VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INŽINERINĖS GRAFIKOS KATEDRA

Andrius Tunikaitis

PANAŠIOS FORMOS DETALIŲ PAIEŠKOS SISTEMA
SIMILAR FORM PART SEARCH SYSTEM

Baigiamasis magistro darbas

Informacinių technologijų studijų programa, valstybinis kodas 6211BX016
Inžinerinės ir kompiuterinės grafikos specializacija
Informatikos inžinerijos studijų kryptis

Vilnius, 2021

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INŽINERINĖS GRAFIKOS KATEDRA

TVIRTINU
Katedros vedėjas
_____
(Parašas)
doc.dr.Daiva_Makutėnienė
(Vardas, pavardė)
_____
(Data)

Andrius Tunikaitis

PANAŠIOS FORMOS DETALIŲ PAIEŠKOS SISTEMA
SIMILAR FORM PART SEARCH SYSTEM

Baigiamasis magistro darbas

Informacinių technologijų studijų programa, valstybinis kodas 6211BX016
Inžinerinės ir kompiuterinės grafikos specializacija
Informatikos inžinerijos studijų kryptis

Vadovas dr. Saulius Dereškevičius _____ _____
(Moksl. laipsnis/pedag. vardas, vardas, pavardė) (Parašas) (Data)

Lietuvių kalbos konsultantas dr. Vaida Buivydienė _____ _____
(Moksl. laipsnis/pedag. vardas, vardas, pavardė) (Parašas) (Data)

Vilnius, 2021

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INŽINERINĖS GRAFIKOS KATEDRA

TVIRTINU
Katedros vedėjas

Informatikos inžinerijos studijų kryptis

Informacinių technologijų studijų programa, valstybinis kodas 6211BX016

Inžinerinės ir kompiuterinės grafikos specializacija

(Parašas)
Daiva Makutėnienė
(Vardas, pavardė)
2018-10-22
(Data)

BAIGIAMOJO MAGISTRO DARBO UŽDUOTIS

2018-10-22 Nr. 2
Vilnius

Studentui (ei) Andriui Tunikaičiui
(Vardas, pavardė)

Baigiamojo darbo tema:

Panašios formos detalių paieškos sistema

patvirtinta 2021m. gegužės 21 d. dekano potvarkiu Nr. 125 fm

Baigiamojo darbo užbaigimo terminas 2021 m. gegužės 31 d.

BAIGIAMOJO DARBO UŽDUOTIS:

Magistrantui keliamos užduotys:

- išnagrinėti nuotraukų atpažinimo ir kompiuterinio mokymosi principus ir atlikti mokslinės literatūros analizę.
- sukurti programos prototipą, pasitelkiant kompiuterinio mokymosi principus.
- įvertinti ir išnagrinėti programą pagal įgytas žinias ir programos tolesnį tinkamumą naudoti
- pateikti išvadas ir rekomendacijas.

Baigiamojo darbo rengimo konsultantai:

.....

(Moksl. laipsnis/pedag. vardas, vardas, pavardė)

Vadovas ..... dr. Saulius Dereškevičius
(Parašas) (Moksl. laipsnis/pedag.vardas, vardas, pavardė)

Užduotį gavau

.....
(Parašas)
Andrius Tunikaitis
(Vardas, pavardė)
.....
(Data)

Vilniaus Gedimino technikos universitetas
Fundamentinių mokslų fakultetas
Inžinerinės grafikos katedra

ISBN ISSN
Egz. sk. .....
Data .....-.....-.....

Antrosios pakopos studijų **Informacinių technologijų** programos magistro baigiamasis darbas 4

Pavadinimas	**Panašios formos detalių paieškos sistema**
Autorius	**Andrius Tunikaitis**
Vadovas	**Saulius Dereškevičius**

Kalba: lietuvių

Anotacija

Baigiamojo magistro darbo tikslas – siekiant palengvinti konstruktorių darbą ir remiantis išnagrinėta moksline literatūra apie panašios formos objektų paieškos sistemas, sukurti nesudėtingą programos prototipą, kuris gali atpažinti detalių formas iš nuotraukų ir pateikti objekto klasės tikimybę.

Darbe iškeltam tikslui pasiekti yra nagrinėjamos kitos 3D objektų paieškos sistemos, jų panaudojimo galimybės. Taip pat yra išnagrinėti kompiuterio mokymosi (angl. machine learning) ir 3D objektų atpažinimo principai.

Atlikus literatūros analizę buvo prieita prie sprendimo sukurti programos prototipą naudojant 3D objektų atpažinimą iš objektų projekcijų vaizdų, o juos nagrinėti ir vertinti pasitelkiant „Google Colab“ aplinką ir konvuliacinius neuroninius tinklus. UML diagramomis pateikiama programos prototipo realizacijos idėja bei atliekamas „Google Colab“ aplinkos įvertinimas ir aptariamas vaizdų atpažinimas.

Programos prototipas geba atpažinti ir pateikti rezultatą ar pateiktas objektas yra varžtas arba profilis. Taip pat yra įvertinama modelio mokymosi kokybė bei tokios programos maketo trūkumai.

Pateikiamos išvados ir rekomendacijos.

Darbo apimtis – 60 p. teksto be priedų, 25 iliustr., 3 lent., 61 bibliografinių šaltinių. Atskirai pridedami darbo priedai.

Prasminiai žodžiai: detalių paieškos sistema, konvuliaciniai neuroniniai tinklai, panašios formos 3D objektų atpažinimas, programos prototipas, kompiuterio mokymasis

Vilnius Gediminas Technical University
Faculty of Fundamental Sciences
Department of Engineering Graphics

ISBN ISSN
Copies No. .....
Date .....-.....-.....

Master Degree Studies **Information Technologies** study programme Master Graduation Thesis 4

Title	**Similar Form Model Search System**
Author	**Andrius Tunikaitis**
Academic supervisor	**Saulius Dereškevičius**

Thesis language: Lithuanian

Annotation

The purpose of the final master thesis is to facilitate engineer work taking into account scientific literature on similar form object search systems and create an easy-to-use prototype of application that can recognize shape parts from photos and provide object class probability

The thesis involves an investigation of existing 3D object search systems and their use. The machine learning and 3D object recognition principles are also investigated.

After literature analysis, it is concluded that program prototype should be created using multi-view 3d object recognition together with Google Colaboratory environment and convolutional neural networks. UML diagrams represent the idea of realization. Also, the assessment of Google Colaboratory and 2D image recognition is made.

Program prototype is capable of recognizing object class and can return a result if the object is screw or profile. Additionally, the quality of learning model and deficiencies of such a program prototype are evaluated

Conclusion and recommendations are presented.

The thesis consists of 60 pages of text without appendixes, 25 pictures, 3 tables, 60 bibliographical entries. Appendixes are included separately.

Keywords: object search system, convolutional neural network, CNN, similar form search system, program prototype, machine learning

Vilniaus Gedimino technikos universiteto egzaminų sesijų ir baigiamųjų darbų rengimo bei gynimo organizavimo tvarkos aprašo
2 priedas

(Baigiamojo darbo sąžiningumo deklaracijos forma)

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Andrius Tunikaitis, 20124187

(Studento vardas ir pavardė, studento pažymėjimo Nr.)

Fundamentinių mokslų fakultetas

(Fakultetas)

Informacinės technologijos, IKGfm-19

(Studijų programa, akademinė grupė)

BAIGIAMOJO DARBO (PROJEKTO)

SĄŽININGUMO DEKLARACIJA

2021 m. gegužės 31 d.

Patvirtinu, kad mano baigiamasis darbas tema „Panašios formos detalių paieškos sistema“ patvirtintas 2021 m. gegužės 21 d. dekano potvarkiu Nr. 125fm, yra savarankiškai parašytas. Šiame darbe pateikta medžiaga nėra plagijuota. Tiesiogiai ar netiesiogiai panaudotos kitų šaltinių citatos pažymėtos literatūros nuorodose.

Parenkant ir įvertinant medžiagą bei rengiant baigiamąjį darbą, mane konsultavo mokslininkai ir specialistai: daktaras Saulius Dereškevičius. Mano darbo vadovas daktaras Saulius Dereškevičius.

Kitų asmenų indėlio į parengtą baigiamąjį darbą nėra. Jokių įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs (-usi).

_____ Andrius Tunikaitis

(Parašas) (Vardas ir pavardė)

TURINYS

PAVEIKSLŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

PRIEDŲ SĄRAŠAS

ĮVADAS

Panašios formos detalių paieškos sistema yra algoritmų rinkinys, kuris surenka duomenis apie detalę, palygina su kitų detalių informacija ir pateikia informaciją apie detalių skirtumus ar panašumus arba atrenka detales, kurios yra panašios formos. Tokia sistema gali būti įvairių paskirčių ir kiekviena sistema skiriasi nuo naudojamų funkcijų ar norimų gauti rezultatų. Rinkoje galima rasti įvairių komercinių arba atvirojo kodo programų. Komercinės programos paieškos sistemos yra slepiamos dėl komercinių tikslų. Atvirojo kodo programos dažnai neatlieka savo funkcijos arba dėl vidinių programų paketų atnaujinimų bei nesuderinamumo su senomis versijomis tiesiog nustoja veikti. Šios programos dažnu atveju pritaikomos specifiniams duomenų failams ir su kitais neveikia. Vertinant programinę įrangą, tik kelios programos turi paieškos funkcijas ir dažniausiai gali atlikti tik palyginimą. Tačiau tokios funkcijos veikia tik su pavienėmis detalėmis arba nėra visiškai išvystytos ar tinkamos. Šio proceso metu greitai apdorojami dideli kiekiai duomenų, todėl tam reikalingi didelio galingumo kompiuteriai. Tokia sistema yra reikalinga dėl to, kad produkto dizaino dalis užima kritinę vietą produktų kūrimo linijoje. 80 % kaštų produktui sukurti sunaudojama šioje dalyje. O 75 % dizaino laiko yra išnaudojama ieškant ir pritaikant kūrėjo patirtį ir jau sukurtų objektų dizaino sprendimus (Chu & Hsu, 2005).

Išnagrinėjus literatūrą ir algoritmus buvo pastebėta, kad visi naudojami sprendimai yra labai sudėtingi ir reikalauja didelių kompiuterio išteklių. Visus algoritmus paleisti yra sudėtinga ir reikalauja daug papildomų žinių norint juos ne tik nagrinėti, bet ir valdyti. Todėl šiame darbe buvo ieškoma paprastų ir lengvų sprendimų paieškos sistemai įgyvendinti.

Darbo tikslas – siekiant palengvinti konstruktorių darbą sukurti bei išnagrinėti nesudėtingą algoritmo programą, naudojantis kompiuterio mokymosi (angl. *machine learning*) principais, kurie gali atpažinti detalių formas iš nuotraukų ir pateikti objekto klasės tikimybę.

Tyrimo objektas – kompiuterio mokymosi (angl. *machine learning*) programos projektavimas „Python“ programavimo kalba ir naudojant „Google colaboratory“.

Siekiant įgyvendinti iškeltą tikslą buvo užsibrėžti šie uždaviniai:

- Išnagrinėti nuotraukų atpažinimo ir kompiuterinio mokymosi principus ir atlikti mokslinės literatūros analizę.
- UML (angl. *Unified Modeling Language*) diagramomis pateikti programos realizacijos idėją.
- Parengti mokymosi bazę kompiuterinio mokymosi algoritmui.
- Sukurti programą su „Google colab“ pasitelkiant kompiuterinio mokymosi principus.

- Įvertinti ir išnagrinėti programą ir tolesnį tinkamumą naudoti pagal įgytas žinias.

Magistro darbo tema yra aktuali, nes augant žaidimų kūrimo, erdvinių kūnų modeliavimo ar 3D spausdinimo technologijoms vis daugiau yra kuriama 3D modelių. Šiuo metu serveriai apkraunami įvairias 3D kūnais, kurie dažnai kartojasi, turi smulkių skirtumų bei užima daug vietos (Chu & Hsu, 2005). Tačiau modelių tekstinės sąsajos gali nepakakti surasti modelio, jeigu nuo pat pradžių modelis nebuvo pažymėtas arba pradėtas kurti pagal tam tikras procedūras. Todėl vienintelis būdas surūšiuoti detales – tai eiti per tūkstančius detalių ir rankiniu darbu atrinkinėti detales. Toks darbas reikalautų didelių laiko išteklių ir neleistų konstruktoriui dirbti darbų, kurie įmonei kuria pridėtinę finansinę naudą. Sukurta panašios formos paieškos sistema leidžia atrinkti ir surūšiuoti detales ir objektus pagal formas. Taip pat leidžia greičiau apskaičiuoti produkto kainą ir pagaminimo patirtį pagal jau esančius sukurtus modelius. Tačiau tokios sistemos yra sudėtingos ir daug kompiuterio išteklių reikalaujančios programos, kurias sukurti reikia daug patirties ir programavimo žinių. Atsižvelgiant į nuolatinę technologijų kaitą tokios sistemos turi būti dažnai atnaujinamos, kad neatsiliktų nuo dabartinių standartų. Naudojant paprastus metodus arba naujausius kompiuterinio mokymosi principus galima greitai ir efektyviai sukurti ir pradėti naudoti paprastas programas tokiam rūšiavimui.

Pasirinkta darbo tema yra nauja, nes paieškos sistemos programos maketui sukurti yra panaudotos skirtingos pavienės išnagrinėtos technologijos leidžiančios įgyvendinti magistro darbo tikslus. Pirmoji technologija, tai „Google colaboratory“, kuri yra internetinė skaičiavimo aplinka skirta paleisti ir apdirbti „Python“ kalba parašytus kodus. Kadangi „Google colaboratory“ naudoja debesijos serverius, tai labai palengvina darbą su programos maketu, kadangi duomenis galima bus imti naudoti tiesiai iš serverio. Tokioje aplinkoje galima paleisti įvairias kompiuterinio mokymosi programas, kurios reikalauja nemažai kompiuterio išteklių. O šiam darbui pasirinkta pritaikyti 3D objektų atpažinimą per 2D vaizdų projekcijas, kurios yra išnagrinėtos mašininio mokymosi principais paremtais konvuliaciniais neuroniniais tinklais. Skirtingai nuo kitų autorių į neuroninį tinklą pateikiamas ne didelis kiekis skirtingų vaizdų, o sujungtos objekto projekcijos į vieną vaizdą. Tai leidžia ne tik pagreitinti neuroninio tinklo mokymosi laiką, bet ir pateikia kitokį objektų nagrinėjimo būdą.

Tyrimo metodai. Darbe naudojant tyrimo ir lyginamosios analizės metodus išnagrinėti užsienio ir lietuvių autorių literatūros šaltiniai, kuriuose nagrinėjami kompiuterinio mokymosi ir vaizdų atpažinimo principus, juos keičiant ir naudojant pagal poreikį.

Šio darbo pabaigoje pateikiamos išvados bei rekomendacijos.

1. LITERATŪROS APŽVALGA

Pastaruoju metu pastebima tendencija, kad 3D detalių kūrimas, naudojimas ir dalinimasis jais nuolat auga. Tokias tendencijas galima pamatyti, kadangi 3D modelių kūrimas ir poreikis didėja įvairiose technologinėse profesijose. Įvairių formų, dydžiu ar kitokių požymių kūnai yra naudojami pramogų industrijoje (filmai, kompiuteriniai žaidimai, animacijos), 3D spausdinimo bei statybos ar pramonės inžinerijos srityse. Dėl technologinės pažangos kompiuteriuose modelių kiekis irgi didėja, nes galima modeliuoti dideles ir sudėtingas sistemas iš daug detalių arba kurti didžiulius kompiuterinių žaidimų pasaulius. Turint tiek įvairių formų detalių kūrėjui arba konstruktoriui yra sunku išrūšiuoti detales, jeigu jos nėra kuriamos pagal kokius nors reikalavimus ir tekstinės sąsajos nepakanka. Tokioms problemoms spręsti yra reikalinga detalių formos paieškos sistema.

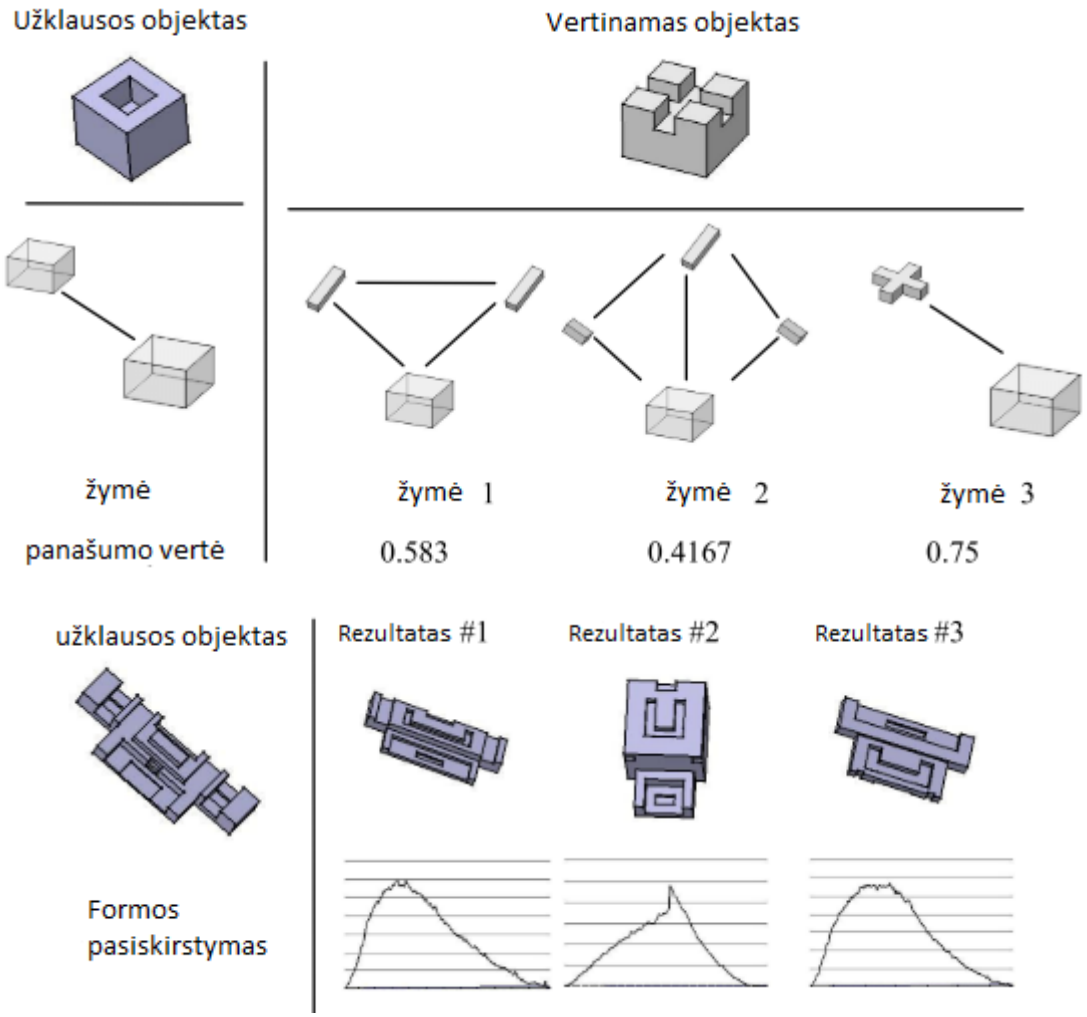
Tačiau tobulėjant technologijoms ir atsirandant poreikiui greitai išnagrinėti didžiulius kiekius duomenų atsiranda papildomi būdai tokioms sistemoms atlikti arba tam tikros dalys yra praleidžiamos arba sukurtos pasinaudojus jau sukurtais programų paketais bei taip pagreitinant visą paieškos ir atpažinimo procesą. Šioje mokslinės literatūros apžvalgoje išnagrinėtos 2 dalys, kurios padės nustatyti ir pasitikrinti darbo gaires, analizės ir vertinimo kriterijus bei tinkamai įgyvendinti išsikeltus darbo tikslus. Pirmoje literatūros analizės dalyje nagrinėjamos esamos panašios formos detalių paieškos sistemos, aptartos mokslinėje literatūroje. Antrojoje dalyje nagrinėjamos esminės panašios formos detalių paieškos sistemos dalys ir jų panaudojimo galimybės.

1.1. Panašios formos detalių paieškos sistemos literatūroje

Mokslinėje literatūroje yra keletas panašios formos detalių paieškos sistemų. Kiekvienu atvejų sistemos skiriasi savo paskirtimis ar esminiais principais, nes kiekvienas autorius bando atrasti vis geresnius sprendimus, o technologijoms sparčiai judant į priekį ieškoma vis naujesnių būdų paieškai atlikti. Pagal autorių Chu, paieškos sistemos išskiriamos pagal tai, kokiais būdais yra atpažįstamos ir atskiriamos objektų formos. Autorius išskiria, kad turėtų būti 6 kategorijos: bruožais paremtos, erdvinių funkcijų, formos histogramos, segmentų vaizdas, topologinės diagramos, formos statistika (Chu & Hsu, 2005). Verta pastebėti, kad priklausomai nuo autorių, tam tikros kategorijos kartojosi, o kai kurios skyrėsi ir visai neatitiko požiūrių (Chen, Tian, Shen, & Ouhyoung, 2003; Chu, Lo, & Cheng, 2017; Tangelder & Veltkamp, 2008). Vertinant išnagrinėtas skirtingas paieškos sistemas, buvo išskirtos taip pat 6 kategorijos pagal kurias buvo atpažįstami objektai. Tačiau svarbu paminėti, kad kai kurios sistemos gali priklausyti kelioms kategorijoms dėl savo naudojamų principų ir autorių noro atrasti vis geresnes ir patogesnes paieškos sistemas. Išskirtos šios paieškos sistemų rūšys:

požymių ir primityvų panaudojimas, eskizų principai, taškų debesys, 3D modeliai, funkcijos grafikai, histogramos ir atpažinimas iš 2D vaizdų.

Primityvų panaudojimas išsiskiria tuo, kad 3D objektas yra supaprastinimas panaudojant įvairias primityvias figūras. Tokios figūros yra ritiniai, kubo pavidalai, sferos ir pan. Autorius Chu sukūrė paieškos sistemą objektus lygindamas su kūnais, kurie yra atimami ar pridedami iš pagrindinės figūros ją modeliuojant ir paverčiant į grafikus (Chu & Hsu, 2005; Ma & Tian, 2014). Paieškos sistemos principai aiškiai pavaizduoti paveikslėlyje (žr. 1.1 pav.). Šiuo principu paieškos sistema nėra apkraunama, gali greičiau nagrinėti mažus nesudėtingus geometrinius kūnus.



1.1 pav. Formos funkcijos kraštinės generavimo grafikas. (Chu & Hsu, 2005)

Tokios sistemos minusai, kad sistema bus jautri modeliuose padarytoms modifikacijoms kaip pavyzdžiui nuožulos ar užapvalinimai. Be to, naudojant tokius primityvus, reikia daugiau informacijos, kad tinkamai atpažinti modelius. Objektus su primityvais ir formos grafikais atpažinti taip pat naudojo ir kitas autorius Ma. Kitas būdas tinkantis prie atpažinimo primityvais yra atpažinimas sferiniais harmonikais (Papadakis, Pratikakis, Perantonis, & Theoharis, 2007).

Toks pasirinkimas leido autoriui sukurti paieškos sistemą, kuri būtų atspari mastelio ir modelio pokyčiams pagal modelio ašį.

Antroji kategorija - objektų atpažinimo sistema iš eskizų. Tai reiškia, kad objektai bus atpažįstami iš vartotojo piešinių ir pagal juos sistema bandys atrinkti paveikslėliui artimiausius variantus. Ši paieškos sistema skiriasi nuo visų tuo, kad vyksta 2 atpažinimo procesai. 1-ojo proceso metu 3D objektai duomenų bazėje yra paruošiami į eskizus ir sudedama atitinkama informacija į bibliotekas. Vartotojui naudojantis, sistema ieško duomenų bazėje ir bando atpažinti ir pateikti galimus variantus. Tokius būdus naudojo 4 autoriai. 2 iš jų problemas sprendė panašiai ir atpažinimui naudojo modelių eskizus arba pridėjo papildomai vartotojų piešinius (Jiantao & Ramani, 2007; Yoon & Yoon, 2017). Qin savo darbe išsiskiria, kad modelių atpažinimą eskizuose taikė išskaidęs objektus pagal jų kūrimo procesus CAD aplinkoje. Tiksliau, kad jis pateikia vertinti ne visą objektą, o palaipsniui įvertinant kiekvieną modelio dalį atskirai. Galima sakyti, kad vertinimas vyksta panašiai, kaip kad būtų kuriamas modelis (Qin, Gao, Yang, Bai, & Zhao, 2017). Paskutinis autorius pabrėžia, kad atpažinimo procesuose naudoja jau konvuliacinius neuroninius tinklus, kurie leidžia jam efektyviau išsspręsti paieškos problemą (Wang, Kang, & Li, 2015). Vertinant visų autorių tyrimus panašu, kad 3D CAD modeliams atpažinti pagal formą naudojantis tik eskizais yra sudėtinga ir tai užima nemažai kompiuterio turimų išteklių. Todėl galima teigti, kad procesams pagerinti vienas teisingesnių sprendimų yra atnaujinti ir mėginti naudotis neuroniniais tinklais.

Trečiojoje kategorijoje priskiriamos panašios formos paieškos pagal turimus taškų debesis. Tokie taškų debesys yra labai dažnai sutinkami ten kur objektai yra skanuojami su 3D skaneriais. Tai įrengimai, kurie leidžia greitai perteikti ne tik objektus, patalpas, bet ir kitus realybėje esančius objektus, kuriuos mes norime pamatyti skaitmeninėje erdvėje. Iš 4 analizuotų darbų, 3 naudoja neuroninius tinklus, kad apdorotų ir atpažintų objektus. Taip yra todėl, kad taškų debesyse yra labai daug taškų ir tam, kad algoritmai galėtų įvertinti objektus reikia daug kompiuterio išteklių. Pirmasis autorius, norėdamas pagerinti atpažinimą iš taškų sukūrė naują variantą, kuris taškus vertina dar ir pagal histogramas (Y. Zhang & Rabbat, 2018). Mokslininkas Pang atlieka daiktų atpažinimą iš taškų debesies, bet vienu metu atpažįsta objektus transformuodamas juos į 2D paveikslus. Nors autorius naudoja konvuliacinius neuroninius tinklus, kurie gerai dirba su 2D nuotraukomis, tačiau tai apkrauna jo tinklą. Tai reiškia, kad sistema sparčiai neveiks ir nebus patogi vartotojui (Pang & Neumann, 2016). Autorius Hong, sukūrė paieškos sistemą ir įvertino jos pajėgumus su konkurencingomis programomis (Hong, Kim, & Lee, 2020). Autoriui pavyko įgyvendinti gerą, daug duomenų suteikiantį algoritmą. Dažnai autoriai ieškantys geriausio būdo, bando apjungti savo ir kitų teorijas, taip siekiant atrasti optimaliausią sprendimą problemai įveikti. Tą atliko Ip su

komanda. Autorius sujungė ne tik taškų debesis, bet ir objekto tinklelius (angl. *mesh*) (Ip & Gupta, 2007). Pagal autoriaus atsiliepimus, panašu, kad jam pavyko sukurti atpažinimo sistemą, kuri yra atspari objektų rotacijai ir leidžia greitai įsivertinti modelius. Išnagrinėjus šiuos pavyzdžius buvo pamatyta, kad neuroniniai tinklai tikrai gali greičiau ir patogiau padėti įvertinti ir atrinkti objektus.

Ketvirtojoje kategorijoje nagrinėjamos paieškos sistemos, kurios atpažįsta objektus vertinant 3D modelius. Šią kategoriją galima išskaidyti į dvi dalis. Kai modelių paviršius yra išskaidomas į tinklelį arba visas modelis yra išdalinamas į vokselius (liet. *tūrinis elementas, reprezentuojantis patį mažiausią galimą diskretizuotos trimatės erdvės elementą*). Dažniausiai modelio tinklelio sukūrimas arba triangulizacija daugiau naudojama modelio spalvos ir tekstūros priskyrimui, taip padarant modelį vizualesnį. Tokį modelio išskaidymą į tinklelį galima panaudoti ir panašių formų paieškai. Tai atliko R.Gal naudojantis 3D modelio tinkleliu ir bruožų atpažinimo algoritmais jis palygina panašius objektus. Skirtingai nuo kitų, autorius lygina ne visus modelius, o išskaido juos į esmines modelio dalis, pagal kurias galima palyginti objektus. Svarbu paminėti, kad nuo tinklo tankumo priklauso objektų atpažinimo savybės. Naujausioje literatūroje Ruggeri išsiskyrė iš kitų naudojančių triangulizacija, nes jis naudoja nebe tinklelį, o „surfels“ (angl. *SURFace ELements*). Tai yra mažiausias paviršiaus dalies elementas, kurį galima naudoti norint atlikti modelio vizualizaciją. Autoriaus remdamasis modelių paviršiaus trianguliacija, juos perdaro į „surfels“ paviršių, taip išlaikydamas modelio formas (Ruggeri, Vranić, & Saupe, 2006). Po to modeliai yra perdaromi į 2D vaizdus, kurie po to ir vertinami. Pagal autorių, šis naujas būdas yra lygiai taip pat geras, kaip ir kiti atpažinimo ir paieškos metodais. Toks skaidymas ir atpažinimas labiau liečia 3D modelių paviršius, skirtingai nuo to, skaidymas į vokselius, pilnai suskaido modelį į mažus kubus. Priklausomai nuo kubelių tankumo galime išryškinti daugiau objekto detalių. Vokselius galima sukurti iš skirtingų pradinių duomenų. Standartiškai tai būtų modelio pavertimas į vokselius, tačiau autorius sukūrė variantą, kad iš anksto jau būtų paruošta tokių vokselių modelių pilna duomenų bazę. Ir lyginant su kitais, atpažinimo procesui atrenka geriausią objekto kampą, padaro pjūvį ir su neuroninio tinklo pagalba įvertina ir analizuoja visus vokselius, kurie yra galimi panašiuose objektuose (Wu et al., 2015). Autoriaus teigimu toks metodas yra 10% pranašesnis už kitus panašius metodus. Kitas tyrėjas vokselių objektų atpažinimą pritaiko taškų debesims, kad būtų lengviau dirbti su objektas robotams realioje aplinkoje. Tam kad visas procesas vyktų sklandžiai, vokselizuotas modelis yra sudedamas ir išskaidomas į sekcijas ir paduodamas į neuroninį tinklą, kur yra nagrinėjamas (Maturana & Scherer, 2015). Panašu, kad toks būdas yra vienas tinkamiausių renkantis atpažinimo sistemą.

Penktoji kategorija yra paieškos sistemos, kurios naudojasi funkcijų grafikais, grafikais ir histogramomis objektams atpažinti. Dažnu atveju atpažinimas ir grafikų lyginamas yra tik dalis paieškos sistemos. Kitas variantas, kad pradiniai duomenys yra standartiniai objektai, kuriuos reikia konvertuoti, kad būtų galima pasinaudoti. Naudoti grafikus paieškos sistemose yra labai patogu, kadangi kompiuteris grafikus gali lengvai išnagrinėti, palyginti ir įvertinti. Vienas iš tokių sprendimų yra D2 formos funkcija, kuri yra vienos dimensijos histograma. Ši histograma yra nejautri variacijoms, ir paviršiaus orientacijoms. Taip yra todėl, kad funkcija pasidaro modelį pakeičiant jį taškus ir suskaičiuojant atstumus tarp taškų bei paverčiant juos į histogramą (Cheng, Lo, Chu, & Kim, 2011; Ohbuchi, Minamitani, & Takei, 2005). Toks būdas taip pat reikalauja mažiau kompiuterio išteklių. Kadangi 3D objektų struktūra ir geometrinės formos dažnai keičiasi, bei sudaro daug iššūkių stengiantis juos išanalizuoti ir atpažinti, panašu, kad signalo sukūrimas yra vienas iš gerų būdų atpažinti ir surasti įvairių formų kūnus. Antrajame variante buvo pasirinkta, kad modelio viršūnės, kraštinės ir kraštinių svoriai buvo apskaičiuoti ir apdirbti su papildomomis funkcijomis, taip gaunant funkcijos grafiką, kuris paduodamas į neuroninį tinklą ir dar kartą apdirbamas. Toliau turint funkcijos grafikus jie gali būti palyginami ir taip atrandamas objektų panašumas (Xie, Fang, Zhu, & Wong, 2015).

Paskutinė, šeštoji, kategorija yra objektų paieškos sistemos naudojančios nuotraukas ir 2 dimensijų vaizdus. Skirtingai nuo kitų kategorijų, šios kategorijos pavyzdžių yra daugiau. Priežastys didesniam kiekiui pavyzdžių yra atsiradę neuroniniai tinklai ir taip pat dėl to, kad atpažinimas iš nuotraukų yra ganėtinai seniai nagrinėjama tema. Labai daug nagrinėta yra 3D objekto nagrinėjimas pasitelkiant objektų projekcijų vaizdus (Bai, Bai, Zhou, Zhang, & Latecki, 2016; Chen et al., 2003; Lian, Godil, & Sun, 2010; Shen, Chen, Tian, & Ouhyoung, 2003; Su, Maji, Kalogerakis, & Learned-Miller, 2015). Šiuose darbuose aišku yra tam tikrų skirtumų. Kai kurie darbai naudoja detalių projekcijas arba tik formos šešėlius. Taip pat daugumą darbų naudoją konvuliacinius neuroninius tinklus dėl jų gero darbo su 2D nuotraukų atpažinimu ir efektyvesnio objektų įvertinimo. Peržiūrėjus darbus visi autoriai teigia, kad naudojant nuotraukų atpažinimą paieškos sistemas, turime efektyvią ir stabilią programą, kurios duomenų nepaveikia prasti ar deformuoti objektai paduoti treniravimuisi į sistemą. Tačiau sistema gali būti apkrauta, dėl gausybės nuotraukų, kadangi vienas objektas turės net keletą projekcijos vaizdų, tam kad atpažinti nuotrauką. Buvo atrastas ir vienas kitoks 3D objekto ir 2D vaizdo gavimo būdas objektams ieškoti ir palyginti. Tai panoraminis formos deskriptorius, kuris priešingai nei anksčiau minėti sukuria 2 panoraminius vaizdus iš modelio ir sujungia juos į bendrą projekciją bei paduoda nagrinėjimui ir atpažinimui (Papadakis, Pratikakis, Theoharis, & Perantonis, 2010). Šis atpažinimo būdas taip pat kaip ir kiti yra tvirtas ir nejautrus mastelio

ar nuotraukos transformacijoms. Be to nuotraukų atpažinimas yra lankstu ir būdas paieškos sistemai, kadangi tau reikia surinkti tik objektų nuotraukas, kad juos atpažinti.

Vertinant visas kategorijas, kiekvienos kategorijos autorių rezultatai yra iš esmės panašūs, jei nenaudojami ypatingi būdai (Bustos, Keim, Saupe, Schreck, & Vranić, 2005). Netgi Shafi yra išnagrinėjęs, kad 2D ir 3D objektų atpažinime, kai yra naudojami konvuliaciniai neuroniniai tinklai turime, kad ar naudosi nuotraukų atpažinimą ar kitą būda, galime pamatyti darbų ir atpažinimo tikslumo panašumus. Pagal autorių tikslumas tarp 2D ir 3D skiriasi mažiau negu pusę procento (Shafi, 2017).

1.2. Esminės paieškos sistemos dalys

Atlikus panašios formos paieškos sistemų analizę buvo pamatyta, kad yra daugybė įvairių būdų paieškos sistemai įgyvendinti. Dėl greitai augančių technologijų ir spartaus mokslininkų ir tyrėjų indėlio, tokių sistemų yra vis daugiau ir jų naudojami metodai pradeda pintis viena su kita. Tokiu principu atrandami vis efektyvesnis ir geresni variantai leidžiantys geriau atpažinti ir suklasifikuoti turimus 3D objektus. Vertinant visus paieškų sistemos pavyzdžius buvo pamatyta, kad paieškos sistema iš esmės susideda iš objektų požymių išskyrimo, objektų analizės ir klasifikacijos. Tačiau analizuojant naujesnes paieškos sistemas matome, kad tam tikros technologijos jau atlieka daugiau ir mes išskirtume dvi esmines dalis, kurios nagrinėtos tam, kad išsirinkti geriausią sprendimą savo sistemai įgyvendinti. Tos dalys yra objektų formos atpažinimas ir konvuliaciniai neuroniniai tinklai. Skirtingai nuo senesnių paieškos sistemų, konvuliaciniai neuroniniai tinklai patys išanalizuoja duomenis ir su jais lengviau galima gauti informacija apie objektų panašumus. Tačiau jiems tiesiog reikia turėti surinktą informacijos duomenų bazę pagal kurią galėtų mokytis ir tikrintis duomenis.

1.2.1. Objektų formos atpažinimas

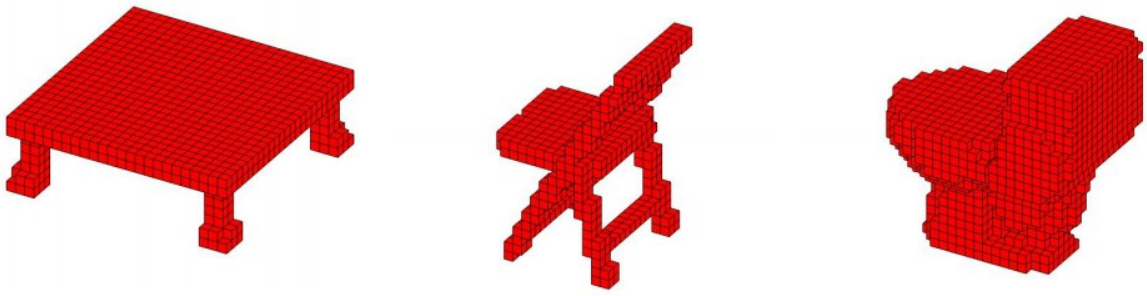
Objektų atpažinimas naudojantis neuroniniais tinklais yra pagrįstas tam tikra duomenų baze pagal kuria mokosi ir po to atpažįsta objektus. Tam kad sukurti tokią duomenų bazę reikia pirmiausia išsiaiškinti kokie yra galimi principai objektui atpažinti. Iyer išnagrinėjo kitų mokslininkų formos atpažinimo tyrimus ir teigi, kad yra 7 kriterijai formai įvertinti ir nustatyti (Iyer, Jayanti, Lou, Kalyanaraman, & Ramani, 2005):

1. Sritis. Algoritmai turi atpažinti visas formos klases. Tai reiškia, kad nesvarbu, kokiu kampu pateikiamas objektas, algoritmas gali teisingai nustatyti jo formą. Netgi nustatyti ar objektas yra kubas ar tai yra plokštuma ar tai yra koks apvalus paviršius.

2. Unikalumas. Kiekvienas objektas ir jo forma turi unikalumą, pagal kurį galima atpažinti ar dvi formos yra tokios pačios ar ne. Todėl algoritmams yra svarbu atpažinti kiekvieno objekto svarbiausius unikalius taškus.
3. Stabilumas. Algoritmas turi būti stabilus atpažįstant forma. Maži formos pokyčiai turėtų sukurti mažus pokyčius atpažinimo algoritme.
4. Jautrumas. Atpažinimo procesas turi gebėti nustatyti net ir mažiausius formos pokyčius. Ši savybė yra prieštaraujanti stabilumui. Tačiau algoritmas turi sugebėti reaguoti į pokyčius ir išlaikyti esmines vietas stabiliai jei nori tinkamai atpažinti objektus jiems besikeičiant.
5. Efektyvumas. Algoritmai turi efektyviai apskaičiuoti ir palyginti formai sukurtus deskriptorius iš pradinės informacijos. Priklausomai nuo objekto ir deskriptorių tai gali būti vektoriai, grafikai ar kitos duomenų struktūros.
6. Kelių lygių vertinimas. Algoritmas turi sugebėti įsivertinti objektus keliais lygiais kaip struktūrą. Tam tikras detalumas gali būti nevertinamas iki kol to nereikalauja išsikelti tikslai.
7. Vietinis vertinimas. Algoritmas turi sugebėti įsivertinti objektą lygyje kuriame yra, bet tuo pačiu turi sugebėti ir išsaugoti tą informaciją apie esamą lygi, vertinant su kitais lygiais, kad būtų galima atlikti detalesnius vertinimus.

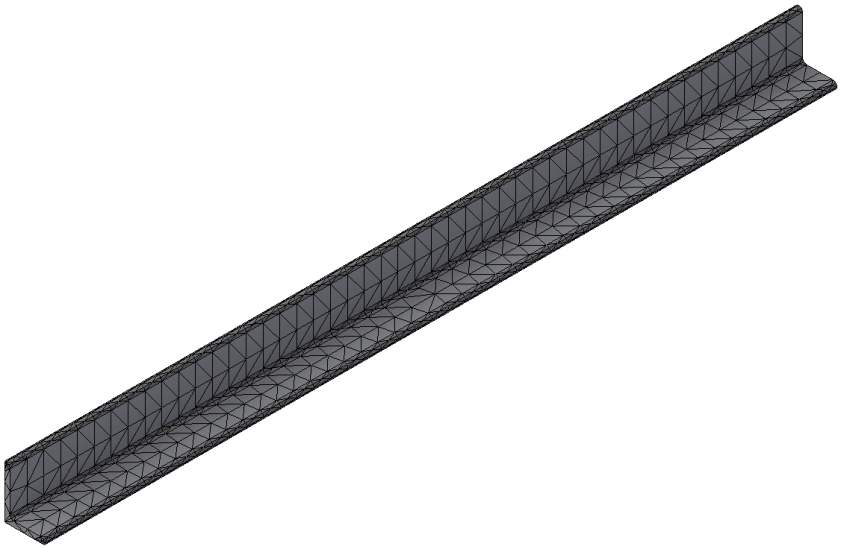
Turint tokius kriterijus yra lengva nustatyti ar objektų atpažinimas yra geras ar dar reikalauja papildomų tyrinėjimų ir geresnių funkcijų paieškos. Tačiau vien tik kriterijų objektams įvertinti nepakanka. Vienas iš dalykų, kas daro objektų atpažinimą sudėtingu yra tai, kad patys 3D objektai gali būti pavaizduoti skirtingai, o dėl to, nagrinėjimo būdas pagal kriterijus anksčiau turi prisitaikyti prie objekto. Pagal Ben-Shabat 3D objektai dažniausiai vaizduojami šiais būdais (Ben-Shabat et al. 2017):

a) Tūrinis tinklelis (angl. *Volumetric grid*)



1.2 pav. Tūrinis tinklelis.3D objektas sudarytas iš mažiausių trimačių elementų - vokselių. (Liu et al. 2018)

b) Trianguliacija, tinklelis (angl. *Mesh*)

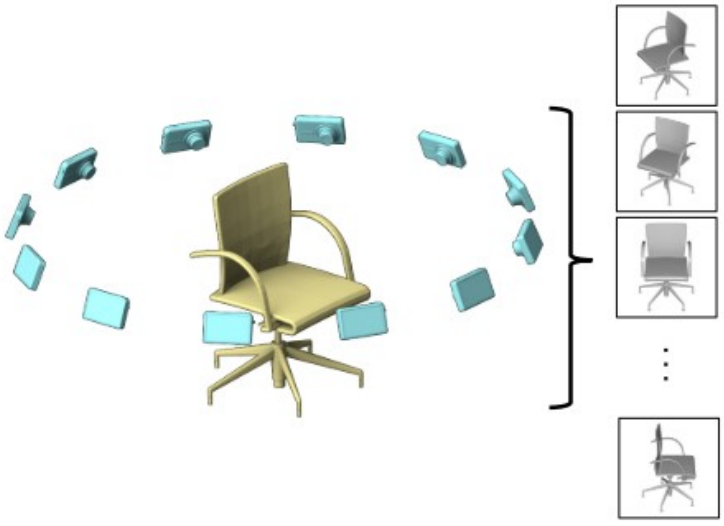


1.3 pav. 3D objektas atvaizduotas modelio tinkleliu.

c) Taškų debesimis (angl. *Point Cloud*)

1.4 pav. Lėktuvo modelis naudojantis iš taškų debesies (Ben-Shabat et al. 2017)

d) Daugialypis vaizdas (angl. *Multi-view*)

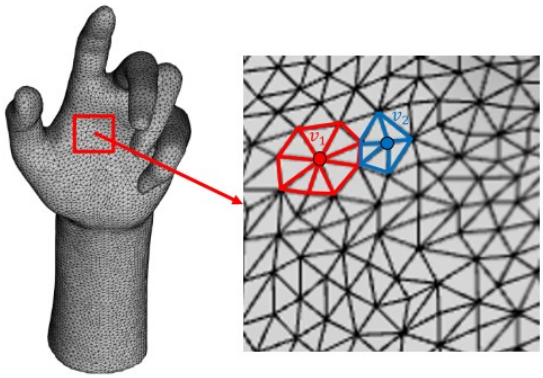


1.5 pav. Daugialypis vaizdas (angl. *Multi-view*) – 3D vaizdas sudaromas iš skirtingų projekcijų ir atiduodamas atpažinimui (Su et al. 2015: 945–953)

Tūrinis tinklelis parodo 3D aplinką objektui, kaip kad nuotrauka parodo kameros projekcijos plokštumą. Tai reiškia, kad kaip ir nuotraukoje esantys pikseliai, taip vokseliai yra tie patys pikseliai tik turi papildomą 3 dimensiją. Patį tūrinį tinklelį mes galime kontroliuoti ir valdyti priklausomai nuo situacijos. Retesnis tinklelis leidžia greitai įsivertinti objekto panašias

formas. Tankesnis tinklelis leidžia geriau įsivertinti objektą. Taip pat toks volumetrinis tinklelis turi ir daugiau gerų savybių. Kadangi atskiras vokselis laiko informaciją, galima daugiau jį apdirbti ir naudoti objektams įsivertinti. Pavyzdžiui vokselius galima spalvinti ir šitai parodyti, kuri dalis objekto matosi ar pavyzdžiui parodyti informacija ar tame vokselyje yra objekto dalis ar yra tuščia vieta, tai reiškia, kad vokselis gali būti dvejetainio formato (Jia, Zhang, Zeng, & Liang, 2014). Taip pat galima sakyti kad naudojantis vokseliais turime stabilų objekto atpažinimo forma, nes jie nėra jautrus objekto tekstūrai, skylėms ir įvairiems įpjovimams. Kadangi vokseliai yra 3 dimensijų, tai savaime juos apdoroti ir išnagrinėti yra sunkiau nei kokias 2 dimensijų nuotraukas ar grafikus. Pagal autorių norint įsivertinti vokselius neprarandant 42% informacijos, reikia didinti objekto rezoliuciją dvigubai dėl ko kompiuterio proceso atminties reikia suvartoti apie 7 kartus daugiau (Liu, Tang, Lin, & Han, 2019). Kitas dalykas, kad priklausomai nuo tinklelio tankumo ar retumo atsiranda ir kitos problemos. Mažos rezoliucijos arba retesnis tinklelis neleidžia tinkamai atpažinti panašių formų tarp objektų ir prarandama tekstūros informacija. Tankesnis tinklelis automatiškai reikalauja daugiau kompiuterio išteklių. Taip pat buvo nustatyta, kad retas tinklelis nėra toks efektyvus, kaip daugialypio vaizdo atpažinimas (Brock, Lim, Ritchie, & Weston, 2016). Be to, dažniausiai modeliai nėra iš karto vokselių formoje, todėl tyrėjams reikia dar susitvarkyti ir persidaryti modelius į vokselius dar net prieš pradedant darbą (Bian, Hu, He, & Cai, 2010).

Kitas variantas yra 3D objekto trianguliacija. Atliekant trianguliacija yra daugiau sunkumų siekiant atrasti svarbiausius unikalius taškus dėl kūno paviršiaus topologijos. Trianguliacijos modelis yra vaizduojamas kaip viršūnių ir paviršių junginiai su informacija apie kaimyninius plotus tarp taškų. Kiekvieną viršūnę galima sujungti su kitomis viršūnių linijomis ir gauti trikampius, todėl modeliai ir yra vadinami trianguliacijos. Modelio topologija yra savavališka, todėl kiekviena viršūnė gali turėti skirtingą kiekį kaimyninių viršūnių skaičių, priklausomai nuo pačio modelio paviršiaus nelygumų. Dėl skirtingų viršūnių kiekio aplink ir paviršiaus skirtumų, plotai ir linijos yra skirtingi, todėl atstumai tarp viršūnių yra visada skirtingi. Vien dėl to yra sudėtinga sukurti tinkama matematinį modelį, kuriuo būtų galima apskaičiuoti ir išspręsti šias problemas. Nes realiai paviršius yra nepastovus ir nuolat kintantis, priklausomai nuo modelio ir jo parametru. Dar didesnį iššūkį kelia ir tai, kad viršūnės, atstumai tarp jų, plotai ir paviršius yra vienintelė informacija pagal kurią galima vertinti modelius (Lin, Zhu, & Liu, 2016). Dėl to svarbiausius taškus nustatyti yra dar sudėtingiau. Tai galima matyti paveikslėlyje numeris 6.



1.6 pav. Rankos 3D modelis ir jo trianguliacijos detalės (Lin et al., 2016).

Nors nustatyti atpažinimui svarbius taškus yra sudėtinga, tačiau yra atlikta nemažai tyrimų objektų atpažinimui naudojant trianguliaciją kaip pagrindą. Tai leidžia turėti visą kūno paviršių ir matyti paviršiaus skirtumus, nes pati trianguliacija ir priklauso nuo kūno paviršiaus, todėl pervertus histogramas ar išklotinių palyginimui galime turėti puikius stiprius kūnų atpažinimo būdus, kurie įvertina visą kūną. Kūno trianguliaciją, kaip ir su vokseliais, galima reguliuoti ir daryti tankesnę arba retesnę. Nuo to priklauso, kūno detalumas, bet žinoma, augant detalumui, kyla ir kompiuterio reikalaujami ištekliai. Be to kūno trianguliacija gali būti koreguojama į kitokius paviršius, pavyzdžiui ne trikampius, o „superquadric“. Tai įvairių formų paviršiai iš 4 taškų, leidžiantys dar lanksčiau geometriškai išnagrinėti objektų paviršius (Biegelbauer, Vincze, & Wohlkinger, 2010). Tai toks lankstumas ir pritaikomumas skirtingiems metodams, ypač, kai daug duomenų bazių yra iš karto jau pateikiamos su kūnų trianguliacija, leidžia teigti, kad tai yra irgi stiprus ir naudingas metodas kūnų formoms atpažinti ir įvertinti.

Taškų debesys – tai taškų rinkinys esantis erdvėje. Taškai atvaizduoja 3D objektą ir jo formas. Kiekvienas taškas turi savo X,Y ir Z koordinates. Tokie taškai dažniausiai sukuriami naudojantis 3D skaneriais arba programine įranga, kuri matuoja objekto paviršiaus taškus ir taip gaunant taškų debesį. Taškų debesys išsiskiria tuo, kad taškų debesys turi lanksčia struktūra, turtinga informacijos aprašą ir efektyvius duomenų apdorojimus. Be to taškų debesys reikalauja mažiau procesoriaus išteklių negu vokseliais paremtų modelių (Liu et al., 2019). Tačiau dėl taškų išsibarstymo kyla problemos, kad taškai nestruktūrizuoti ir taškų atžvilgiu jie neturi tvarkos. Kadangi taškai be tvarkos norint išsaugoti kaimyniniu taškų duomenis, sistemos turi nereguliariai ir nestruktūrizuotai prisijungi prie atminties ir išsaugoti informaciją apie taškus (Poux, Hallot, Neuville, & Billen, 2016). Dėl pastovios atminties apkrovos gali būti pasiekti skaičiavimų efektyvumo duomenų kamščiai, kurie reikalaus daugiau laiko duomenis tinkamai sustruktūrizuoti. Pagal Liu tai gali užpimti nuo 36% iki 57%, viso duomenų apdorojimo laiko (Liu et al., 2019). Kadangi taškų modeliuose yra daug ir ne visi taškai pateikia reikiama informaciją, dažniausiai taškus reikia apdoroti norint juos naudoti objektų atpažinime (X.

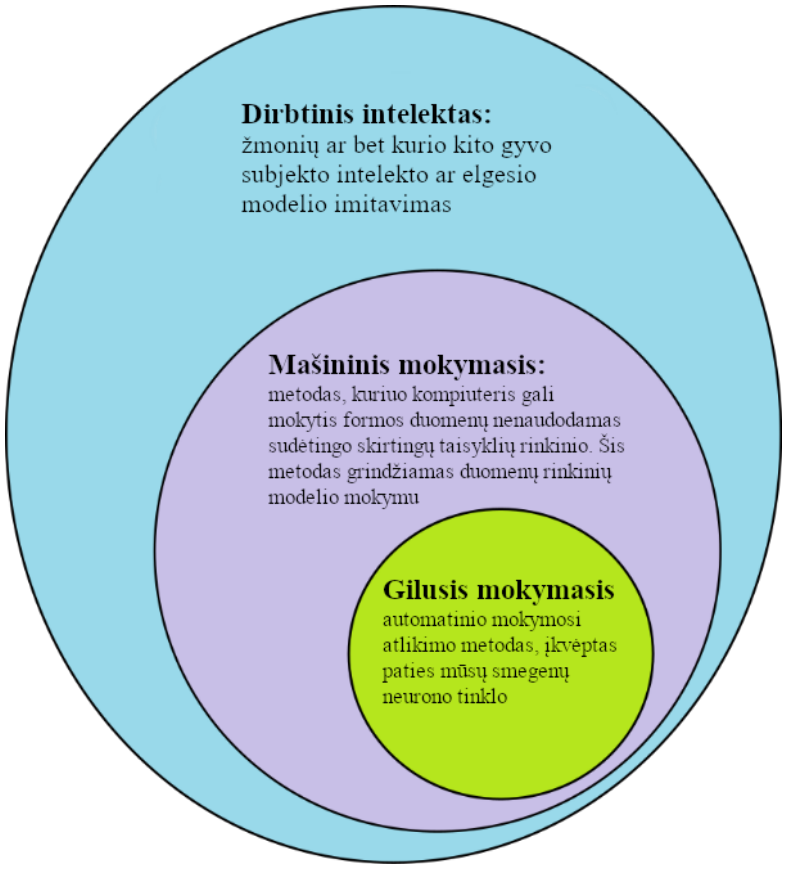
Zhang, Fu, & Zhao, 2021). Apdirbimas taip pat reikalingas dėl to, kad dažnai skanuojant objektus į taškų debesis gali pasitaikyti daug papildomų taškų, kurie nepriklauso objektui ir yra tik lazerio klaidos (Sharif, Nahangi, Haas, & West, 2017). Tokie papildomi taškai tik padidina objekto sudėtingumą ir gali iškraipyti formos atpažinimo procesus ir padaryti juos neteisingus. Dar blogesnis variantas, kad objektas gali būti net pilnai nepavaizduotas visais taškais dėl skanavimo klaidos (Che, Jung, & Olsen, 2019). Todėl apdirbimas prieš atpažinimo procesus yra svarbus taškų debesims. Bet taškų panaudojimas yra vienas iš galimų objektų atpažinimo būdų, jei objektai yra pateikiami naudojanti skaneriais ar tiesiog tai vienintelis būdas gauti informacija apie modelius.

Paskutinis būdas yra iš turimo 3D objekto sukurti daugialypius skirtingų projekcijų 2D vaizdus. O 2D dimensija atpažinimų procesuose yra stipriai išnagrinėta Galima pritarti autoriaus nuomonei, kad daugialypio vaizdo metodu visa informacija apie objektą yra pateikiama objekto vaizdo projekcijomis iš visų pusių, kurios dalinai kompensuoja informacijos praradimą verčiant 3D objektą į 2 dimensijų nuotraukas (Ben-Shabat et al. 2017). Kalbėdamas apie 2 dimensijų vaizdų panaudojimą autorius Hang Su nurodo keletą svarbių dalykų (Su et al. 2015: 945–953). Pirma, kad atpažinimo procesas yra tikslesnis, greitesnis ir efektyvesnis negu lyginant su 3D objektų atpažinimu. Toks tikslumas atsiranda dėl to, kad išlaikoma kokybė, kadangi nuotraukos užima mažiau vietos ir nereikia apdirbti, o norint paleisti 3D objektus į neuroninį tinklą, reikia juos supaprastinti, kadangi jie užima per daug vietos ir būna labai sudėtingi. Antras privalumas, kad objektų nuotraukas yra gauti paprasčiau negu objektų 3D modelius, kadangi yra sukurta nemažai duomenų bazių su nemažais kiekiais nuotraukų duomenų. Taip pat svarbu paminėti, kad programos vartotojas turėtų būti inžinierius arba kūrėjas. Todėl galime teigti, kad naudojantis paieškos sistema, reiktų gauti kuo tikslesnius duomenis, kadangi tai sutaupytų jo laiką ir sumažintų jo darbo laiką. Tokiu atveju reikia atsižvelgti ir į tai, su kokia programine įranga dirba vartotojas. Inžinerinėje srityje, sakykime baldų gamyboje, konstruktoriai naudoja viena iš dviejų programinių įrangų: „Dassault Solidworks“ arba „Autodesk Inventor“. Abi programos veikia tais pačiais principais ir darbui įgyvendinti gali būti panaudotos abi. Šioje vietoje galima taip pat pamatyti perspektyvą, jog 2D dimensijų vaizdų naudojimas leidžia būti lankstesniems ir naudoti skirtingą programinė įrangą iš kurios galima kaupti duomenų bazei reikalingas nuotraukas. Nuotraukos taip pat leidžia surinkti labai daug duomenų apie objektą ir tai nėra tik informacija apie projekcijas, bet galimybė pateikti informacija ir apie pjūvius ar kitus objekto formą nusakančius parametrus. Vieną tik kaip trukumą autoriui Su norėtume išskirti, kad autorius į neuroninius tinklus nuotraukas pateikia atskirai į atskirus neuroninių tinklų sluoksnius (Su et al. 2015: 945–953).

Kiekvienas iš šių 3D objektų reprezentacijų turi teigiamų ir neigiamų savybių vertinant panašios formos objektus. Kiekvienas algoritmas yra pritaikytas skirtingiems atvejams ir veikia geriau su skirtingais kriterijais. Šiame darbe mes pasirenkame naudoti ir nagrinėti daugialypio vaizdo metodą objektams atpažinti. Toks pasirinkimas yra dėl nuotraukų lankstumo ir vietos užėmimo. Be to nuotraukų panaudojimas leidžia lanksčiai žiūrėti į programas, kurios naudotos duomenų bazei sukurti. Vertinant esamą literatūrą, manome, kad teisingesnis sprendimas yra vieno nagrinėjamo objekto projekcijų vaizdus sujungti į vieną. Tai leidžia supaprastinti ir palengvinti neuroninių tinklų atpažinimo procesus ir tuo pačiu išlaiko visas tas pačias galimybes apie ką rašo Hang Su. Todėl manome, kad mūsų darbo atveju geriausia yra naudotis nuotraukų atpažinimu, dėl aukščiau išvardintų priežasčių ir dėl to, kad tai puikiai išvystyta ir išnagrinėta tema, kuri turi labai gerų sąsajų su konvuliaciniais neuroniniais tinklais.

1.2.2. Kompiuterio mokymas. Konvuliaciniai neuroniniai tinklai

Siekiant išnagrinėti konvuliacinius neuroninius tinklus, pirmiausia reiktų aptarti kas yra kompiuterio mokymasis, o po to ir gilusis mokymasis, kadangi jiems priklauso konvuliaciniai neuroniniai tinklai (žr. 1.7 pav.).



1.7 pav. Dirbtinis intelektas ir jo sudedamosios dalys

Kompiuterio mokymas (angl. *machine learning*) – dirbtinio intelekto metodų klasė, kuriai būdingas ne tiesioginis problemos sprendimas, o mokymasis, kaip pritaikyti daugelio

panašių problemų sprendimus. Šis principas sukuria sistemos modelį pagal pavyzdinius treniravimosi duomenis pagal kuriuos ir atlieka savo sprendimus (Holzinger, 2017). Tačiau sistema nėra specialiai suprogramuota atlikti paskirtus darbus, o pati išsimokina pagal išsikeltus tikslus.

Kompiuterio mokymasis turi akivaizdžius ryšius su kognityvine ir elgsenos psichologija. Jos modelius dažnai įkvepia psichologijos teorijos. Daugybė kompiuterio mokymosi algoritmuose esančių problemų matomos ir žmonių mokymosi procese. Tokios problemos yra bendrų interesų, abstrakcijos problemos, neturimos ankstesnės mokymosi žinios, pavėluota gradacija (Lison, 2012). Taip pat yra svarbu žinoti ir kada yra pritaikomas kompiuterio mokymasis. Pagal O.Simeone yra išskiriami 8 kriterijai (Simeone, 2018):

1. Užduotis gali turėti funkciją, kuri gali parodyti gerai apibrėžtas įvestis su išvestimis
2. Turime didelius duomenų rinkinius arba juos galima sukurti naudojant įvesties ir išvesties poras
3. Užduotis pateikia aiškų grįžtamąjį ryšį su aiškiai apibrėžtais tikslais ir metrika
4. Užduočiai atlikti nereikia sudėtingų ir ilgų logikos grandinių, kurios priklauso nuo įvairių bendrųjų žinių ar sveiko proto
5. Užduočiai nereikia išsamių paaiškinimų, kaip buvo gautas sprendimas
6. Užduotis toleruoja klaidas ir nereikia įrodyti teisingumo ar ieškoti optimalių sprendimų
7. Išmoktos funkcijos ir procesai netūrėtų greitai kisti laikui bėgant
8. Nereikia specializuoto kruopštumo, fizinių įgūdžių ar mobilumo.

Tokie kriterijai yra naudingi atpažinti ir pasitikrinti ar norima atlikti užduotis yra patogi kompiuterio mokymosi principams. Šios gairės taip pat leidžia atskirti kompiuterio mokymąsi nuo dirbtinio intelekto.

Šie kompiuterio mokymosi algoritmai naudojami skirtingose paskirtyse: medicinos, elektroninių laiškų filtravimo ar kompiuterio atpažinimo, kur yra sunku sukurti algoritmus, kurie galėtu atlikti reikiamas užduotis. Tačiau tokiose srityse yra pilna informacijos apie kitų atliktas realias užduotis. Tai leidžia sukurti sudėtingų problemų sprendimus naudojantis tikrais duomenimis, vietoj intuicijos. Taip pat, naudojantis kompiuterio mokymusi, programos gali lanksčiai prisitaikyti prie naujų situacijų, tereikia surinkti papildomus duomenis (Lison, 2012). Teoriškai kiekvienas mokymosi procesas gali būti suformuluotas į 2 sudėtingas dalis: įvestis ir išvestis. Teoriškai mes norime išmokti koks yra geriausias išvesties variantas įvesčiai. Bet visi mokymosi procesai priklauso nuo duomenų, kuriuos galime gauti. Duomenys, kuriuos galime gauti skirstoma į 3 rūšis:

1. Duomenų pavyzdžiai, kurie turi įvestis ir išvestis.
2. Duomenys, kuriuose yra tik įvestys, bet nėra išvesčių.
3. Duomenys, kai neturi gerų išvesčių, bet pagal įvesčių pokyčius galime nuspręsti apie išvesčių kokybiškumą.

Jeigu turime 1 variantą, tai toks algoritmų mokymasis yra vadinamas prižiūrimu mokymu. Tai kaip ir minėjome anksčiau turime duomenų įvestis ir išvestis, bet nuo kitų mokymosi būdų skiriasi dar ir tuo, kad teisingos išvestys pasirenkamos rankiniu būdu (Learned-miller, 2014). Prižiūrimas mokymosi pavyzdžiai būtų „šlamšto“ filtravimas laiškuose, teksto vertimas, veido atpažinimas. Tačiau, nors ir teisingos išvestys pasirenkamos algoritmo kūrėjo, bet pats mokymosi procesas reikalauja didelio kiekio duomenų ir parametrų. O algoritmo tikslas pasitaisyti parametrus taip, kad jie atitiktu turimus duomenis. Pavyzdžiui, „šlamšto“ filtravimo sistema galimus šlamšto turinio žodžius įvertina skaičiais. Kuo aukštesnis skaičius pagal žodį, tuo didesnė tikimybė, kad laiškas yra „šlamštas“. Tai mokymosi algoritmas pakeis žodžių vertes, taip, kad atitiktų pagal duomenis.

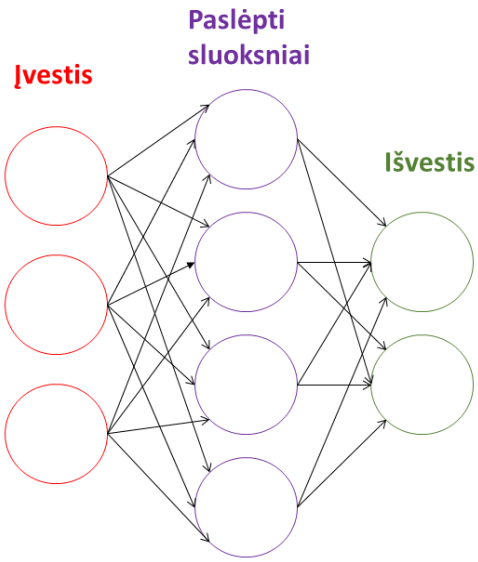
Naudojantis antruoju variantu turime neprižiūrimą mokymąsi. Kai neturime išvesčių ir turime tik įvestus duomenis, bandome suprasti kažkokia duomenų struktūra, pagal kurią galėtume išsinagrinėti duomenis. Tokiu atveju ieškome koreliacijos tarp duomenų arba bandome sujungti duomenis į skirtingas grupes, taip ieškant parametrų, pagal kuriuos būtų galima nagrinėti duomenis. Taigi skirtingai nuo prižiūrimo mokymosi, šiuo atveju algoritmas pats suranda galimus variantus. Yra 2 pagrindiniai metodai naudojami: pagrindinis komponentas ir grupinė analizė (Dayan, 2008). Grupinė analizė yra naudojama sugrupuoti duomenis panašiomis charakteristikomis, kad atrasti algoritminius ryšius tarp duomenų. Tokia analizė suranda panašumus duomenyse ir pagal tai vertina kitus naujus duomenis. Šis būdas leidžia surasti išskirtinius duomenų taškus kurie netinka nei vienai grupei.

Paskutinis, trečiasis aprašymas aprašo pastiprinamąjį mokymąsi. Kadangi mes neturime teisingų išvesčių, tai naudojame kokybės matavimais, kiek yra gera įvestis. Kokybės matavimai gali būti vertė, kuria galime vadinti apdovanojimu arba bausme. Apdovanojimas dažnai parodo užduoties paskirtį. Pavyzdžiui, jei algoritmas norėtų išmokti apversti blynus. Tai apdovanojimas galėtų duoti +3 taškus, už tai kad blynas buvo apverstas, -1 už tai, jei verčiant blynas liko keptuvėje ir -5 jei nepavyko apversti ir blynas ne keptuvėje. Tai algoritmo tikslas yra išmokti ir surinkti kuo daugiau taškų per tam tikrą mokymosi laiką. Pastiprinamasis mokymasis nepriskiriamas nei prižiūrimam mokymuisi, nes mokymosi procesui nepateikiamos įvestys ir išvestys pagal kurias mokytis. Tačiau taip pat nepriklauso ir neprižiūrimam mokymuisi, vertinant grįžtamojo ryšio prieinamumą apie pasirinkto veiksmo kokybę.

Pastiprinamąjį mokymąsi galima išskirti dėl ankstesnių veiksmų įtakos būsimoms būsenoms ir atlygiams mokymosi procese (Simeone, 2018).

Tai geras mokymosi modelis yra tas kuris gerai prisitaiko prie naujų duomenų. Tai reiškia kad modelis gali atrasti pagrindines struktūras nagrinėjimui. Tai tokių modelių kūrimas, mokymas ir testavimas yra vienas iš pagrindinių mašinos mokymosi dalių. Tačiau jei modelis tinkamai neatpažįsta struktūrų arba išmoksta blogai, tai vadinama „perkrovimu“. Jeigu tiksliau, tai reiškia, kad modelis labai prisiderino prie esamų pavyzdžių, kuriuos įvertino, tačiau veiks labai prastai jei jam bus pateikti anksčiau nematyti pavyzdžiai. Nors mūsų tikslas yra turėti programa, kuri tiksliai pati gali nuspręsti, net ir matydama naujus duomenis. Dažnai naudodamiesi įvairiais duomenimis, mes turime tam tikras žinias apie tuos duomenis, kas gali padėti modeliui dar geriau įsivertinti duomenis.

Dirbtinis intelektas apima labai daug sričių ir vien tik tai nagrinėti nenaudinga. Taigi norint atlikti mašinos mokymąsi, reikia sukurti modelį, kuris mokysis, analizuos ir atliks prognozes. Įvairūs modelių tipai buvo tyrinėti ir išbandyti kompiuterinio mokymosi sistemose. Vienas iš labai populiarių metodų tai **giliojo mokymo principai**, kurie sukurti pasiremiant biologiškai žmogaus smegenyse vykstančiais procesais, o tiksliau neuroniniais tinklais (Shrestha & Mahmood, 2019). Sistemos mokosi atlikti užduotis pagal pavyzdžius, dažniausiai be jokių tiksliai nustatytų programavimų ir taisyklių. Tokios sistemos vadinamos dirbtiniais neuroniniais tinklais. Jis pagrįstas sujungtų blokų ar mazgų, vadinamų dirbtiniais neuronais, rinkiniu, kuris netiksliai modeliuoja biologinių smegenų neuronus taip, kad jungčių struktūra primena neuronus struktūras gyvūnų vizualinėje smegenų žievėje (Voulodimos, Doulamis, Doulamis, & Protopapadakis, 2018). Kaip ir smegenyse pavieniai neuronai reaguoja į dirgiklius tik ribotoje regėjimo lauko srityje, vadinamoje receptiniu lauku. Skirtingų neuronų receptiniai laukai iš dalies persidengia taip, kad apimą visą vizualinį lauką. Ir tada kiekvienas ryšys, kaip ir biologinių smegenų sinapsės perduoda signalus arba kitaip informaciją, iš vieno dirbtinio neurono į kitą.



1.8 pav. Pilnai sujungtas neuronų tinklas su įvesties, paslėptais ir išvesties mazgais

Dirbtinis neuronas, kuris gauna signalą, gali jį apdoroti ir persiųsti naują signalą kitiems su juo sujungtiems neuronams. Įprastuose dirbtiniuose neuroniniuose tinkluose, signalas tarp neuronų yra realusis skaičius, o kiekvieno neurono išvestį apskaičiuoja netiesinė įvesčių sumos funkcija. Jungtys tarp dirbtinių neuronų vadinamos kraštais. Dirbtiniai neuronai ir kraštai turi reikšmes, kurios reguliuojasi vykstant mokymosi procesams. Reikšmės koreguoja signalo stiprumą neuronų jungtyse. Pavyzdžiui dirbtiniai neuronai gali turėti tokią ribinę reikšmę, kad signalas būtų siunčiamas tik tada, kai bendras signalas kerta šią reikšmę. Paprastai dirbtiniai neuronai yra su sugrupuojami į sluoksnius. Skirtingi sluoksniai gali atlikti skirtingas įvesčių transformacijas. Signalai toliau keliauja iš pirmo, įvesties, sluoksnio į paskutinį, išvesties, sluoksnį, dažniausiai pereidamas per skirtingus sluoksnius keletą kartų. Pradinis dirbtinių neuroninių tinklų tikslas buvo spręsti problemas taip pat, kaip tai darytų žmogaus smegenys. Tačiau laikui bėgant dėmesys buvo skiriamas konkrečioms užduotims, lemiančioms tinklo nukrypimą nuo biologijos. Dirbtiniai neuroniniai tinklai jau yra naudojami įvairioms užduotims atlikti, įskaitant kompiuterine rega, balso atpažinimą, automatinį vertimą, socialinio tinklo filtravimą, stalo ir vaizdo žaidimų automatinį žaidimą bei medicininę diagnostiką (Bengio, Courville, & Vincent, 2013). Siekiant vis atrasti geresnius būdus ir tobulinant neuroninius tinklus ir atsirado giliojo mokymosi principai, kurie nusako, kad dirbtiniame neuroniniame tinkle turime keletą paslėpt sluoksnių. Šis metodas bando imituoti žmogaus smegenų procesus, kai jie apdoroja šviesą ir garsus pagal vaizdą ir klausą. Sėkmingi pritaikymo būtai yra kompiuterinė rega ir balso atpažinimo sistemos.

Vienas iš stipriai nagrinėjamų metodų vaizdo atpažinimo problemoms spręsti yra **konvuliaciniai neuroniai tinklai**, kurie ir priklauso giluminio mokymosi neuroninių tinklų klasei. Šie tinklai yra tvarkingos formos daugiasluoksniai tinklai, kurie turi pilnai sujungtus

tinklus. Tiksliau tai reiškia, kad kiekvienas neuronas viename sluoksnyje yra sujungtas su visais kitais neuronais kitame sluoksnyje (Zhao, Zheng, Xu, & Wu, 2018). Dėl pilno šio tinklo sujungimo, jie yra linkę „perkrauti“ duomenis. Įprasti reguliavimo būdai arba apsisaugojimas nuo perkrovimo apima: bausmės parametrams mokymosi proceso metu, kaip reikšmių mažinimas (angl. *weight decay*), arba ryšių nukirtimas (angl. *trimming connectivity*), kaip praleistos jungtys (angl. *skipped connections*) arba duomenų nubyrėjimai (angl. *dropout*). Konvuliaciniams neuroniniams tinklams taikomi kitokie metodai skirti reguliavimui. Jie pasinaudoja hierarchine duomenų struktūra ir surenka sudėtingesnes struktūras naudojantis mažesniais ir paprastesniais modeliais esančiais struktūrų filtruose. Todėl esant dideliam ir sudėtingam kiekiui neuronų jungčių, konvuliaciniai neuroniniai tinklai turi mažesnius ekstremumo ribas. Palyginti su kitais vaizdų klasifikavimo algoritmais, šie tinklai naudoja santykinai mažą išankstinį apdorojimą. Tai reiškia, kad tinklas mokosi optimizuoti filtrus ar branduolius naudodamas automatinį mokymąsi, o tradiciniuose algoritmuose šie filtrai kuriami ranka. Šis savarankiškumas yra didelis pranašumas prieš anksčiau turimas žinias ir senesnius žmogaus būdus išskirti ir išanalizuoti bruožų išgavime ir analizėje. Konvuliacijų neuroninio tinklo pavadinimas nurodo, kad tinkle bus naudojamas matematinė operacija, vadinama konvuliacija. Šis tinklas yra specializuotas neuroninių tinklų tipas, kuris naudoja konvuliacijas vietoj tradicinės matricos daugybos, bent viename iš tinklo sluoksnių. Architektūra yra sudaryta iš krūvos atskirų sluoksnių, kurie per diferencijuojamą funkciją keičia įvesties tūrį į išvesties tūrį. Dažniausiai naudojami keli skirtingi sluoksniai (Voulodimos et al., 2018).

Tam, kad tinklas veiktų tvarkingai ir efektyviai, reikalinga tam tikra tinklo architektūra. Konvuliacinis neuroninis tinklas susideda iš įvesties sluoksnio, vidinių sluoksnių ir išvesties sluoksnio (Kavzoglu, 2009). Bet kokiame neuroniniame tinkle, kur įvestys yra pradžioje, vidiniai sluoksniai vadinami paslėptais, dėl to, kad įvestys ir išvestys yra užmaskuotos aktyvacijos funkcija ir galutine konvuliacija. Tai konvuliaciniame neuroniniame tinkle vidiniuose sluoksniuose yra atliekamos konvuliacijos.

1.9 pav. Konvuliacinio neuroninio tinklo sluoksniai

Paprastai tai yra ir sluoksnis, kuris atlieka skaliarinę sandaugą tarp konvuliacijos branduolio ir sluoksnio pradinėmis įvestimis. Realiai įvyksta operacija, kuri iš dviejų reikšmių atiduoda vieną reikšmę. Tuo pačiu dar naudojama ir aktyvavimo funkcija. Konvuliacijos branduoliui judant sluoksnio įvesties matrica, konvuliacijos operacija sugeneruoja funkcijų schemą, kuri savo ruožtu yra perduodama kitam sluoksniui ir tampa jo įvestis. Po to eina kiti sluoksniai, kaip pavyzdžiu „pooling“ (angl. *pooling layers*), visiškai sujungti (angl. *fully connected* layers) ir normalizavimo sluoksniai (angl. *normalization layers*) (Voulodimos et al., 2018). Šie sluoksniai bus detaliau aptarti žemiau.

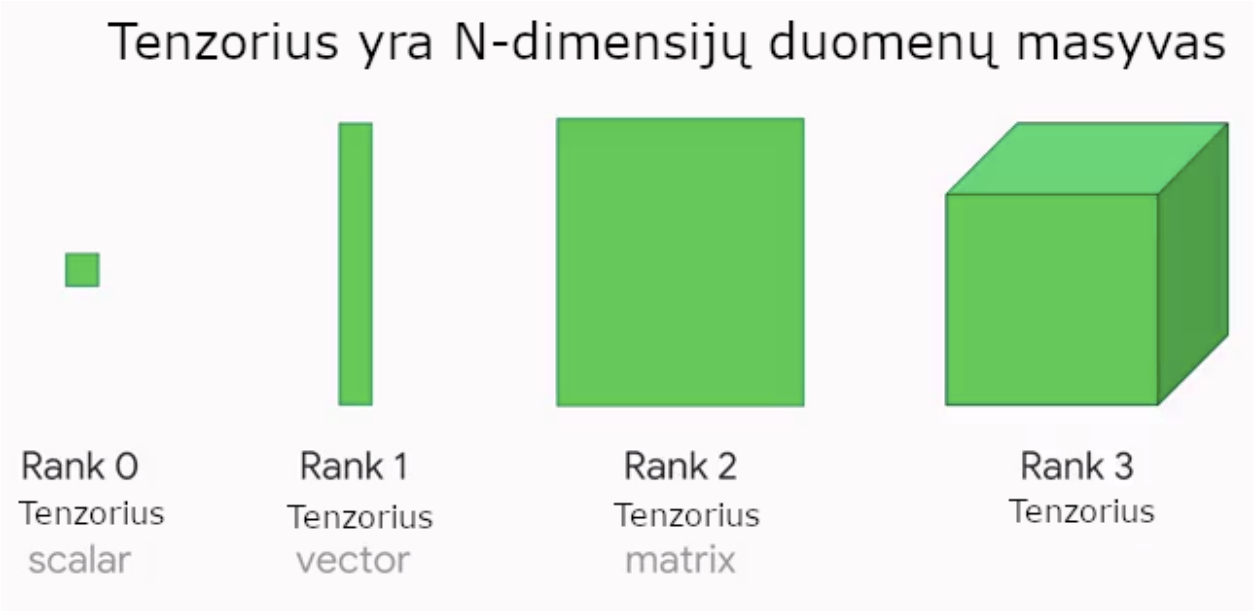
Šie tinklai naudoja daugiau hyperparametrų negu standartiniai daugiasluoksniai perceptronai. Nors vis dar taikomos įprastos mokymosi normos ir reguliarizavimo konstantų taisyklės, optimizuojant reikėtų atsižvelgti į šiuos papildomus parametrus (Shrestha & Mahmood, 2019):

1. **Filtrų skaičius.** Kadangi bruožų žemėlapio dydis sumažėja su gyliu, sluoksniai prie įvesties sluoksnio paprastai turi mažiau filtrų, o aukštesniuose sluoksniuose gali būti daugiau. Norint suvienodinti

kiekvieno sluoksnio skaičiavimą, funkcinių verčių produktas su pikselių padėtimi yra maždaug pastovus visuose sluoksniuose. Norint išsaugoti daugiau informacijos apie įvestį, reikėtų, kad bendras aktyvinimų skaičius nemažėtų iš vieno sluoksnio į kitą. Bruožų žemėlapių skaičius tiesiogiai valdo pajėgumą ir priklauso nuo turimų pavyzdžių skaičiaus ir užduočių sudėtingumo.

2. **Filtrų dydis.** Bendri filtrų dydžiai labai skiriasi randamoje literatūroje ir paprastai pasirenkami pagal duomenų rinkinį. Iššūkis yra rasti teisingą detalumo lygį tam kad būtų sukurtos reikiamo mastelio abstrakcijos ir jos netaptų perteklinės, duomenų atžvilgiu.
3. **Telkimo tipas ir dydis.** Šiuolaikiniuose tinkluose paprastai naudojams maksimalus telkinys, dažnai 2x2 dydžio, 2 žingsnių. Tai reiškia, kad sanaudos sumažinamos, o tai dar labiau padidina skaičiavimo efektyvumą. Jei įvestis ir didelė, gali reikėti ir didesnio telkimo proceso. Tačiau pasirinkus didesnes figūras, signalo matmuo labai sumažės, o informacijos perteklius gali būti prarastas. Dažnai geriausiai veikia nepersidengiantys telkimo sluoksniai.

Konvuliacijų sluoksniai yra pagrindiniai konvuliacinių tinklų kūrimo blokai. O tinklų įvestis yra tenzorius (angl. *tensor*) su forma. Tenzorius yra geometrinis objektas, kurio struktūra sudaro komponenčių sumą, kurios yra transformuojamos pagal tiesinius sąryšius. Tenzorių pavyzdžiai yra vektorius, skaliaras, matrica.



1.10 pav. Tenzoriaus galimi variantai pagal rangą

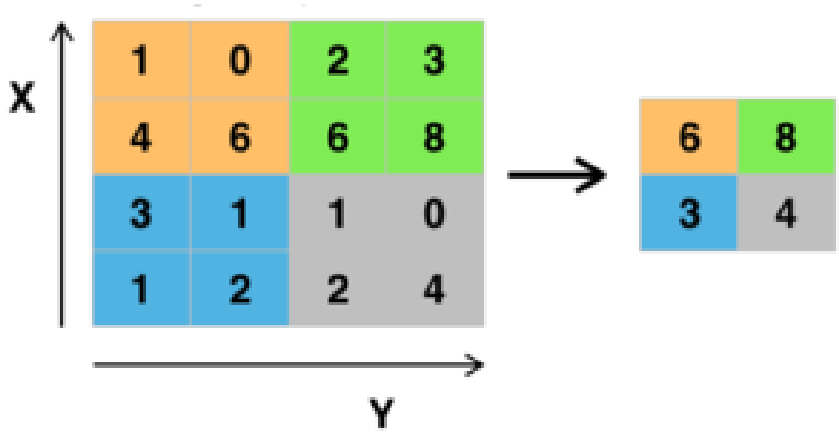
Neuroniniuose tinkluose tenzorių sudaro įvesčių skaičius * įvesčių aukštis * įvesčių plotis * įvesčių kanalai. Kanalas šiame kontekste yra bespalvė nuotrauka, tokio paties dydžio kaip įvesties nuotrauka, tik padaryta iš pirminių spalvų. Kadangi spalvotos nuotraukos yra padarytos iš pikselių, o pikseliai yra padaryti iš pradinių spalvų kombinacijų, kurios atstovaujamos kodo eilučių. Pavyzdžiui, tradicinė nuotrauką iš kameros turės raudona, žalią ir mėlyną kanalus, o bespalvė pilka nuotrauka turės tik 1 kanalą. Tai tokios išskaidytos nuotraukos pereina per konvuliacijų sluoksnius ir nuotrauka tampa atskirta į bruožų žemėlapį (angl. *feature map*), kuris dar kitaip vadinamas aktyvacijos žemėlapiu, su forma: įvesčių kiekį * bruožų žemėlapio aukštis * bruožų žemėlapio plotis * bruožų žemėlapio kanalai. Konvuliacijų sluoksniai konvuliuoja įvestis ir rezultatus perduoda kitam sluoksniui. Tai yra panašu į neuronus vizualinėje smegenų žievėje į tam tikrus dirgiklius. Tai kiekvienas konvuliacijos neuronas apdirba duomenis tik savo jautrumo zonoje. Vykstant procesams, kiekvienas filtras sukasi per įvesties tūrio plotį ir aukštį ir apskaičiuojamas taškų produktas tarp filtro įrašų ir įvesties, sudarydamas to filtro 2 dimensijų aktyvinimo žemėlapį. Tokiu būdu tinklas sužino filtrus, kurie suaktyvėja, kai aptinka tam tikrą konkrečios rūšies funkciją tam tikroje įvesties erdvinėje padėtyje (Indolia, Goswami, Mishra, & Asopa, 2018). Visų filtrų aktyvinimo žemėlapių sudėjimas pagal gylį sudaro viso konvuliuojamo sluoksnio išvesties tūrį. Todėl kiekvieną išvesties tūrio įrašą taip pat galima interpretuoti kaip neurono, nagrinėjančio nedidelį įvesties regioną, ir tame pačiame aktyvinimo žemėlapyje dalijančio parametrus su neuronais išvestį. Tokia architektūra paprastai yra nepraktiška didesniems duomenims, kaip pavyzdžiui aukštos skiriamosios gebos nuotraukoms. Tam reikėtų labai daug neuronų net ir mažai sluoksnių turinčioje architektūroje, dėl didelio nuotraukos dydžių, kur kiekvienas pikselis yra svarbus bruožams atpažinti (Guo et al., 2016). Pavyzdžiui, visiškai sujungta tinklas, 100 x 100 dydžio nuotraukai turi 10000 reikšmių kiekvienam neuronui antrame sluoksnyje. Tačiau konvuliacijos sumažina laisvų parametrų skaičių ir tai leidžia tinklui būti gilesniam. Pavyzdžiui, nepriklausomai nuo vaizdo dydžio, naudojant 5x5 dengimo regioną, kurio kiekvieno reikšmės tokios pačios, tereikia 25 mokymosi parametrų. Naudojantis pasirinktomis reikšmėmis lyginant su mažiau parametrų algoritmais, išvengiama nykstančių gradientų ir sprogstančių gradientų problemų, pastebimų „backpropagation“ metu tradiciniuose neuroniniuose tinkluose. Be to, konvuliaciniai neuroniniai tinklai idealiai tinka duomenims su į tinklelį panašia topologija, kaip pavyzdžiui, nuotraukas, nes erdvės ryšiai tarp atskirų bruožų yra įvertinami vykstant konvuliacijoms arba telkimo (angl. *pooling*) metu. Trys hyperparametrai konvuliacijų sluoksnio išvesties dydžius:

- Išvesties tūrio ***gylis*** valdo neuronų skaičių sluoksnyje, kuris susijungia su tuo pačiu įvesties tūrio regionu. Šie neuronai išmoksta aktyvuoti dėl

skirtingų įvesties funkcijų. Pavyzdžiui, jei pirmasis konvuliacinis sluoksnis paima neapdirbtą nuotrauką kaip įvestį, tai skirtingi neuronai išsidėstę gylio dimensija gali suaktyvėti esant įvairiai orientuotiems kraštams arba spalvų dėmėms.

- ***Žingsnis*** (angl. *stride*) kontroliuoja, kaip paskirstomi gylio stulpeliai aplink plotį ir aukštį. Jeigu žingsnis yra 1, tada filtrus perkeliame po vieną pikselį. Dėl to, tarp stulpelių labai sutampantys imlūs laukai ir dideli išvesties svoriai. Bet kuriam sveikajam skaičiui S>0, žingsnis S reiškia, kad filtras yra koreguojamas S vienetais vienu metu per išvestį. Didesnis žingsnis reiškia mažesnį sutapimą su imlumo laukais ir mažesnius išvesties tūrio dimensijas išvesties tūriui.
- Kartais patogu įvestis pateikti nuliais (arba kitomis reikšmėmis, pvz.: regiono vidurkiu) ant įvesties tūrio krašto. Šio ***užpildymo*** (angl.*padding*) dydis yra trečias hyper parametras. Užpildymas reguliuoja išvesties tūrio erdvinį dydį. Ypač kartais pageidautina tiksliai išsaugoti įvesties tūrio erdvinį dydį, kuris paprastai vadinamas tuo pačiu užpildymu.

Kita svarbi konvuliacinių neuroninių tinklų koncepcija yra **telkimas** (angl. *pooling*), kuris yra netiesinio mažinimo forma. Konvuliaciniai tinklai gali naudoti vietinius ar globalius telkimo sluoksnius kartu su tradiciniais konvuliaciniais sluoksniais. Telkdami sluoksnius sumažiname duomenų matmenis. Mažinimas vyksta jungiant neuronų grupių išvestis viename sluoksnyje į vieną neuroną kitame sluoksnyje. Vietiniame telkinyje sujungiamos mažos grupelės, paprastai naudojamos 2x2 matricos.



1.11 pav. 1 gylio maksimalaus telkimo pavyzdys

Globalieji telkimo procesai vyksta su visais neuronais esančiais bruožų žemėlapyje. Yra kelios nelinijinės funkcijos skirtos atlikti telkimui ir standartiškai naudojami 2 telkimo

metodai: maksimumas ir vidurkis (Voulodimos et al., 2018). Maksimumas telkimas (angl. *max pooling*) naudoja didžiausią vertę kiekvienos neuronų grupės vertę bruožų žemėlapyje. Jei tiksliau, tai jis perskirsto įvesties nuotrauka į stačiakampių rinkinį ir kiekvienam tokiam sub-regionui išskiria maksimumą. Intuityviai, tiksli funkcijos vieta yra mažiau svarbi nei grubi, palyginti su kitomis funkcijomis. Tokia yra sumanymo, kaip naudoti telkimą konvuliaciniuose neuroniniuose tinkluose. Telkiamasis sluoksnis padeda palaipsniui sumažinti atstovavimo erdvinį dydį, sumažinti parametrų skaičių, atminties apkrovimą ir skaičiavimų kiekį tinkle bei kontroliuoti perteklinį mokymąsi ir prisitaikymą. Neretai tinklo architektūroje periodiškai įterpiamas telkimo sluoksnis tarp vienas po kito einančių konvuliacijų sluoksnių, kurios paprastai lydimos aktyvacijos funkcija, kaip pvz. „ReLU“ sluoksnis. Nors telkimo sluoksniai prisideda prie vietos vertimo nevienodumo, jie neužtikrina visuotinės transformacijų nekintamumo tinkluose, nebent naudojama globalaus telkimo forma. Telkimo sluoksnis paprastai veikia nepriklausomai kiekviename įvesties gylyje ar pjūvyje ir keičia jų mastelį erdvėje. Labai dažna maksimumo telkimo forma yra 2x2 dydžio filtras, taikomas su 2 žingsniu, kuris kiekvieną įvesties gylio pjūvį išbando 2 reikšme pagal plotį ir aukštį, atmesdamas 75% aktyvumo. Kaip ir minėta, kad be maksimalaus telkimo, telkimui galima naudoti ir kitas funkcijas, kaip pavyzdžiui vidutinis telkimas. Vidutinis telkimas (angl. average pooling) naudoja vidurkio vertę. Vidutinis telkimas dažnai buvo naudojamas istoriškai, tačiau pastaruoju metu tapo nepalankus, palyginti su maksimaliu telkimu, kuris praktiškai yra geresnis.

Visiškai sujungti sluoksniai (angl. *fully connected layers*) jungia kiekvieną neuroną viename sluoksnyje su kiekvienu kito sluoksnio neuronu. Tai tas pats, kas tradicinis daugiasluoksnis perceptronų (angl. perceptron) neuroninis tinklas. Perceptronas - žmogaus smegenų veiklą atkartojanti adaptyvioji informacinė sistema, kuri, keisdama savo struktūrą, keičia ir savo elgseną ir tokiu būdu geba prisitaikyti prie funkcinės veiklos. Kompiuterio mokyme perceptronas yra algoritmas skirtas prižiūrimame mokyme dvejetainiams klasifikavimams. Dvejetainis klasifikatorius yra funkcija, kuri gali nuspręsti ar įvestis, atstovaujama skaičių vektoriaus, priklauso kokiai nors skaičių klasei (Voulodimos et al., 2018). Tai yra klasifikavimo algoritmas, kuris savo prognozes formuoja funkcija, derinančia svorių rinkinį su bruožų funkcijų vektoriumi. Šiuo sluoksniu atliekama paskutinė klasifikacija.

Imliajame lauke, neuroniniuose tinkluose, kiekvienas neuronas gauna informacija iš tam tikrų vietų ankstesniame sluoksnyje. Konvuliaciniame sluoksnyje, kiekvienas neuronas gauna įvestis iš riboto ankstesnio sluoksnio ploto, vadinamo neurono imliuoju lauku. Paprastai plotas yra kvadratas, 5x5 neuronų. O visiškai sujungtame sluoksnyje , imlus laukas yra visas ankstesnis sluoksnis. Taigi kiekviename kontūriniame sluoksnyje kiekvienas neuronas įneša indėli iš didesnio įvesties ploto nei ankstesniuose sluoksniuose. Taip yra dėl to, kad

konvoliucija taikoma nuolat atsižvelgiant į pikselio vertę ir aplinkinius pikselius. Naudojant išplėstus sluoksnius, imliajame lauke pikselių skaičius išlieka pastovus, tačiau, derinant kelių sluoksnių efektą, laukas yra rečiau apgyvendinamas, nes jo matmenys didėja.

Kiekvienas neuronas tinkle apskaičiuoja išvesties **vertę**, taikydamas konkrečia funkciją įvesties reikšmėms, gautoms iš ankstesnio sluoksnio imliojo lauko. Įvesties reikšmėms taikoma funkcija yra nustatoma reikšmių vektoriumi ir šališkumu (paprastai realiais skaičiais). Mokymasis susideda iš pasikartojančio šių tendencijų ir reikšmių reguliavimo. Reikšmių vektorius ir šališkumas yra vadinami filtrais ir atspindi tam tikras įvesties savybes, kaip pavyzdžiui konkrečią formą. Skiriamasis konvuliacinių tinklų bruožas yra tas, kad daugelis neuronų gali turėti tą patį filtrą. Tai sumažina atminties pėdsaką, nes visuose imliuose laukuose kuriuose yra filtras, naudojamas vienas šališkumas ir vienas svorio vektorius, o ne kiekvienas imlus laukas turintis savo šališkumą ir vektorių reikšmes.

Reguliarizacija – tai papildomos informacijos pateikimas, siekiant išspręsti netinkamą problemą arba užkirsti kelią duomenų analizės pertekliui. Konvuliaciniai neuroniniai tinklai taiko įvairus reguliarizacijos būdus: išmetimas (angl. *dropout*), „Dropconnect, tikimybinis telkimas (angl. *stochastic pooling*), dirbtiniai duomenys (angl. *artificial data*), ankstyvas sustojimas (angl. *early stopping*), parametrų skaičius, reikšmių nykimas (angl. *weight decay*), maksimalių normų ribojimai (angl. *max norm constraints*).

Kadangi pilnai sujungtas sluoksnis turi beveik visus parametrus, tikimybė, kad jis taps su pertekliniais duomenimis ir mokysis blogai yra didelė tikimybė. Todėl vienas metodas sumažinti pertekliniam mokymuisi yra **išmetimas**. Kiekviename mokymosi etape, individualūs mazgai yra išmetami iš tinklo arba ignoruojami su tikimybe, kad būtų paliktas sumažintas tinklas (Shrestha & Mahmood, 2019). Tik sumažintas tinklas yra mokomas to etapo duomenų. Pašalinti mazgai iš naujo įstatomi į tinklą su pirminėmis reikšmėmis. Vengiant mokyti visus mazgus su visais mokymosi duomenimis, išmetimas sumažina perteklinio mokymosi tikimybe. Dėl to modelio derinys tampa praktiškas net ir giliems nerviniams tinklams. Šis metodas mažina mazgų sąveiką, todėl jie išmoksta patikimesnes funkcijas kurios geriau apibendrina naujus duomenis (Wan, Zeiler, Zhang, LeCun, & Fergus, 2013). Reikia nepamiršti, kad pagrindinis trūkumas yra tas, kad išmetimo metodas neturi tokios pat naudos konvuliaciniems sluoksniams, kur neuronai nėra visiškai sujungti.

„Dropconnect“ metodas yra išmetimo apibendrinimas, kai kiekvienas ryšys, o ne kiekvienas išvesties vienetas, o ne kiekvienas išvesties vienetas, gali būti atmestas su tikimybe. Todėl kiekvienas vienetas gauna įvestį iš atsitiktinio ankstesnio sluoksnio vienetų poaibio. Šis metodas yra panašus į išmetimą, nes modelyje atsiranda dinaminis retumas, tačiau skiriasi tuo, kad retėjimas atsiranda reikšmėse negu sluoksnių išvesčių vektoriuose. Reiškia su

„dropconnect“ pilnai sujungti sluoksniai tampa retai sujungtais sluoksniais, kurie pasirenkami pagal tikimybę algoritmų mokymosi metu, taip išvengiant perteklinio mokymosi (Wan et al., 2013).

Tikimybiniame telkime (angl. stochastic pooling) įprastinės deterministinio telkimo operacijos pakeičiamos tikimybinėmis procedūromis, kai aktyvinimas kiekviename telkiamajame regione pasirenkamas atsitiktine tvarka pagal daugianarį pasiskirstymą, kurį lemia veikla telkiamajame regione. Šis metodas neturi hyper parametrų ir gali būti derinamas su kitais reguliavimo metodais, pavyzdžiui išmetimu ar duomenų korekcija. Kad būtų aiškiau, šitas telkimas prilygsta maksimumo telkimui, bet turi daugiau įvesties vaizdo kopijų, kurių kiekviena turi nedideles vietines deformacijas (Guo et al., 2016). Naudojant tikimybinį telkimą daugiasluoksniame modelyje deformacijų skaičius yra eksponentiškas, nes atranka aukštesniuose sluoksniuose yra nepriklausoma nuo toliau pateikt.ų

Dirbtiniai duomenys (angl. artificial data) tai duomenų analizės metodas, naudojamas duomenų kiekiui padidinti, pridedant šiek tiek modifikuotas jau esamų duomenų kopijas arba naujai sukurtus sintetinius duomenis iš esamų duomenų. Kadangi modelio perpildymo laipsnis priklauso ir nuo jo galios, ir nuo mokymo, kurį jis gauna, apimančio tinklo, kuriame yra daugiau mokymo pavyzdžių, naujų duomenų sukūrimas gali sumažinti perpildymą ir veikia kaip reguliatorius (Nusrat & Jang, 2018). Pavyzdžiui, įvesties vaizdai gali būti asimetriškai apkarpyti keliais procentais, kad būtų sukurti nauji pavyzdžiai su ta pačia etikete kaip ir originalas (Guo et al., 2016).

Ankstyvas sustojimas (angl. early stopping) yra tam tikra regulerizavimo forma tam kad išvengti perteklinio mokymosi, kai kompiuteris mokosi kartotinio metodu. Tokie metodai atnaujina mokymosi modelį, kad modelio duomenys geriau atitiktų kiekvieną pakartojimą. Tai iki tam tikros ribos pagerina modelio mokymosi rezultatus, susijusius us duomenimis, nepriklausančiais mokymo rinkiniui. Tačiau, peržengus šią ribą, pagerinti modelio gebėjimus atpažinti ir klasifikuoti objektus yra nebenaudinga ir tai tik didina analizės ir apibendrinimo klaidas. Ankstyvas sustojimo taisyklės leidžia nusistatyti, kiek epochų gali būti paleista, kol modelis pradės veikti pertekliuje. Šis metodas gali būti taikomas skirtinguose metoduose.

Kitas paprastas būdas išvengti perteklio, tai riboti **parametrų skaičių**. Paprastai ribojama paslėptų vienetų skaičius kiekviename sluoksnyje arba apribojamas tinklo gylis. Konvuliaciniuose neuroniniuose tinkluose, filtro dydis taip pat turi įtakos parametrų skaičiui. Ribojant parametrų skaičių, tiesiogiai ribojama tinklo prognozinė galia, dėl ko yra mažinamas funkcijų sudėtingumas, kurios nagrinėja duomenis ir todėl ribojamas duomenų analizės perteklius.

Reikšmių nykimas (angl. weight decay) yra paprasta pridėtinio reguliarizavimo forma, kuri tiesiog prie kiekvieno mazgo klaidos prideda papildomą klaidą, proporcinga reikšmių sumai (L1 norma) arba reikšmės vektoriau kvadratiniam dydžiui (L2 norma). Priimtino modelio sudėtingumo lygį galima sumažinti didinant proporcingumo konstantą, taip padidinant baudą stambių reikšmių vektoriams. L2 reguliarizacija yra dažniausiai naudojama forma. Ji gali būti įgyvendinama nubaudžiant visų parametrų, tiesiogiai susijusių su tikslu, kvadratinį dydį. L2 įteisinimas intuityviai aiškina, kaip smarkiai baudžiami aukščiausių reikšmių vektoriai ir pirmenybė teikiama difuziniams reikšmių vektoriams. Dėl daugkartinės reikšmės ir įvesties sąveikos, tai naudinga priemonė skatinti tinklą naudoti visas savo įvestis po truputi, o ne kai kurias įvestis ir daug. Taip pat dažnas yra ir L1 reguliarizavimas. Optimizavimo metu reikšmių vektoriai sumažėja. Kitap tariant, neuronai sureguliavę L1, naudoja tik retą svarbiausių savo įvesčių pogrupį ir tampa nejautriu triukšmingoms įvestims (Nusrat & Jang, 2018). L1 ir L2 reguliarizavimai gali būti sujungti.

Maksimalių normų ribojimai (angl. max norm constraints) užtikrina absoliučią viršutinę kiekvieno neurono reikšmės vektoriaus dydžio ribą ir naudoti prognozuojamą gradiento nusileidimą suvaržymui vykdyti. Praktiškai tai atitinka parametrų atnaujinimo atlikimą, kaip įprasta, ir apribojimo vykdymą, pritaikant kiekvieno neurono reikšmės vektoriui šiuos apribojimus.

Norint įsivertinti neuroninio tinklo efektyvumą reikia vertinti 2 dalykus: **nuostolius** ir **tikslumą**. Tikslumas ir nuostoliai yra dvi labiausiai žinomos ir aptartos mašinų mokymosi metrikos Kuo mažesnis nuostolis tuo geresnis mokymosi modelis, išskyrus kai modelis per daug prisitaikė prie mokymosi duomenų. Skirtingai nuo tikslumo, nuostoliai nėra procentinis dydis – tai kiekvieno mėginio klaidų, padarytų mokymo ar patvirtinimo rinkiniuose, apibendrinimas. Jų interpretacija parodo, kaip gerai sekėsi modeliui dirbti su šiais 2 duomenų rinkiniais. Natūralu, kad pagrindinis mokymosi modelio tikslas yra sumažinti praradimo funkcijos vertę, palyginti su modelio parametrais, keičiant reikšmės vektoriaus vertes įvairiais optimizavimo metodais. Nuostolių vertė reiškia, kaip gerai arba prastai veikia tam tikras modelis po kiekvieno optimizavimo pakartojimo. Geriausia būtų tikėtis nuostolių sumažinimo po kiekvieno ar kelių pakartojimų. Dažniausiai pasitaikančios praradimo funkcijos yra „log loss“ ir „cross-entropy loss“ (kurios duoda tokį patį rezultatą skaičiuojant klaidų lygius nuo 0 iki 1), taip pat vidutinė kvadratinė klaida (angl. *mean squared error*) ir tikimybės praradimas (angl. *likelihood loss*.). Skirtingai nuo tikslumo, praradimas gali būti naudojamas ir klasifikavimo, ir regresijos problemų atveju. Modelio tikslumas paprastai nustatomas išmokus ir nustačius modelio parametrus ir modeliui nesimokant, paprastai jis išreiškiamas procentais. Tada bandiniai paduodami modeliui ir palyginus su tikraisiais duomenimis, fiksuojamas klaidų

skaičius, kurį daro modelis. Tam tikras mėginys yra dvejetainis: teisingas arba klaidingas. Tada apskaičiuojama klaidingo klasifikavimo procentinė dalis. Pavyzdžiui, jei bandinių skaičius yra 1000, o modelis teisingai klasifikuoja 952 mėginius, modelio tikslumas yra 95,2. Taip pat yra tam tikrų subtilybių mažinant nuostolių vertę. Paprastai galime pastebėti kad tikslumas didėja, kai nuostoliai mažėja, tačiau taip būna ne visada. Tikslumas ir nuostoliai turi skirtingus apibrėžimus ir matuoja skirtingus dalykus. Dažnai jie atrodo atvirkščiai proporcingi, tačiau tarp šių dviejų metrikų nėra matematinio ryšio. O dažniausia problema, kuri gali iškraipyti rezultatus, yra standartinis modelio perteklinis prisitaikymas jau minėtas prieš tai darbe.

Atlikus visą mokslinės literatūros analizę buvo priimtas sprendimas paieškos sistemai naudoti 2D vaizdų atpažinimą. Viena iš priežasčių, kad darant multi-projekcijas padarome objektą atsparų rotacijai ir masteliui, kadangi taip išsaugomas vienas tas pats dydis ir asocijuojami skirtingai kampai pagal nuotraukas (Tangelder & Veltkamp, 2008). Be to naudojantis nuotraukomis mūsų nevaržo jokia programinė įranga ar formatas ir norint atlikti savo funkcijas tereikia sukaupti tinkamą kiekį nuotraukų. Be to naudojantis konvuliaciniais neuroniniais tinklais galime žymiai efektyviau įgyvendinti savo tikslus ir išspręsti problemas. Be to vienas iš svarbių faktorių, kad nuotraukų nagrinėjimas yra stipriai plėtojamas ir galima išbandyti ir pasinaudoti daugybę kitų autorių jau atrasta patirtimi, kas leidžia bandyti įgyvendinti stipresnius tikslus ir sukurti geresnę paieškos sistemą

2. SISTEMOS PROJEKTAS

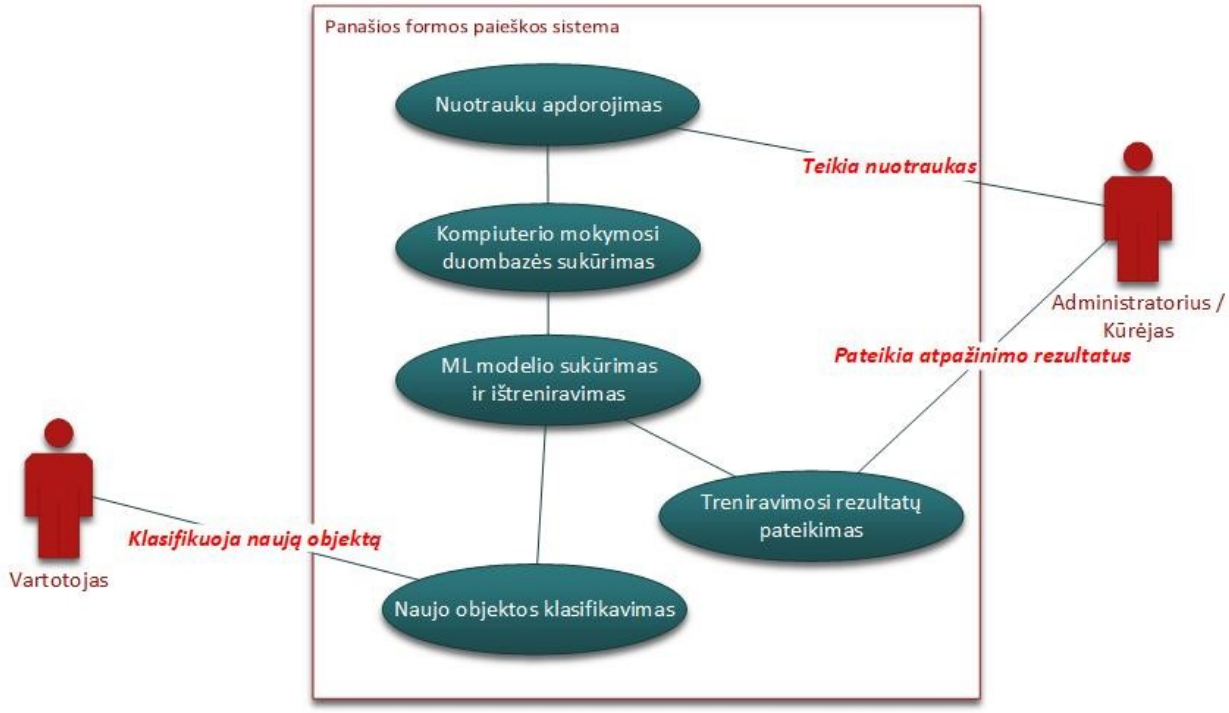
Šis darbo skyrius skirtas pasiruošti paieškos sistemos kūrimui ir naudojimui realizacijoje. Pirmiausia paruošiame sistemos diagramas, kurios yra skirtos aiškiai pateikti sistemos veikimo principus. Diagramos sukurtos pagal dažniausiai naudojamą informacinių sistemų modeliavimo, UML (angl. *unified modeling language*), kalbos principus. Taip pat aprašėme ir naudojamą programinę įrangą ir jos privalumus mūsų sistemoje.

2.1. Diagramos

Atliekame užduočių, ansamblių ir sekų diagramas, kurios aprašo pagrindines sistemos funkcijas, sąveikaujančių objektų elgesį, aktyvumą ir poveikį kitiems objektams.(Sokas 2012). Šiame darbe yra pasirinktos šios diagramos ir UML kalba dėl to, kad leidžia parodyti programos veikimo principus, paruošti sistemą ir išgryninti problemas bei funkcijas, kurias atliekame savo paieškos sistemoje.

2.1.1. Užduočių diagrama

Užduočių diagramose yra nurodomos svarbiausios sistemos funkcijos bei aktoriai ir jų roles programoje (Booch, G., J. Rumbaugh, 2000). Mūsų programoje turime 2 aktorius: vartotoją inžinierių ir duomenų administratorių arba programos kūrėją.



2.1 pav. Paieškos sistemos užduočių diagrama

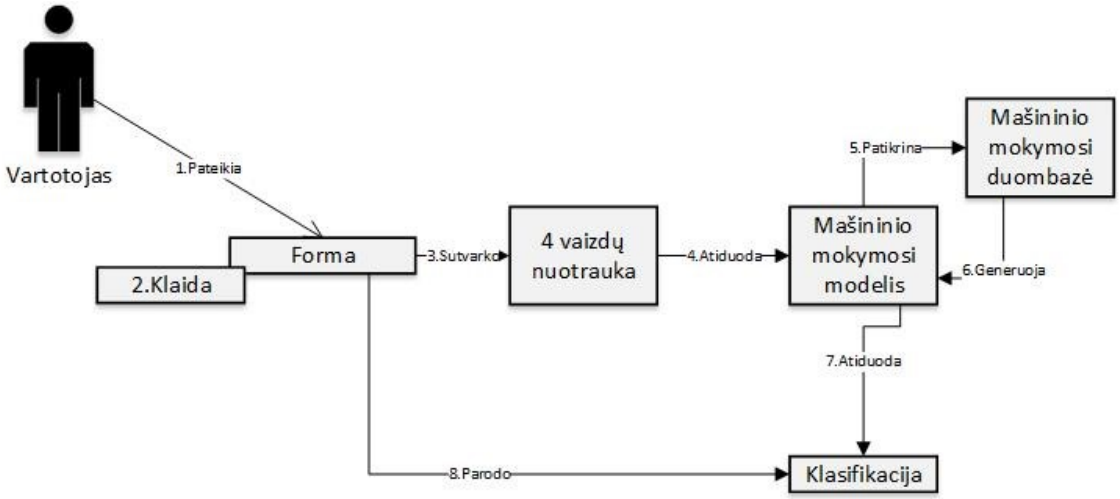
Paveikslėlyje 2.1 paveikslėlyje matome, kad paieškos sistema prasideda nuo pirmos užduoties – nuotraukų apdorojimo. Nuotraukų apdorojimas atliekamas tada kai kūrėjas ar

administratorius pateikia visą reikalingą informaciją. Tai yra svarbi dalis, nes pagal ją, sistema apdorojusi duomenis, toliau dirbs ir mokysis atpažinti ir klasifikuoti objektus. Kad programa galėtų mokytis iš apdorotų nuotraukų yra atliekamas antras žingsnis ir yra sukuriama kompiuterinio mokymosi duomenų bazė. Turint pakankama mokymosi bazė programa galima atlikti trečiąjį žingsnį ir sukurti modelį bei pagal jį išsitreniruoti tinkamai klasifikuoti objektus. Kadangi pati sistema gali išsimokyti neteisingai dėl prasto duomenų kiekio, papildomai reikalinga kūrėjo patikra. Pagal turimus rezultatus kurie gaunami atlikus 3 žingsnį, kūrėjas įsivertina ar sukurtas modelis yra tinkamas naudoti tolimesniame klasifikavime. Paskutiniame žingsnyje vartotojas paduoda naujus objektus ir turimas modelis įvertina objektus, juos suklasifikuoja ir pateikia duomenis vartotojui.

2.1.2. Ansamblių diagrama

Ansamblių diagrama yra diagrama, kuri parodo sąveikaujančių objektų elgesį. Viena iš išskirtinių jos savybių yra kelias, kuris nurodomas kaip susiję objektai. Antra, tai atliekamų veiksmų numeracija kuri parodo eilės tvarką (Booch, G., J. Rumbaugh, 2000). Norint tinkamai sukurti šią diagramą turime nusistatyti sistemoje esančius objektus, kurie ir turės veikti tarpusavyje.

Pagal pradinę užduočių diagramą sistemoje turime tokius objektus: forma, 4 vaizdu nuotrauka, mašininio mokymosi modelis, mašininio mokymosi duomenų bazė, klasifikatorius. Rodyklėmis parodyti objektų perduodami pranešimai ir informacija. Taip pat pateikiame veiksmo pavadinimus ir atliekamų veiksmų numeracija.



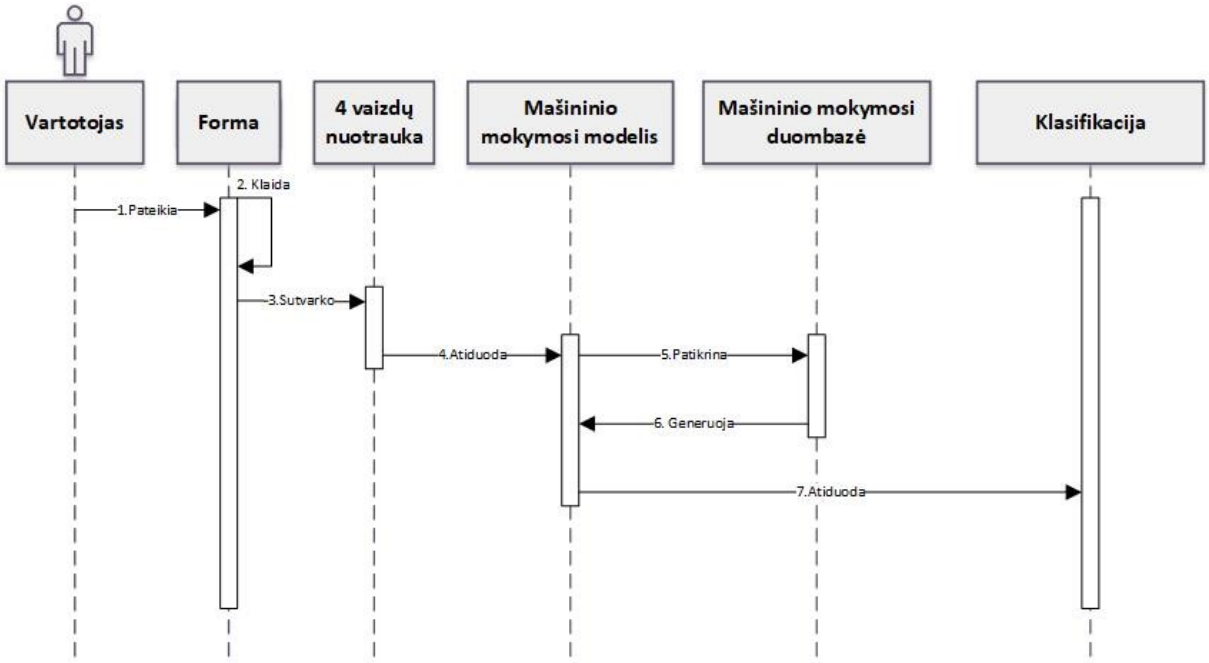
2.2 pav. Programos ansamblių diagrama

Aukščiau pateiktame paveikslėlyje matome, kad vartotojas pateikia formai objektą ar objektus, kuriuos tikriname. Forma aptvarko gautas objektų nuotraukas ir padaro 4 vaizdų

nuotrauką, kuri atiduodama mašininio mokymosi modeliui išnagrinėti. Modelis patikrina nuotrauką tarp jau turimos duomenų bazės ir gauna sugeneruotą atsakymą. Toliau vyksta klasifikacija ir forma parodo rezultatus – kam priklauso objektas arba kokia tikimybė, kad objektas priklauso kokiai nors kategorijai.

2.1.3. Sekų diagrama

Sekų diagramos parodo ne tik objektus arba objektų sąveiką, bet ir objektu poveikį kitiems objektams laiko atžvilgiu bei jų aktyvumą. Ši diagrama labiausiai akcentuoja pranešimų užsakymo laikus ir trukmę. Tai parodo paveikslėlyje 2.3 vertikaliose ašyse esantys objektai. Šioje diagramoje matome siunčiamus pranešimus ir kiekvienos operacijos laiką ir trukmę. Diagramoje pateikiami objektai atitinka ansamblių diagramą ir jų atliekamus pranešimus.



2.3 pav. Sekų diagrama

Kad būtų aiškesni sekų diagramoje esantys objektai ir jų atliekami pranešimai atliekame trumpus jų paaiškinimus:

1. Pateikia – vartotojas pateikia modelį ar modelių nuotraukas, kurias nori patikrinti sistemoje.
2. Klaida – esant, klaidai grąžinama į formą
3. Sutvarko – modeliai arba nuotraukos yra sutvarkomos, kad būtų patogiau naudojamos mašininio mokymosi modelyje. Šioje sistemoje yra sutvarkoma taip, kad 1 nuotraukoje matome 4 objekto vaizdus: frontalinis, kairinis, viršutinis ir izometrinis vaizdai.
4. Atiduoda – paruoštas vaizdas ir atiduodamas mašininio mokymosi modeliui

5. Patikrina – modelis susitvarko savo procesus. Optimizuoja nuotrauką ir duomenis ir atiduoda mokymosi duomenų bazei, kuri pasipildo dar vienu įrašu apie objektą ateičiai ir tuo pačiu patikrina papildomus duomenis.
6. Generuoja – Nesant objektu pasikartojimui. Duomenų bazė generuoja visą turimą informaciją ir atiduoda modeliui
7. Atiduoda – modeliui atlikus savo procesus yra sukuriamas ir atiduodama klasifikavimo analizė. Joje galima pamatyti kiek faktiškai objektas yra panašu į kokią nors formą ar detalę.

2.2. „Google colaboratory“ (Colab) platforma

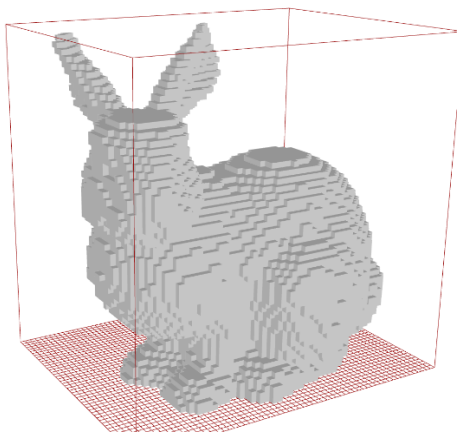
Sistemos projektui įgyvendinti vien tik diagramų nepakanka. Tam kad visa paieškos sistema veiktų mes naudojame mašininio mokymosi principus. Tai reiškia, kad yra sukurtas modelis, kuris gali mokytis iš daugybės gerai sutvarkytų duomenų. Mažas duomenų kiekis ir jų kokybė gali sugadinti visą mašininio mokymosi procesą (Yin et al., 2018). Didelis duomenų kiekis automatiškai apkrauna kompiuterius ir taip sulėtina visą procesą. Todėl šiame darbe naudojame „Google colaboratory“ (kitaip žinomą, kaip Colab). Tai yra debesijos paslauga sukurta pagal „Jupyter notebooks“, kuri skirta mašininio mokymosi mokymams ir tyrimams atlikti. Ši paslauga turi visiškai nemokama priėjimą prie aukštų standartų kompiuterio įrangos bei suteikia pilnai paruošta aplinka mašinų mokymuisi. Taip pat visa paslauga yra prijungta prie „Google drive“, dėl to yra lengva prieiga prie duomenų iš bet kokios vietos, kur yra interneto ryšis. (Google, n.d.) Kadangi ši aplinka yra padaryta pagal „Jupyter notebook“, tai ir „colab“ veikimo principas yra toks pats. Kiekvienas dokumentas yra sudarytas iš daugybės langų, kuriuose gali būti kodas ir kitokia informacija. Kitokia informacija, tai tekstas, nuotraukos, lentelės ar kiti grafiniai elementai. Dėl tokios struktūros yra lengviau pateikti ir aprašyti visą programinį kodą. O kadangi ši aplinka lengvai prieinama, todėl ja galima naudotis atliekant komandinius projektus. Pagal T.Cameiro straipsnį, kuriame jis nagrinėjo „Colab“, kaip įrankį atlikti mašininio mokymosi užduotims atlikti buvo padarytos šios išvados (Carneiro et al., 2018):

- „Colab“ teikiami techninės įrangos išteklių pakaks įgyvendinti nuotraukų atpažinimo poreikius mašininio mokymosi principais.
- Šios aplinkos efektyvumas yra lygus kaip naudojantis dedikuota technine įranga su panašiais ištekliais.
- Tai yra puiki aplinka dalintis žiniomis.

Ši aplinka ypač tinka mums dėl aukščiau išvardintų priežasčių. Be to T.Cameiro straipsnyje buvo naudojama tokie nustatymai, kurie bus naudojami ir mūsų programoje.

2.3. 2D vaizdų pasirinkimas

Atlikus literatūros analizes buvo pastebėta, kad daugiausia paieškos sistemos buvo naudoti taškų debesų algoritmai arba ten kur naudota figūrų paieška pagal nuotraukas, tai naudoti figūrų šešėliai (Chen et al., 2003; Lian et al., 2010). Tačiau paieškos sistema turėtų naudotis inžinieriai arba vartotojai, kuriems reikia paieškos sistemos su detalumu, tada tokie variantai gali netikti. Šešėliai apibūdina tik formą, kuri nėra aiški, o debesų taškai turi kitas problemas. Pagal Zhang atliktą analizę, taškų debesų nagrinėjimas yra atliktas įvairiais būdais: nuo nuotraukų kūrimo, kur nuotraukose, skirtingose projekcijose yra sudedami taškai arba taškų pakeitimas į mažiausius trimatės erdvės elementus (angl. *voxel*)(Y. Zhang & Rabbat, 2018). Tačiau visi iš būdų reikalauja didelio duomenų kiekio, apdirbimo ir daug skaičiavimo ir analizės, dėl ko gali atsirasti labai daug klaidų, kurios gali sugadinti visą atpažinimo procesą. Taip pat duomenų gali būti nepakankamai, kad atpažinti 3D duomenis. Tokia problema atsiranda dėl to, kad sąsūkų neuroniniai tinklai (angl. *convolution neural network*) yra sukurti įeitiniems duomenims, kurie yra surikiuoti fiksuoto dydžio matricose, kurioms galima atlikti sąsūka (angl. *convolution*). Tačiau taškų debesyse yra nestruktūrizuoti, netvarkingi ir įvairaus kiekio taškai. Todėl natūraliai jie nėra tinkami į erdvines matricas (Ben-Shabat, Lindenbaum, & Fischer, 2018). Pagal Hong, mažiausi trimatės erdvės elementai irgi nėra tinkami, kadangi turi retų ir neaiškių dalių ir yra pateikti per tokia forma, kad gali prarasti daug informacijos apie objektą.



2.4 pav. 3D Objektas sudarytas iš vokselių

Taip pat paveikslėlyje galime matyti, kad tokie objektai reikalauja daug techninių išteklių, kad galima būtų apdoroti visą informaciją (Hong et al., 2020). Paskutinis būdas yra objektų paviršiaus suskaidymas į tinklą (angl. *mesh*), kuris yra labiau naudingas kompiuterinei grafikai ir objektų atpažinimo procesuose bus per daug sudėtingas.

Dėl aukščiau išvardintų priežasčių šiame darbe ir pasirinkome atpažinimą ir paiešką pagal nuotraukas. Kadangi pagal mokslinės literatūros analizę, tai yra detaliai išnagrinėtas procesas ir pagal (Shafi, 2017) atpažinimas iš nuotraukų pagal objektų atpažinimo vertinimo kriterijus beveik nesiskiria nuo kitų būdų. Iš išnagrinėtos literatūros mes darome prielaidą, kad objektų atpažinimas iš nuotraukų turi ir daugiau galimų pliusų, kaip kad galimybe pateikti daugiau duomenų apie objektus. Kadangi programos vartotojas yra inžinierius, tai mes teigiame, kad naudojantis paieškos sistema, reiktų gauti kuo tikslesnius duomenis, kadangi taip būtų sutaupytas laikas ir sumažintos darbo sąnaudos. Tokiu atveju reikia atsižvelgti ir į tai, su kokia programine įranga dirba vartotojas. Inžinerinėje srityje, pavyzdžiui baldų gamyboje, konstruktoriai naudoja viena iš dviejų programinių įrangų: „Dassault solidworks“ arba „Autodesk inventor“. Abi programos veikia tais pačiais principais ir darbui įgyvendinti gali būti panaudotos abi. Darbui atlikti pasirinkome „Autodesk inventor“ programinė įrangą dėl jos naudojimosi patirties.

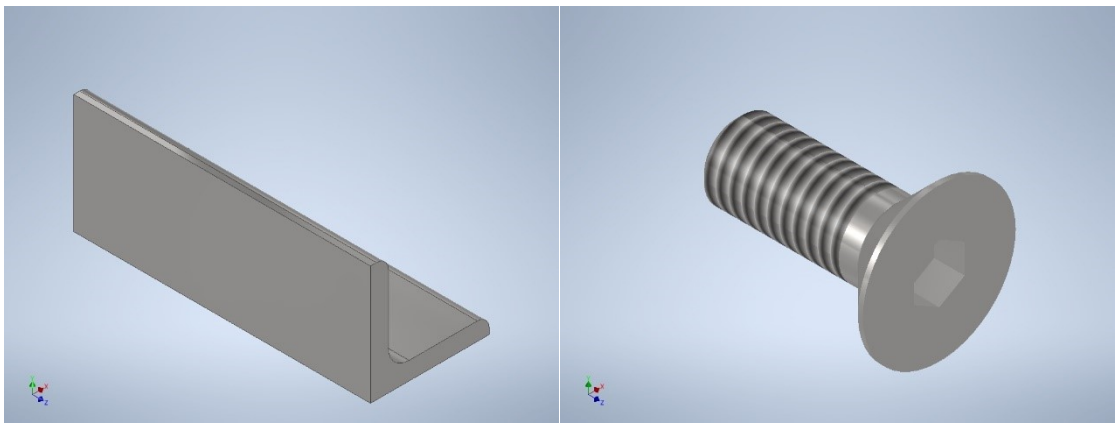
Pirminei programai sukurti naudojame „Autodesk inventor“ turimomis modelių duomenų bazėmis. Iš jų pasirenkame tam tikras modelių klases, kurios ir bandytos atpažinti programoje. Tačiau tam kad turėti pradines objektų nuotraukas reikės papildomai pasidaryti dar vieną programą, kuri iš objektų padaro skirtingų projekcijų nuotraukas, kurios po to bus paduodamos į formą ir į tolimesnius paieškos sistemos algoritmo procesus (Rosebrock, 2019). Vertinant šią programą, manome, kad tokia programa gali būti pritaikyta ir kitoms 3D objektų kūrimo programoms, kadangi į paieškos sistema paduodame objekto projekcijų nuotraukas. Todėl mūsų neriboja jokie modelių formatai ar jų programinė įranga.

3. REALIZACIJA

Šioje dalyje sistema susidarys iš skirtingų dalių ir sprendimų išnagrinėtų prieš tai. Tam kad būtų paprasčiau išnagrinėti ir pateikti realizacija, programa yra skaidoma į 3 etapus:

1. Mokymosi duomenų bazės paruošimas
2. Algoritmo mokymasis
3. Rezultatai

Mokymosi duomenų bazės pasiruošimas yra iš dalies atliktas naudojantis „Autodesk inventor“ programa, kuria sukurtos reikalingos objektams atpažinti nuotraukos. Toliau visa likusi demonstracinės sistemos dalis yra atlikta su „Google colab“ aplinka, kurios privalumai buvo pristatyti 2.2 skyriuje. Kadangi pradiniai šios sistemos vartotojai inžinieriai, buvo nuspręsta, kad kuriamoje sistemoje iš formų bus atpažintos 2 įvairaus dydžio ir formos objektai: varžtai ir profiliai. Pradžiai pasirinkti šie 2 objektai (žr. 3.1 pav.), kadangi jie yra dažnai naudojami konstruktorių darbe ir turi daug skirtingų konfigūracijų.



3.1 pav. Profilis ir varžtas

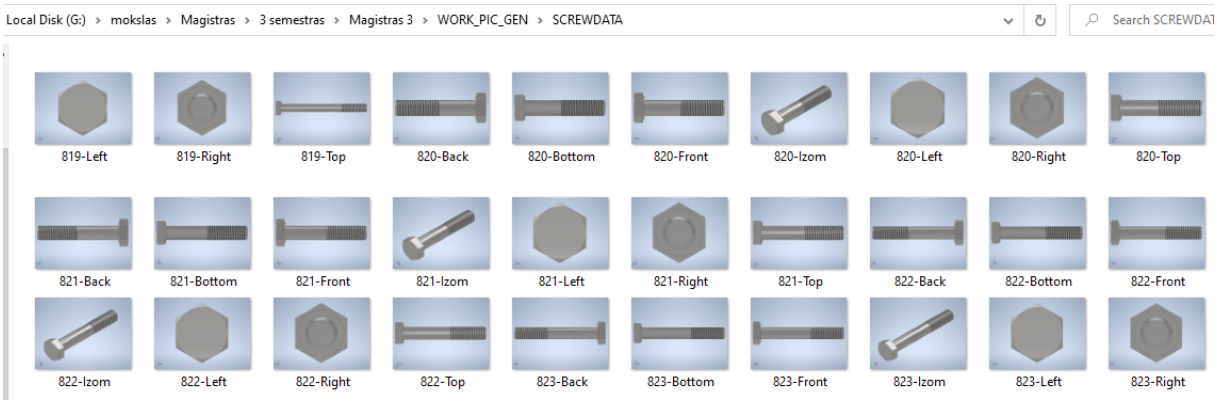
Susitvarkius nuotraukas ir pasiruošus mokymo duomenų bazę, atliekamas antrasis etapas, tai algoritmo mokymosi pagal mašininio mokymosi principus pasitelkiant sąsūkų neuroninį tinklą. Išmokius ir sukūrus modelį, įgyvendinamas trečiasis etapas, kuriame įvertinami programos realizacijos rezultatai.

3.1. Mokymosi duomenų bazės paruošimas

Pirmiausia, norint sukurti tinkamą duomenų bazę, reikia nusistatyti pradinį surinktų duomenų kiekį. Nuo duomenų kiekio ir vaizdų sudėtingumo proporcingai skiriasi ir reikalingų duomenų kiekis. Šiame darbe pasirinkta, kad kiekvienos klasės objektų užteks iki 1000 pavyzdžių per klasę. Iš šio duomenų kiekio yra ne tik kuriamas modelis, bet ir atliekami patikros ir analizės procesai.

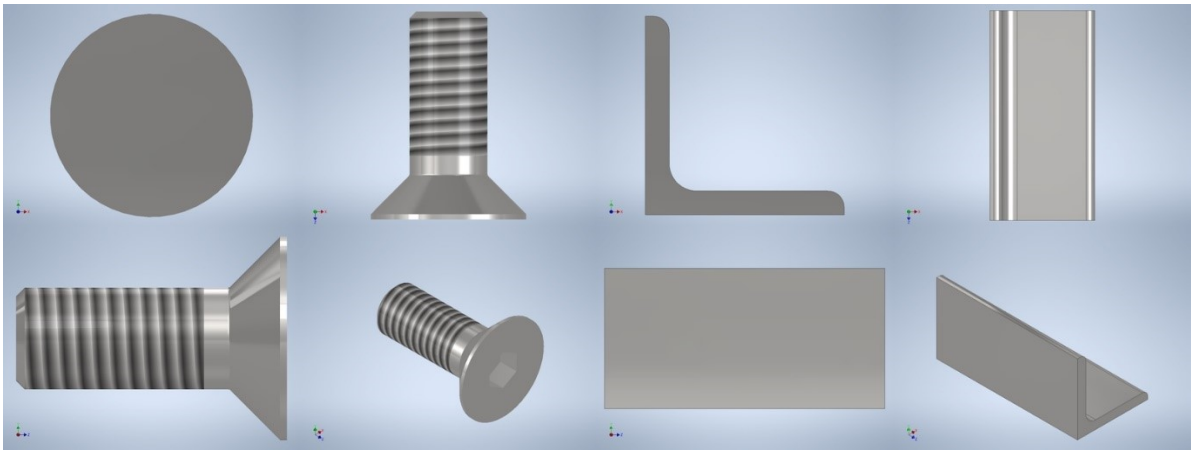
Mokymosi duomenų bazei paruošti reikia iš „Autodesk inventor“ standartinių komponentų duomenų bazės pasirinktį profilio ar varžto modelį, kažkurią jo konfigūraciją ir išsaugoti vaizdus tolimesniems procesams. Svarbu paminėti, kad nuotraukos turi nesikartoti,

kadangi tai iškraipys mokymosi procesą ir pats atpažinimo modelis gali tapti netikslus. Norint paspartinti šį nuotraukų kūrimo procesą buvo sukurtas algoritmas VBA (angl. *visual basic for applications*) kalba. Kodo eilutės yra pridėtos priede (žr. priedą nr.1). Ši kalba buvo pasirinkta, kadangi ją naudoja „Autodesk inventor“ ir su ja patogu parašyti kodą. Taigi dėl šios programos, vienu paspaudimu galima išsaugoti visas objekto projekcijas į nuotraukas.



3.2 pav. Objekto projekcijos

Pradinei demonstracinei versijai nustatėme, kad saugotų 6 projekcijas. Tačiau tolimesniuose procesuose bus naudojami tik 4 projekcijų vaizdai: izometrinis, dešinys, viršus ir priekis.



3.3 pav. Sujungtas 4 projekcijų vaizdas. Varžtas ir profilis

Svarbu pastebėti, jog nuotraukose nėra jokio užrašo, iš kurios būtų galima suprasti, kad tai yra varžtas ar koks kitas objektas. Taip yra todėl, kad atsirandant etiketėms ant duomenų, programa gali pati automatiškai išmokti, kad lengviausias būtas atpažinti objektą yra pasižiūrėti į etiketę. Tiriamuoju atveju reikia, kad modelis peržiūri nuotraukas ir jas pats įvertinų. Tačiau mokslinės literatūros analizės metu buvo išsiaiškinta, kad pateikti kiekvieną nuotrauką į sąsūkų neurono tinklų modelį yra prastas pasirinkimas dėl to, kad tai labai apkraus modelį ir sulėtins sistemą dėl didelio duomenų kiekio ir reikiamų sluoksnių modeliams apdirbti (Rosebrock,

2019). Todėl prieš pateikiant nuotraukas į sistemą, jos sujungiamos į vieną nuotrauką ir išsaugoma į katalogą. Tokia nuotrauka leis ne tik pagreitinti procesus bet ir matyti bendrą detalės vaizdą iš karto. Papildomas pliusas, kad tęsiant šiame tema darbus toliau, įmanoma išnagrinėti ir daugiau skirtingų vaizdų kombinacijų. Taip pat į vaizdus galima prijungti net pjūvius, nes dažnai gali būti, kad detalė panašiu formų iš išorės, bet vidus detalės visiškai kitoks. O kadangi sąsūkų neuroninių tinklų modeliui geriausia, kad informacija nepasikartotų, tai yra galimybė padaryti kuo daugiau skirtingu nepasikartojančių objektų nuotraukų.

3.2. Algoritmo mokymasis

Paruošus visas nuotraukas atliekame antrąjį ir vieną iš svarbiausių etapų – tai yra mokymosi modelio sukūrimas ir treniravimas. Svarbu priminti, kad prieš pradedant kurtį tokį modelį reikia pasitikrinti ir įsivertinti ar tikrai surinkti duomenys yra nepasikartojantys ir geri ir ar tokie duomenys tinkamai ištreniruos modelį. Iš patikrintų duomenų toliau kuriame duomenų rinkinius. Tam kad juos sukurti naudoajme „TensorFlow“ paketus. Tai yra atviro kodo platforma skirta kompiuterio mokymuisi. Ši aplinka turi sukaupusi didelį kiekį įvairių įrankių, bibliotekų ir paketų leidžiančių pagreitinti ir palengvinti mokymosi procesus. Viena iš tokių yra „image_dataset_from_directory“ paslaugų programa. Programa iš aplanko paima nuotraukas ir sukuria duomenų rinkinį (TensorFlow, 2021). Nuotraukos turi būti sudėtos tokioje struktūroje kaip pav.3.4.

```
main_directory/
...class_a/
......a_image_1.jpg
......a_image_2.jpg
...class_b/
......b_image_1.jpg
......b_image_2.jpg
```

3.4 pav. Duomenų struktūra reikalinga programai veikti

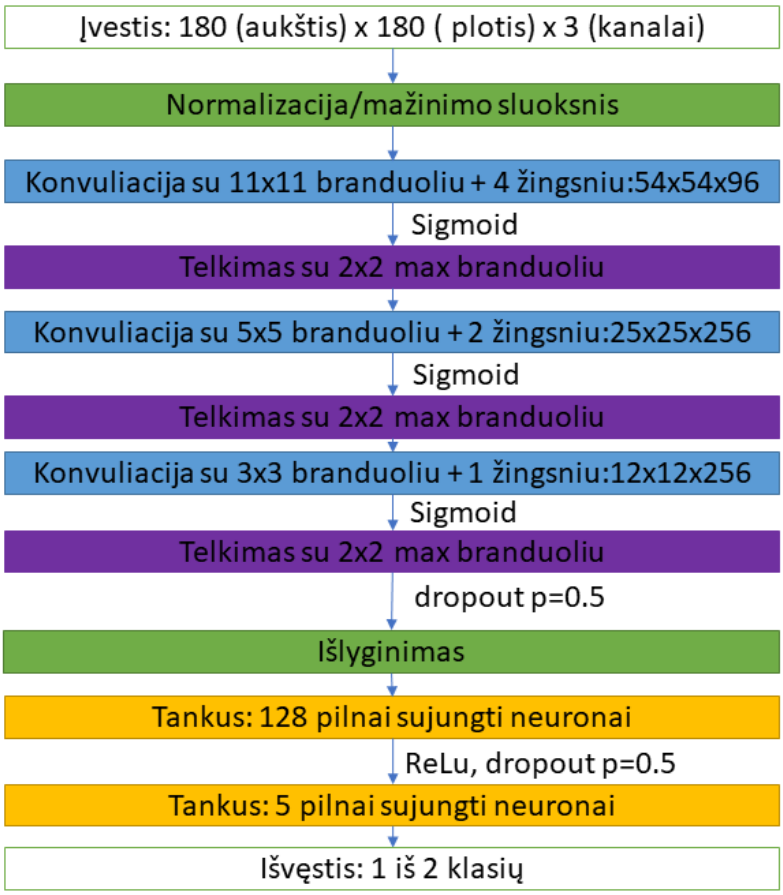
Pagal tokia struktūrą sudėti ir mūsų 4 projekcijų vaizdai. Nežymėtos nuotraukos sudėtos į varžtų ir profilių klases. Iš šių klasių padarėme 2 duomenų rinkinius: vieną iš kurio algoritmas gali mokytis ir kitą iš kurio tikrinasi kiek gerai išmoko. Su šia pagalbine programa klasės rekonstruotos vienetinio kodavimo (angl. *one-hot encoding*) formatu, kuris yra dvejetainis (angl. *binary*) vektorius, kurio vienas iš elementų yra 1, o visi kiti yra 0 (Li, Si, Xu, & Jiang, 2018). Toks kodavimas yra reikalingas, nes tinklas gali pasakyti nuotrauka yra varžtas ar profilis, tačiau kadangi tinklas naudoja tik skaičius, jis negali atiduoti rezultato žodžiais, todėl ir turime priskirti vertes žodžiams (Karbhari, 2020). Toliau pagal mašininio mokymosi principus šias nuotraukas skaidome į 80 % skirtų mokymuisi ir 20 % skirtų pasitikrinimui.

Pagalbinė programa šias nuotraukas sumaišo, kad jos kuo mažiau kartotųsi skirtinguose rinkiniuose. Nuotraukų dydis pasirenkama 180x180 pikselių ir nuotraukų partijos dydis 15. Realizacijoje galime pamatyti kiek iš viso turime duomenų per 2 klases ir kiek duomenų yra skirta treniravimuisi ir patikrai. Taip pat galime pasitikrinti ir kokias klases pagalbine programa pasirinko.

3.5 pav. Pradinės nuotraukų etiketės

Tai leidžia vartotojui ir programos kūrėjui pasitikrinti ar pradinė programa veikia teisingai ir nėra kažkokių tai klaidų. Tai yra reikalinga patikra, kadangi mokymosi algoritmas gali mokytis tik iš tų duomenų, kuriuos jam pateikia ir jeigu jie kartosis arba bus pateikti neteisingai, tai bus neteisingai ir blogai išmokyta programa, kuri negalės atlikti norimų funkcijų. Toliau turimus duomenų rinkinius pakoreguojame, kad jie efektyviau dirbtų. Visus treniravimosi ir patikros duomenis sukeliame į spartinančią atmintį (angl. *cache*) ir atliekame išankstinius duomenų išrinkimus, kurie leidžia modeliui greičiau pasirinkti duomenis iš

atminties ir tuo pačiu pradeda ruošti duomenis kitam žingsniui atlikti (TensorFlow, 2021). Tačiau sudėtos nuotraukos duomenų rinkikliuose yra spalvotos arba kitaip gali atitikti RGB spalvų modelį, kurių vertės yra srityje [0, 255]. Toks platus verčių spektras nėra tinkamas neuroniniams tinklams ir jos turėtų būti normalizuotos taip, kad būtų srityje [0, 1]. Tam atlikti buvo panaudotas mažinantis sluoksnis, po kurio pikselių vertės tampa tarp 0 ir 1. Tai pagal T.Carneiro buvo pirmas etapas objektams atpažinti neuroniniais tinklais (Carneiro et al., 2018). Antrame etape yra pastatomas, sukompiliuojamas ir ištreniruojamas konvuliacinio neuroninio tinklo modelis. Modelis susideda iš įvairių sluoksnių, kurie turi skirtingas funkcijas ir paskirtis. Paveikslėlyje žemiau galime pamatyti grafiškai modelio sluoksnius ir jų eiliškumą.



3.6 pav. Modelio sluoksniai ir eiliškumas

Tinklas susideda iš pirminio mažinimo sluoksnio, minėto anksčiau šiame darbe. Toliau 3 sluoksnius konvuliacijų, po kiekvienos iš jų atliekant „MaxPooling“ sluoksnį. Jame, naudojantis 2x2 judančia matrica gauname aukščiausias vertes iš matricos, kurios paduodamos toliau apdoroti. Konvuliaciniuose sluoksniuose panaudoti 16, 32 ir 64 filtrai su „Sigmoid“ aktyvavimo funkcija. Pagal ją ir gauname vertę tarp 0 ir 1 bei tikimybė, kiek detalė panaši paduodamą pavyzdį. Toliau turime atmetimo (angl. *dropout*) sluoksnį, kuris turi 0,2 atmetimo

tikimybę. Tai yra 20% duomenų vertės, pagal kurias bus mokomasi bus paverstos į 0 ir išimtos iš mokymosi proceso, tam kad modelis neišmoktų neteisingų verčių ir nedarytų spėjimų pagal perteklinius duomenis, kurie užspaudžia esminius faktorius. Susimažinus duomenų kiekį, jį paduodame toliau, kur lyginimo sluoksnyje bus gauta perkeisti duomenys į 1 dimensiją. Pačiam gale „tankiajame“ (angl. *dense*) sluoksnyje yra 128 neuronai, kurie naudos „ReLU“ (angl. *Rectified Linear Unit*) aktyvavimo funkciją. Turint sluoksnius ir visą informaciją galima sukompiliuoti mokymosi modelį. Mūsų darbe naudojamas optimizatorius yra „adam“ ir naudojama nuostolių „SparseCategoricalCrossentropy“ funkcija. Turint sukurtą modelį galime pateikti apžvalga apie sluoksnius.

```
1  model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rescaling_1 (Rescaling)      (None, 180, 180, 3)       0
_________________________________________________________________
conv2d (Conv2D)              (None, 180, 180, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 90, 90, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 90, 90, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 45, 45, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 45, 45, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 22, 22, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 22, 22, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 30976)             0
_________________________________________________________________
dense (Dense)                (None, 128)               3965056
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 645
=================================================================
Total params: 3,989,285
Trainable params: 3,989,285
Non-trainable params: 0
_________________________________________________________________
```

3.7 pav. Sluoksnių apžvalga

Turint visą informacija, einame prie paskutinio mokymosi etapo, tai modelio treniravimas. Tam, kad modelis išmoktų atpažinti objektus, reikia jį ištreniruoti. Su turimu duomenų kiekiu, mažų objektų klasių kiekiu ir paprastais objektais mums pakanka treniruoti modelį 5 epochas po 15 nuotraukų per partiją. Tokiu atveju mes turime dar gera modelio tikslumą, tačiau mokymasis nebūna perteklinis ir duomenys nėra iškraipomi.

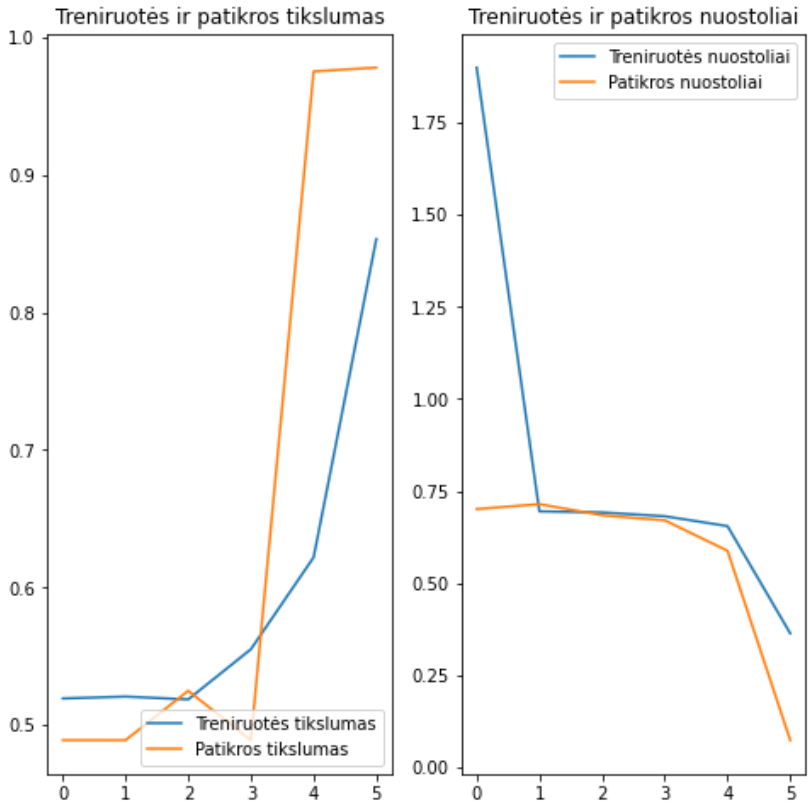
```
Epoch 1/7
97/97 [==============================] - 178s 2s/step - loss: 3.1341 - accuracy: 0.4661 - val_loss: 0.6931 - val_accuracy: 0.5110
Epoch 2/7
97/97 [==============================] - 1s 13ms/step - loss: 0.6960 - accuracy: 0.4987 - val_loss: 0.6913 - val_accuracy: 0.5110
Epoch 3/7
97/97 [==============================] - 1s 12ms/step - loss: 0.6944 - accuracy: 0.5069 - val_loss: 0.6879 - val_accuracy: 0.9199
Epoch 4/7
97/97 [==============================] - 1s 12ms/step - loss: 0.6919 - accuracy: 0.5149 - val_loss: 0.6795 - val_accuracy: 0.4890
Epoch 5/7
97/97 [==============================] - 1s 12ms/step - loss: 0.6670 - accuracy: 0.5790 - val_loss: 0.7114 - val_accuracy: 0.4890
Epoch 6/7
97/97 [==============================] - 1s 12ms/step - loss: 0.4626 - accuracy: 0.7640 - val_loss: 0.0084 - val_accuracy: 1.0000
Epoch 7/7
97/97 [==============================] - 1s 12ms/step - loss: 0.0224 - accuracy: 0.9965 - val_loss: 0.0013 - val_accuracy: 1.0000
```

3.8 pav. Epochų duomenys

Algoritmas išmokytas ir galima įsivertinti algoritmo tikslumus ir išbandyti jo objektų atpažinimą ir kitus kriterijus.

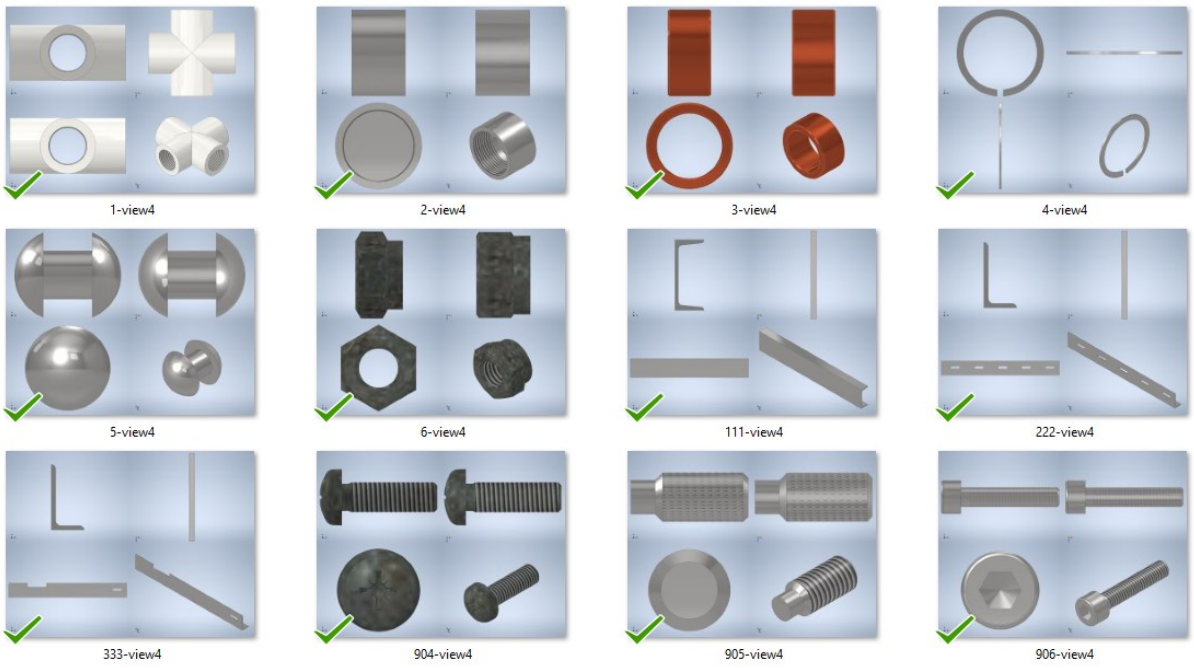
3.3. Rezultatai

Išmokyto algoritmo veikimo efektyvumas yra nustatomas pagal tikslumą ir nuostolius. Kuo didesnis mokymosi ir patikros tikslumas, tuo galime sakyti, kad algoritmas geriau atliks savo funkciją. Pagal mūsų darbe ištreniruotą modelį matome, kad nuostolių rodiklis yra labai žemas. Tai reiškia, kad patikros metu paduoti duomenys po visų mokymosi epochų yra arti ir parodo, kad modelio spėjimai yra arti nuo tikrųjų verčių ir po kiekvienos epochos turėtų mažėti. Tikslumas yra nustatomas po kiekvienos mokymosi epochos pabaigos, kuris parodo, kiek iš pateiktų bandymo duomenų modelis teisingai nuspėja objekto klasę. Paveikslėlyje 25 matome mūsų mokymosi ir treniravimosi grafikus.



3.9 pav. Algoritmo treniruotės ir patikros mokymosi grafikas

Pagal jį galime pasakyti, kad algoritmas ištreniruotas gerai pagal visus kriterijus ir juo galime naudotis tolimesniems žingsniams. Norint pilnai ištikrinti sistemą reikia vis dėl to išbandyti ją su naujais algoritmui nematytais objektais. Tam sukūrėme naujus objektus, kurie pateikti algoritmui nustatyti kokia objekto klasės tikimybė. Algoritmui išbandyti turime 3 sukurtų objektų kategorijas: varžtai, profiliai ir kiti objektai. Paveikslėlyje 3.11 galime pamatyti visus objektus ir jų 4 projekcijų vaizdus.



3.10 pav. Testavimams sukurti objektai

Visi naujai sukurti objektai skiriasi nuo esančių duomenų bazėje ir yra visiškai nauji. Tarp objektų galime pamatyti netik varžtus ar profilius bet ir kitus objektus, kaip vamzdis, kniedė, veržlė ir kita. Bandymo tikslais paleidome visus objektus į algoritmo sistemą ir gavome rezultatus pateiktus lentelėje.

3.1 lentelė. Atpažinimo rezultatų apibendrinimo lentelė varžto objektams

Pavadinimas	906	905	904
Objektas	Varžtas		
Algoritmo spėjama klasė	Varžtas		
Algoritmo spėjama tikimybė	99,86	100	100

3.2 lentelė. Atpažinimo rezultatų apibendrinimo lentelė profilio objektams

Pavadinimas	333	222	111
Objektas	Profilis		
Algoritmo spėjama klasė	Profilis		
Algoritmo spėjama tikimybė	99,86	99,86	99,86

3.3 lentelė. Atpažinimo rezultatų apibendrinimo lentelė kitiems objektams

Pavadinimas	1	2	3	4	5	6
Objektas	Kitas objektas					
Algoritmo spėjama klasė	Profilis	Varžtas	Varžtas	Profilis	Varžtas	Varžtas
Algoritmo spėjama tikimybė	88,01	94,08	99,97	99,56	99,93	100

Pagal lentelę (žr. 3.1 ir 3.2 lentelę) galime pamatyti, kad varžtus ir profilius programa labai gerai atpažįsta. Tačiau jei pasitaikys kiti objektai, kurie yra labai panašus į jau turimus (žr. 3.3 lentelę), kaip pavyzdžiui veržlė ar kamštelis, tai programa įvertina juos taip pat kaip varžtus. Arba kryžminė vamzdžių jungtis laikoma profiliu pagal programą. Taip atsitinka dėl to, kad mokantis programai buvo pateiktos tik 2 kategorijos ir pagal dvi kategorijas programa iš turimų domenų pasirinko, kas galėtų būti arčiausiai galimos klasės. Galima teigti, kad programa veikia gerai, tačiau tokie rezultatai neleidžia daryti patikimų išvadų, kadangi algoritmas neskirsto objektų, jei atsiras nauji objektai, kurie nebuvo nagrinėti. Tačiau to galima išvengti, jei bus sukurta daugiau klasių ir įtraukta papildoma kitų arba nesuprantamų objektų klasė, kuri liestų dar labiau atrūšiuoti gaminius. Taip pat ateityje yra matoma galimybė, kad priklausomai nuo poreikio pakeisti patiekiamų projekcijų kiekį, nes dabar nagrinėtiems objektams 4 vaizdai yra per daug, kadangi pasirinkti objektai nebuvo tokie sudėtingi, kad reikėtų 4 ar daugiau vaizdų projekcijų.

IŠVADOS IR PASIŪLYMAI

Išanalizavus literatūros šaltinius, sukūrus sistemos projektą, įsivertinus 2D vaizdus, naudojamas aplinkas ir pristačius panašios formos detalių paieškos sistemos pradinį demonstracinį algoritmą, pateikiamos magistro darbo teorinės ir praktinės darbo išvados bei pasiūlymai.

Išvados:

1. Teorinė mokslinių šaltinių analizė atskleidė, kad pagrindinė panašios formos detalių paieškos sistemos dalis yra objektų atpažinimas ir analizė. O objektų atpažinimas yra dažnas mokslinių tyrimų objektas. Svarbu pastebėti, kad šioje srityje yra atlikta daugybė įvairiausių tyrimų ir bandymų, kurie jau naudojami įvairiose srityse – nuo automobilio numerių atpažinimo iki balso į tekstą rašymo ir panašiai. Taip pat stipriai yra nagrinėjami mašininio mokymosi principai bei sąsūkų neuroniniai tinklai, kurie stipriai pritaikomi ir objektų atpažinimo srityje. Priklausomai nuo funkcijos ir iškeltų tikslų kiekvienas autorius skirtingai taiko mašininio mokymosi principus ir duomenų apdirbimą. Tarp atrastų pavyzdžių buvo galima rasti objektų atpažinimą iš nuotraukų, objekto projekcijų, 3D modelių, taškų debesų ar mažiausių trimatės erdvės elementų. Tačiau objektų nagrinėjimas 3D modeliais ar mažiausiai trimatės erdvės elementais yra dar tyrinėjimų stadijos ir nėra daug viešai paskelbtos medžiagos, kurią būtų galima pritaikyti ir nagrinėti. Tačiau 2D nuotraukų panaudojimas objektų atpažinimo srityje yra stipriai išvystyta sritis, todėl galima rasti daugybę įvairių sričių pavyzdžių, kuriuos galima pritaikyti ir šiame darbe tiriamoje paieškos sistemoje.
2. Sistemos projekto dalyje UML (angl. *Unified Modeling Language*) diagramomis pateiktos paieškos sistemos realizacijos idėjos. Jos leidžia aiškiai parodyti vartotoją ir jo sąveiką su sistema bei nurodyti svarbias sistemos dalis. Be to, matomos esminės demonstracinės programos dalys.
3. Atlikus literatūros analizę buvo pastebėta ir aplinkos naudojamos sistemai sukurti ar atlikti svarbumas, kadangi neuroniniams tinklams naudotis reikalingi dideli kompiuterio ištekliai, kurie auga dar ir priklausomai nuo nagrinėjamų duomenų kiekio. Norint išspręsti šią problemą, buvo pasirinkta išnagrinėti „Google Colaboratory“ aplinka, kuri leistų sistemos projektą supaprastinti ir taip pat naudotis „Google“ kompiuterių ištekliais, kurie yra daug aukštesnių standartų ir pajėgumų. Taip pat aplinka iš karto yra pritaikyta mašininio mokymosi principams. Vienas iš papildomų pliusų naudotis šia aplinka yra jos sąsaja su „Google drive“ duomenų saugykla, į kurią lengvai galima sudėti visus didelio kiekio duomenis. Tai dažnai užima daug vietos, kadangi

norint išmokyti neuroninius tinklus atpažinti objektus reikia kuo didesnį unikalių duomenų kiekį.

4. Kadangi mūsų duomenys yra objektų nuotraukos, sistemos projekte buvo įsivertinta ir pasirinkta, kad „Autodesk inventor“ programinė įranga yra patogiausia surinkti objektų nuotraukoms ir užpildyti duomenų bazę demonstracinei paieškos sistemai dėl aukšto kiekio įvairių objektų bibliotekos. Be to, buvo pasirinkta dėl to, kad tai yra viena iš dažniausiai naudojamų programų kurti 3D modeliams. Vertinant programą ir 2D atvaizdų panaudojimą buvo atrasta, kad iš tikro sistemos nevaržo programinė įranga, kadangi svarbiausia gauti objekto skirtingų projekcijų vaizdus, kad sistema veiktų. Tai leidžia būti lankstesniems ir atveria galimybę tokia sistema naudotis didesniam vartotojų skaičiui.
5. Išnagrinėjus pavyzdžius buvo pasirinkta sukurti ir išbandyti paieškos sistemą, kuri naudotų sąsūkų neuroninius tinklus, tačiau skirtingai nuo kitų tyrimų į duomenų bazę pateikti objekto vaizdų projekcijas sujungtas į vieną nuotrauką vietoj to, kad kiekviena skirtinga projekcija būtų sudėta į atskirus neuroninio tinklo sluoksnius. Toks pasirinkimas leidžia sutrumpinti algoritmo analizės laiką, sumažina išteklių apkrovą ir sumažina reikalingo kodo kiekį.
6. Paskutiniame etape programos realizacijoje buvo parašyta programa „Google colab“ aplinkoje, kuri apdoroja prieš tai padarytas nuotraukas ir jas paverčia į 4 projekcijų vaizdus, kuriuos sukelia į duomenų bazę. Toliau parenkami reikalingi nustatymai ir sistema ištreniruojama atpažinti 2 klasių objektus: profilį arba varžtą. Įsivertinus tokią atpažinimo sistemą iš grafikų atrodo, kad programa gerai išmoko ir gali atrinkti pagal objektų nuotraukas varžtus ir profilius. Norint išsamiai patikrinti sistemą, buvo sukurti papildomi niekur kitur nematyti objektai. Šie objektai taip pat nepriklauso nei vienai iš 2 klasių. Tačiau paleidus per algoritmą buvo pastebėta, kad kadangi tėra 2 klasės, programa ir bando pritaikyti objektus kuo arčiau 2 klasių. Neatsižvelgiant į tai, galima teigti, kad programa atlieka savo funkcijas tinkamai. Tačiau norint ją naudotis reiktų atlikti papildomų tyrimų.

Pasiūlymai ir pastebėjimai:

1. Dabartinė sukurta sistema yra tik demonstracinė. Norint įsivertinti visą jos perspektyvą, reiktų padidinti duomenų kiekį ir pridėti daugiau klasių ir objektų. Ypač reikėtų išbandyti sistemą sudedant objektus, kuriuos sukūrė pats vartotojas, kadangi tokie objektai yra sudėtingesni atpažinti ir taip pat užima daugiau vietos ir juos išrūšiuoti vartotojui užtrunka ilgiau.

2. Rekomenduojama būtų išnagrinėti ir įsivertinti optimaliausią ir naudingiausią projekcijų vaizdų skaičių objektui nusakyti. Dabartinėje sistemoje 4 projekcijų vaizdai tokiems paprastiems objektams yra pakankamai ir per daug. Kadangi sistema labai lengvai ir per mažą epochų skaičių išmoko objektų skirtumus. Tačiau esant sudėtingesniems objektams didesnis projekcijų skaičius bei papildomi pjūvių vaizdai leistų lengviau ir detaliau išanalizuoti kiekvieną objektą.
3. Taip pat norint, kad sąsūkų neuroniniai tinklai dar geriau išsitreniruotų, reiktų surinkti dar didesnę unikalių nuotraukų duomenų bazę. Tolesniems darbams būtų naudinga įtraukti ne tik dabar naudotas „Autodesk inventor“ programinės įrangos nuotraukas, bet ir kitų 3 dimensijų objektų kūrimo programų detalių nuotraukas.
4. Duomenims surinkti užtrunka daug laiko, ypač kai yra renkamos nuotraukos iš objekto ir dar skirtingų vaizdų. Tai užtrunka nemažą tyrimo dalį, norint susirinkti pakankamą duomenų kiekį, kad sąsūkų neuroniniai tinklai galėtų pradėti efektyviai veikti. Tam rekomenduotume susikurti papildoma algoritmą, kuris leistų iš skirtingų programų be mažai papildomų korekcijų išsaugoti objektų projekcijas į nuotraukas. Manome, kad dabartinis sukurtas algoritmas išsaugoti nuotraukas iš „Autodesk inventor“ turi perspektyvų, kadangi naudoja VBA programine kalba, kuri gali būti naudojama ir kitose programose, kaip „Dassault solidworks“.

LITERATŪROS SĄRAŠAS

1. Bai, S., Bai, X., Zhou, Z., Zhang, Z., & Latecki, L. J. (2016). GIFT: A real-time and scalable 3D shape search engine. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 5023–5032. https://doi.org/10.1109/CVPR.2016.543
2. Ben-Shabat, Y., Lindenbaum, M., & Fischer, A. (2018). 3DmFV: Three-dimensional point cloud classification in real-time using convolutional neural networks. *IEEE Robotics and Automation Letters*, *3*(4), 3145–3152. https://doi.org/10.1109/LRA.2018.2850061
3. Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828. https://doi.org/10.1109/TPAMI.2013.50
4. Bian, Q., Hu, Z., He, Y., & Cai, H. (2010). Bling: A new sketch based 3D model search engine. *Proceedings - 3rd International Conference on Information Sciences and Interaction Sciences, ICIS 2010*, 142–147. https://doi.org/10.1109/ICICIS.2010.5534731
5. Biegelbauer, G., Vincze, M., & Wohlkinger, W. (2010). Model-based 3D object detection: Efficient approach using superquadrics. *Machine Vision and Applications*, *21*(4), 497–516. https://doi.org/10.1007/s00138-008-0178-3
6. Booch, G., J. Rumbaugh, I. J. (2000). *Unified Modeling Language User Guide*. *ResearchGate* (2nd ed., Vol. 2nd). Addison-Wesley Professional; 2nd edition (May 19, 2005).
7. Brock, A., Lim, T., Ritchie, J. M., & Weston, N. (2016). Generative and Discriminative Voxel Modeling with Convolutional Neural Networks. Retrieved from http://arxiv.org/abs/1608.04236
8. Bustos, B., Keim, D. A., Saupe, D., Schreck, T., & Vranić, D. V. (2005). Feature-based similarity search in 3D object databases. *ACM Computing Surveys*, *37*(4), 345–387. https://doi.org/10.1145/1118890.1118893
9. Carneiro, T., Da Nobrega, R. V. M., Nepomuceno, T., Bian, G. Bin, De Albuquerque, V. H. C., & Filho, P. P. R. (2018). Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. *IEEE Access*, *6*, 61677–61685. https://doi.org/10.1109/ACCESS.2018.2874767
10. Che, E., Jung, J., & Olsen, M. J. (2019). Object recognition, segmentation, and classification of mobile laser scanning point clouds: A state of the art review. *Sensors (Switzerland)*, *19*(4). https://doi.org/10.3390/s19040810

11. Chen, D. Y., Tian, X. P., Shen, Y. Te, & Ouhyoung, M. (2003). On Visual Similarity Based 3D Model Retrieval. *Computer Graphics Forum*, *22*(3), 223–232. https://doi.org/10.1111/1467-8659.00669
12. Cheng, H. C., Lo, C. H., Chu, C. H., & Kim, Y. S. (2011). Shape similarity measurement for 3D mechanical part using D2 shape distribution and negative feature decomposition. *Computers in Industry*, *62*(3), 269–280. https://doi.org/10.1016/j.compind.2010.09.001
13. Chu, C. H., & Hsu, Y. C. (2005). An Integrated Approach for 3D Part Search with Multiple Shape Signatures. *Computer-Aided Design and Applications*, *2*(1–4), 183–192. https://doi.org/10.1080/16864360.2005.10738366
14. Chu, C. H., Lo, C. H., & Cheng, H. C. (2017). Cognitive shape similarity assessment for 3D part search. *Journal of Intelligent Manufacturing*, *28*(7), 1679–1694. https://doi.org/10.1007/s10845-016-1211-4
15. Dayan, P. (2008). Unsupervised Learning. *Encyclopedia of Neuroscience*, 1–7. https://doi.org/10.1007/978-3-540-29678-2_6202
16. Google. (n.d.). Colaboratory: Frequently Asked Questions. Retrieved from https://research.google.com/colaboratory/faq.html
17. Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep learning for visual understanding: A review. *Neurocomputing*, *187*, 27–48. https://doi.org/10.1016/j.neucom.2015.09.116
18. Holzinger, A. (2017). Introduction to Machine Learning & Knowledge Extraction (MAKE). *Machine Learning and Knowledge Extraction*, *1*(1), 1–20. https://doi.org/10.3390/make1010001
19. Hong, J., Kim, K., & Lee, H. (2020). Faster Dynamic Graph CNN: Faster Deep Learning on 3D Point Cloud Data. *IEEE Access*, *8*, 190529–190538. https://doi.org/10.1109/access.2020.3023423
20. Indolia, S., Goswami, A. K., Mishra, S. P., & Asopa, P. (2018). Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach. *Procedia Computer Science*, *132*, 679–688. https://doi.org/10.1016/j.procs.2018.05.069
21. Ip, C. Y., & Gupta, S. K. (2007). Retrieving Matching CAD Models by Using Partial 3D Point Clouds. *Computer-Aided Design and Applications*, *4*(5), 629–638. https://doi.org/10.1080/16864360.2007.10738497
22. Iyer, N., Jayanti, S., Lou, K., Kalyanaraman, Y., & Ramani, K. (2005). Three-dimensional shape searching: State-of-the-art review and future trends. *CAD Computer Aided Design*, *37*(5 SPEC.ISS.), 509–530. https://doi.org/10.1016/j.cad.2004.07.002
23. Jia, J. Y., Zhang, Q., Zeng, L., & Liang, S. (2014). Voxel-encoded descriptor for 3D

model retrieval by exploring model's spatial information. *Journal of Mechanical Science and Technology*, *28*(7), 2459–2467. https://doi.org/10.1007/s12206-014-0603-7

24. Jiantao, P., & Ramani, K. (2007). An Integrated 2D and 3D Shape-based Search Framework and Applications. *Computer-Aided Design and Applications*, *4*(6), 817–826. https://doi.org/10.1080/16864360.2007.10738514
25. Karbhari, V. (2020). Why do we need one-hot encoding? Retrieved April 5, 2021, from https://medium.com/acing-ai/why-do-we-need-one-hot-encoding-7bcb456d49df
26. Kavzoglu, T. (2009). Increasing the accuracy of neural network classification using refined training data. *Environmental Modelling and Software*, *24*(7), 850–858. https://doi.org/10.1016/j.envsoft.2008.11.012
27. Learned-miller, E. G. (2014). Introduction to Supervised Learning, 1–5. https://doi.org/10.1142/9789814287319_0007
28. Li, J., Si, Y., Xu, T., & Jiang, S. (2018). Deep Convolutional Neural Network Based ECG Classification System Using Information Fusion and One-Hot Encoding Techniques. *Mathematical Problems in Engineering*, *2018*. https://doi.org/10.1155/2018/7354081
29. Lian, Z., Godil, A., & Sun, X. (2010). Visual Similarity based 3D Shape Retrieval Using Bag-of-Features. *IEEE INTERNATIONAL CONFERENCE ON SHAPE MODELING AND APPLICATIONS (SMI)*.
30. Lin, X., Zhu, C., & Liu, Y. (2016). Mesh Interest Point Detection Based on Geometric Measures and Sparse Refinement, 1–17. Retrieved from http://arxiv.org/abs/1604.08806
31. Lison, P. (2012). An Introduction to Machine Learning (p. 35).
32. Liu, Z., Tang, H., Lin, Y., & Han, S. (2019). Point-voxel CNN for efficient 3d deep learning. *ArXiv*, (NeurIPS).
33. Ma, S. H., & Tian, L. (2014). Hierarchical 3D mechanical parts matching based-on adjustable geometry and topology similarity measurements. *Journal of Central South University*, *21*(1), 89–99. https://doi.org/10.1007/s11771-014-1920-9
34. Maturana, D., & Scherer, S. (2015). VoxNet: A 3D Convolutional Neural Network for real-time object recognition. *IEEE International Conference on Intelligent Robots and Systems*, *2015-Decem*, 922–928. https://doi.org/10.1109/IROS.2015.7353481
35. Nusrat, I., & Jang, S. B. (2018). A comparison of regularization techniques in deep neural networks. *Symmetry*, *10*(11). https://doi.org/10.3390/sym10110648
36. Ohbuchi, R., Minamitani, T., & Takei, T. (2005). Shape-similarity search of 3D models

by using enhanced shape functions. *International Journal of Computer Applications in Technology*, *23*(2–4), 70–85. https://doi.org/10.1504/IJCAT.2005.006466

37. Pang, G., & Neumann, U. (2016). 3D point cloud object detection with multi-view convolutional neural network. *Proceedings - International Conference on Pattern Recognition*, *0*, 585–590. https://doi.org/10.1109/ICPR.2016.7899697
38. Papadakis, P., Pratikakis, I., Perantonis, S., & Theoharis, T. (2007). Efficient 3D shape matching and retrieval using a concrete radialized spherical projection representation. *Pattern Recognition*, *40*(9), 2437–2452. https://doi.org/10.1016/j.patcog.2006.12.026
39. Papadakis, P., Pratikakis, I., Theoharis, T., & Perantonis, S. (2010). Panorama: A 3D shape descriptor based on panoramic views for unsupervised 3d object retrieval. *International Journal of Computer Vision*, *89*(2–3), 177–192. https://doi.org/10.1007/s11263-009-0281-6
40. Poux, F., Hallot, P., Neuville, R., & Billen, R. (2016). SMART POINT CLOUD: DEFINITION and REMAINING CHALLENGES. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *4*(2W1), 119–127. https://doi.org/10.5194/isprs-annals-IV-2-W1-119-2016
41. Qin, F. wei, Gao, S. ming, Yang, X. ling, Bai, J., & Zhao, Q. hong. (2017). A sketch-based semantic retrieval approach for 3D CAD models. *Applied Mathematics*, *32*(1), 27–52. https://doi.org/10.1007/s11766-017-3450-3
42. Rosebrock, A. (2019). Keras, Regression, and CNNs. Retrieved from https://www.pyimagesearch.com/2019/01/28/keras-regression-and-cnns/
43. Ruggeri, M. R., Vranić, D. V., & Saupe, D. (2006). Shape Similarity Search for Surfelbased Models. *Computer Vision and Graphics*, 131–140. https://doi.org/10.1007/1-4020-4179-9_20
44. Shafi, U. M. T. M. A. (2017). MORE FOR LESS : INSIGHTS INTO CONVOLUTIONAL NETS FOR 3D POINT CLOUD RECOGNITION. In *2017 IEEE International Conference on Image Processing (ICIP)* (pp. 1607–1611).
45. Sharif, M. M., Nahangi, M., Haas, C., & West, J. (2017). Automated Model-Based Finding of 3D Objects in Cluttered Construction Point Cloud Models. *Computer-Aided Civil and Infrastructure Engineering*, *32*(11), 893–908. https://doi.org/10.1111/mice.12306
46. Shen, Y., Chen, D., Tian, X., & Ouhyoung, M. (2003). 3D Model Search Engine Based on Lightfield Descriptors. *Forum American Bar Association*, 23–30.
47. Shrestha, A., & Mahmood, A. (2019). Review of deep learning algorithms and architectures. *IEEE Access*, *7*, 53040–53065.

https://doi.org/10.1109/ACCESS.2019.2912200

48. Simeone, O. (2018). *A brief introduction to machine learning for engineers*. https://doi.org/10.1561/2000000102
49. Su, H., Maji, S., Kalogerakis, E., & Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3D shape recognition. *Proceedings of the IEEE International Conference on Computer Vision*, *2015 Inter*, 945–953. https://doi.org/10.1109/ICCV.2015.114
50. Tangelder, J. W. H., & Veltkamp, R. C. (2008). A survey of content based 3D shape retrieval methods. *Multimedia Tools and Applications*, *39*(3), 441–471. https://doi.org/10.1007/s11042-007-0181-0
51. TensorFlow. (2021). TensorFlow Core v2.4.1. Retrieved April 4, 2021, from https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image_dataset_from_directory
52. Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience*, *2018*. https://doi.org/10.1155/2018/7068349
53. Wan, L., Zeiler, M., Zhang, S., LeCun, Y., & Fergus, R. (2013). Regularization of Neural Networks using DropConnect. In *Proceedings of the 30 th International Conference on Machine Learning* (Vol. 28). Atlanta, Georgia, USA,.
54. Wang, F., Kang, L., & Li, Y. (2015). Sketch-based 3D shape retrieval using Convolutional Neural Networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07-12-June*, 1875–1883. https://doi.org/10.1109/CVPR.2015.7298797
55. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. (2015). 3D ShapeNets: A deep representation for volumetric shapes. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07-12-June*, 1912–1920. https://doi.org/10.1109/CVPR.2015.7298801
56. Xie, J., Fang, Y., Zhu, F., & Wong, E. (2015). Deepshape: Deep learned shape descriptor for 3D shape matching and retrieval. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07-12-June*, 1275–1283. https://doi.org/10.1109/CVPR.2015.7298732
57. Yin, Z., Kong, D., Shao, G., Ning, X., Jin, W., Jin, W., & Wang, J. (2018). A-optimal convolutional neural network, 2295–2304.
58. Yoon, G. J., & Yoon, S. M. (2017). Sketch-based 3D object recognition from locally optimized sparse features. *Neurocomputing*, *267*, 556–563.

https://doi.org/10.1016/j.neucom.2017.06.034

59. Zhang, X., Fu, C., & Zhao, Y. (2021). An improved volumetric grid deep network model for point cloud segmentation. *Systems Science and Control Engineering*, *9*(S1), 161–167. https://doi.org/10.1080/21642583.2020.1826004
60. Zhang, Y., & Rabbat, M. (2018). A Graph-CNN for 3D Point Cloud Classification. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, *2018-April*, 6279–6283. https://doi.org/10.1109/ICASSP.2018.8462291
61. Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2018). Object detection with deep learning: A review. *ArXiv*, *30*(11), 3212–3232.

A. PRIEDAI

1 Priedas. Objekto projekcijų nuotraukų kūrimo kodas

```
Public Sub CreateImage()

Dim a As Application
Set a = ThisApplication
Dim izo As View
Dim Path As String
Path = "G:\mokslas\Magistras\3 semestras\Magistras 3\WORK_PIC_GEN\DATA\"

' Atrandame paskutinį objekto numerį, kad galima būtų tęsti nuotraukų generavimo eiliškumą ir vaizdai nebūtų perrašomi.
Dim folderPath As String, tableName As String, latestTblName As String
Dim latestModified As Date, modifiedDate As Date

' Nurodome kelią pagal tai kokį objektą išsaugome
folderPath = "G:\mokslas\Magistras\3 semestras\Magistras 3\WORK_PIC_GEN\DATA\"

tableName = Dir(folderPath)

Do While tableName <> vbNullString
    modifiedDate = FileDateTime(folderPath & tableName)
    If latestModified < modifiedDate Then
        latestModified = modifiedDate
        latestTblName = tableName

    End If
    tableName = Dir()
Loop

MsgBox latestTblName

' Sukuriame skaičių sekos nuotraukų numeravimui taisykles
Dim numbergen As Integer
numbergen = Val(latestTblName)
MsgBox numbergen

Dim addon As Integer
addon = numbergen + 1
MsgBox addon

Dim FileNameGen As String
FileNameGen = addon & "-"

' Saugojame objekto skirtingų projekcijų vaizdus
' Pasirenkame aktyvią kamerą
Dim cam As Camera
Set cam = ThisApplication.ActiveView.Camera

' Parenkame izometrinį projekcijos vaizdą

cam.ViewOrientationType = kIsoTopLeftViewOrientation
cam.Fit
cam.ApplyWithoutTransition
Set izo = a.ActiveView
'Išsaugome vaizdą su automatiškai sukurtu pavadinimų ir nustatytu spalvos fonu.
Call izo.SaveAsBitmap(Path & FileNameGen & "Izom" & ".bmp", 1024, 768)

' Parenkame priekinį projekcijos vaizdą
cam.ViewOrientationType = kFrontViewOrientation
cam.Fit
cam.ApplyWithoutTransition
Set izo = a.ActiveView
' Išsaugome vaizdą su automatiškai sukurtu pavadinimų ir nustatytu spalvos fonu.
Call izo.SaveAsBitmap(Path & FileNameGen & "Front" & ".bmp", 1024, 768)

' Set up the BACK camera.
cam.ViewOrientationType = kBackViewOrientation
cam.Fit
cam.ApplyWithoutTransition
Set izo = a.ActiveView
'Išsaugome vaizdą su automatiškai sukurtu pavadinimų ir nustatytu spalvos fonu.
Call izo.SaveAsBitmap(Path & FileNameGen & "Back" & ".bmp", 1024, 768)

' Set up the RIGHT camera
cam.ViewOrientationType = kRightViewOrientation
cam.Fit
```

```
cam.ApplyWithoutTransition
Set izo = a.ActiveView
'Išsaugome vaizdą su automatiškai sukurtu pavadinimų ir nustatytu spalvos fonu.
Call izo.SaveAsBitmap(Path & FileNameGen & "Right" & ".bmp", 1024, 768)

' Set up the LEFT camera
cam.ViewOrientationType = kLeftViewOrientation
cam.Fit
cam.ApplyWithoutTransition
Set izo = a.ActiveView
' Išsaugome vaizdą su automatiškai sukurtu pavadinimų ir nustatytu spalvos fonu.
Call izo.SaveAsBitmap(Path & FileNameGen & "Left" & ".bmp", 1024, 768)

' Set up the BOTTOM camera
cam.ViewOrientationType = kBottomViewOrientation
cam.Fit
cam.ApplyWithoutTransition
Set izo = a.ActiveView
'Išsaugome vaizdą su automatiškai sukurtu pavadinimų ir nustatytu spalvos fonu.
Call izo.SaveAsBitmap(Path & FileNameGen & "Bottom" & ".bmp", 1024, 768)

' Set up the TOP camera
cam.ViewOrientationType = kTopViewOrientation
cam.Fit
 cam.ApplyWithoutTransition
 Set izo = a.ActiveView
 'Išsaugome vaizdą su automatiškai sukurtu pavadinimų ir nustatytu spalvos fonu.
 Call izo.SaveAsBitmap(Path & FileNameGen & "Top" & ".bmp", 1024, 768)

 ' Išsaugomi duomenys į "excel" failą patikrai ir istorijai.
 Dim oApp As Application
 Set oApp = ThisApplication

 Dim oDoc As Document
 Set oDoc = oApp.ActiveDocument

 ' Patikrinimas kokiame dokumente yra vartotojas, kadangi programa veikai tik detalės
 lange.
 If oDoc Is Nothing Or TypeOf oDoc Is DrawingDocument Then
 End If

 Dim oCompDef As ComponentDefinition
 ' Gaunam komponentų apibrėžimus
 Set oCompDef = oDoc.ComponentDefinition

 Dim oParams As Parameters
 Dim oParam As Parameter
 ' Gaunam objekto parametrus
 Set oParams = oCompDef.Parameters

 Dim oExcel As Object
 ' Sukuriamas excel objektas
 Set oExcel = CreateObject("Excel.Application")

 ' Sukuria excel langą, kad matyti ar viskas veikia
   oExcel.Visible = True

 ' Nustato lapo pavadinimą, kur bus keliama visa informacija
 Dim sSheetName As String
 sSheetName = "data"
 Dim sFilePath

 ' Nustatoma excel dokumento vieto
  sFilePath = "G:\mokslas\Magistras\3 semestras\Magistras
  3\WORK_PIC_GEN/data_hist.xlsx"
'sFilePath = oExcel.GetOpenFilename(FileFilter:="Excel Files (*.xls*),*.xls*",
Title:="Select Excel File to Read From")
 Dim oWkbk As Workbook
 ' Atidaromas excel failas
 Set oWkbk = oExcel.Workbooks.Open(sFilePath, False)

 Dim oSheet As WorkSheet
 ' Pasirenkamas reikiamas lapas
```

```
143   Set oSheet = oWkbk.Sheets(sSheetName)
144   oSheet.Unprotect
145
146
147   ' Nustatoma riba, kur yra duomenys lape
148   Dim sParamRange As String
149   sParamRange = "A2:E1"
150
151   oSheet.Cells(addon + 1, 1) = addon
152
153   Dim oCell As Range
154   ' Atliekamas ciklas tarp parametrų esančių lape
155   For Each oCell In oSheet.Range(sParamRange)
156       ' Pereinama tarp parametrų pavadinimų excel lape ar jie atitinka turimus objektų
          parametrų pavadinimus
157       For Each oParam In oParams
158           ' Jei pavadinimas atitinka, nukopjuojama informacija iš objekto parametrų į
              excel duomenų lapus po atitinkamu parametru
159           If oCell.Value = oParam.Name Then
160               oCell.Offset(addon, 0).Value = oParam.Expression
161           End If
162       Next oParam
163   Next oCell
164
165
166
167   '  Išsaugojimas
168     ' Išjungiame patvirtinimo langą, jeigu failas jau egzistuoja ir reikia jį perrašyti.
169     oExcel.DisplayAlerts = False
170     Call oWkbk.SaveAs("G:\mokslas\Magistras\3 semestras\Magistras
        3\WORK_PIC_GEN/data_hist.xlsx")
171
172
173   ' Išjugniamas workbook
174   Call oWkbk.Close
175   oExcel.Quit
176   Set oExcel = Nothing
177   Set oWkbk = Nothing
178   Set oSheet = Nothing
179   Set oCells = Nothing
180
181
182   ' Atnaujinamas detalės modelis
183   oDoc.Update
184
185   End Sub
186
```

2 Priedas. „Google colab“ aplinkos programos maketo kodas

Pasiruošimas

Prijungiamas "Google Drive"diskas

```
from google.colab import drive # import drive from google colab

ROOT = "/content/drive/"    # default location for the drive
print(ROOT)                 # print content of ROOT (Optional)

drive.mount(ROOT)           # we mount the google drive at /content/drive
```

Nuotraukų apdorojimas ir sujungimas.

```
# Pasitikrinti ar tikrai last file number atitinka paskutini failu numeri papk4je

from __future__ import print_function
import os

from PIL import Image
import glob
import os
import re
import sys


list_of_files = glob.glob('/content/drive/MyDrive/Magistras/DATA/SCREW/*')
latest_file = max(list_of_files, key=os.path.getctime)
print(latest_file)

last_number = re.findall('\d+', latest_file )
print(re.findall('\d+', latest_file ))
print(last_number)
endnumber = int("".join(map(str, last_number)))
print(endnumber)
```

```
# Sujungiamos nuotraukos į 1 bendrą iš skirtingų projekcijų. Varžtų nuotraukos

from __future__ import print_function
import os

from PIL import Image
import glob
import os
import re
import sys
```

```

list_of_files = glob.glob('/content/drive/MyDrive/Magistras/DATA/SCREW/*')
latest_file = max(list_of_files, key=os.path.getctime)
print(latest_file)
last_number = re.findall('\d+', latest_file )
print(re.findall('\d+', latest_file ))
print(last_number)
endnumber = int("".join(map(str, last_number)))
print(endnumber)
endnumber = 915

for b in range(1,endnumber) :
  files = [
  '/content/drive/MyDrive/Magistras/DATA/SCREW/'+str(b)+'-Front.bmp',
  '/content/drive/MyDrive/Magistras/DATA/SCREW/'+str(b)+'-Left.bmp',
  '/content/drive/MyDrive/Magistras/DATA/SCREW/'+str(b)+'-Top.bmp',
  '/content/drive/MyDrive/Magistras/DATA/SCREW/'+str(b)+'-Izom.bmp']

  result = Image.new("RGB", (800, 600))


  for index, file in enumerate(files):
    path = os.path.expanduser(file)
    img = Image.open(path)
    img.thumbnail((400, 400), Image.ANTIALIAS)
    x = index // 2 * 400
    y = index % 2 * 300
    w, h = img.size
    #print('pos {0},{1} size {2},{3}'.format(x, y, w, h))
    result.paste(img, (x, y, x + w, y + h))

  result.save(os.path.expanduser('/content/drive/MyDrive/Magistras/INPUT/SCREW/'+str(
  print(b)
```

```
# Sujungiamos nuotraukos į 1 bendrą iš skirtingų projekcijų. Profilių nuotraukos
from __future__ import print_function

import os

from PIL import Image
import glob
import os
import re
import sys


list_of_files = glob.glob('/content/drive/MyDrive/Magistras/DATA/PROFILE/*') # * mear
latest_file = max(list_of_files, key=os.path.getctime)
print(latest_file)
last_number = re.findall('\d+', latest_file )
print(re.findall('\d+', latest_file ))
print(last_number)
endnumber = int("".join(map(str, last_number)))
print(endnumber)
```

```
endnumber = 915 # nes neveikia last number normaliai, reikia pakeisti getctime

for b in range(1,endnumber) :
  files = [
  '/content/drive/MyDrive/Magistras/DATA/PROFILE/'+str(b)+'-Front.bmp',
  '/content/drive/MyDrive/Magistras/DATA/PROFILE/'+str(b)+'-Left.bmp',
  '/content/drive/MyDrive/Magistras/DATA/PROFILE/'+str(b)+'-Top.bmp',
  '/content/drive/MyDrive/Magistras/DATA/PROFILE/'+str(b)+'-Izom.bmp']

  result = Image.new("RGB", (800, 600))

  for index, file in enumerate(files):
    path = os.path.expanduser(file)
    img = Image.open(path)
    img.thumbnail((400, 400), Image.ANTIALIAS)
    x = index // 2 * 400
    y = index % 2 * 300
    w, h = img.size
    #print('pos {0},{1} size {2},{3}'.format(x, y, w, h))
    result.paste(img, (x, y, x + w, y + h))

  result.save(os.path.expanduser('/content/drive/MyDrive/Magistras/INPUT/PROFILE/'+st
  print(b)
```

Konvuliacininio neuroninio tinklo kūrimas

contains `train` and `validation` subdirectories for the training and validation datasets for a refresher on training, validation and test sets), which in turn each contain `profile` and `screw` subdirectories.

Sukeliami reikalingi duomenų paketai, bibliotekos ir programos

```
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import models
from tensorflow.keras.models import Sequential
from PIL import Image
```

```
import pathlib
data_dir = pathlib.Path('/content/drive/MyDrive/Magistras/INPUT/')
```

Patikra. Pateikiamos profilio nuotraukos

```
PROFILE = list(data_dir.glob('PROFILE/*'))
PIL.Image.open(str(PROFILE[0]))
```

```
PIL.Image.open(str(PROFILE[1]))
```

Patikra. Pateikiamos varžto nuotraukos

```
SCREW = list(data_dir.glob('SCREW/*'))
PIL.Image.open(str(SCREW[0]))
```

```
PIL.Image.open(str(SCREW[1]))
```

Sukuriama duomenų bazė

Užkrauname apdorotas nuotraukas iš disko ir sukuriu duomenų bazę tolimesniems programos veiksmams

Nustatomi parametrai, nuotraukos dydis ir partijos dydis

```
batch_size = 15 # startinis - 32
img_height = 180
img_width = 180
```

Nuotraukos išskaidomos į 2 dalis: mokymosi ir patikros. 80% nuotraukų naudojama mokymosi duomenų rinkiniui sukurti, likusi 20% nuotraukų skirtas patikros duoemenų rinkiniui sukurti.

```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="training",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

Parodomi gauti objektių klasių pavadinimai sukurtuose duomenų rinkiniuose. Jie atitinka papkių pavadinimus, kuriuose buvo nuotraukos.

```
class_names = train_ds.class_names
print(class_names)
```

Duomenų vizualizacija

Pateikiamos 9 nuotraukos iš mokymosi duomenų rinkinio.

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i]])
    plt.axis("off")
```

```
for image_batch, labels_batch in train_ds:
  print(image_batch.shape)
  print(labels_batch.shape)
  break
```

Nuotraukų rinkinys yra tenzorius su tokia forma `(15, 180, 180, 3)`. Tai yra 15 nuotraukų rinkis su 180x180 dydžios nuotraukas su RGB kanalų spalvomis.

Duomenų apdirbimas prieš mokymosi modelio kūrimą

Kad programa veiktų sklandžiau naudoju buferinį išankstinį išvkietimą, kad gaučiau duomenis iš disko neblokuodmas įvesčių ir išvesčių.

`Dataset.cache()` - išsaugo nuotraukas atmintyje, pirmą kartą įkėlus vaizdus iš disko. Taip užtikrinama, kad mokymo metu duomenų rinkinio duomenų perdavimas netaptų "butelio kakliuku".

`Dataset.prefetch()` - mokymo metu atlieka duomenų perdengimo funkciją ir perdegnia išankstinio apdorojimo ir modelio vykdymo duomenis.

```
AUTOTUNE = tf.data.AUTOTUNE

train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

Duomenų standartizavimas

RGB kanalų vertės yra `[0, 255]` diapazone. Tai nėra tinkama neuroniniams tinklams. Tinklai veikia efektyviau, jei paduodamos vertės yra mažos. Todėl turimas vertes standartizuojame ir sumažiname iki `[0, 1]` diapazono naudodamiesi Rescaling layer arba kodu esančiu apačioje.

```
normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
image_batch, labels_batch = next(iter(normalized_ds))
first_image = image_batch[0]
# Notice the pixels values are now in `[0,1]`.
print(np.min(first_image), np.max(first_image))
```

```
0.043142524 0.9975307
```

Sukuriamas neuroninio tinklo modelis

Pateikiami modelio sluoksniai.

```
num_classes = 5

model = Sequential([
  layers.experimental.preprocessing.Rescaling(1./255),
  layers.Conv2D(16, 3, padding='same', activation='sigmoid'), # 'sigmoid' 'relu' - la
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='sigmoid'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='sigmoid'),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

Modelis sukompiliuojamas

Naudojamas `optimizers.Adam` optimizatorius ir `losses.SparseCategoricalCrossentropy` nuostolių funkcija

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Modelis mokomas

```
# Užkraunamas "Tensorboard" papildymas duomenims lengviau įvertinti
%load_ext tensorboard

from datetime import datetime
from packaging import version

import tensorflow as tf
from tensorflow import keras

print("TensorFlow version: ", tf.__version__)
assert version.parse(tf.__version__).release[0] >= 2, \
    "This notebook requires TensorFlow 2.0 or above."
```

```
logdir="logs/fit/" + datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = keras.callbacks.TensorBoard(log_dir=logdir)
```

```
epochs = 6
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs,
  callbacks= [tensorboard_callback])
```

Modelio apžavalga

Parodomi visi modelio sluoksniai ir išvęstys bei parametrai

```
model.summary()
```

Paleidžiamas "TensorBoard" paketas leidžianti apžiūrėti mokymosi modelio duomenis ir patogiai įsivertinti parametrus: tikslumą bei nuostolius.

```
%tensorboard --logdir logs
```

Modelio mokymosi rezultatų grafikas

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)
```

```
plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Treniruotės tikslumas')
plt.plot(epochs_range, val_acc, label='Patikros tikslumas')
plt.legend(loc='lower right')
plt.title('Treniruotės ir patikros tikslumas')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Treniruotės nuostoliai')
plt.plot(epochs_range, val_loss, label='Patikros nuostoliai')
plt.legend(loc='upper right')
plt.title('Treniruotės ir patikros nuostoliai')
plt.show()
```

Modelio testavimas

Modeliui atpažinti ir išbandyti atiduodama naujos nuotraukos, kurios nebuvo pradžioje sukurtuose duomenų rinkiniuose.

```
#object_path = '/content/drive/MyDrive/Magistras/TEST DATA/OUTPUT/906-view4.bmp'
#object_path = '/content/drive/MyDrive/Magistras/TEST DATA/OUTPUT/905-view4.bmp'
#object_path = '/content/drive/MyDrive/Magistras/TEST DATA/OUTPUT/904-view4.bmp'
#object_path = '/content/drive/MyDrive/Magistras/TEST DATA/OUTPUT/111-view4.bmp'
object_path = '/content/drive/MyDrive/Magistras/TEST DATA/OUTPUT/222-view4.bmp'
#object_path = '/content/drive/MyDrive/Magistras/TEST DATA/OUTPUT/333-view4.bmp'
#object_path = '/content/drive/MyDrive/Magistras/TEST DATA/OUTPUT/1-view4.bmp'
#object_path = '/content/drive/MyDrive/Magistras/TEST DATA/OUTPUT/2-view4.bmp'
#object_path = '/content/drive/MyDrive/Magistras/TEST DATA/OUTPUT/3-view4.bmp'
#object_path = '/content/drive/MyDrive/Magistras/TEST DATA/OUTPUT/4-view4.bmp'
#object_path = '/content/drive/MyDrive/Magistras/TEST DATA/OUTPUT/5-view4.bmp'
#object_path = '/content/drive/MyDrive/Magistras/TEST DATA/OUTPUT/6-view4.bmp'


img = keras.preprocessing.image.load_img(
    object_path, target_size=(img_height, img_width)
)
img_array = keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

plt.figure()
plt.imshow(img)
plt.show()


print(
    "Šis objektas priklauso {} klasei. Spėjimo tikimybė yra {:.2f} procentai."
   .format(class_names[np.argmax(score)], 100 * np.max(score))
)
```

SUSITVARKYMAS

Paleisti, kad būtų atlaisvinama atmintis, tolimesniems veiksmams.

```
import os, signal
os.kill(os.getpid(), signal.SIGKILL)
```

SUTIKIMAS DĖL ASMENS DUOMENŲ NAUDOJIMO

2021-05-31
(Data)

Šiuo sutikimu aš, [Andrius Tunikaitis] (toliau – Duomenų subjektas) sutinku, kad Vilniaus Gedimino technikos universitetas, juridinio asmens kodas 111950243, adresas Saulėtekio al. 11, LT-10223 Vilnius (toliau – Duomenų valdytojas), tvarkytų mano asmens duomenis kitų studentų mokymosi tikslu. T. y. tvarkytų (*pažymėkite tinkamą*):

☐ vardą, pavardę, bakalauro baigiamąjį darbą;
☐ bakalauro baigiamąjį darbą, nenurodant vardo, pavardės;
☐ vardą, pavardę, magistro baigiamąjį darbą;
☒ magistro baigiamąjį darbą nenurodant vardo, pavardės.

Šiuo tikslu tvarkomų asmens duomenų Duomenų valdytojas neperduos jokiems tretiesiems asmenims, studentams su baigiamaisiais darbais bus leidžiama susipažinti vidinėje informacinėje sistemoje. Duomenų subjekto asmens duomenys šiuo tikslu bus naudojami ne ilgiau nei 5 metai.

Šiuo sutikimu Duomenų subjektas patvirtina, kad yra supažindintas su šiomis teisėmis:

- Susipažinti su savo duomenimis ir kaip jie yra tvarkomi (teisė susipažinti);
- Reikalauti ištaisyti arba, atsižvelgiant į asmens duomenų tvarkymo tikslus papildyti asmens neišsamius asmens duomenis (teisė ištaisyti);
- Savo duomenis sunaikinti arba sustabdyti savo duomenų tvarkymo veiksmus (išskyrus saugojimą) (teisė sunaikinti ir teisė „būti pamirštam“);
- Reikalauti, kad asmens duomenų valdytojas apribotų asmens duomenų tvarkymą (teisė apriboti);
- Teise į duomenų perkėlimą (teisė perkelti);
- Nesutikti, kad būtų tvarkomi asmens duomenys, kai šie duomenys tvarkomi ar ketinami tvarkyti kitais tikslais;
- Pateikti skundą Valstybinei duomenų apsaugos inspekcijai;

Duomenų subjektas turi teisę bet kuriuo metu atšaukti savo sutikimą.

__Andrius Tunikaitis_____
[Duomenų subjekto vardas, pavardė, parašas]

Duomenų valdytojo rekvizitai:
Vilniaus Gedimino technikos universitetas
Juridinio asmens kodas: 111950243
Adresas: Saulėtekio al. 11, LT-10223 Vilnius
Tel. (8 5) 274 5030
Faks. (8 5) 270 0112
El. paštas: vgtu@vgtu.lt
PVM mokėtojo kodas: LT119502413
Duomenų apsaugos pareigūno tel. (8 5) 251 2191, el. paštas: dap@vgtu.lt