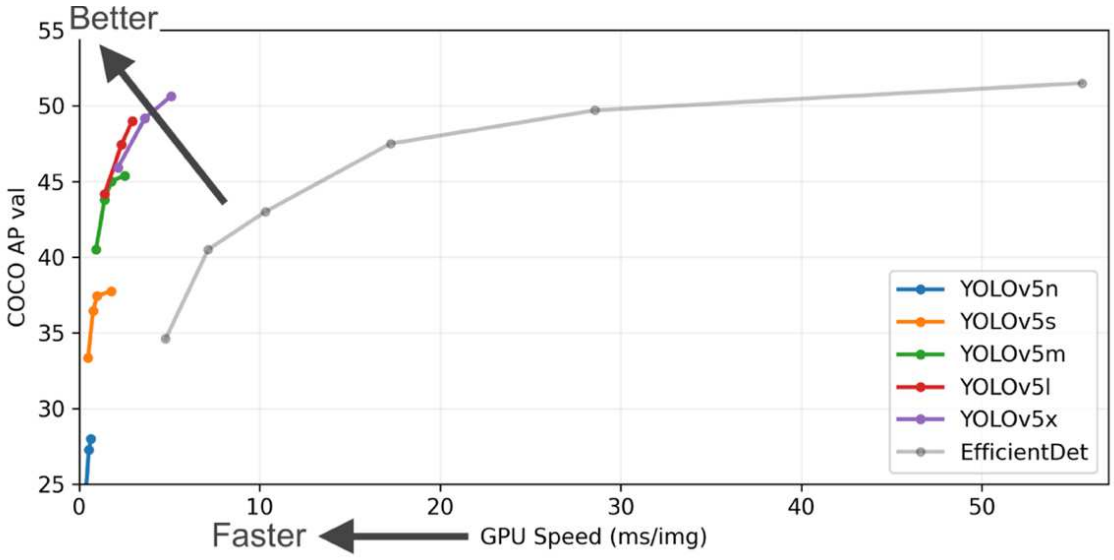
- Binarni Cross-Entropy gubitak klasa gdje se za svaku klasu mjeri pogreška klasifikacije, to jest je li objekt dobro (TRUE) ili krivo (FALSE) klasificiran.
- Binarni Cross-Entropy gubitak prisutnosti objekta koji promatra nalazi li se određeni objekt na slici (TRUE 1) ili ne nalazi (FALSE 0). Time se modela uči kako prepoznati ćelije (grid) gdje se objekt nalazi.

- Complete Intersection over Union (CIoU) – mjeri koliko se predviđene lokacije objekta podudaraju sa stvarnim lokacijama objekata.

To se može zapisati formulom (3.1.).

$$Loss = \lambda_1 L_{cls} + \lambda_2 L_{obj} + \lambda_3 L_{loc} \tag{3.1}$$

YOLOv5 postoji u 5 glavne verzije: nano (n), mala (s), srednja (m), velika (l) i ekstra velika (x). Ove verzije razlikuju se po broju parametara koje koriste za učenja. Veći modeli postižu veću točnost, no vrijeme treniranja i broj parametara je isto veći. Ulaz koji se daje YOLOv5 modelu je 640 piksela. Na slici (Slika 4.5) vidljivo je da sve verzije YOLOv5 modela postižu veću točno u kraćem vremenu od algoritma EfficientDet na podatkovnom skupu COCO (enlg. *Common Objects in Context*) [28]. COCO je opsežan skup podataka dizajniran za prepoznavanja objekata. Sadrži 328 000 slika svakodnevnih prizora s 91 vrstom objekata označenih segmentacijom po instancama. Ukupan broj označenih instanci je 2,5 milijuna [28].



Slika 4.5 Prikaz točnosti različitih algoritama ovisno o brzini [8]

Prije početka treniranja potrebno je instalirati biblioteku YOLO i sve njezine zavisnosti kako bi programsko okruženje bilo spremno za treniranje modela. Treniranje je napravljeno u Google Colabu kako bi se mogle koristiti jače grafičke kartice i time ubrzati proces treniranja. Korištena je T4 – Tesla 4 grafička kartica.

Idući korak je uvoz pripremljenog podatkovnog skupa za treniranje modela. Korištena je Python biblioteka Roboflow koja omogućava jednostavno preuzimanje već pripremljenih podataka. Nakon toga potrebno je odabrati arhitekturu i verziju YOLOv5 modela koji će se

koristiti. Za potrebe ovog modela odabrana je srednja (m) verzija jer podatkovni skup nije prevelik. Odabirom arhitekture i verzije modela konfigurira se .yaml datoteka u kojoj je napisana konfiguracija modela. Tu datoteku moguće je urediti, no u većini slučajeva to nije potrebno jer je arhitektura optimizirana da radi dobro s gotovo svim vrstama podataka.

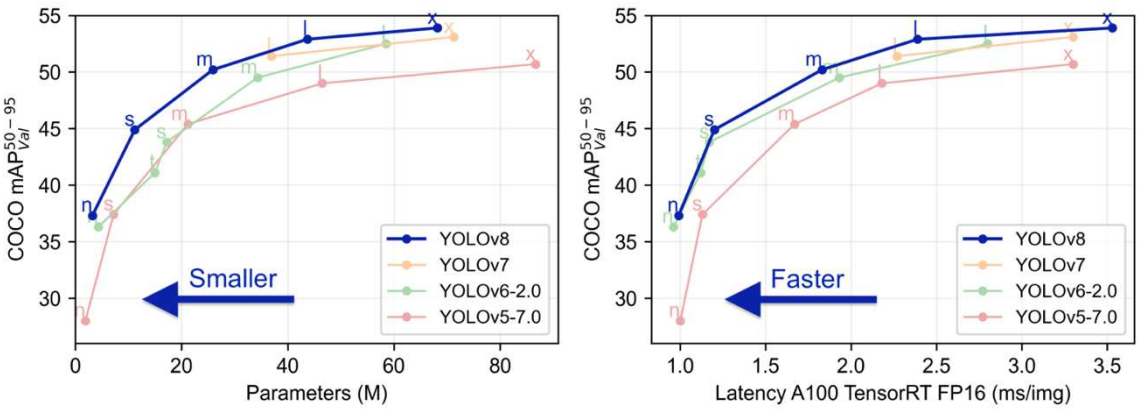
Idući korak je treniranje modela. Moguće je odabrati ulazne hiperparametre za treniranje i započet s treniranjem modela. Neki od hiperparametara su veličina slike, veličina serije (engl. *batch size*) i broj epoha. Također je moguće modelu dodijeliti neke početne težine kako bi se ubrzao proces učenja. Prikaz koda kojim se pokreće učenje modela prikazan je na slici (Slika 4.6).

```
!python train.py --img 640 --batch 16 --epochs 800 --data
{dataset.location}/data.yaml --cfg ./models/custom_yolov5m.yaml --weights '' --
name yolov5m_results  --cache
```

Slika 4.6 Kod za učenje modela korištenjem YOLOv5 algoritma

4.2.2. YOLOv8

YOLOv8 se može koristiti za detekciju objekata, segmentaciju instanci i za klasifikaciju slika. Razvijen je od strane tima Ultralytics, a glavni autor mu je Glenn Jocher. Objavljen je početkom 2023. godine i u usporedbi s modelom YOLOv5 sadrži mnoga poboljšanja. Dizajniran je za jednostavno korištenje putem CLI (engl. *Command Line Interface*) sučelja [7]. Na slici (Slika 4.7) vidljivo je da YOLOv8 postiže najbolje performanse u usporedbi s ostalim YOLO modelima na podatkovnom skupu COCO. U kraćem vremenu i s manje parametara, mAP metrika je najveća.



Slika 4.7 Usporedba točnosti o brzini i veličini parametara YOLO algoritama [7]

Postoji 5 verzija YOLOv8 modela: nano (n), mala (s), srednja (m), velika (l) i ekstra velika (x). Ulazna veličina slika koja je preporučena je 640 x 640 piksela.

YOLOv8 također koristi mozaik povećanje, to jest kombinira četiri različite slike kako bi naučio detektirati objekte na novim lokacijama i različitim veličinama. Koristi PyTorch TXT format anotacije koji se koristi i kod YOLOv5 modela za označavanje objekta na slici [7].

Treniranje YOLOv8 modela je veoma jednostavno jer se može koristiti CLI sučelje s kojim se u nekoliko naredba pokreće treniranje. Prvo je potrebno pripremiti programsko okruženje i učitati potrebne biblioteke. Korišten je Google Colab i dostupna grafička T4. Nakon toga potrebno je dohvatiti podatke. To je napravljeno s pomoću biblioteke Roboflow koja omogućuje jednostavni dohvat podataka s alata Roboflow gdje su podaci anotirani. Treniranje je moguće započeti naredbom prikazanom na slici (Slika 4.8). U kodu je vidljivo da je korištena srednja (m) verzija arhitekture YOLOv8 za izgradnju modela. Korištena je osnovna konfiguracija modela YOLOv8m što je zapisano u dokumentu data.yaml. Broj epoha je postavljen na 150 jer se ispostavilo da se postižu maksimalni rezultati za mAP-50. U kodu se definira i veličina ulaznih slika te je odabrano 640 x 640 piksela. Zadnji argument *plots* omogućuje da se spremaju podaci i grafovi tijekom treniranja kako bi se mogle napraviti detaljnije analize.

```
!yolo task=detect mode=train model=yolov8m.pt data={dataset.location}/data.yaml
epochs=150 imgsz=640 plots=True
```

Slika 4.8 Kod za učenje modela korištenjem YOLOv8 algoritma

4.3. Vrednovanje i tumačenje rezultata

U ovom poglavlju opisane su neke od metrika koje se primjenjuju kod različitih modela za detekciju objekata. Formule metrika prikazane su u tablici (Tablica 4.2). IoU (engl. *Intersection over Union*) mjeri preklapanje između predviđenog okvira i stvarnog okvira objekta čime se ispituje točnosti lokalizacije objekata. AP (engl. *Average Precision*) računa površinu ispod krivulje preciznosti i odziva. Mjera mAP (engl. *Mean Average Precision*) proširuje koncept AP-a izračunavanjem prosječnih AP vrijednosti za više klasa objekata. Preciznost (engl. *Precision*) se računa kao udio točnih pozitivnih predikcija među svim pozitivnim predikcijama. Odziv (engl. *Recall*) izračunava udio točnih pozitivnih predikcija među svim stvarnim pozitivnim slučajevima. F1 vrijednost (engl. *F1 Score*) je harmonijska sredina preciznosti i odziva. Ona uzima u obzir lažno pozitivne i lažno negativne rezultate [18].

Tablica 4.2 Popis metrika za validaciju detekcije objekata i njihove formula

Metrika	Formula
IoU	$IoU = \frac{Presjek \text{ predviđenog i stvarnog okvira}}{Unija \text{ predviđenog i stvarnog okvira}}$
AP	$AP = \int_{r=0}^{1} p(r)dr$
mAP	$mAP = \frac{1}{n} \sum_{k=1}^{k=n} AP_k$
Preciznost	$precision = \frac{TP}{TP + FP}$
Odziv	$recall = \frac{TP}{TP + FN}$
F1 vrijednost	$F1 = 2\,\frac{precision \times recall}{precision + recall}$

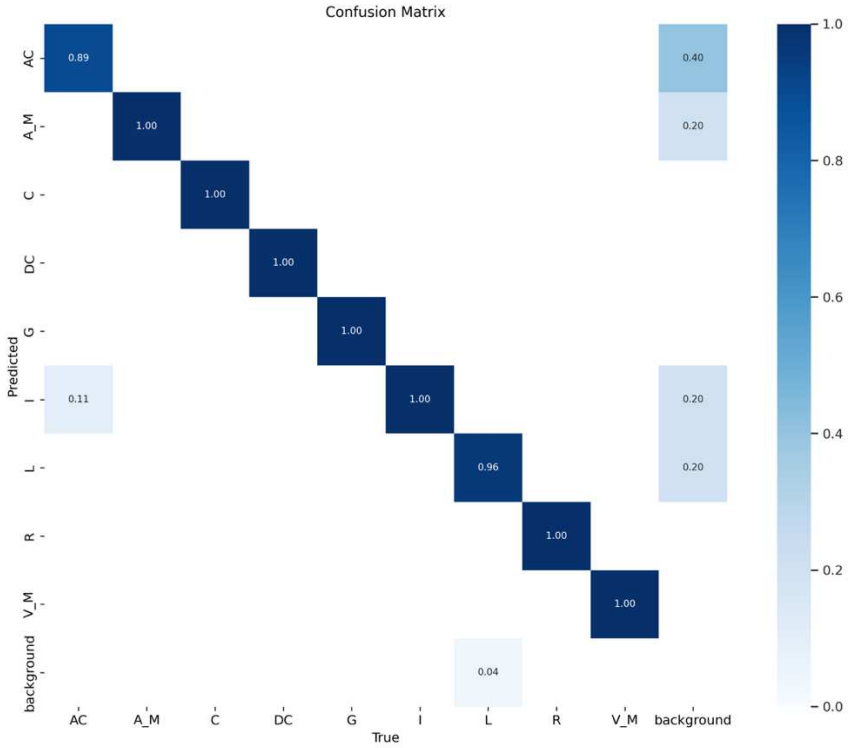
Srednja prosječna preciznost izračunata na pragu preklapanja (IoU) od 50 % naziva se mAP50. Mjera mAP50-95 uzima prosječne vrijednosti srednje prosječne preciznosti izračunate na pragovima od 50 % do 95 % IoU. S tom mjerom se postiže sveobuhvatniji pregled performansi modela na različitim razinama težine detekcije.

Matrica zabune (engl. *confusion matrix*) prikazuje brojeve točnih pozitivnih, točnih negativnih, lažno pozitivnih i lažno negativnih za svaku klasu. Često se koristi i normalizirana matrica zabune gdje su podaci prikazani kao proporcije, a ne kao sirovi brojevi. To olakšava usporedbu performansi između različitih klasa pogotovo ako su klasa jako nebalansirane.

Oba algoritma generiraju dvije datoteke last.py i best.py koje sadrže težine (engl. *weights*) zadnje epohe i epohe u kojoj su postignuti najbolji rezultati. Nastavak .py sugerira da je to spremljeno za korištenje s pomoću PyTorcha.

4.3.1. YOLOv5

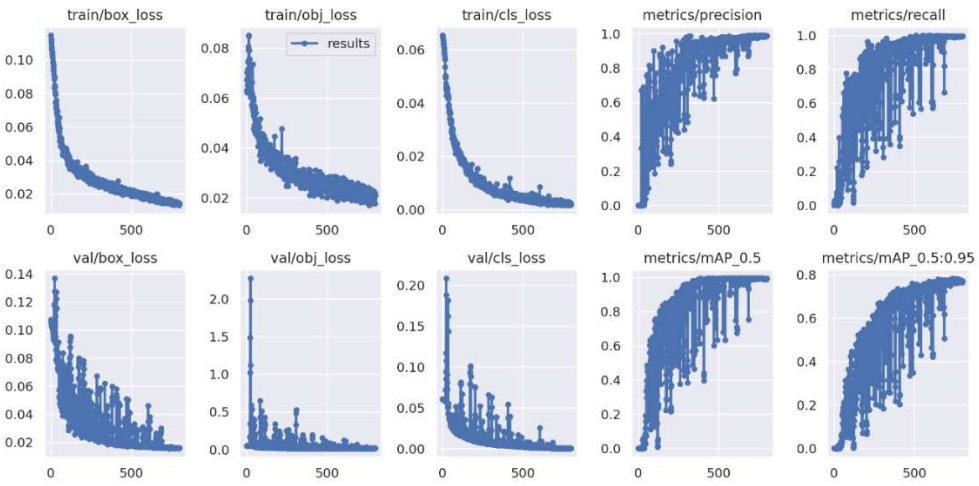
Na slici (Slika 4.9) prikazana je normalizirana konfuzijska matrica čiji stupci prikazuju stvarnu klasu, a reci predviđenu klasu. Većina elemenata je dobro locirana i klasificiran. Primjer pogrešne klasifikacije je 11 % elemenata izvora izmjeničnog napona koji su krivo prepoznati kao strujni izvori. Razlog je vjerojatno okrugli oblik oba elementa. Najveće odstupanje događa se za pozadinu, to jest da neka lokacija nije nijedan element, no dodijeljena je nekom elementu. Matrica konfuzije napravljena je tijekom učenja na podacima za validaciju.



Slika 4.9 Normalizirana matrica konfuzije za model YOLOv5

Na slici (Slika 4.10) prikazani su različiti gubici, odziv, preciznost, mAP-0.50 i mAP-0.95 za svaku epohu. Neke od metrika su jako varirale tijekom epoha što je vidljivo po širini grafova.

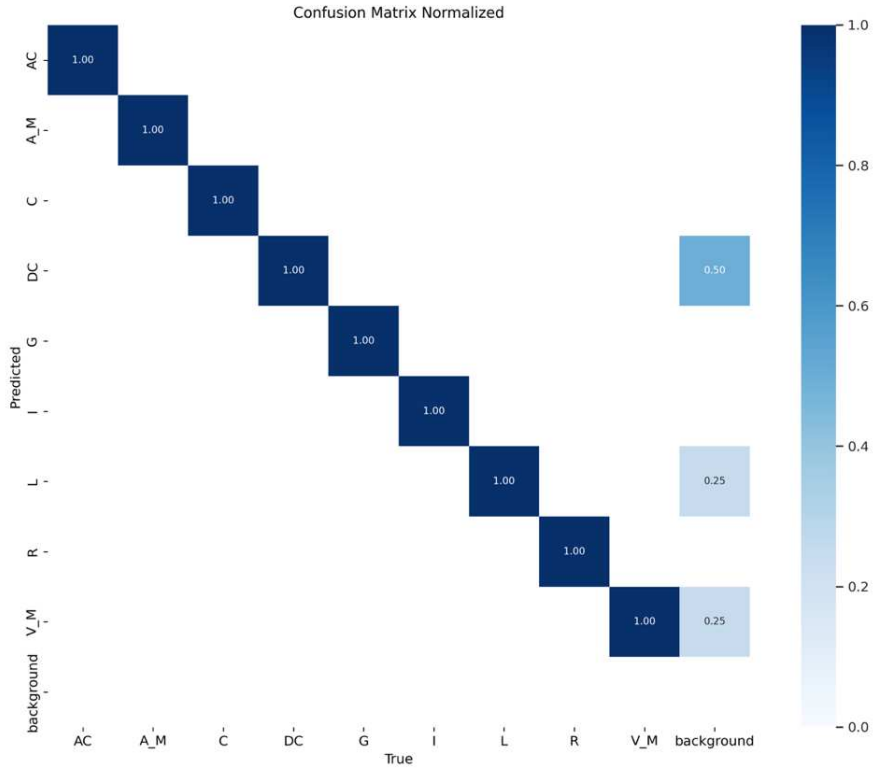
Tijekom učenja model prolazi kroz 291 sloj te konačan broj parametara i gradijenata iznosi 20 903 646.



Slika 4.10 Prikaz grafova za gubitke, odziv, preciznost, mAP-0.50 i mAP-0.95 kroz epohe modela YOLOv5

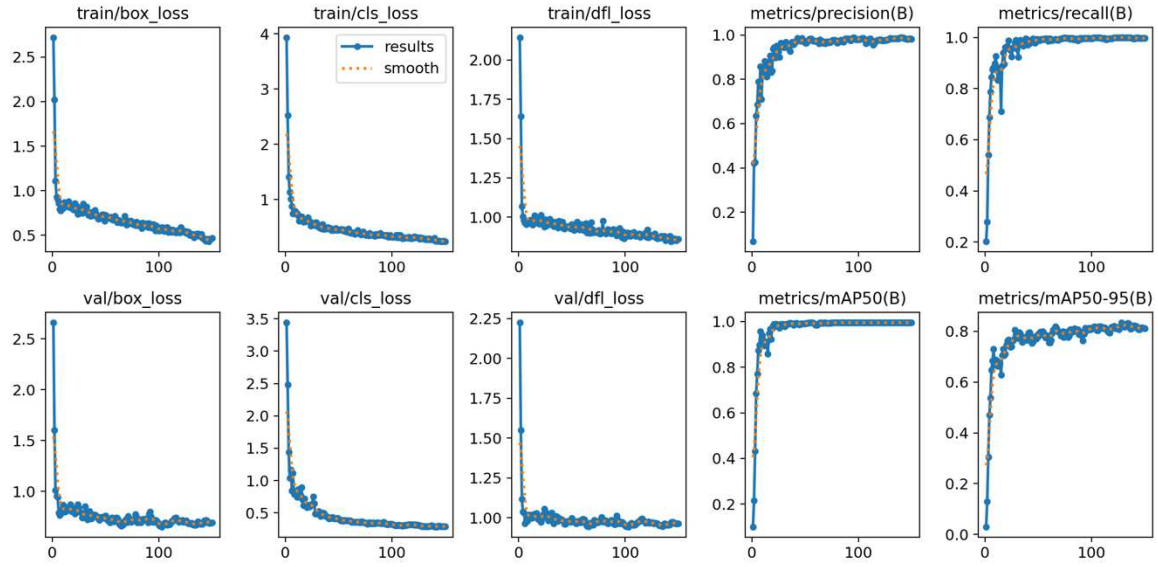
4.3.2. YOLOv8

Na slici (Slika 4.11) prikazana je normalizirana matrica konfuzije čiji stupci prikazuju stvarnu klasu, a retci predviđenu klasu. Vidi se da su svi locirani elementi dobro klasificirani, jedino je u 4 slučaja locirana površina bez elemenata te klasificirana dvaput kao izvor istosmjernog izvora, jednom kao zavojnica i jednom kao voltmetar.



Slika 4.11 Normalizirana matrica konfuzije za model YOLOv8

Idući grafovi (Slika 4.12) prikazuju različite metrike koje se prate tijekom treniranja kroz 150 epoha. Vidljivo je da sve vrijednosti nakon otprilike 30-e epohe krenu lagano stagnirati s blagim padom ili rastom. Nema velikih varijacija tijekom treniranja kao kod YOLOv5 modela. Moglo bi se zaključiti da je moguće dosta dobro naučiti model i u manje epoha.



Slika 4.12 Prikaz grafova za gubitke, odziv, preciznost, mAP-0.50 i mAP-0.95 kroz epohe modela YOLOv8

Tijekom učenja model prolazi kroz 295 sloja te konačan broj parametara iznosi 25 861 531, a od toga 25 861 515 čine gradijenti.

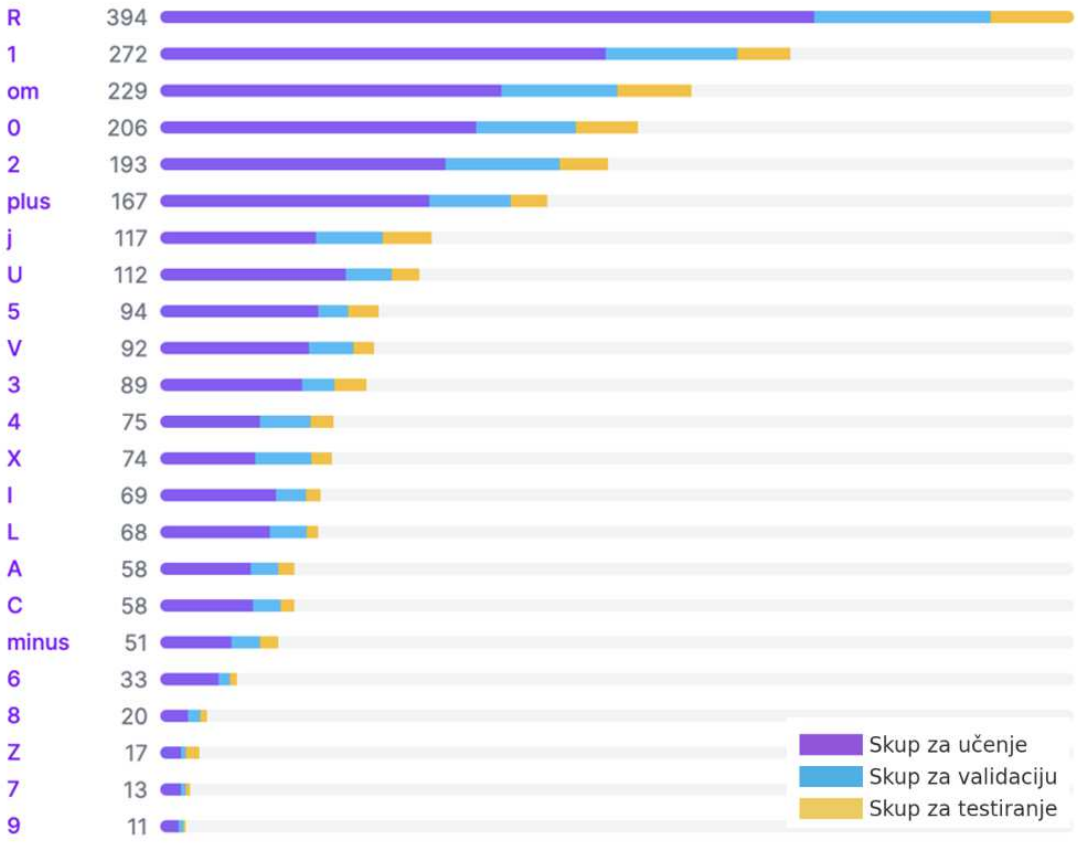
4.4. Optičko prepoznavanje znakova

Optičko prepoznavanje znakova (OCR) je vrlo razvijeno područje u računalnom vidu. Postoje mnogi algoritmi koji mogu prepoznati tekst na slici. Većina njih je optimizirana za prepoznavanje teksta u redovima ili jednoj liniji. U ovom odlomku opisano je kako se na slikama prepoznaju slova i brojevi.

Isprobana su **dva poznata algoritma: Tesseract i Keras OCR**. Tesseract model ima nekoliko verzija, a verzija 11 namijenjena je za prepoznavanje teksta koji nije u redovima, već je nasumično razbacan po slici. Pokazalo se da model može prepoznati neka slova i brojeve, ali ne može se prepoznati grčko slovo omega. Također, većina modela, uključujući ovaj, očekuje da su sva slova iste veličine i u istom retku. Na slikama shema to često nije slučaj jer gotovo sve slike na kojima su oznake imaju kraj sebe malu brojku koja često nije

točno u istoj ravnini kao i slovo, već malo niže. Ispostavilo se da ovaj algoritam ne prepoznaje tekst dovoljno dobro kako bi se koristio za daljnji rad. Sljedeći isprobani algoritam je Keras OCR. Nažalost ni ovaj algoritam ne uspijeva s visokom točnošću prepoznati tekst na slikama. Zbog navedenih razloga **niti jedan od dva perspektivna algoritma nije korišten za daljnji rad**.

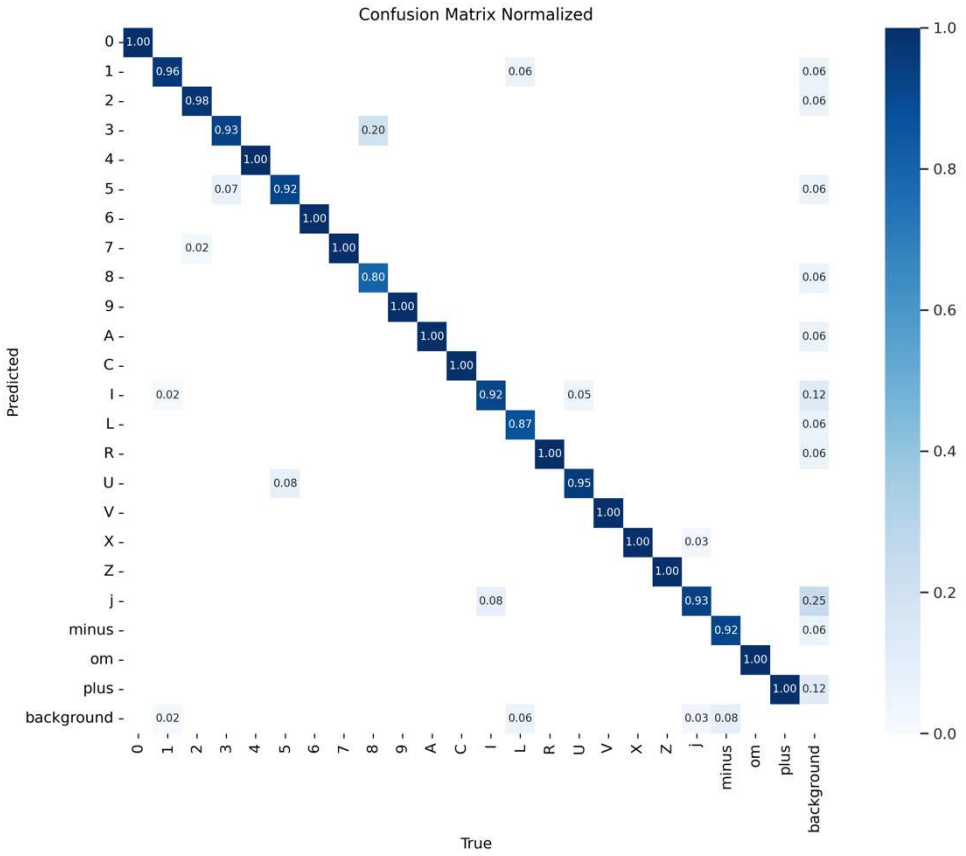
Izrađen je vlastiti model za prepoznavanje znakova na slikama shema. Korišten je algoritam YOLOv8x. Odabrana je ekstra velika (x) verzija algoritma jer ima više klasa i teži je problem od samog prepoznavanja elemenata. Slike od kojih je napravljen podatkovni skup su slike shema koje su također korištene za izradu modela za detekciju, uz par dodatnih slika koje su uklonjene iz početnog podatkovnog skupa. To su na primjer slike koje sadrže kose vodove. Ovaj podatkovni skup sadrži 155 slika i ukupno 2512 anotacija. U prosjeku se na slici nalazi 16,2 locirana i klasificirana objekta. Ove objekte možemo svrstati u 23 klase koje su vidljive na slici (Slika 4.13). Na slici (Slika 4.13) također je vidljiv ukupan udio određene klase te udio klase u skup za učenje, validaciju i testiranje. Omjer klasa je ponovo 70 % za učenje (ljubičasta boja), 20 % slika za validaciju (plava boja) i 10 % za testiranje (narančasta boja).



Slika 4.13 Prikaz klasa po skupovima za model OCR

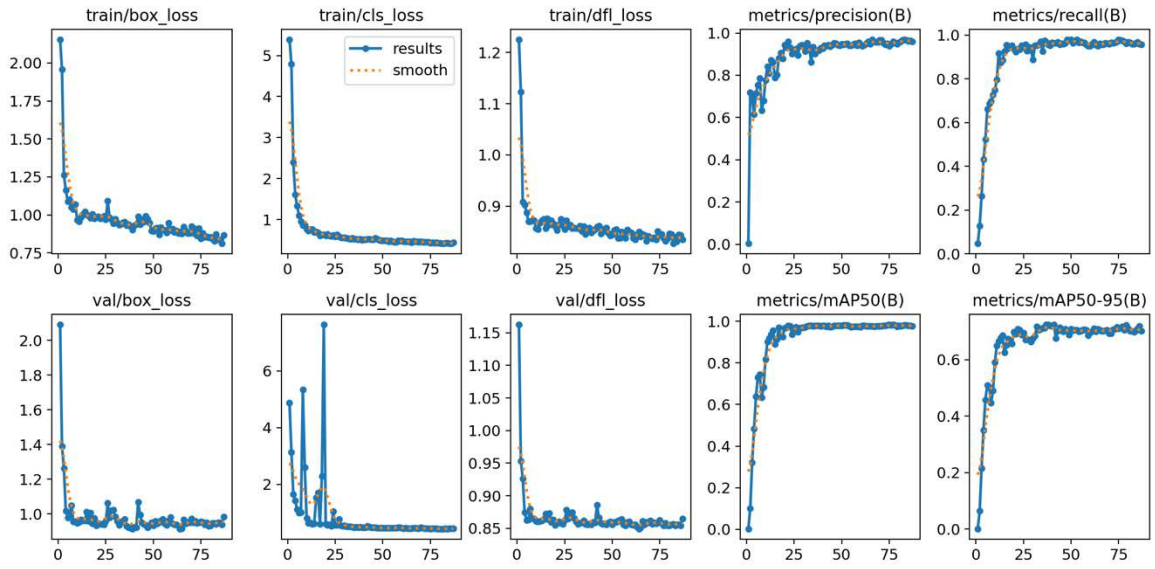
U podatkovnom skupu prevladava klasa R koja označava da se na slici nalazi otpornik. Nekih klasa nažalost nema dovoljno te se za njih očekuju lošiji rezultati.

Veličina slika koja se koristila za učenje je 640 x 640 piksela. Broj odabranih epoha je 200. Zbog većeg broja klasa i verzije algoritma učenje je trajalo duže, nego za prethodne modela. Na slici (Slika 4.14) prikazana je normalizirana matrica konfuzije. Za sve klase je postignuta točnost veća od 80 %.



Slika 4.14 Normalizirana matrica konfuzije za model OCR korištenjem algoritma YOLOv8

Na slici (Slika 4.15) prikazani su gubici, preciznost, odziv i mAP kroz epohe. Učenje je završilo u 136 epohi jer nije bilo napretka u zadnjih 50 epoha. To se zove ranije zaustavljanje (engl. *Early Stopping*).



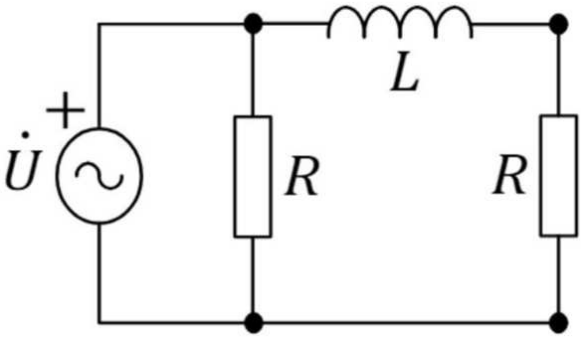
Slika 4.15 Prikaz grafova za gubitke, odziv, preciznost, mAP-0.50 i mAP-0.95 kroz epohe modela OCR korištenjem algoritma YOLOv8

Ovim modelom postignuto je da model preciznije detektira slova i brojeve na slikama shema. Nažalost kako se u podatkovnom skupu nalazila samo jedna slika koja je imala oznaku H za Henry ta klasa nije naučena. Izradom vlastitog modela uklonjena je mogućnost da se detektiraju neka slova za koja nikako nije moguće da se pojave na slici. Na primjer slovo D nije moguće nikada pronaći na shemi jer to slovo nema nikako značenje u području elektrotehnike. Svi brojevi se nalaze na shemama te za svakog postoji zasebna klasa. Dodatna prednost vlastitog modela za detekciju teksta je mogućnost prepoznavanja grčkog slova omega (Ω) što je oznaka za otpor (om)

Kombinacijom oba nastala modela (model za detekciju elemenata i model za detekciju karakteristika) mogu se dodatno potvrditi detektirane stvari. Na primjer, ako je model za detekciju elemenata detektirao otpornik, a model za detekciju karakteristika slovo R, s većom sigurnošću možemo utvrditi da je to točno. Isto tako ako je model za detekciju elemenata detektirao otpornik, a model za detekciju karakteristika neki broj i oznaku Ω, isto možemo s većom sigurnošću utvrditi da je to točno.

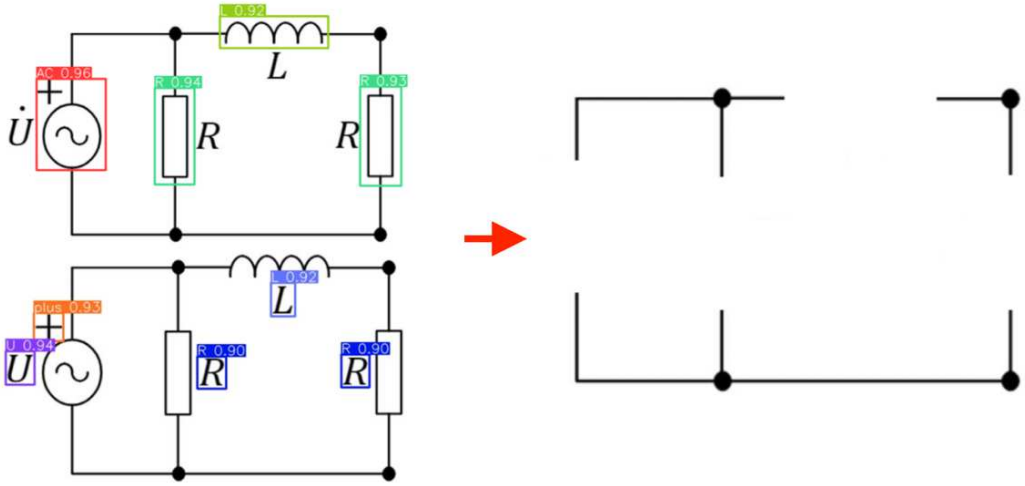
4.5. Pretvorba u netlist

Za prikaz izrade netlista koristit će se shema na slici (Slika 4.16). Svi koraci opisani su na toj slici, no postupak se ekvivalentno može provesti na bilo kojoj drugoj slici sheme.



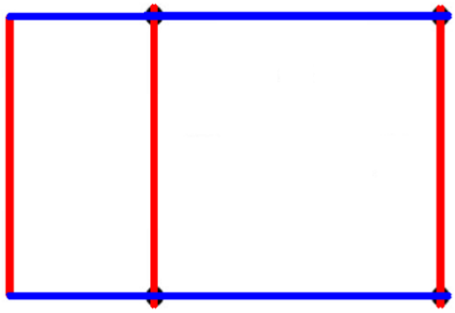
Slika 4.16 Odabrana shema za izradu netlista

Za stvaranje netlista potrebno je ukloniti sa slike sve elemente i oznake kraj elemenata. To se uklanja tako da se lokacije detektiranih objekata izrežu iz slike, to jest oboje u bijelo. Shema koja nastaje nakon uklanjanja objekata prikazana je na slici (Slika 4.17) s desne stane, a lokacije koje se uklanjaju označene su pravokutnicima što je vidljivo na slici (Slika 4.17) s lijeve strane. Vidljivo je da su ostali prikazani samo vodovi.



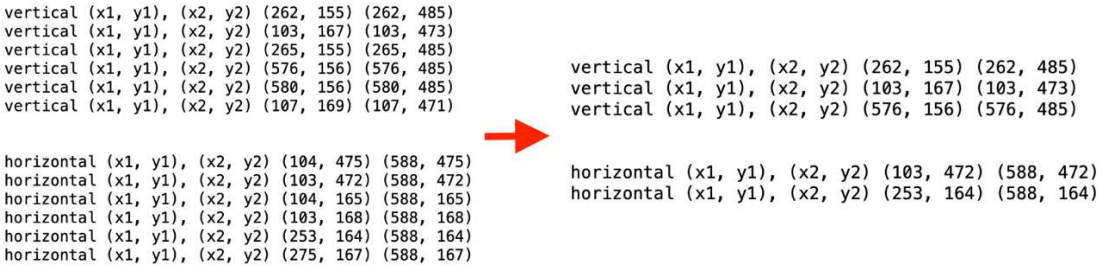
Slika 4.17 Prikaz uklanjanje lociranih objekata sa slike sheme

Nakon toga se na slici koja sadrži samo vodove traže horizontalne i vertikalne linije. To je napravljeno s pomoću Houghove transformacije kao i u radu [13]. Minimalan broj potrebnih piksela da bi se detektirala linija postavljen je na 40. Minimalan dužina linije postavljena je na 70 piksela. Maksimalni razmak između linija da bi se linije spojile u jednu liniju je 200 piksela. Na slici (Slika 4.18) vertikalne linije su označene crvenom bojom, a horizontalne linije plavom bojom.



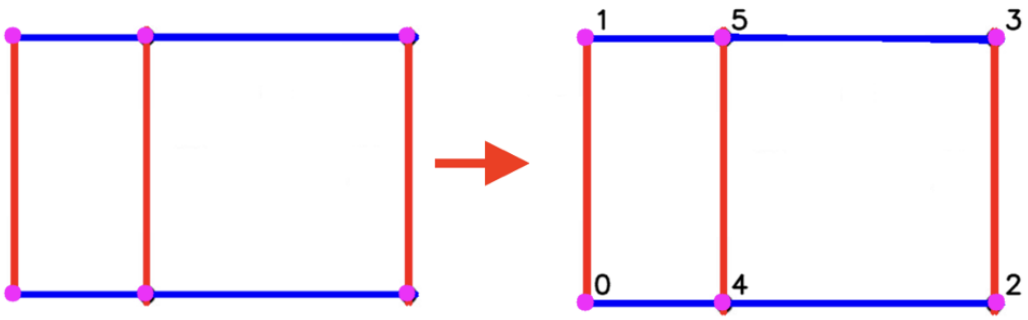
Slika 4.18 Prikaz pronađenih vertikalnih (plavo) i horizontalnih (crveno) linija korištenjem Houghove transformacije

Detektirane linije se spremaju u listu horizontalnih i vertikalnih linija. Tu listu je dodatno potrebno optimizirati da susjedne linije budu objedinjene u jednu. Na primjer, gornja horizontalna linija je zapravo skup od nekoliko horizontalnih linija, no zbog malih piksela to izgleda kao jedna. Potrebno je optimizirati linije, to jest želi se postići da je svaki vod predstavljen samo s jednom linijom. Optimizacija linija za primjer gornje sheme vidljiva je na slici (Slika 4.19). Vidljivo je da je na početku detektirano 6 vertikalnih i 6 horizontalnih linija. Optimizacijom su te linije svedene na 3 vertikalne i 2 horizontalne što je također vidljivo na slici (Slika 4.19) s desne strane.



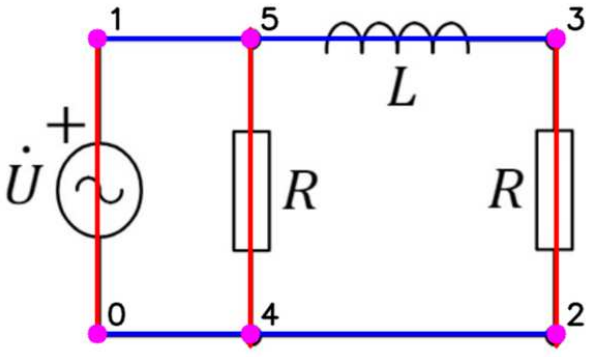
Slika 4.19 Detektirane linije (lijevo), optimizirane linije (desno)

S pomoću tih linija određuje se sjecišta kao presjek horizontalnih i vertikalnih linija. Ta sjecišta su zapravo čvorovi električne sheme. Čvorovi su prikazani ružičastom bojom na slici (Slika 4.20). Svaki čvor ima nasumičnu brojčanu oznaku. Dodatno se u kasnijem koraku dodaju čvorovi ako neki vod ima više od dva elementa na istoj liniji. Taj postupak je opisan u nastavku.



Slika 4.20 Detekcija čvorova na shemi

U sljedećem koraku se između svaka dva povezana čvora provjerava nalazi li se neki element. Na primjer ako promatramo čvor 5 i njegove susjedne čvorove, između čvora 5 i 1 se ne nalazi nijedan element što je vidljivo na slici (Slika 4.21). Između čvorova 5 i 4 se nalazi otpornik te se ta vrijednost zapisuje u rječnik gdje su ključevi točke 5 i 4, a vrijednost je otpornik. Između čvorova 5 i 3 nalazi se zavojnica pa se u rječnik zapisuju 5 i 3 kao ključ, a zavojnica kao vrijednost. Ako se između neke dvije točke nalazi više od dva elementa potrebno je dodati čvor između ta dva elementa. Na primjer da se s desne strane zavojnice nalazi kondenzator u rječnik će se dodati ključ 5 i broj čvorova što je u ovom slučaju broj 6 i vrijednost će biti zavojnica. Iduća vrijednost koja će se dodati je ključ s vrijednostima 6 i 3, a vrijednost će biti kondenzator. Ovo je potrebno napraviti kako bi se shema mogla pravilno konstruirati iz netlist zapisa. Postupak se ponavlja za preostale čvorove i elemente.



Slika 4.21 Prikaz sheme s označenim vodovima i čvorovima

Idući korak je da se za svaki detektirani element Euklidskom udaljenošću pronađe tekstualni zapis. Prije toga tekstualni zapisi su grupirani u manje grupe ako se nalaze dovoljno blizu. Korištena je također Euklidska udaljenost te je pritom uzeto u obzir da se slovo koje je s lijeve strane na slici zapiše i s lijeve stane u tekstualnom zapisu.

Konačnim prolaskom kroz sve susjedne čvorove dobiveni je rječnik prikazan na slici (Slika 4.22). Prva vrijednost u rječniku označuje da se izvor izmjeničnog napona nalazi između čvorova 1 i 0 te je detektiran najbliži tekst slovo „U“.

```
{(1, 0): ('AC', 'U'),
 (3, 2): ('R', 'R'),
 (5, 4): ('R', 'R'),
 (0, 4): [],
 (1, 5): [],
 (4, 2): [],
 (5, 3): ('L', 'L')}
```

Slika 4.22 Rječnik koji sadrži kao ključ oznake čvorova, a kao vrijednost klasu detektiranog elementa i karakteristiku detektiranu uz taj element

Linija koja se nalazi između točke 0 i 4 nema element pa to možemo promatrati kao jedan čvor jer je iznos struje i napona jednak u obje točke. Stvara se pomoćni rječnik u kojem se zapisuju točke koje su istog potencijala te se koristi rekurzija kako bi se one objedinile u jedan čvor. Rječnik zamjena vidljiv je na slici (Slika 4.23) te se u rječniku sve četvorke mogu zamijeniti s nulama, sve petice s jedinicama i sve dvojke s četvorkama, to jest nulama.

```
{4: 0, 5: 1, 2: 4}
```

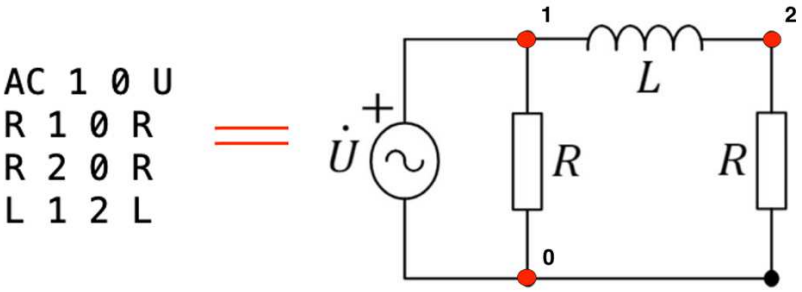
Slika 4.23 Prikaz rječnika za zamjenu čvorova istih potencijala

Kad se provede zamjena u rječniku sa slike (Slika 4.22) ostaju samo točke koje se ne nalaze u ključevima rječnika zamjena sa slike (Slika 4.23). Vidljivo je da su točke koje nemaju elemenata između sebe svedene na isti čvor. U ključevima novonastalog rječnika sa slike (Slika 4.24) ostala su 3 različita čvora što znači da se navedeni netlist može jednoznačno prikazati s 3 čvora. Čvorovi su 0, 1 i 3, no prije stvaranja netlista napravljena je optimizacija kako bi čvorovi bili od nula do broja različitih čvorova umanjen za jedan. U rječniku se 3 zamjenjuje s 2.

```
{(1, 0): [('R', 'R'), ('AC', 'U')],
 (3, 0): ('R', 'R'),
 (0, 0): [],
 (1, 1): [],
 (1, 3): ('L', 'L')}
```

Slika 4.24 Optimiziran rječnik koji sadrži kao ključ oznake čvorova, a kao vrijednost klasu detektiranog elementa i karakteristiku detektiranu uz taj element

Iz novonastalog rječnika se stvara netlist kao tekstualni zapis gdje se za svaki element prvo zapisuje klasa elementa. Nakon toga se zapisuje položaj elementa, to jest između koja dva čvora se nalazi element. Na kraju se zapisuje karakteristika elementa u osnovnoj mjernoj jedinici, to jest najbliža oznaka uz detektirani element. Primjer netlist zapisa dan je na slici (Slika 4.25). Vidljivo je da promatrana shema nema vrijednosti elemenata i mjerne jedinice, nego samo oznake. Te oznake se mogu ručno mijenjati ako korisnik želi zadati proizvoljan iznos napona, otpora ili induktiviteta. Tako formuliran zapis pogodan je za izvršavanje simulacija uz male preinake ovisno o alatu za simulacije.



Slika 4.25 Prikaz sheme netlistom i grafički

5. Razvoj aplikacije

Aplikacija je razvijena za mobilnu platformu iOS koristeći programski jezik Swift. Razvojno okruženje korišteno za izradu aplikacije je Xcode, koje omogućuje pokretanje aplikacije na iOS simulatorima. Aplikaciju je moguće pokretati i na vlastitom iPhone uređaju, no potrebno je namjestiti određene postavke. Na primjer, potrebno je uključiti razvojni (engl. *developer*) način rada u postavkama uređaja. Trenutna najmanja podržana verzija aplikacije je iOS 17, što znači da uređaji s verzijom manjom od 17 ne mogu preuzeti niti pokrenuti aplikaciju. U tom slučaju potrebno je ažurirati operacijski sustav.

Aplikacija je pisana koristeći arhitekturu MVP što je skraćenica od engleskog izraza Model View Presenter. Ova arhitektura je opisana u idućem poglavlju 5.1. gdje su također opisane i funkcionalnosti aplikacije. Za prikaz korisničkog sučelja koristi se biblioteka SwiftUI čiji je detaljniji opis dan u poglavlju 5.2. Stvaranje servera te njegova integracija u aplikaciju opisana je u poglavlju 5.3. Korištenje aplikacije prikazano je u zadnjem poglavlju 5.4.

5.1. Arhitektura i funkcionalnosti aplikacije

MVP (engl. *Model View Presenter*) ili arhitektura sastoji se od tri komponenti od kojih svaka ima svoju specifičnu ulogu. Model predstavlja podatke i poslovnu logiku. Zadužen je za pohranjivanje, dohvaćanje i manipulaciju nad podacima. View je odgovoran za prikaz na ekranu te osluškivanje korisničkih akcija koje prosljeđuje Presenteru. Presenter ima ulogu mosta između Modela i Viewa. On upravlja korisničkim akcijama i podacima, obrađuje ih te šalje Viewu, koji ih zatim prikazuje [14].

U nastavku su nabrojane funkcionalnosti aplikacije te njihov detaljni opis.

Funkcionalnosti aplikacije:

- **Testiranje rada aplikacije na skupu za testiranje:**

Kao što je spomenuto podatkovni skup je podijeljen na skup za učenje, skup za validaciju i skup za testiranje. U aplikaciji su na početnom ekranu prikazane sheme iz skupa za testiranje koje korisnik može odabrati ako nema u blizini fotografiju neke sheme, a želi ispitati rad aplikacije. To je ukupno 14 slika shema koje su poredane u 3 stupca.

- **Fotografiranje slike sheme kamerom uređaja**

Korisnik ima mogućnost fotografiranja proizvoljne sheme odabirom gumba s ikonom kamere i natpisom „*Camera*“. Klikom na navedeni gumb otvara se nativna kamera te se fotografirana slika može koristiti za izradu netlista.

- **Učitavanje slike sheme iz galerije mobitela**

Korisnik ima mogućnost učitavanja proizvoljne sheme odabirom gumba s ikonom galerije i natpisom „*Gallery*“. Klikom na navedeni gumb otvara se galerija uređaja gdje se nalaze sve korisnikove slike. Učitana slika može koristiti za izradu netlista.

- **Izrezivanje fotografirane ili učitane slike sheme**

Fotografirana ili učitana slika može se izrezati tako da je u fokusu crtana shema, bez nepotrebnog okolnog prostora.

- **Prepoznavanje električnih elemenata modelom YOLOv5**

Nakon što je odabrana shema, korisnik može kliknuti na gumb „*YOLOv5“* čime će se na odabranoj shemi detektirati električni elementi korištenjem modela YOLOv5.

- **Prepoznavanje električnih elemenata modelom YOLOv8**

Nakon što je odabrana shema, korisnik može kliknuti na gumb „*YOLOv8*“ čime će se na odabranoj shemi detektirati električni elementi korištenjem modela YOLOv8.

- **Prepoznavanje slova i brojeva modelom YOLOv8**

Nakon što je odabrana shema, korisnik može kliknuti na gumb „*OCR*“ čime će se na odabranoj shemi detektirati tekst korištenjem modela YOLOv8.

- **Prikaz vjerojatnosti pouzdanosti**

Klikom na oznaku detektiranog elementa ili teksta prikazuje se vjerojatnost pouzdanosti s kojom je model siguran da se detektirani element ili tekst nalazi na slici.

- **Dodavanje novih oznaka na postojeću shemu**

U slučaju neuspješne detekcije, korisnik može dodati novi neprepoznati element ili tekst klikom na gumb “Add element“ ili “Add text“.

- **Brisanje detektiranih ili novo dodanih oznaka**

U slučaju pogrešne detekcije, korisnik može ukloniti pogrešno detektiranu oznaku elementa ili teksta klikom na označeni pravokutnik i odabirom gumba „*Delete*“ za brisanje.

- **Uređivanje detektiranih ili novo dodanih oznaka**

U slučajno ispravne lokalizacije, a pogrešne klasifikacije elementa ili teksta, korisnik može urediti element ili tekst klikom na označeni pravokutnik i odabirom gumba „*Edit*“. Otvara se skočni izbornik u kojemu se nalaze ponuđeni elementi ili tekst.

- **Prikaz binarne slike sheme**

Ako korisnik dugo drži pritisnutu sliku (engl. *long press*), ona će se pretvoriti u binarnu sliku, što je korisno za ručno crtane sheme.

- **Stvaranje netlista**

Nakon detekcije elemenata i teksta pojavljuje se gumb „*Create Netlist*“ čijim odabirom se izvršava upit na server te se vraća netlist koji se prikazuje na ekranu.

5.2. Izrada aplikacije

Aplikacija je pisana u programskom jeziku Swift. Swift je predstavljen 2014 godine te je nastao kao zamjena za Objective C koji se prije koristio za Apple proizvode. Swift podržava razvoj aplikacija za sve Apple platforme: iOS, macOS, watchOS i tvOS [2].

Za izradu korisničkog sučelja (engl. *user interface* UI) korištena je SwiftUI biblioteka koja omogućuje dinamičko stvaranje prikaza. SwiftUI omogućuje programerima da kreiraju UI komponente koristeći jednostavan i intuitivan kod pisanjem deklarativne sintakse. SwiftUI također omogućuje pregled uživo (engl. *live preview*) čime su promjene u kodu vezane uz UI vidljive u stvarnom vremenu tijekom izrade aplikacije [2].

5.3. Integracija modela u aplikaciju

Za izgradnju servera korištena je programska platforma Heroku koja omogućava pokretanje aplikacija u oblaku. Heroku pruža jednostavno okruženje za razvoj, testiranje i implementaciju aplikacija.

Za postavljanje servera na Heroku, prvo je potrebno kreirati Heroku račun. Nakon prijave, potrebno je povezati željeni GitHub direktorij s Herokom te odabrati ime aplikacije. Heroku također omogućuje kreiranje i upravljanje aplikacijama iz naredbenog retka koristeći CLI. Nakon uspješnog stvaranja nove aplikacije na Heroku platformi, generira se jedinstveni URL za pristup aplikaciji.

API (engl. *Application Programming Interface*) je razvijen koristeći programski jezik Python i Flask framework. Flask framework omogućuje brzo i jednostavno kreiranje web aplikacija i API-ja. Glavni zadatak API-ja je omogućiti komunikaciju između mobilne aplikacije i modela za stvaranje netlista iz dobivene slike. Potrebno je definirati API rute, na primjer za stvaranja netlista koristi se „**/createNetlist“** te su u funkciji izvršene sve naredbe za uspješno stvaranje netlista. Rad API-ja testiran je lokalnim pokretanjem servera te korištenjem Postamana za slanje API poziva na lokalni server.

Nakon razvoja i testiranja API-ja, potrebno je postaviti (engl. *deployment*) aplikaciju na Heroku server. Potrebno je kreirati Procfile datoteku koja specificira kako pokrenuti aplikaciju. Za Flask aplikacije to se može učiniti jednom linijom kao što je „**web: gunicorn circiutRecognition:app“** gdje je circuitRecognition.py naziv Python datoteke u kojoj su napisane API rute i funkcije koje se izvršavaju njihovim pozivima. Potrebno je kreirati i requirements.txt datoteku koja sadrži sve potrebne biblioteke i verzije koje aplikacija koristi.

Kako bi se na Heroku napravilo postavljanje, potrebno je na GitHub postaviti sve lokalne promjene koje su prethodno napravljene i testirane. Moguće je odabrati git granu (engl. *branch*) s kojeg se želi postaviti. Obično se koristi glavna (engl. *main*) grana. Moguće je podesiti da se postavljanje na server radi automatski svaki put kad se na zadanoj grani dogode promjene. Funkcije koje se izvršavaju na Heroku serveru su funkcije za obradu slika, to jest traženje vodova, čvorova, optimizacije i povezivanje u netlist. Modeli su zbog velikih veličina postavljeni na server koristeći alat Roboflow. On nudi mogućnost jednostavnog postavljanja modela koji su napravljeni koristeći YOLO algoritme. Razlog tome je ograničena veličina Heroku servera od 500 MB.

Mobilna aplikacija koristi URLSession za slanje HTTP zahtjeva API-ju na Heroku i Roboflow serveru. Podaci se šalju u JSON formatu, a odgovori API-ja također se vraćaju u JSON formatu. Dodatno se šalje i slika. API poziv za stvaranje netlista se izvršava koristeći POST zahtjev na „**/createNetlist"** rutu. API obrađuje sliku, koristi funkcije za prepoznavanje elemenata, teksta, linija, čvorova i vraća netlist aplikaciji.

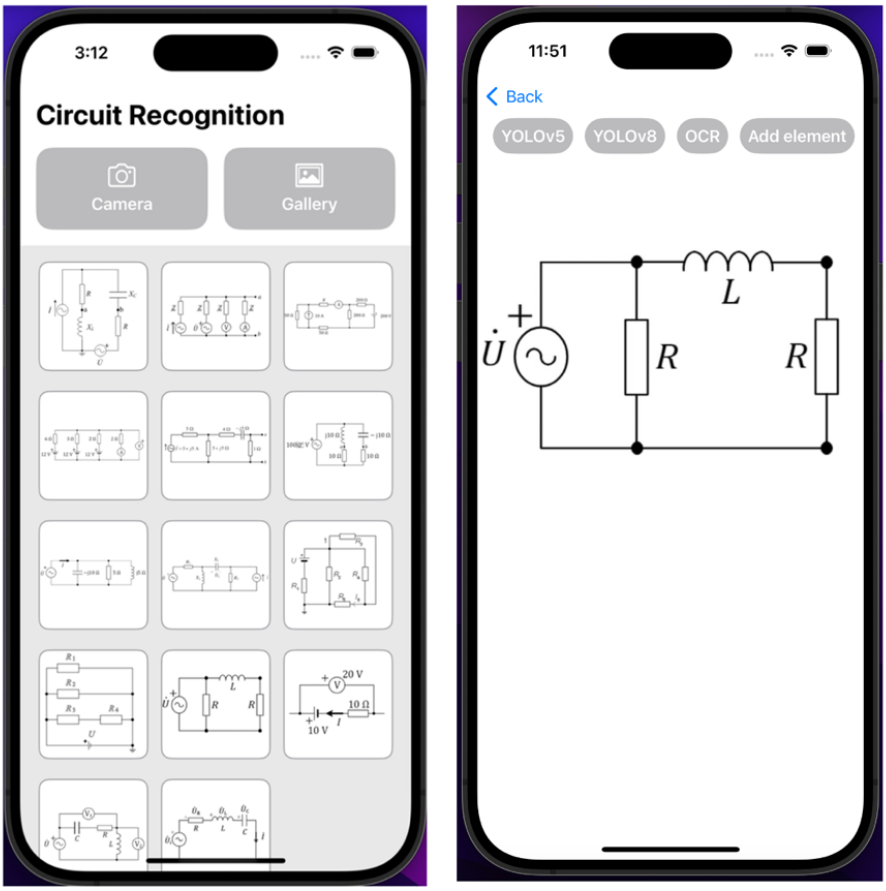
5.4. Korištenje aplikacije

Aplikacija je nazvana „Circuit Recognition“ te se otvara pritiskom na ikonu koja je prikazana na slici (Slika 5.1). Ikona aplikacije je napravljen u besplatnom alatu Logo Maker - Looka. Odabrana je jednostavna bijela pozadina aplikacije, dodana je mala sličica proizvoljne sheme te je odabran tekst koji izgleda kao da je napravljen od vodova što sve ukupno sugerira za što je aplikacija namijenjena.

Slika 5.1 Ikona i logo aplikacije Circuit Recognition

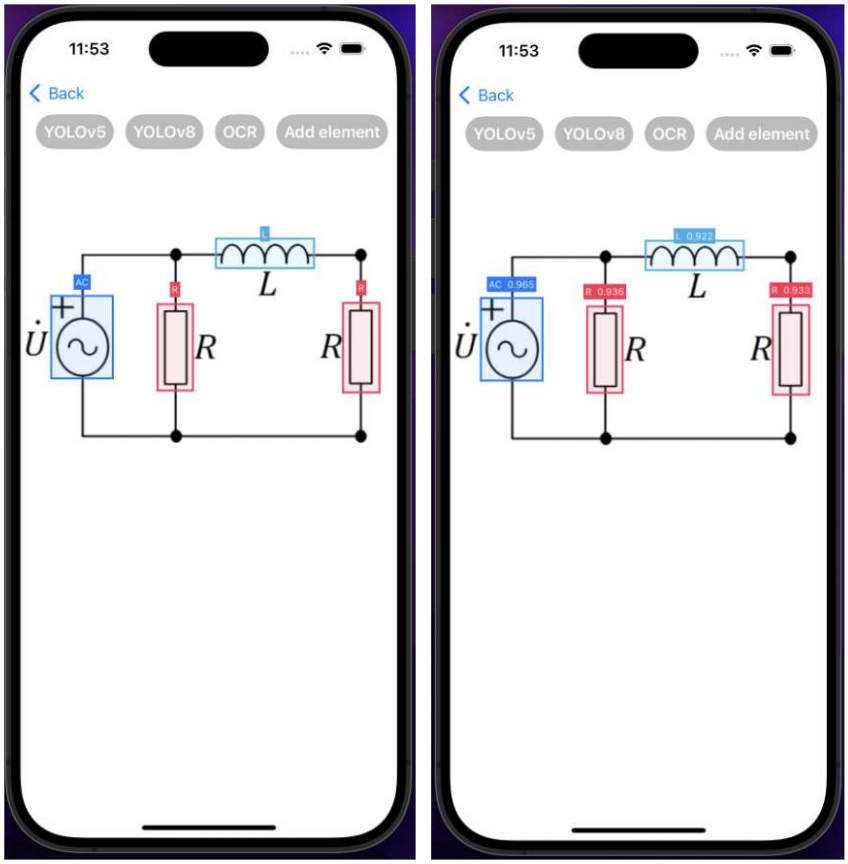
Početni ekran aplikacije prikazan je na slici (Slika 5.2) s lijeve strane. Sastoji se od naslova „Circuit Recognition“, dva gumba i pomičnog dijela u kojem su prikazane sheme iz testnog podatkovnog skupa. Odabirom lijevog gumba „*Camera*“ otvara se nativna kamera uređaja s kojom je moguće fotografirati željenu shemu, a odabirom gumba „*Gallery*“ otvara se galerija uređaja te je moguće odabrati prethodno fotografiranu ili neku spremljenu shemu.

Kad se odabere shema otvara se novi ekran koji je prikazan na slici (Slika 5.2) s desne strane. Ekran ima gumb „*Back*“ s kojim je moguće vratiti se na prethodni (lijevi) ekran te ponovo odabrati, fotografirati ili učitati neku shemu. Na vrhu ekrana ispod gumba za povratak nalazi se pomični izbornik s 5 gumba: „*YOLOv5*“, „*YOLOv8*“, „*OCR*“, „*Add element*“ i „*Add text*“.



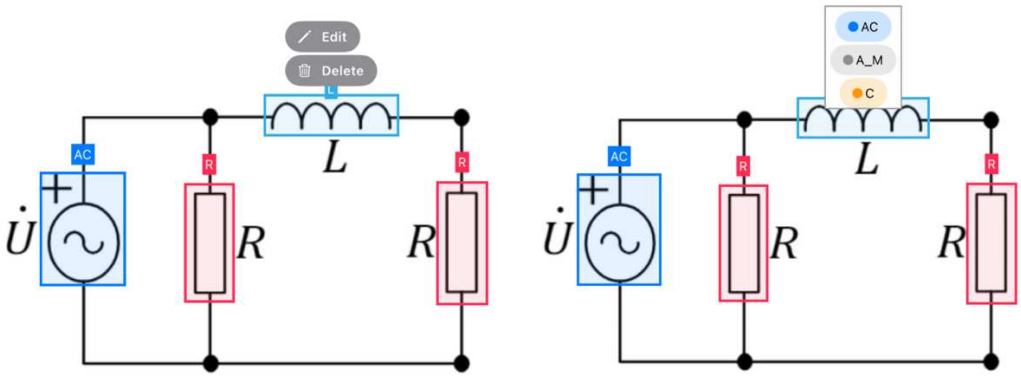
Slika 5.2 Početni ekran aplikacije (lijevo) i ekran nakon odabira sheme (desno)

Pritiskom na gumb „*YOLOv5*“ ili „*YOLOv8*“ poziva se API poziv na Roboflow server kojemu se šalje odabrana shema, a vraća se JSON file u kojemu su sadržane informacije o veličini slike te detektiranim elementima. Svaki detektirani element sadrži sljedeće informacije: centar na x osi, centar na y osi, širina, visina, klasa, ID klase te vjerojatnost da se navedeni element nalazi na slici. Na temelju tih informacija iscrtani su pravokutnici koji se izračunavaju dinamički ovisno o širini i visini ekrana te o širini i visini slike. Klikom na slovo koje označuje klasu kojoj element pripada prikazuju se vjerojatnosti pouzdanosti pripadnosti klasi. Iscrtani pravokutnici prikazani su na slici (Slika 5.3) s lijeve strane, a prikaz pouzdanosti modela na slici (Slika 5.3) s desne strane. Za vrijeme čekanja odgovora od servera prikazana je animacija za učitavanje te nije moguće klikati po ekranu. Boje pravokutnika su definirane unutar enuma kako bi korisnik lakše uočio klase elemenata.



Slika 5.3 Prikaz detektiranih elemenata (lijevo) i pouzdanosti detektiranih elemenata (desno)

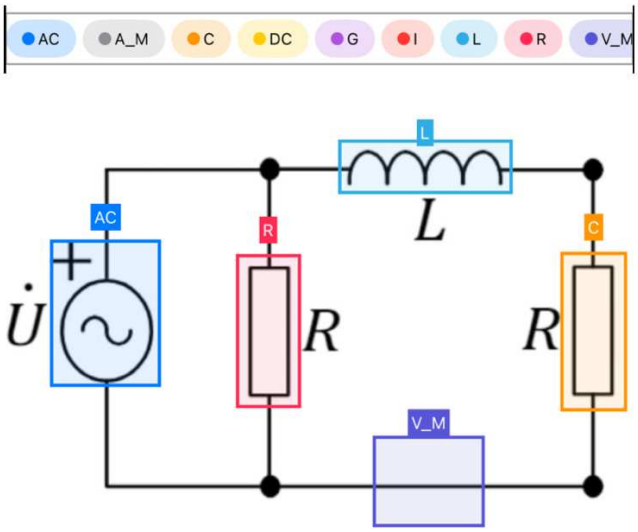
Klikom na bilo koji od iscrtanih pravokutnika otvara se pomoćni izbornik vidljiv na slici (Slika 5.4). Moguće je obrisati neželjeni element ili promijeniti oznaku detektirane klase.



Slika 5.4 Prikaz pomoćnog izbornika za uređivanje ili brisanje detektiranog elementa (lijevo) i prikaz pomoćnog izbornika za promjenu klase (desno)

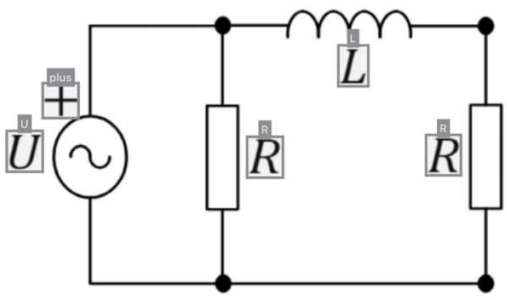
Isto tako moguće je dodati na bilo kojem mjestu proizvoljni element odabirom gumba „*Add element*“. Klikom na taj gumb otvara se horizontalna lista u kojoj su ponuđene sve moguće klase te korisnik odabire jednu od njih. Nakon odabira klase iscrtava na shemi lokaciju gdje

se nalazi navedeni element. Na primjer, zamislimo da korisnik želi izmjeriti napon između dva otpornika te dodaje voltmetar između odabranih otpornika. Dodan voltmetar prikazan je na slici (Slika 5.5) ljubičastom bojom. Također, desni otpornik je zamijenjen s kondenzatorom. Da se sad izvrši API poziv, dobio bi se netlist koji ima 5 elemenata: izvor izmjeničnog napona, otpornik, zavojnicu, kondenzator i voltmetar. Sve promjene koje korisnik napravi koriste se kasnije kod stvaranja netlista. Ovime se omogućuje ispravljanje pogrešaka modela te izrada netlista s malim varijacijama po vlastitoj želji.



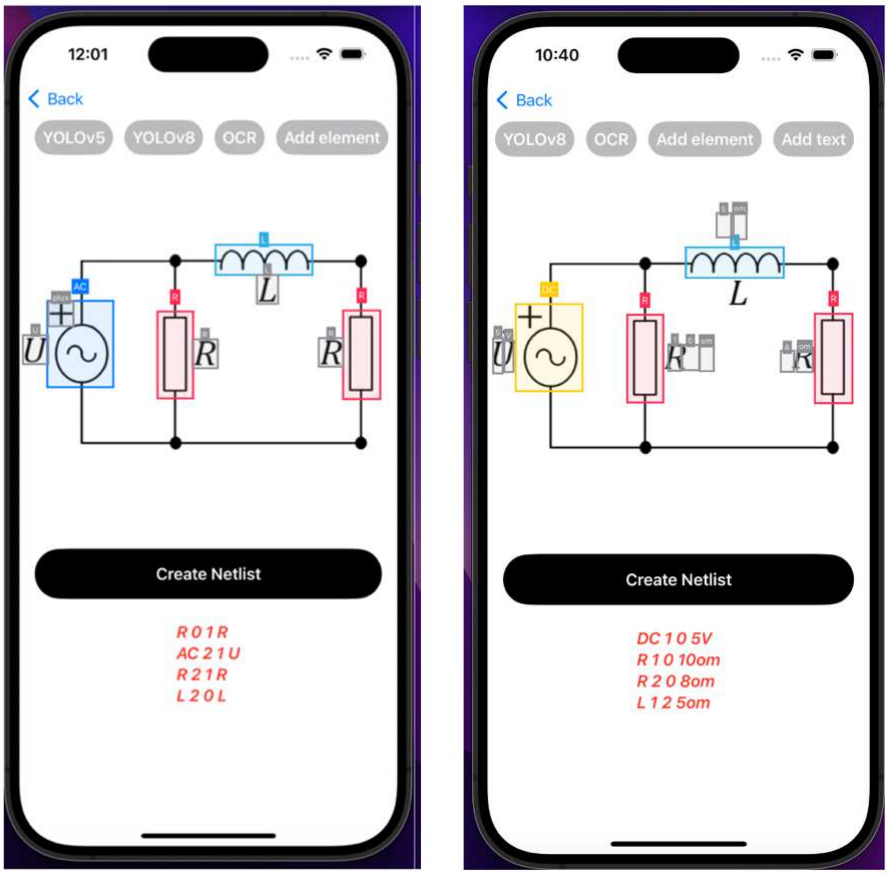
Slika 5.5 Pomična traka koja se otvara odabirom gumba za dodavanje elemenata i dodane promjene na shemu

Na slici (Slika 5.6) prikazan je izgled ekrana pritiskom na gumb OCR. Ponovo se šalje slika ako kroz API poziv te su vraćene iste informacije kao i za električne elemente. Moguće je uređivati, brisati i dodavati neispravno detektirane karakteristike ili ako želimo promijeniti neke od karakteristika za stvaranje netlista i simulacija.



Slika 5.6 Prikaz detektiranih karakteristika na shemi

Primjer netlista bez promjena prikazan je lijevo na slici (Slika 5.7), a s desne strane prikazan je netlist ako korisnik promijeni neke karakteristike u mreži. Na primjer promijenjen je izvor napona iz izmjeničnog u istosmjerni i dodane u numeričke vrijednosti za svaki element.



Slika 5.7 Prikaz netlista s detektiranim podacima (lijevo) i promijenjenim podacima (desno)

6. Simulacija električne sheme korištenjem netlista

U aplikaciji je nemoguće pokretati simulacije, no dobiven netlist moguće je pokretati u drugim alatima kao što su LTspice i Multisim. U nastavku je prikazano simuliranje prethodno dobivenog netlista u LTsipce alatu. Simuliran je desni primjer sa slike (Slika 5.7). Primjer je potrebno dodatno prilagoditi LTspice alatu kao na slici (Slika 6.1). Prvi redak označava kao i u prethodnom netlistu da se istosmjerni napon nalazi između čvorova 1 i 0 te iznosi 5 volti. Vidljivo je da se zapis malo razlikuje, no logika je ista. Na primjer, LTspice u prvi stupac uvijek za izvor napona koristi oznaku *v* i redni broj, a nakon čvorova dodatno zahtjeva da se specificira koji izvor napona se koristi. Izmjenični napon bi se zapisao ekvivalentno samo bi umjesto *dc* pisalo *ac* te je potrebno napisati sinusnu funkciju [20]. Zapis za otpornik i zavojnicu je ostao isti jedino su oznake napisane malim slovima i dodan je redni broj. Na kraj netlista je potrebno dodati naredbu .end koja označava kraj netlist zapisa.

```
v1 1 0 dc 5
r1 1 0 10
r2 2 0 8
l1 1 2 5

.end
```

Slika 6.1 Primjer netlist mreže razumljiv za LTspice alat

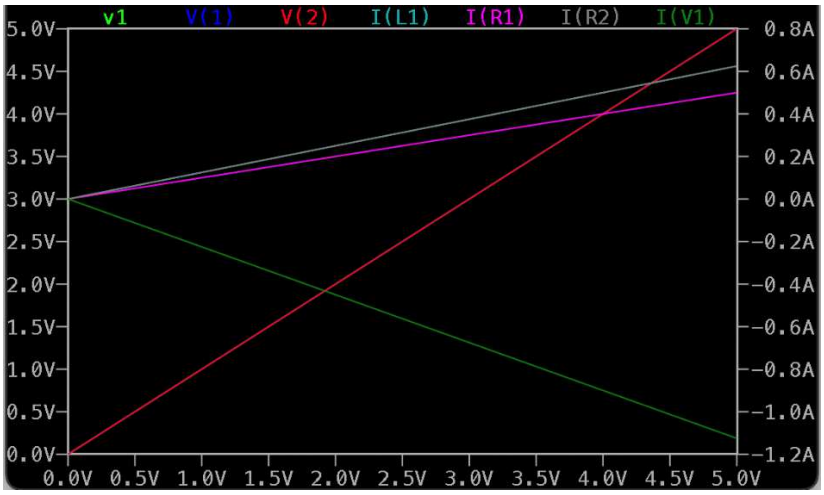
Pokretanjem programa neće se dogoditi ništa jer je dodatno potrebno unijeti neke naredbe koje se žele ispitati. Na primjer, ako se žele ispitati potencijali na svim čvorovima i iznosi struja dodaje se naredba *.op* kao što je prikazano lijevo na slici (Slika 6.2). Ako se želi ispitati promjena napona i struje u krugu ovisno o iznosu izvora istosmjernog napona dodaje se naredba prikazana na slici (Slika 6.2) s desne strane. Tom naredbom se programu zadaje da mijenja iznos izvora napona od 0 do 5 volti s korakom od 1 volt.

```
v1 1 0 dc 5
r1 1 0 10
r2 2 0 8
l1 1 2 5

.op
.end
```

```
v1 1 0 dc 5
r1 1 0 10
r1 2 0 8
l1 1 2 5

.dc v1 0 5 1
.end
```

Slika 6.2 Primjer naredba za simuliranje netlist mreže

Na slici (Slika 6.3) prikazan je graf koji je nastao pokretanjem simulacije. Oznaka v1 označava promjenu iznosa izvora napona od 0 do 5 volti, oznaka V(1) označava iznos potencijala u čvoru 1, a oznaka V(2) označava iznos potencijala u čvoru 2. Te sve tri oznake su na slici prikazane crvenom linijom. Linije za v1 i V(1) se ne vide jer su ispod crvene linije. I(L1) označava iznos struje kroz zavojnicu, I(R1) i I(R2) iznos struje kroz otpornike, a I(V1) iznos struje kroz izvor. Iznos struje na otpornicima raste s porastom izvora napona. Ako se promatra I(R1) kada je iznos izvora 5 volti, onda je iznos struje 0.5 ampera. To je dobiveno prema Ohmovom zakonu tako da se 5 V podijeli s 10 Ω i dobije se 0.5 A. Slično se može provesti izračun za preostale elemente. Iz opisanog je uočena korisnost stvaranja netlista i jednostavnost simuliranja.



Slika 6.3 Iznos potencijala na čvorovima i struje kroz element ovisno o iznosu izvora istosmjernog napona

7. Diskusija

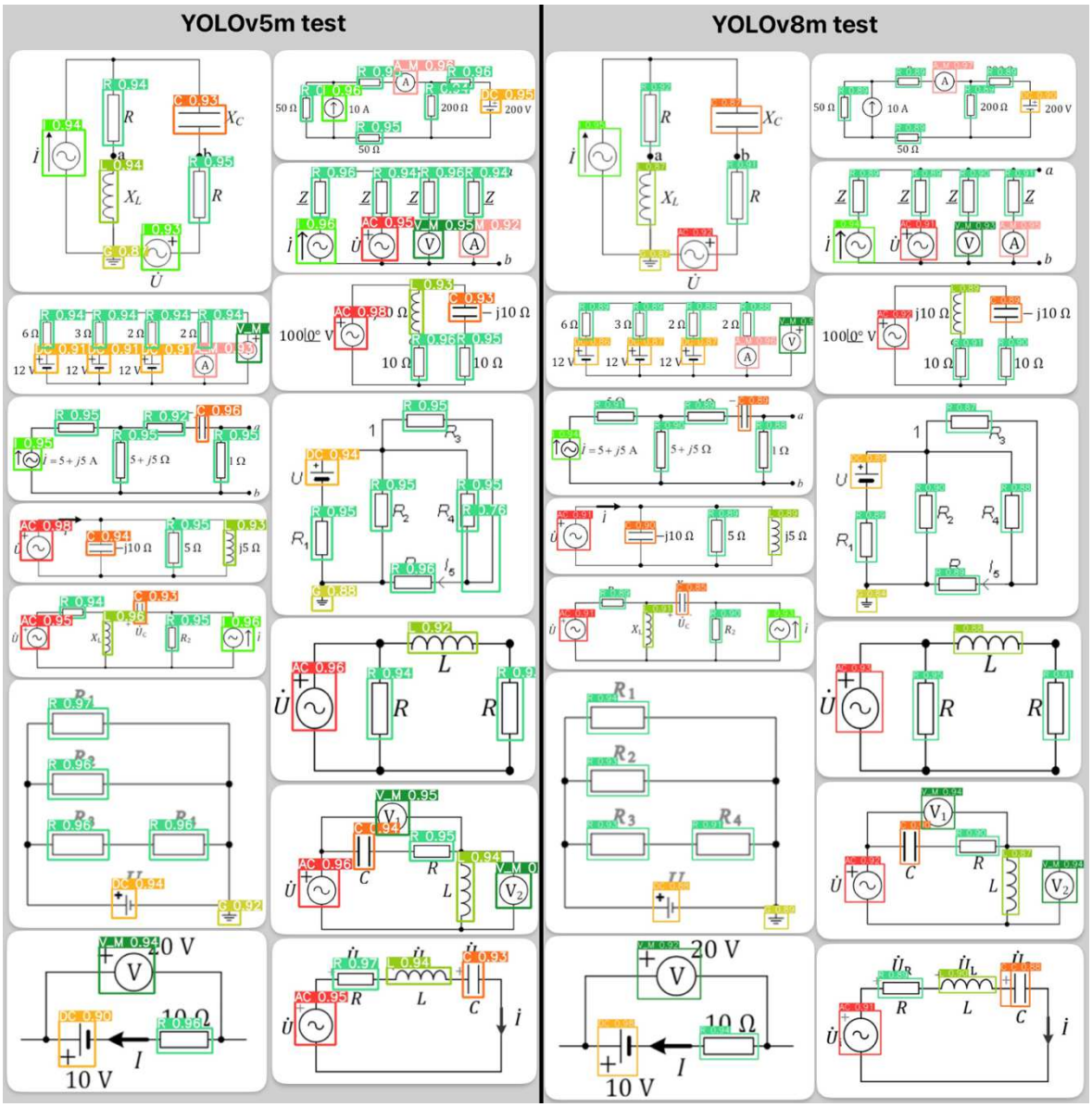
U ovom radu razvijen je računalni algoritam za detekciju elemenata i njihovih karakteristika na slikama električnih shema. U nastavku su rezultati ovog istraživanja uspoređeni s sličnim radovima na području detekcije elemenata električnih shema i stvaranja formalnog zapisa.

U tablici (Tablica 7.1) prikazane su mjere postignute razvijenim modelima. Svijetlo plavom bojom su označeni modeli za detekciju elemenata, a tamno plavom bojom model za detekciju karakteristika. Oba modela za detekciju elemenata postižu veoma uspješne rezultate, no YOLOv8 je malo bolji u usporedbi s modelom YOLOv5. Bitno je napomenuti da YOLOv8 postiže ove rezultate treniranjem u samo 150 epoha, a YOLOv5 u 800 epoha. To čini model YOLOv8 puno bržim. Model za detekciju karakteristika postiže također visok mAP. Taj model sadrži više parametara i klasificira više klasa. Modeli za detekciju elemenata imaju bolji odziv, a model za detekciju karakteristika ima bolju preciznost.

Tablica 7.1 Prikaz mAP, preciznosti i odziva ovisno o modelu

	mAP	Preciznost	Odziv
YOLOv5m	99,4 %	97,8 %	99,1 %
YOLOv8m	99,5 %	98 %	100 %
YOLOv8x	99 %	97, 2%	96, 9 %

Na slici (Slika 7.1) prikazana je detekcija modela YOLOv5 i YOLOv8 na skupu za testiranje. Detekcija modela YOLOv5 prikazana je s lijeve strane slike te je od 83 elementa koji se nalaze u skupu za testiranje njih 82 točno detektirano. Pogrešno detektiran element vidljiv je na prvoj slici u prvom retku. To je element izvora izmjeničnog napona koji je detektiran kao izvor struje. Dodatno je na četvrtoj slici u drugom stupcu detektiran otpornik iako je to zapravo pozadina sheme. Modelom YOLOv8 uspješno je detektirano također 82 elemenata. Jedan element izvora struje nije uopće detektiran što je vidljivo na prvoj slici u zadnjem stupcu. Preostali elementi su dobro detektirani. Ako se usporede pouzdanosti ova dva modela, model YOLOv5 gotovo za sve elemente ima veću pouzdanost.



Slika 7.1 Prikaz usporedbe detektiranih elemenata YOLOv5 i YOLOv8 modela na testnom skupu

Usporedba modela detekcije s drugim radovima prikazan je u tablici (Tablica 7.2). Većina istraživanja bavi se samo detekcijom električnih elemenata zbog složenosti detekcije vodova i čvorova te njihovog povezivanja. U radu [13] navodi se da je točnost rekonstrukcije sheme uspješna u 80 % slučajeva, no nije prikazan nijedan primjer konačnog izgleda netlista. Slično se u radu [5] provodi postupak detekcije vodova i čvorova, no ne stvara se netlist. U radu [17] prikazana je izrada netlista, no moguće je rekonstruirati samo RLC sheme koje imaju izvor istosmjernog napona. RLSA je algoritam koji je opisan u radu [3] kako bi se locirali elementi. To se radi tako da se primjenom RLSA uklanjaju vertikalni i horizontalni vodovi te ostavljene mrlje na slici označuju mjesta elemenata. Nije proveden postupak klasifikacije lociranih elemenata. Rad [4] provodi samo detekciju elemenata u dva koraka koristeći CNN.

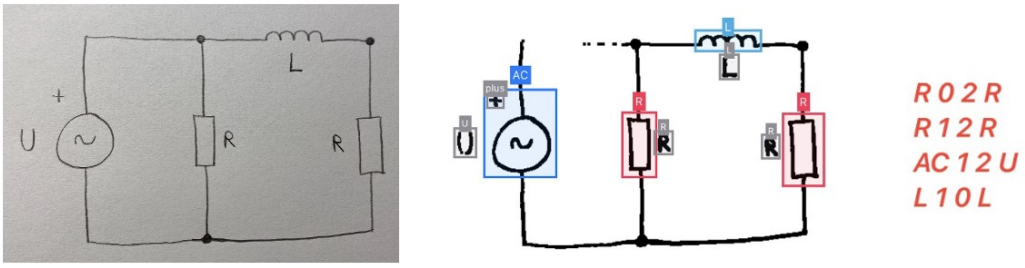
U prvom koraku se grupiraju objekti sličnih oblika, a u drugom se provodi klasifikacija. U radu [5] je postignuta točnost prepoznavanja čvorova od 92 % i točnost prepoznavanja komponenti od 86 %, no ne stvara se netlist. Radovi [21], [23] i [26] koriste SVM algoritam za klasifikaciju elemenata. Dodatno se u radu [26] opisuje segmentacija elemenata, no nije proveden postupak stvaranja netlista. Rad [22] je najsličniji trenutnom radu jer se detektiraju elementi i karakteristike te se generira netlist. Prednost navedenog rada je rad s ručno crtanim slikama shema. U radu [24] korišten je algoritam KNN i točnost klasifikacije iznosi 90 %.

Tablica 7.2 Usporedba rezultata postignutih u dosadašnjim istraživanjima

Rad	Algoritam	mAP/točnost/ F1-score	Ručno crtana shema	Netlist
[13]	YOLOv5	98,2 %	Da	Da
	Netlist	80 %		
[3]	RLSA algoritam za lokalizaciju	91,28 %	Da	Ne
[4]	CNN	97,33 %	Da	Ne
[5]	Sintaktička analiza (prepoznavanje čvorova)	92 %	Da	Ne
	Kombinacija operacija nad slikama (klasifikacija)	86 %		
[17]	OCR – strvaranje netlista	80 %	Da	Da
[21]	SVM	99 %	Da	Ne
[22]	SSD-ResNET, SSD-Mobilenet, Faster RCNN	87 %	Da	Da
	Netlist	82 %		
[23]	SVM	/	Da	Da
[24]	KNN	90 %	Da	Ne
[25]	ANN	83,64 %	Da	Ne
[26]	SVM	87,7 %	Da	Ne
	Segmentacija	92 %		
Trenutni rad	YOLOv8 – detekcija elemenata	99,5 %	Ne	Da
	YOLOv8 – detekcija karakteristika	99 %		
	Algoritam za stvaranje netlista	85,71 %		

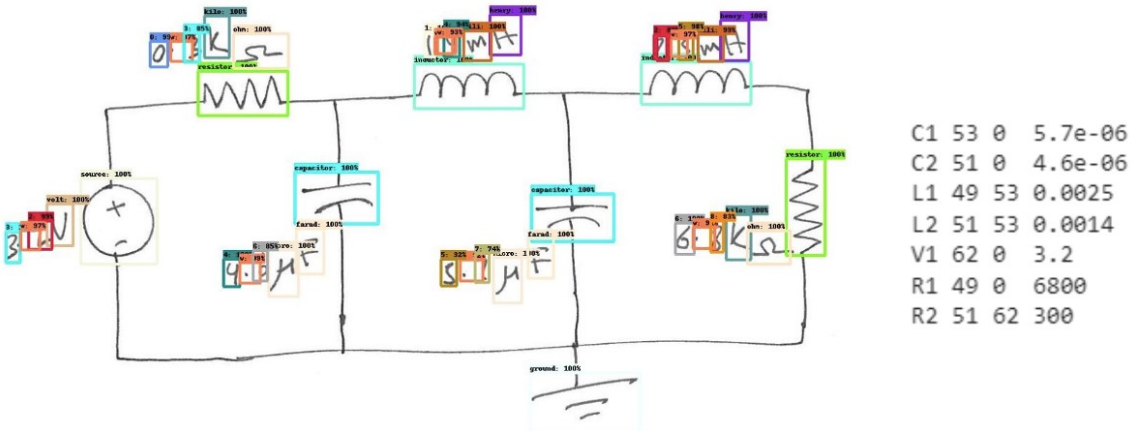
Algoritam izrađen u ovom radu za stvaranje netlista testiran je na testnom skupu od 14 shema te je za ukupno njih 12 ispravno kreiran netlist čime se postiže točnost od 85,71 %. Detekcija elemenata i karakteristika ima mAP vrijednost od oko 99 %. Razvijen algoritam gotovo

savršeno detektira električne elemente na slikama električnih shema. Ispitana je njegova uspješnost detekcije elemenata na ručno crtanim shemama te se model također pokazao uspješnim. Glavni nedostatak ovog istraživanja je korištenje računalno kreiranih slika shema što je jednostavniji problem, nego ručno crtane sheme. Zbog navedenog stvaranje netlista iz ručno crtanih shema postiže nezadovoljavajuće rezultate na složenijim shemama. Potrebna je dodatna optimizacija kako bi se uzela u obzir nesavršenost crtanja shema ovisno o crtaču, kvaliteti papira i olovke te slikanju ili skeniranju slike. Na slici (Slika 7.2) prikazana je uspješna detekcija i stvaranje netlista unutar aplikacije na jednostavnijoj shemi.



Slika 7.2 Prikaz detekcije i stvaranje netlista u aplikaciji

Od promatranih 11 istraživanja, njih samo 4 razvija algoritam za stvaranje netlista. Od tih istraživanja samo u radu [22] se radi potpuni postupak kao i u ovom radu. U ostala 3 istraživanja se stvara netlist, no bez pripadajućih karakteristika, to jest samo se definira između kojih čvorova se nalaze detektirani elementi. Još jedna razlika u radu [22] i ovom radu je da se tamo koristi jedan model za detekciju i elemenata i karakteristika, a u ovom radu su to dva različita modela. Prikaz detekcije i stvorenog netlista iz rada [22] se nalazi na slici (Slika 7.3).



Slika 7.3 Prikaz detekcije na shemi te njezin netlist [22]

Zaključak

Cilj ovog rada bio je razviti algoritam za detekciju elemenata električnih shema i njihovih karakteristika te formirati formalni zapis sheme za daljnje simulacije i testiranja. U radu su provedeni svi potrebni koraci kako bi se uspješno ostvario postupak rekonstrukcije netlista iz slike električne sheme. Za detekciju vodova koristi se Houghova transformacija kojom se pronalaze vodoravne i horizontalne linije te se njihovim presjekom utvrđuju mjesta gdje se nalaze čvorovi. Razvijeni su modeli za detekciju elemenata i detekciju karakteristika koristeći algoritme iz YOLO obitelji. Euklidskom udaljenošću su detektirane karakteristike pridružene detektiranim elementima.

Rezultati pokazuju da algoritam uspješno prepoznaje elemente i njihove karakteristike za uspješnu izgradnju netlista, čime su postignuti glavni ciljevi rada. Na skupu za testiranje postignuta je mAP vrijednost od 99,5 % za detekciju elemenata i 99% za detekciju karakteristika, a uspješnost rekonstrukcije električne sheme u netlist iznosi 85,71 %. Razvijeni algoritam integriran je u mobilnu aplikaciju operacijskog sustava iOS u kojoj je moguća manipulacija nad detektiranim podacima. Automatizacija ovog procesa korisna je jer omogućava bržu analizu električnih shema. Doprinos i primjena ovog rada vidljiva je u unapređenju procesa analize i simulacije električnih shema, što može biti korisno za inženjere ili studente elektrotehnike.

Za daljnji rad preporučuje se povećanje podatkovnog skupa prikupljanjem većeg broja slika eklektičnih shema te dodavanjem novih klasa. Također, razvoj algoritama za detekciju vodova i čvorova testiran je i na ručno crtanim shemama, no potrebno je provesti dodatne optimizacije kako bi se postigla veća uspješnost nad ručno crtanim shemama koje su nepravilnije te predstavljaju teži problem za stvaranje netlista.

Literatura

[1] Alat za kreiranje netlist mreže. Poveznica: https://www.circuit-diagram.org/editor/ ; pristupljeno: ožujak, 2024.

[2] Apple. *SwiftUI*. Poveznica: https://developer.apple.com/xcode/swiftui/ ; pristupljeno: lipanj, 2024.

[3] Bhattacharya, A., Roy, S., Sarkar, N., Malakar, S., & Sarkar, R. *Circuit Component Detection in Offline Handdrawn Electrical/Electronic Circuit Diagram*. In 2020 IEEE Calcutta Conference (CALCON) IEEE. (2020, veljača). str. 80-84.

[4] Dey, M., Mia, S. M., Sarkar, N., Bhattacharya, A., Roy, S., Malakar, S., & Sarkar, R. *A two-stage CNN-based hand-drawn electrical and electronic circuit component recognition system*. Neural Computing and Applications, *33*, (2021), str. 13367-13390.

[5] Edwards, B., & Chandran, V. *Machine recognition of hand-drawn circuit diagrams.* In 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100) IEEE. 6 (2000). str. 3618-3621.

[6] Jocher, G., & Waxmann, S. *Ultralytics YOLOv5 Architecture.* Ultralytics Inc. (2023, studeni). Poveznica: https://docs.ultralytics.com/yolov5/tutorials/architecture_description/; pristupljeno: travanj, 2024.

[7] Solawetz, J., Francesco. *What is YOLOv8? The Ultimate Guide*. Roboflow Blog. (2023, siječanj). Poveznica: https://blog.roboflow.com/whats-new-in-yolov8/; pristupljeno: travanj, 2024.

[8] Solawetz, J., Nelson, J. *How to Train a YOLOv5 Model On a Custom Dataset*. Roboflow Blog. (2020, lipanj). Poveznica: https://blog.roboflow.com/how-to-train-yolov5-on-a-custom-dataset/; pristupljeno: travanj, 2024.

[9] Solawetz, J. *What is YOLOv5? A Guide for Beginners*. Roboflow Blog. (Jun 29, 2020). Poveznica: https://blog.roboflow.com/yolov5-improvements-and-evaluation/; pristupljeno: travanj, 2024.

[10] Nelson, J. *What is YOLO? The Ultimate Guide*. Roboflow Blog. (2021, lipanj). Poveznica: https://blog.roboflow.com/guide-to-yolo-models/; pristupljeno: travanj, 2024.

[11] Khan, A. A., Laghari, A. A., & Awan, S. A. *Machine learning in computer vision: a review*. EAI Endorsed Transactions on Scalable Information Systems, 8(32), (2021). e4-e4.

[12] Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. *A survey of convolutional neural networks: analysis, applications, and prospects*. IEEE transactions on neural networks and learning systems, 33(12), (2021). str. 6999-7019.

[13] Rachala, R. R. & Panicker, M. R. *Hand-Drawn Electrical Circuit Recognition using Object Detection and Node Recognition*. SN Computer Science, 3(3), (2022), 244.

[14] Rigens, G. Swift MVP: *A step-by-step guide for clean code*. Medium. (2023, lipanj). Poveznica: https://medium.com/@gbrigens/swift-mvp-a-step-by-step-guide-for-clean-code-869ec8eb4b22; pristupljeno: lipanj, 2024.

[15] Roboflow. *YOLOv5 PyTorch TXT*. Roboflow Blog. Poveznica: https://roboflow.com/formats/yolov5-pytorch-txt?ref=blog.roboflow.com; pristupljeno: svibanj, 2024.

[16] Sebe, N. *Machine learning in computer vision* (Vol. 29). Springer Science & Business Media. (2005).

[17] Sridar, S. & Subramanian, K. *Circuit recognition using netlist*. In 2013 IEEE Second International Conference on Image Information Processing (ICIIP-2013), IEEE. (2013). str. 242-246.

[18] Ultralytics. *Performance Metrics Deep Dive*. (2023). Poveznica: https://docs.ultralytics.com/guides/yolo-performance-metrics/; pristupljeno: travanj, 2024.

[19] Škopljanac-Mačina, F. *Osnove elektrotehnike*. Interaktivni nastavni materijali. Poveznica: https://osnove.tel.fer.hr/vjezbeoe/DC_5.htm?x=1; pristupljeno: veljača, 2024.

[20] EETech Media. *Example Circuits and Netlists.* All About Circuits. Poveznica: https://www.allaboutcircuits.com/textbook/reference/chpt-7/example-circuits-and-netlists/; pristupljeno: lipanj, 2024.

[21] Lakshman Naika, R., Dinesh, R., & Prabhanjan, S. *Handwritten electric circuit diagram recognition: An approach based on finite state machine.* Int J Mach Learn Comput, 9, (2019). str. 374-380.

[22] Uzair, W., Chai, D., & Rassau, A. *Automated Netlist generation from offline hand-drawn circuit diagrams.* In 2023 International Conference on Digital Image Computing: Techniques and Applications (DICTA) IEEE. (2023). str. 364-370.

[23] Mohan, A., Mohan, A., Indushree, B., Malavikaa, M., & Narendra, C. P. *Generation of Netlist from a Hand drawn Circuit through Image Processing and Machine Learning.* In 2022 2nd International Conference on Artificial Intelligence and Signal Processing (AISP) IEEE. (2022). str. 1-4.

[24] Dewangan, A., & Dhole, A. *KNN based hand drawn electrical circuit recognition.* Int J Res Appl Sci Eng Technol, 6, (2018). str. 1-6.

[25] Rabbani, M., Khoshkangini, R., Nagendraswamy, H. S., & Conti, M. *Hand drawn optical circuit recognition.* Procedia Computer Science, 84, (2016). str. 41-48.

[26] Moetesum, M., Younus, S. W., Warsi, M. A., & Siddiqi, I. *Segmentation and recognition of electronic components in hand-drawn circuit diagrams*. EAI Endorsed Transactions on Scalable Information Systems, 5(16), (2018). e12-e12.

[27] Lazar, L. *Računalna detekcija elemenata strujnog kruga na električnim shemama.* Završni rad. Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva, 2022.

[28] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., & Zitnick, C. L. *Microsoft COCO: Common objects in context*. In Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, Proceedings, Springer International Publishing. Part V 13 (2014). str. 740-755.

Razvoj aplikacije za stvaranje formalnog računalnog zapisa električnih shema uz razvoj računalnog algoritma za detekciju električnih elemenata i njihovih karakteristika na slikama shema električnih krugova

Sažetak

U ovom radu opisan je razvoj aplikacije za generiranje formalnog računalnog zapisa električnih shema koji se naziva netlist. Korištenjem YOLO algoritama razvijen je računalni algoritma za detekciju električnih elemenata i njihovih karakteristika na slikama shema električnih krugova. Model je razvijen i testiran na slikama shema korištenih na kolegiju Osnove elektrotehnike na Fakultetu elektrotehnike i računarstva. Integriranjem razvijenog modela u aplikaciju omogućena je ručna prilagodba detektiranih elemenata i njihovih karakteristika. Kao rezultat, aplikacija generira formalni zapis sheme koji može služiti za izvođenje simulacija. Programsko rješenje implementirano je u programskim jezicima Python i Swift.

Ključne riječi: strojno učenje, računalni vid, detekcija, YOLO, prepoznavanje električnih krugova, električna shema, netlist, iOS

The development of an application for creating a formal computer record of electrical schematics, along with the development of a computer algorithm for the recognition of electrical elements and their characteristics on images of electrical schematics

Abstract

This paper describes the development of an application for generating a formal computer record of electrical schematics, known as a netlist. Using YOLO algorithms, a computer model was developed for the detection of electrical elements and their characteristics on images of electrical schematics. The model was developed and tested on images from the Fundamentals of Electrical Engineering course at the Faculty of Electrical Engineering and Computing. The developed model is integrated into an application that allows for manual adjustment of detected elements and their characteristics. As a result, the application generates a formal schematic representation suitable for performing simulations. The software solution is implemented in Python and Swift programming languages.

Keywords: machine learning, computer vision, detection, YOLO, circuit recognition, electrical schematic, netlist, iOS