Tartu Ülikool
Arvutiteaduse instituut
Informaatika õppekava

Joonas Tiitson

PCB komponentide tuvastamine

Bakalaureusetöö (9 EAP)

Juhendaja:

Kristo Raun, PhD

Tartu 2025

PCB komponentide tuvastamine

Lühikokkuvõte:

Võtmesõnad: pildituvastus, YOLO, ResNeXt, suunatuvastus

Sisukord

1. Sissejuhatus

Elektroonikatööstuses toodetakse trükkplaate, mille peal on erinevaid elektroonikakomponendid näiteks transistorid, takistid, pistikud, jne. Komponendid on üsna väikesed ja neid on iga trükkplaadi peal palju, sellepärast on nende montaaži kvaliteedi hindamine täpne ning rutiinne töö. Samuti võiks olla igale trükkplaadile kuluv aeg võimalikult väike, kuna siis on võimalik testimisel sammu pidada tootmisega, ilma protsessi aeglustamata.

Tänapäeval eksisteerivad mitmed automaatsed lahendused, et vähendada manuaalset tööd trükkplaatide hindamisel. Enamikud neist on kahjuks üsnagi kallid ning nende ülesseadmine ei ole lihtne. Odavamat lahendust pakub kaamerapõhine kontroll, mis ei pruugi olla nii täpne kui muud meetodid, kuid võimaldaks lahendada osa probleemist automatiseerimisega. Sellist lahendust võimaldavad erinevad masinõppel põhinevad pildituvastus meetodid. Eesmärgiks on luua mudel, mis on täpne ning usaldusväärne trükkplaadi komponentide tuvastamisel. Andes ette trükkplaadil leiduvate komponentide kogumi ja trükkplaadi pildi, peaks lahendus ütlema, mis on puudu. Paljude komponentide puhul on oluline ka tuvastada selle orientatsiooni.

Tähtis on ka trükkplaatide kiire analüüsimine, müra eiramine ning väikeste komponentide märkimine, mis teeb lahenduste loomise keeruliseks. Ei saa kasutada pika analüüsimise ajaga meetodeid, mis aeglustaks kogu tootmist. Ega ei saa ka kasutada väga kiireid, kuid tihti vigu tegevaid mudeleid ega meetodeid. Selleks, et oleks mõõdetav eesmärk on see subjektiivselt määratud, kuna puudus määratud mõõdupuu. Eesmärgiks on märkida vähemalt 90% vigastest plaatidest vigaseks ning lasta 90% terveid plaate läbi ilma valesti vigaseks märkimata.

Kuna ideaalis peaks olema loodud tarkvara vabalt ja tasuta kasutatav kõigile, kes seda tahavad, siis võiksid olla ka kõik lisarakendused ja masinõppe mudelid tasuta kasutatavad kõigile.

Järgnevalt on kirjeldatud töö struktuur. Teises peatükis on kirjutatud töö teoreetilisest taustast, kus on uuritud masinõppimise ja objekti tuvastuse kohta. Samuti erinevate abistavate protsesside kohta ning ka andmete märgestuse kohta. Sellele järgneb metoodika peatükk, kus on kirjeldatud lahenduse loome protsesse ning on toodud välja toodud ka lahenduse käivituslikku ajakulu ning täpsust komponentide tuvastusel. Peale seda on tulemuste analüüs kus tehakse järeldusi ning arutletakse saadud tulemuste üle. Viimases peatükis võetakse kokku tulemused ning arutletakse, mis tulevikus peaks edasi arendama, mida peaks lahenduse juures muutma ning mis tulemused ütlevad üleüldiselt.

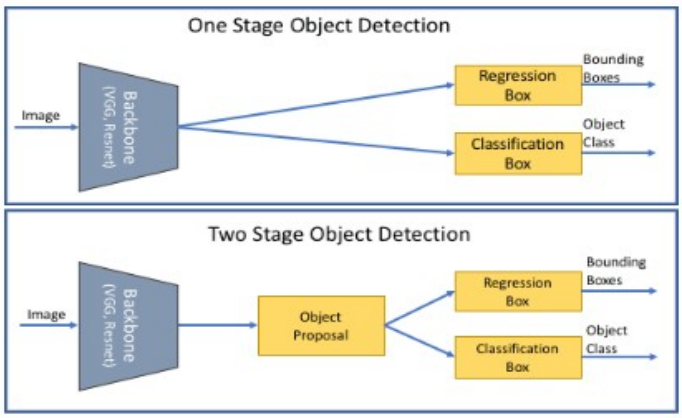
2. Kirjanduse ülevaade

Töö keskendub trükkplaatidelt (*PCB*) komponentide tüübi ning orientatsiooni tuvastusele masinõppe lahendustega. See on oluline tööstusliinide suurema automatiseerimise ning üldistusvõimsuse jaoks. Samuti on tähtis, et tuleviku jaoks oleks olemas ka meetodeid, et vähendada mudelite arenduses manuaalset tööd, kuna lahendus võiks olla võimalikult laialt rakendatav.

Nende eesmärkide täitmiseks on vajalik kirjeldada võimalikke lahendusi lahenduse loomise käigus vajalikest lahendustest ning nende taustadest. Kirjanduse ülevaate esimeses alampeatükis on kirjeldatud erinevaid tuntud ning laialt kasutatud pildituvastus mudeleid ning nende töö põhimõtteid. Teises alampeatükis on juttu YOLO mudelist eraldi, kuna see on tänapäeval kujunenud pildituvastuse eestvedajaks tööstuses rakendamisel. Kolmandas peatükis tuuakse välja pooljuhendatud õppe võimalused, mida on varasemalt loodud, et edaspidisel arendusel oleks vaja vähem inimeste manuaalset tööd mudeli parandamisel. Neljandas alampeatükis on toodud välja ka *Zero-Shot* masinõppe võimalus, mis samuti aitaks tulevikus rakendades vähendada manuaalset tööd. Viimane alampeatükk kirjeldab kuidas on võimalik kasutada masinõppe mudeleid, et määrata objektide suunda.

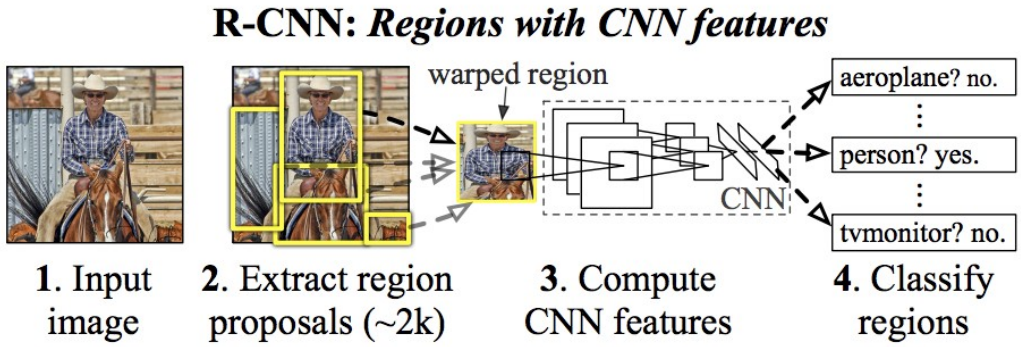
2.1 Pildituvastuse mudelite ülevaade

Laialdaselt on kasutuses tänapäeval kahe erineva struktuuriga pildituvastuse mudelit (Andreu 2018). Nendeks on üheastmelised objekti tuvastuse mudelid nagu näiteks YOLO (*you only look once*) ning kaheastmelised objekti tuvastuse mudelid nagu näiteks R-CNN (*Region-based Convolutional Neural Network*) (Andreu 2018). Nende peamine erinevus tuleneb sellest, et kaheastmelised mudelid kõigepealt valivad piirkonnad, millelt tuvastada, ja seejärel klassifitseerivad leitud ala, kuid üheastmelised mudelid teevad mõlemad sammud korraga (Joonis 1).



(Joonis 1. ühe- ja kaheastmelise mudeli üldine struktuur, Andreu 2018)

Järgnev seletus R-CNN tööpõhimõttest, toetub peamiselt „14.8. Region-based CNNs (R-CNNs)" peatükile, raamatust „Dive into deep learning" (Zhang 2024). R-CNN mudelid töötavad mitmeastmeliselt, tuvastuseks läbivad terve pildi, kust valivad mitu huvi pakkuvat regiooni. Neid regioone omakorda muudetakse, nii et oleksid vajalike mõõtmetega närvivõrgus (*neural network*) töötlemiseks. Igat tüüpi objekti jaoks luuakse toetav vektormasin (*Support vector machine*), mis määrab kas valitud alas on selle vektormasina tuvastatav objekt. (Joonis 2)



(Joonis 2. RCNN mudeli ülesehitus, Girshick 2014)

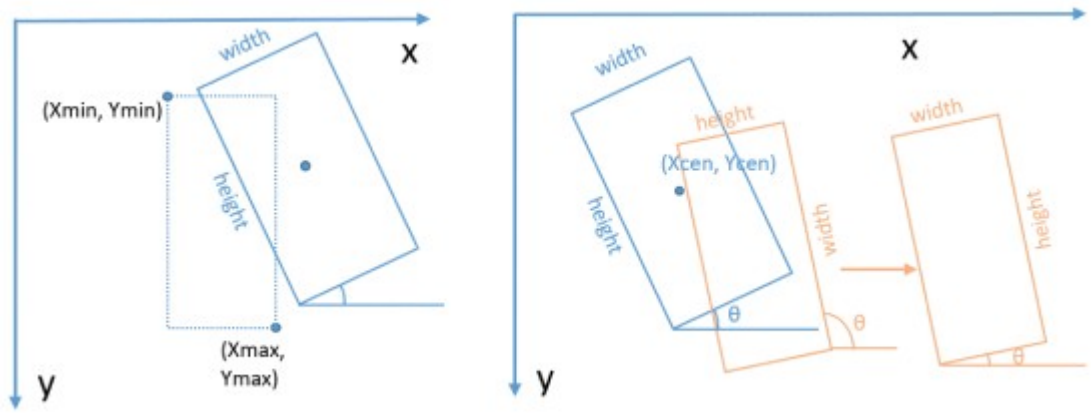
Kui kõik vektormasinad on oma tulemused saanud tuvastatava ala kohta, antakse need edasi lineaarregressiooni mudelile, mille abil on võimalik lõpuks leida, mis objekt asub vaatluse all oleval alal. Nii tehakse iga algselt huvi pakkunud alaga. Kuna töötlemine on mitmeastmeline, siis tihti on see väga aeglane ja ressursikulukas.

YOLO mudel on uuem ning loodud kiireks pildituvastuseks. Selle struktuur on lihtsam kui R-CNN mudelitel, omades ainult ühte närvivõrku, mis töötleb pilti otse (Redmon 2016). Mudel tuvastab samaaegselt piltidelt piirdekaste (*Bounding box*) ning ka ennustab, mis objektiga on tegu (Redmon 2016). Seega on mudeli sisendiks pilt, millelt on vaja tuvastada objekte ja tulemuseks on pildilt tuvastatud objektide piirdekastid ning klassid. Mudeli eelisteks on selle kiirus, kuid võrreldes R-CNN mudelitele on see vähem täpne.

2.2 YOLO võrreldes teiste mudelitega

Uurimistöös on kasutusel YOLO mudel, et tuvastada PCB pealt erinevaid komponente. Mõnel komponendil on oluline selle suund. Kuna YOLO mudelid ei saa mitu korda pilti läbida, siis tekib probleem selle ülesandega ja tuleb kasuks kaheastmeline mudel (Haryono 2023). Algseks objekti tuvastamiseks võib ikka kasutada YOLO mudelit, aga suuna määramiseks saab lisada teise astme (Haryono 2023). Kuid selline kaheastmeline tegevus on arvutustelt keeruline ja aeglasem kui üheastmeline meetod, kasutades muu arhitektuuri mudelit, näiteks ResNeXt101

(Haryono 2023). Objekti ja selle suuna tuvastamiseks on vaja kõigepealt märkida selle asukoht horisontaalse piirdekasti (*HBB*) ja suund orienteeritud piirdekastiga (*OBB*) (Haryono 2023).



(Joonis 3. Näide horisontaalsest ja suunatud piirdekastist. Wang 2019)

YOLO mudel on aga väga kasulik, kui pildilt on vaja lihtsalt tuvastada kiirelt otsitavaid komponente (Yang 2023). YOLO mudel on nii kiire kuna kasutab väheseid kõrvalisi teeke ning on kirjutatud otse masinkeelele lähedastes C ja CUDA keeles (Yang 2023). C keel on tuntud kiire keelena, kuna erinevalt teistest levinud kõrge taseme programmeerimiskeeltest, nagu näiteks Python, on C keel suhteliselt väheste abstraktsiooni kihtidega, seega on tegu masinkeelele lähedasema keelega, mis kiirendab selle tööd (C Standards Committe 2025). Samuti jooksutatakse C programme kompileerituna, mis eemaldab jooksutamise ajal keele kompileerimisele kuluva aja (C Standards Committe 2025). CUDA on NVIDIA poolt loodud programmeerimisplatvorm, mis laseb koodi jooksutada graafikakaartidel ning on loodud paralleelprogrammeerimise lihtsustamiseks (NVIDIA 2025). Kuna graafikakaardid on loodud tegelema mitmete lihtsate ja sarnaste ülesannetega korraga, siis pildituvastuse mudelite treenimise ja tuvastamise üleviimine graafikakaardile aitab kiirendada mainitud protsesse. ResNeXt ja YOLO lähenevad samale ülesandele eri nurkade alt. Nende struktuuriline ülesehitus on vägagi erinev, millega kaasnevad nii eelised kui ka omad puudused. Näiteks peaks YOLO olema küll kiirem kui ResNeXt, kuid see-eest ka vähem täpne ning suunatuvastusel halvem.

YOLO mudeli eelisteks on veel andmete säilitamine treenimisprotsessi vältel. See aitab ära hoida valede seoste tekkimist mudelis. Samuti aitab andmete alleshoidmine tulevikus mudeli ümbertreenimisel, kui näiteks tahta lisada mingit objekti, mida see tuvastaks hiljem lisaks (Wang 2024). Tänu GELAN ja PGI objekti tuvastuse koos kasutamisele saavutab YOLOv9 (üheksas versioon YOLO mudelist, mis on ka vabavara MIT litsentsiga) teistest reaalajas tuvastavatest algoritmidest kõrgema täpsuse (Wang 2024).

GELAN ja PGI on YOLO tiimi poolt arendatud pildiandmete töötlemis metoodikad, nagu ka ResNet, mille eesmärk on pildil välja tuua tähtsad andmed, ehk tuua objekt esile (Wang 2024). PGI ja GELANi loomise põhimõtteks oli vähendada pildituvastus nõudlikkust tehisnärvivõrkude sügavuse osas (Wang 2024). GELAN suudab effektiivsemalt kasutada madalamaid närvivõrke kui sellele eelnevalt eksisteerinud lahendused (Wang 2024). PGI tööpõhimõtteks on kasutada eelnevalt töötlemata andmeid lisa kihis koos juba töödeldud andmetega, et hoida ära ekstra töötlemisaega, mis laiemate kihtide dubleerimine toob endaga kaasa, samas aitab see ka hoida ära andmete liigset kadu mis tihti juhtub kui närvivõrk järjest kitsamaks muutub (Wang 2024).

2.3 Pooljuhendatud õpe

Pooljuhendatud õpe on oluline, et vähendada manuaalset tööd andmete märgendamisel. Põhimõtteks on kasutada koos nii eelnevalt märgendatud või klassifitseeritud andmeid kui ka eelnevalt märgendamata andmeid, millel on siiski samad omadused märgendatud andmestikuga (Bergmann 2025). See on kasulik tulevikus aja kokkuhoidmiseks ja mudelite täpsuse parandamiseks ilma suure inimtööta. Pooljuhendatud õpet on laialdasemalt kasutatud R-CNN mudelites, kuid YOLO mudelites pole see nii levinud. Peamiseks põhjuseks on R-CNN mudelite kaheastmeline ülesehitus, mis teeb pooljuhendatud õppe rakendamise lihtsamaks (Zhou 2022). Kaheastmelistes mudelites saab kasutada algse mudeli esimese etapi regioonide soovitusi, mida saab hiljem kasutada modifitseeritud sisendite, näiteks hägustatud piltide puhul (Zhou 2022). Edu on saavutatud YOLOv5-le õpetaja-õpilase mudeli loomisega. Õpetaja-õpilase mudeli põhimõtteks on treenida algselt üks masinõppe mudel ainult inimmärgendatud andmete põhjal, nii nimetatud õpetaja mudel. Seda õpetaja mudelit kasutatakse omakorda enne tundmatute andmete märgendamisel. Kui on valminud masinõppe mudeli märgitud andmestik, ühendatakse see andmestik siis eelnevalt loodud inimmärgendatud andmestikuga. See ongi lõplik ehk õpilas mudeli andmestik, mida kasutatakse, et treenida välja uus ehk õpilas mudel.

YOLOv5 õpetaja-õpilas mudelis kasutati üle 1300 videot, millest treenimiseks valiti välja 12 924 kaadrit 21-st koolist ühes rajoonis, kus käsitsi märgendati ära haigutavad inimesed (Zhou 2022). Pärast treeniti mudel nende piltide peal ning kasutati seda mudelit, et ära märgistada ka teisest rajoonist pärit videod (Zhou 2022). Tekkinud andmekogumit kasutati, et omakorda treenida veel üks uus YOLOv5 mudel, mis saavutas testhulgal parema tulemuse, kui algne mudel (Zhou 2022). Pooljuhendatud õpe on kasulik, et luua täpsemaid mudeleid ning vähendada andmekogumi loomiseks kulutatavat aega.

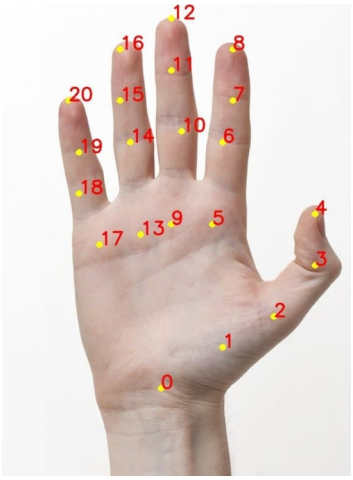
2.4 *Zero-Shot* õpe

Zero-Shot õppe põhimõte võtab eeskujuks inimlaadse õppimise (Larochelle 2008). Põhimõte sai ametlikult avaldatud 2008. aastal, kuid praktilisi lahendusi ning teste tehti juba varasemalt (Larochelle 2008). *Zero-Shot* õppe peamine eesmärk on võimaldada info tuvastamist ja klassifitseerimist, mida eelnevas treeninghulgas ei leidunud. Selline lähenemine aitab vähendada manuaalse märgendamise hulka (Larochelle 2008). Tuvastatavaid objekte on võimalik läbi üldisemate seletuste või abstraktsemate kujunditega lihtsustada ning selliste lihtsustuste abil on võimalik ka täiesti uut infot tuvastada. Artiklis Zero-data Learning of New Tasks on mainitud näiteks uute kirjamärkide tuvastamist, kui on antud abstraktsem märgi üldine struktuur, ja treeninghulgas on sarnaselt juba treenitud välja mudel teisi kirjamärke seondama sarnastele abstraktsetele kujutistele, tekib mudelil võimalus ka uusi varem nägemata objekte õigesti tuvastada.

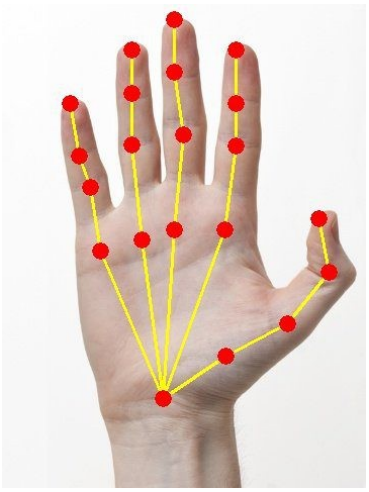
2.5 Suuna tuvastus YOLO mudeliga

Yolo tuvastus kasutab üldiselt eelnevalt mainitud piirkaste (*bounding box*), kuid objektide suuna tuvastuseks on kasu võtmepunktidest (*Keypoints*) ning skelettidest (*Skeleton*). Võtmepunktid toimivad kui lihtsalt koordinaadid pildil, näiteks joonisel 4 märgitud punktid. Võtmepunktid on ka osa skelettidest, kus nendele on lisatud relatsioon mingi teise võtmepunktiga, nii on võimalik kujutada keerulisemaid struktuure näiteks inimese kätt, illustreeritud joonisel 5.

Võtmepunktide ja skelettide abil saab tuvastada objekti servi ning olenevalt nende asetsusest saab määrata ka objektide suundi. Näiteks on võimalik teha vahet paremal ja vasakul asetsevatel munajuhadel ning tuvastada osalise peitumisega objekte (Boonkong 2024). Suurim eelis oli osaliselt peituvate objektide tuvastamine, mis ei peaks selle projekti puhul probleemi tekitama, kuid nende lahendus kasutada võtmepunktidel baseeruvat tuvastust arvatavasti lihtsustaks ka objektide suuna tuvastust.



(Joonis 4. Inimese käe peale märgitud võtmepunktid, Gupta 2018)



(Joonis 5. Inimese käe peal märgitud käe asetus skelett, Gupta 2018)

3. Metoodika

Selle peatüki eesmärk on anda lugejale ülevaade loomisprotsessist ning lõplikust valminud rakendusest. Rakenduse eesmärk on lahendada sissejuhatuses püstitatud probleem. Riistvaraliselt on tulemuste loomiseks kasutatud kaamerat, mille kaadrisagedus on 30 kaadrit sekundis ning arvutit, millel on protsessoriks i5-11400f ja graafikakaardiks RTX 3060.

Kuigi kasutatavad meetodid on arvutuslikult väga nõudlikud, peab süsteem olema kergesti teisaldatav ja kopeeritav või siis kaugelt ligipääsetav. Kuna tooteid võidakse filmida hoones mitmes erinevas kohas, siis on mõistlik, kui kogu süsteem on lihtsalt ülesseatav või video saadetaks ühte kesksesse arvutisse, et seal saaks kaugelt tuvastust kasutada. Miinusteks oleks muidugi suurenenud latentsus, kuid arvutuskiirus, mis nii oleks võimalik saavutada, korvaks selle ära.

3.1 Andmekogumi loomine

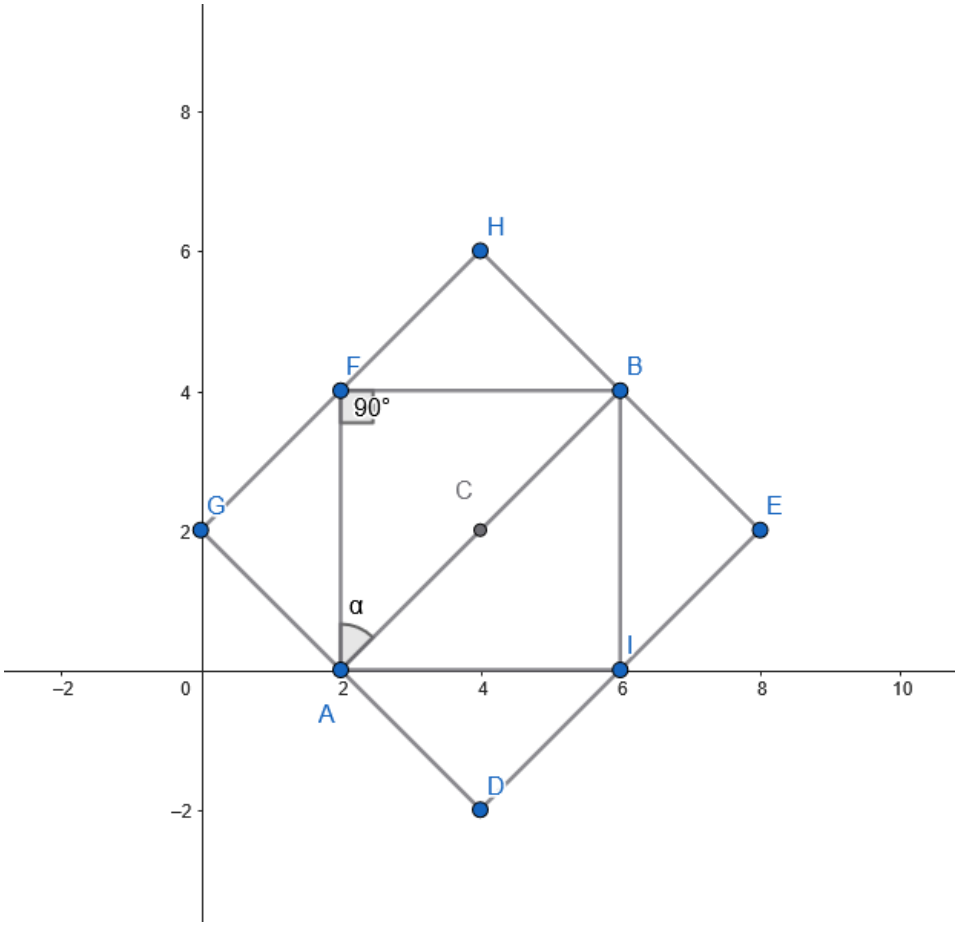
Andmekogumiks kasutatakse kombinatsiooni juba eksisteerivatest märgendatud PCB kogumitest, üksikkomponentide piltidest ja olemasolevate trükkplaatide piltidest. Märgenduse teostas töö autor käesoleva töö raames. Andmekogum ise koosneb piltidest ja nendelt vastavalt leitud komponentide loeteluga failidest, mille saab eraldada treeningkogumiks ja testkogumiks.

3.2 Andmehulga märgendamine

Andmehulk loodi kasutades Pythoni skripti, mis filmis käivitamisel ees olevat trükkplaati, iga kaader filmimisest salvestati eraldi pildina. Sellega pildistati erinevate suundade alt võimalikult palju erinevaid kättesaadavaid trükkplaate. Andmehulga märgendamiseks kasutati CVAT-i. Peamiseks eeliseks CVAT-il oli selle lai märgenduse formaatide valik ning kergesti mõistetav UI. Lõplikuks andmehulgaks valmis 868 pilti, mille peal on märgitud skeleti põhiselt kondensaatorite polaarsus. Seda andmehulka kasutati ka *OBB* sünteesimiseks, mida oli vaja, et ResNeXt põhist mudelit treenida.

OBB sünteesimiseks kasutati iga skeleti kahte otspunkti. Järgneva *OBB* sünteesimise seletuse käiguks tuleb abiks joonis 6. Kahetipulise skeleti otspunktideks on joonisel A ja B. Nende kauguse üksteisest võtsin ruudu kujulise *OBB* laiuseks ja kõrguseks. Otspunktide vahelise joone keskpunkt C võeti ruudu keskpunktiks. Nurk (joonisel 6 kui α) piirdkasti kalde saamiseks loodi arktangensi abil, kasutades otspunkide A ja B vastavate koordinaatide vahesid, et saada lõikude AF ja BF pikkused. Kui arktangens tuli negatiivne siis sellele summeeriti 360 kraadi et tegu oleks alati positiivse nurgaga. Tehtud *OBB*-d läksid kirja kujul: keskpunkti X koordinaat,

kekspunkti Y koordinaat, *OBB* kõrgus ehk A ja B kaugus, *OBB* laius ehk samuti A ja B kaugus, nurk α kraadidena.



(Joonis 6. OBB sünteesimise illustreerimiseks abistav joonis)

3.3 YOLO ja FastRCNN tulemused

Validatsiooni hulga puhul saavutas YOLO mudel kondensaatorite märkimisel 39% täpsuse. Tausta tuvastamisega sai mudel hakkama 100% ajast, mis tähendab, et mudel oli pigem konservatiivne oma tuvastusega ning selle täpsust aitaks madalam tuvastuslävend. Kasutades mitmekordset käivitust ühe trükkplaadi vaatamisel, on võimalik parandada tulemusi. Mudel on väga tundlik valgustingimuste muutustele, seega on mudeli täpsusele suureks abiks ühtlane valgustus. Jooksutades suudab mudel töötada reaalajaliselt.

FasterRCNN mudel vajas palju rohkem ettevalmistust. FasterRCNN mudeli selgrooks kasutati ResNet50-t, kuna see võimaldas mõistliku ajaga treenida mudelit et iteratiivselt parandada tekkivaid vigu. Tuli ümber teha mudeli väljundkiht, et see suudaks ennustada erikujul olevaid suunatud piirdkaste. Piirdkastid on määratud keskpunkti x ja y koordinaatidega, laiuse, pikkuse ja kraadidega. Käivitusel oli eelnevalt mainitud riistvara peal iga kaadri töötlemisaeg umbes

sekund. Seega on tegu kõvasti aeglasema käivitusega ning vähem võimsa riistvara puhul on see problemaatiline.

3.4 Ülevaade praktilisest rakendusest

Praktilise lahendusena eksisteerib praegu YOLO ja ResNeXt treenimiseks andmehulgad vajalikes formaatides. Samuti mõlema mudeli treenimiseks on loodud eraldi skriptid, YOLO kasutab tuvastamiseks skelette ning ResNeXt kasutab orienteeritud piirdkaste. Mõlema mudeli testimiseks on loodud funktsioonid kaamerapildilt objektide tuvastamiseks ning tuvastatud tulemused märgitakse kaamera pildile ära. YOLO mudeli puhul on võimalik muutuja abil määrata mitme eelneva tuvastuse märgendust tuleb kuvada, et oleks võimalik stabiilsemalt jälgida tuvastusi, kuna muidu on iga pilt veidi erinevad nii et tuvastus võib lünklik olla.

Tulevikus saab luua praktiliseks lahenduseks GUI. Selle idee oleks kasutada etalonipõhist kontrolli, kus õigesti loodud trükkplaat antakse kaamerale ette, valitakse sobiv tuvastus, kus on ideaalis märgitud kõik vajalikud komponendid õigesti. Siis järgnevalt võrreldakse kõiki järgnevaid trükkplaate tuvastatud etaloniga. Kui järgneval plaadil on tuvastusel midagi väga erinev etalonist, siis rakendus annaks veast teada, ning seda saaks kontrollida töötaja.

4. Tulemuste analüüs

Järgnevas peatükis on ResNeXt ja YOLO täpsuse ning riistvaralise koormuse tulemuste analüüs, samuti võrdlus kaamerapõhise kontrolli ja tööstuses eelnevalt levinumate lahendustega. Esimeses alampeatükis arutlen kaamerapõhise kontrolli plussidest ja miinustest käesoleva lahenduse kontekstis. Pärast seda võrreldakse ResNeXt ja YOLO tulemusi ennustamisel, ning arutletakse nende tuvastustäpsuse üle ning arutletakse natuke mudelite koostamise raskuse üle. Viimasena arutlen ResNeXt-i ja YOLO kiiruse ja riistvaralise koormuse üle.

4.1 Kaamerapõhine kontroll

Üldiselt on levinud tööstuses mitmekesisemad lahendused, kus ei kasutata ainult kaamerapilti vigade tuvastuseks. Kuigi sellised süsteemid on kõvasti laiemad ning kontrollivad trükkplaate kasutades mitmeid erinevaid meetodeid, on nad suures osas ikkagi manuaalselt vaja üles seada iga erineva trükkplaadi kohta (Rehman 2019). Ühe kaamera põhise kontrolli puhul on suurimateks eelisteks selle odavus ning lai üldistusvõimalus. Kuna komponendid, mida erinevatel trükkplaatidel tuvastada, on sageli samasugused, siis on võimalik kontrollida ka eelnevalt tundmatuid plaate. Suurimaks probleemiks jäi minu lahenduse puhul vähene täpsus. Peamiselt on vähene tuvastustäpsus tingitud treeninghulga ja tuvastushulga suurest varieeruvusest. Ideeks oli luua valgustuselt ja positsiooniliselt varieeruvat treeninghulk, et treenitud mudelid oleks kasutatavad eri tingimuste all. Sellepärast jäi mudeli validatsiooni täpsus ka üsnagi kesiseks.

4.2 ResNeXt võrreldes YOLO mudeliga

Mõlemad treenitud mudelid jäid 40% täpsuse piiridesse, seega 60% ajast ei suutnud mudelid õigesti märkida komponentide asukohtasid. Positiivsest küljest enamus märgendatud komponente olid ka õigesuunaliselt tuvastatud. Samuti on suureks probleemiks mõlemal mudelil keskkonnas toimuvad muutused. Näiteks lambivalguse all tuvastab mudel hästi, kuid kohe, kui valgus natuke muutub, ei tuvasta mudelid enam midagi. Seda olukorda üritati vältida treeninghulka luues valgust, tuvastuskaugust ja nurka varieerides, kuid siiski jäi mudelite üldistusvõime väheseks. YOLO tulemused olid samuti paremad üldistamisel. Seega ResNeXt mudel jäi oma täpsete tuvastustingimuste tõttu liiga kitsaks. Eeldatavasti on ResNeXt mudelil vaja palju rohkem treeningandmeid, et saavutada aktsepteeritav tulemus.

ResNeXt mudelitega tegelemine on kordades keerulisem kui YOLO mudelitega tegelemine. Kui ei kasuta standardset tagastussüsteemi, mis ResNeXt puhul on tavaline horisontaalne piirdkast, siis on vaja teha eri *ROI* – pea (*Region of Pooling* pea, mis on vajalik *Faster-RCNN*

treenimiseks, valib huvi alasid, millelt objekte tuvastada), kaofunktsioon ning tagastus kiht. Loodud *Faster-RCNN* lahenduses kasutasin S. Ma 2024 aasta artiklis kasutatud FPDIoU sarnast kaofunktsiooni, kuna nende orienteeritud piirdkastid esinesid sarnasel kujul käesoleva töö omadele.

4.3 Mudelite nõudlikkus

Selle peatüki eesmärk on anda ülevaade YOLO ja ResNeXt mudelite individuaalsest käivituskiirusest ning arutleda, mis kontekstis on need kasutatavad ning mis kontekstis võib tekitada mudelite käivitamis koormus probleeme.

YOLO tuvastus on kümnetes kordades kiirem kui ResNeXt-i struktuuril põhinev mudel. YOLO saavutas 30 kaadrit sekundis ehk reaalajas tuvastuse. ResNeXt mudeli tuvastus võttis kaadri kohta keskmiselt 2 sekundit. Seega, kui on vaja kiiremas korras tuvastada, on YOLO nendest kahest õigem. Kuid kuna iga trükkplaadi tuvastuseks võib kuluda rohkem aega täpsuse nimel, siis ka ResNeXt on sobiv lahendus. Võimalik, et vähem võimsa riistvara puhul tekib ResNeXt-il probleeme. Kui tahta tuvastuseks kasutada lokaalseid vähe võimsaid arvuteid, siis jääb ainukeseks lahenduseks YOLO tuvastus.

Eelnevatele tulemustele toetuvalt võib väita, et suurel skaalal rakendamisel on ResNeXt tuvastus kümnetes kordades kallim ning selle täpsus ei õigusta lisakulu, mida võimsam riistvara nõuaks. ResNeXt-l põhinev lahendus nõuaks kas tsentraalset serverit, mis saaks mitme erineva tootmisliini tooteid ükshaaval tuvastada. Kuna igale liinile luua oma arvuti, mis ainult töötaks selle liini objektide töötlemisel oleks väga kallis.

5. Tulemuste arutelu

Järgnevates peatükkides on arutlus, mida saaks tulevikus edasi arendada. Samuti toovad need välja mõningad probleemid, mille lahendamiseks oleks vaja erinevalt tulevikus läheneda, et edasised loodud lahendused oleksid täpsemad ja kasulikumad.

5.1 Märgendatud andmekogu

Manuaalse märgendamise hulka tuleks tulevikus vähendada kasutades pooljuhendatud õppimist. Selle kasutamine aitaks kaasa suure andmekogumi loomisel. Ideaalselt saaks suure osa andmekogumist nii valmis teha peale algse kogumi tegemist, ning kasutada seda edaspidi, et saavutada paremat täpsust mudelis. Veel üheks võimaluseks on kasutada ka *Zero-Shot* õppimist, sellega saaks vähendada tulevikus manuaalse märgendamise vajadust, kuid eeldatavasti selline lähenemine tõstaks ka riistvaralisi nõudeid kogu süsteemile. Kuna selle töö maht ei jõudnud praeguste mudelite abiga *Zero-Shot* ja pooljuhendatud õppe rakendamiseni, jääb see tulevikku edasi arendamiseks. Selliste lahenduste rakendamiseks oleks vaja teha ka muudatus andmete märgenduskujule ja süsteemi struktuurile, et süsteemil oleks võimekus võrrelda etaloni ja tuvastatavat trükkplaati eraldiseisvalt objektituvastusele.

5.2 Mudelite täpsus ning parandused mida rakendada tulevikus

Kuna tulemused jäid allapoole soovitut, oleks vaja muuta tuvastamistingimusi. Kui tulevikus edasi arendada treenitud mudeleid, siis tuleks kasutada mingit kergesti ülesseatavat keskonda, milles treenimise ning tegelikud andmed oleks võimalikult sarnastes tingimustes. Samuti tuleks tulevikus kasutamiseks märgendada veel rohkemaid andmeid ning ka teisi komponente märkida, kuna praegune lahendus määrab ainult trükkplaatide kondensaatorite asukohti ja suundi.

ResNeXt tulemused jäid kehvaks tänu halvale RoI muudatuste implementeerimisele, enamus ajast ei tuvastanud mudel midagi kuna puudus kaotusfunktsioonis karistus komponendi märkimiseta jätmise jaoks. Samuti kõrgemate epohhide arv aitaks ResNeXt-i tööd kuna kõrgeim epohhide arv mida treenimisel kasutati oli vaid 10, testimise jaoks oli tarvilik kasutada väikest epohhide arvu, kuna muidu oleks treenimis aeg liiga suur olnud, et programmi iteratiivselt parandada.

6. Lisad

6.1 Mõisted, terminid ja lühendid

Zero-Shot learning – masinõppes kasutatav meetod, kus mudel kasutab enne olemasolevaid seoseid, et klassifitseerida uut infot.

Printed circuit board – trükkplaat, elektroonikas kasutatav montaažiplaat, millele on võimalik paigaldada elektroonikakomponendid ja need elektriliselt ühendada.

Neural network – eesti keeles närvivõrk, arvutisüsteem, mis on modelleeritud inimaju ja bioloogilise närvivõrgu järgi.

Support vector machine – vektormasin, mis on loodud andmete analüüsimiseks ning klassifitseerimiseks, kasutades suurima erinevuse leidmist.

Bounding box – piirkast, kasutatakse mingi ala märkimiseks.

UI – user interface, eesti keeles kasutajaliides, visuaalne element rakendusel, et võimaldada seda kasutada.

Convolutional neural network – (lühendatult CNN) närvivõrk mis kasutab tunnuste õppimiseks filtrite optimiseerimist. Proovib filtreerida välja ebavajalikke andmeid, et jõuda ennustuseni. Sageli kasutuses pildituvastus ülesannete jaoks.

Skeleton – eesti keeles skelett, üks märgendustüüp, mida kasutatakse piltidel olevate objektide märkimiseks.

Viidatud kirjandus

1. Haryono, A.; Jati, G.; Jatmiko, W. Oriented object detection in satellite images using convolutional neural network based on ResNeXt. ETRI J. 2023.

2. Yang, C-H; Lin, Y-N; Wang S-K; Shen, V.R.L; Tung, Y-C; Lin J-F. An Edge Computing System for Fast Image Recognition Based on Convolutional Neural Network and Petri Net Model. Springer Nature. 2023

3. Wang C-Y; Yeh I-H; Liao, H-Y. YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. Arxiv 2024

4. Zhou, H; Jiang, F; Lu, H; SSDA-YOLO: Semi-supervised Domain Adaptive YOLO for Cross-Domain Object Detection. Arxiv 2022

5. Boonkong A; Khampitak K; Kaewfoongrungsi; Namkhun S; Hormdee D; Applying Deep Learning for Occluded Uterus and Fallopian Tube Detection for Laparoscopic Tubal Sterilization. IEEE 2024

6. Zhang, A; Lipton, Z; Li, M; Smola, A J; Dive into deep learning. Cambridge New York Port Melbourne New Delhi Singapore: Cambridge University Press. 2024

7. Redmon, J; Divvala S; Girshick R;Farhadi A; You Only Look Once: Unified, Real-Time Object Detection. Arxiv 2016

8. Larochelle, H; Erhan, D; Bengio Y; Zero-data Learning of New Tasks. Université de Montréal. IEEE 2008

9. Rehman, S; Fei, K; Lai N S; Automated PCB identification and defect detection system (APIDS). IJECE 2019

10. Girshick, R; Donahue, J; Darrell, T; Malik, J; Rich feature hierarchies for accurate object detection and semantic segmentation. Arxiv 2014

11.Andreu O. C; Smeraldi F; Semantic Image Cropping, Queen Mary University of London, Arxiv 2018

12. Y. Wang, Y. Zhang, Y. Zhang, L. Zhao, X. Sun and Z. Guo, SARD: Towards Scale-Aware Rotated Object Detection in Aerial Imagery, IEEE 2019

13. NVIDIA, „What is CUDA?”, https://blogs.nvidia.com/blog/what-is-cuda-2/ , 01.05.2025

14. C Standards Committe, „The C programming language”, https://www.c-language.org/ , 01.05.2025

15. D. Bergmann, „What is semi-supervised learning?”, https://www.ibm.com/think/topics/semi-supervised-learning , 01.05.2025

16. V. Gupta, „Hand Keypoint Detection using Deep Learning and OpenCV”, https://learnopencv.com/hand-keypoint-detection-using-deep-learning-and-opencv/ , 08.10.2018, külastatud 09.05.2025

17. S. Ma, Y. Xu, „FPDIoU Loss: A Loss Function for Efficient Bounding Box Regression of Rotated Object Detection”, Arxiv 16.04.2024