Rīgas 64. Vidusskola

3D modeļa iegūšana no attēla ar mašīnmācīšanos

Pētnieciskais darbs programmēšanā

Darba autors:
Rīgas 64. vidusskolas 12. DIT klases skolnieks
Eduards Žuga

Darba vadītājs:
Rīgas 64. vidusskolas programmēšanas skolotājs
Edvards Bukovskis

Rīga 2022

Anotācija

3D modeļa iegūšana attēla ar mašīnmācīšanos. Eduards Žuga, Rīgas 64. vidusskolas 12.DIT klases skolēns. Darba vadītājs Rīgas 64. vidusskolas programmēšanas skolotājs Edvards Bukovskis.

Darba mērķis – izpētīt mašīnmācīšanos, konvolucionālos neironu tīklus, to darbību attēla pārveidošanā par 3D modeli, iegūt 3D modeli no attēla ar paša trenētu sistēmu, izmantojot atvērto pirmkodu. Darbā analizēta mašīnmācīšanās un KNT darbības principi un nozīme attēla apstrādē objekta 3D modeļa iegūšanai, 3D modeļu digitālās attēlošanas metodes. Darba ietvaros tiks apmācīta sistēma mājas apstākļos. Darba uzdevumi ir: izpētīt mašīnu dziļo mācīšanās pamatprincipus, izpētīt konvolucionālos neironu tīklus, izpētīt, kā no attēla var iegūt 3D modeli, to attēlošanas metodes, veikt eksperimentu, uztrenējot sistēmu, ar atvērto pirmkodu mājas apstākļos un iegūstot 3D modeli no attēla, izmantojot uztrenēto sistēmu. Pētījumā secināts, ka mašīnmācīšanās pielietojums ir daudzveidīgs un aktuāls, KNT pamatprincipi balstās uz bioloģisku nervu sistēmu darbību, KNT algoritms ir visatbilstošākais attēlu apstrādei, 3D informācijas attēlošanai ir vairākas metodes, efektīvai attēlu apstrādei priekš mašīnmācīšanas nepieciešamas modernas tehnoloģijas.

Atslēgas vārdi: mašīnmācīšanās, KNT, 3D modelis.

Abstract

Obtaining a 3D model from an image using machine learning. Eduards Žuga, a student of the 12th DIT class of Riga Secondary School № 64. Supervisor Edvards Bukovskis, programming teacher at Riga Secondary School № 64.

The aim of the work is to research machine learning, convolutional neural networks, their operation in transforming an image into a 3D model, to obtain a 3D model from an image with a self-trained system using open-source code. In the paper are analyzed the principles of machine learning and CNN operation and their importance in image processing for obtaining a 3D model of an object, methods of digital representation of 3D models. As part of the work a system will be trained in home conditions. The tasks are: to research the basic principles of machine deep learning, to research convolutional neural networks, to research how a 3D model can be obtained from an image, the methods of their representation, to conduct an experiment by training the system, with open source code in home conditions and obtaining a 3D model from the image using the trained system. The research concludes that the application of machine learning is diverse and relevant, the basic principles of CNN are based on the functioning of biological nervous systems, the CNN algorithm is the most suitable for image processing, there are several methods for displaying 3D information, modern technologies are required for effective image processing for machine learning.

Keywords: machine learning, CNN, 3D model.

Saīsinājumi un terminoloģija

2D/3D	Dimensionalitāte (dividimensionāls, trīsdimensionāls)
GPU	*(Graphics Processing Unit)* – Videokarte. Specializēts procesors grafiku renderēšanas pāatrināšanai.
CPU	*(Central Processing Unit)* – Centrālais procesors. Datorsistēmas sastāvdaļa, kas kontrolē instrukciju interpretāciju un izpildi.
RGB	*(Red, Green, Blue)* – (Sarkans, zaļš, zils) krāsu sistēma digitālos displejos. Izmanto lielākajā daļā ekrānu.
AVX	*(Advanced Vector Extensions)* – CPU arhitektūras papildinājums, kas ļauj CPU veikt apjomīgākas darbības ar saderīgu programmatūru.
Vērtība	Matemātikā: skaitļos izteikts kvantitatīvais raksturojums
Svars	*(Weight)* – Šī darba kontekstā: vērtība, uz kuras balstās viss KNT. Vērtība, kas tiek piereizināta ievades datiem neironos.
Sistēma	Šī darba kontekstā: datorsistēma.
Skripts	Instrukciju virkne, kas nosaka, kā programmatūrai jāveic kāda specifiska procedūra.
Troksnis	Datorzinātnē: nevēlama uzvedība funkcijās, kas samazina datu kvalitāti.
Matrica	Vairāku vērtību sakārtojums 2 dimensijās – rindās un kolonnās.
Iezīme	*(Feature)* – Datorzinātnē: elements attēlā, piemēram, objekta mala, krāsa, apgaismojuma līmenis.
Iezīmes vektors	*(Feature vector)* – Datorzinātnē: iezīmes vērtība, piemēram, pikseļa vērtība.
Iezīmes karte	*(Feature map)* – Matrica, kas sastāv no iezīmes vektora vērtībām.
Iterācija	Atkārtota operācijas vai funkcijas izpildīšana.
Rezolūcija	Attēla kvalitāte, kopējais pikseļu skaits, ko ekrāns var parādīt (piemēram, 1920x1080).
Renderēšana	Datu pārveidošana vizuālā veidā, iegūstot, piemēram, attēlu vai modeli.
Pamatpatiesība	*(Ground-Truth, GT)* – reāla vērtība, kuru sistēmai ir mērķis sasniegt ar mašīnmācīšanos, vērtība, ar kuru sistēma salīdzina iegūtos rezultātus.
Atpakaļpropagācija	Algoritms, kas paredzēts, lai pārbaudītu kļūdas, atgriežot datus atpakaļ no izvades neironiem uz ievades neironiem. Uzlabo mašīnmācīšanās prognožu precizitāti.
Atvērtais pirmkods	Pirmkods (kods), kas ir brīvi pieejams apskatei, lietošanai un rediģēšanai visiem patērētājiem.
Darbvirsma	Displeja ekrāna apgabals, kas imitē galda virsmu. Darbvirsma dod iespēju lietotājam pārvietot objektus, sākt un beigt uzdevumu izpildi tāpat kā uz reālas galda virsmas, tādējādi atvieglojot lietotāja darbu ar datoru.

Saturs

Ievads

Mūsdienās tehnoloģijas ir attīstījušās tik ļoti, ka programmas, roboti un mākslīgie intelekti spēj veikt cilvēka darbības vai attēlot domāšanas procesus tik pat labi, ja ne labāk. Šie tehnoloģiskie brīnumi ir panākti izmantojot mašīnmācīšanos – veids, kā datoriem mācīties, bez programmatūras, kas tieši norāda, kā ir jārīkojas. To pielietojums šobrīd gan ir ļoti specifisks – tas atvieglo darbu, bet neaizstāj to pilnībā, vai arī kalpo kā pamats vai papildinājums citam mākslīgajam intelektam.

Viens no mašīnmācīšanās pielietojumiem ir automātiska attēla apstrāde, to izmanto teksta nolasīšanai no attēla, objektu atpazīšanai, bildes kvalitātes uzlabošanai, pat medicīnā, palīdzot dažādu slimību un audzēju atpazīšanā, kā arī automatizēta 3D modeļa izveide no attēla, kas šajā darbā tiks apskatīts.

3D modeļi tiek izmantoti praktiski jebkur datorgrafikā – filmās efektu un mākslīgu objektu veidošanā, videospēļu vides izveidē, virtuālajā un paplašinātajā realitātē pasaules radīšanā vai papildināšanā. Šajā jomā 3D modeļa izveide no attēla var atvieglot un paātrināt procesu, jo vairs nav nepieciešams manuāli to konstruēt, jo dators to izdara automātiski. 3D modelēšana no attēla ir arī nepieciešama robotu mākslīgajiem intelektam apkārtējās vides izpratnē, lai bez sarežģījumiem par to varētu pārvietoties vai ar to mijiedarboties. Aktuāls piemērs – pašbraucošajiem automobiļiem ir nepieciešams "izprast" vidi, lai tie nesatriektos ar šķērsli. Lai no tā izvairītos, datoram tiek veidota digitāla apkartējās vides kopija, lai dators spētu aprēķināt optimālo trajektoriju. Arī medicīnā ķermeņa daļu vai audzēju 3D modelēšana var atvieglot operēšanu. Vajadzība pēc ātras un automātiskas 3D modeļa iegūšanas ir aktuāla un, iespējams, nākotnē nepieciešamība var pieaugt.

Šajā darbā tiks pētīta 3D modeļa izveide no attēla izmantojot mašīnmācīšanos, kā attēls tiek apstrādāts, kādas metodes tiek pielietotas, kā tiek nolasīta informācija no attēla un kādi dati ir nepieciešami, lai veiksmīgi iegūtu 3D modeli, kas atbilst attēlā esošajam objektam. Tiks arī veikts eksperiments, kurā tiks apmācīta sistēma ar kā palīdzību tiks iegūts 3D modelis no attēla, izmantojot atvērto pirmkodu.

Darba mērķis: iepazīties un izpētīt mašīnmācīšanos, konvolucionālos neironu tīklus, to darbību attēla pārveidošanā par 3D modeli, iegūt 3D modeli no attēla ar paša trenētu sistēmu, izmantojot atvērto pirmkodu.

Pētāmā problēma: objekta 3D modeļa konstruēšana no tā fotoattēla ar mašīnmācīšanās palīdzību.

Darba uzdevumi:

1. Izpētīt mašīnumācīšanās pamatprincipus
2. Izpētīt konvolucionālos neironu tīklus
3. Izpētīt, kā no attēla var iegūt 3D modeli
4. Veikt eksperimentu, uztrenēt sistēmu, ar atvērto pirmkodu mājas apstākļos
5. Iegūt 3D modeli no attēla, izmantojot uztrenēto sistēmu

Izmantotās darba metodes: Literatūras analīze, lai izpētītu mašīnmācīšanos un kā to pielietot 3D objekta ieguvei no attēla. Salīdzināšanas metode, lai varētu salīdzināt, 3D modeļu atbilstību minētajam uzdevumam. Atvērtā pirmkoda pielietošana, WSL lietošana, "Python" programmēšanas valodas pielietošana eksperimenta veikšanai.

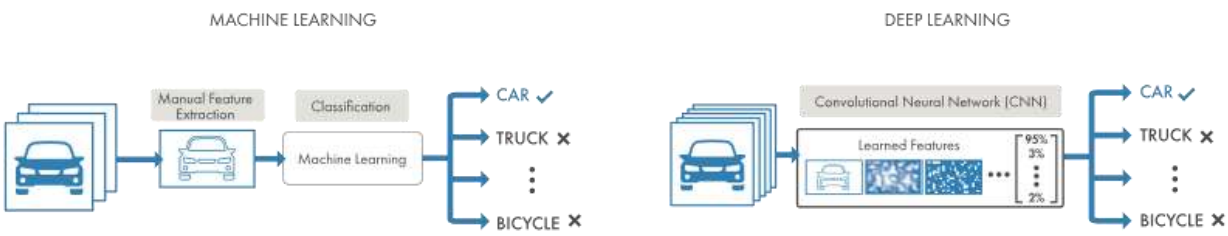
1. Literatūras apskats

1.1. Mašīnmācīšanās principi

Mašīnmācīšanās ir mākslīgā intelekta nozare, kas māca datoriem mācīties no pieredzes. Mašīnmācīšanās mērķis ir panākt, ka sistēma veic prognozes vai darbības, balstoties uz datiem un informāciju, bez īpaši vai specifiski ieprogrammētas uzdevuma veikšanas. Tas tiek panākts ar dažādu algoritmu un statistisko modeļu izmantošanu, lai sistēma "mācītos" informāciju tieši no iedotajiem datiem, nepaļaujoties uz iepriekš noteiktu vienādojumu kā modeli. Citiem vārdiem sakot, sistēma nav balstīta uz konkrētām, iepriekš noteiktām, formulām vai noteikumiem, tā vietā sistēma izmanto algoritmus un statistiskos modeļus, lai mācītos no datiem un veiktu lēmumus vai prognozes balstoties uz no datiem iegūtajām zināšanām. Kā arī laika gaitā, ja sistēmai dos vairāk datu, no kā mācīties, jo precīzāki būs rezultāti. Mašīnmācībā tiek izmantoti divu veidu paņēmieni: pārraudzītā mācīšanās (*supervised learning*) un nepārraudzīta mācīšanās (*unsupervised learning*).[1.]

Pārraudzītā mācīšanās apmāca sistēmu ar zināmiem ievades un izvades datiem, lai tā varētu radīt pamatotas prognozes par iespējamo pareizo atbildi no jauniem, iepriekš nezināmiem ievades datiem jeb datiem, kas netika izmantoti apmācībā. Vistipiskākie piemēri ir objektu, krāpšanu un spama atpazīšana – sistēmai iedod informāciju par konkrēto lietu, un tā mēģina atpazīt to no citiem datiem. Savukārt nepārraudzītajā mācīšanās sistēma meklē ievades datos modeļus, sakarības vai raksturīgākās struktūras iezīmes, lai izveidotu pēc iespējas labākos vai efektīvākos grupējumus. Piemēram, mobilo sakaru operatora uzņēmums vēlas optimizēt mobilo tālruņu bāzes staciju izvietojumu – uzņēmums var izmantot nepārraudzīto mašīnmācīšanos, dodot sistēmai datus par iespējamajām lokācijām, no kurienes tālrunis var mēģināt sazināties ar bāzes staciju, skaitu, cik daudz tālruņi var vienlaikus mēģina savienoties, un retumu, cik bieži šajā vietā tālrunis mēģina savienoties ar bāzes staciju un sistēma izstrādā labāko bāzes staciju izvietojumu, balstoties uz sakarībām iedotajā informācijā. [2.]

Dziļā mācīšanās ir specializēts mašīnmācīšanās veids, kas visbiežāk tiek izmantots attēlu atpazīšanas rīkos, runas atpazīšanas programmatūrās un "dabiskās" valodas apstrādē *(natural language processing)*. Tā imitē veidu, kā cilvēki iegūst noteikta veida zināšanas. Piemēram, bērnam ir pateikts vārds "suns", un viņš norāda uz priekšmetiem, sakot vārdu "suns", un vecāks atbild, ka tas ir vai nav suns. Bērns, turpinot šo darbību uz dažādiem priekšmetiem, iepazīstas ar iezīmēm, kas piemīt vai nepiemīt sunim, izveidojot sarežģītu abstrakciju – suņa jēdzienu, kas tika iegūts hierarhiski, pa slāņiem iegūstot zināšanas (skatīt 1. attēlu).[3.]



(1. attēls: mašīnmācīšanās (pa kreisi) un dziļās mācīšanās (pa labi) pieeju salīdzinājums transportlīdzekļu klasificēšanā. [2.])

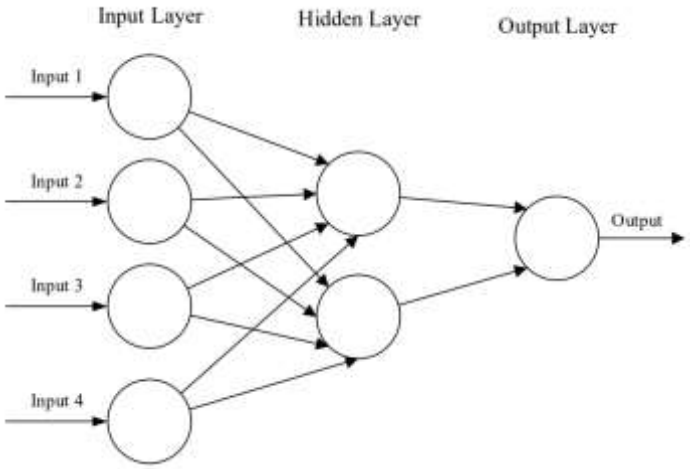
Tradicionālajā mašīnmācīšanā programmētājam ir jābūt ārkārtīgi specifiskam, pastāstot datoram, kādas lietas vai iezīmes tam vajadzētu meklēt, lai izlemtu, vai attēlā ir suns

vai nav suns. [3.] Šo procesu sauc par iezīmju ieguvi – neapstrādāti dati, piemēram, attēla pikseļu vērtības, krāsas vai malas, tiek pārveidoti piemērotā iekšējā datu attēlojumā vai iezīmju vektorā, no kā tālāk sistēma var klasificēt ievades datus. Tas ir darbietilpīgs process, un rezultātu precizitātes līmenis ir pilnībā atkarīgs no programmētāja spējas precīzi definēt objekta, šajā gadījumā suņa, iezīmju kopu. Dziļās mācīšanās priekšrocība ir tāda, ka sistēma pati veido iezīmju kopu ar nepārraudzīto mācīšanos, izmantojot atpakaļpropagācijas algoritmu, kas norāda, kā sistēmai jāmaina iekšējie parametri, uz kuriem dators balstās pārveidojot ievades datus iekšējā datu attēlojumā.[4.] Tas sanāk ne tikai ātrāk, bet parasti arī precīzāk, bet trūkumi ir nepieciešamība pēc jaudīgiem un vairākiem GPU, lai strauji apstrādātu datus, un pēc milzīga datu daudzuma, piemēram, tūkstošiem dažādu attēlu, lai apmācītu datoru. [2.]

1.2. Konvolucionālie neironu tīkli

Konvolucionālais neironu tīkls (KNT) ir dziļās mācīšanās algoritms, kas var uztvert ievades attēlu, piešķirt nozīmi dažādām attēla iezīmēm, aspektiem un objektiem un spēj atšķirt vienu no otra. KNT nav nepieciešama liela iepriekšēja datu apstrāde salīdzinājumā ar citiem klasifikācijas algoritmiem, ar pietiekamu apmācību KNT var apgūt šīs īpašības un iezīmes, lai programmētājam nav jāveido tās manuāli.[5.]

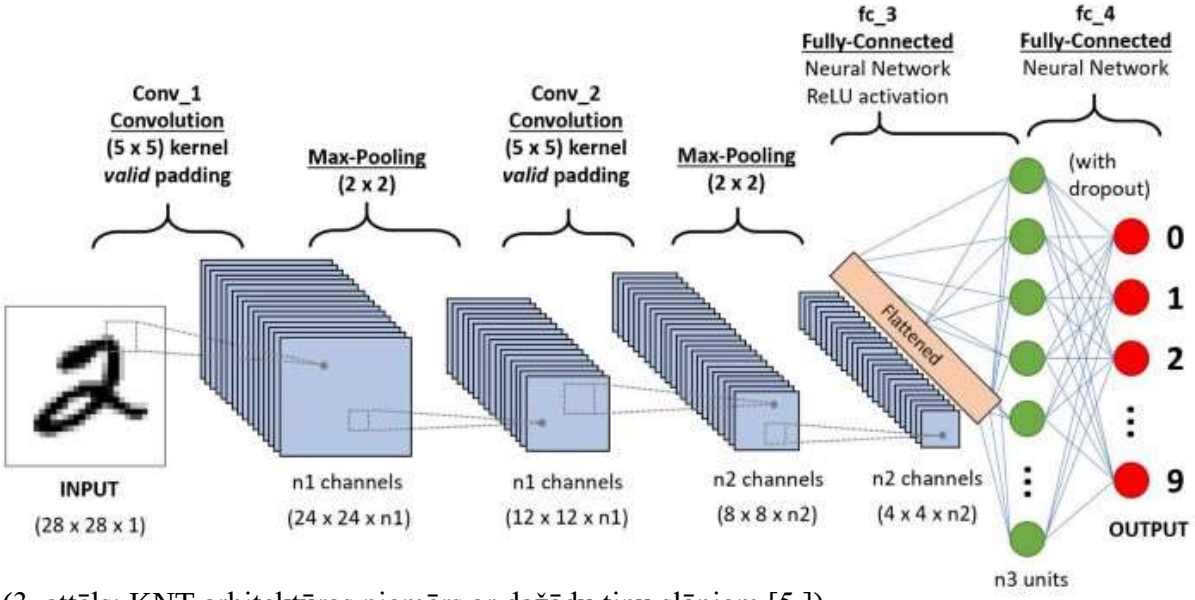
KNT arhitektūras pamatā ir cilvēka smadzeņu neironu sistēmas darbības. KNT galvenokārt sastāv no liela skaita savstarpēji sasaistītiem skaitļošanas mezgliem jeb neironiem, kas darbojas atsevišķi, lai kolektīvi mācītos no iedotajiem datiem, lai optimizētu rezultātu. KNT neironi mācoties paši optimizējas. Ievades slānis sadala ievades datus pa slēptajiem slāņiem, un slēptie slāņi pieņem lēmumu no iepriekšējā slāņa un izsver, vai stohastiskas izmaiņas sevī kaitēs vai uzlabos rezultātus. Piemēram, dators izvēlas nejaušu svara vērtību, un ievades dati iet caur slāņiem, kur šī vērtība tiek pievienota. Dati tiek laisti cauri slāņiem, un izvades slānī tie tiek noregulēti, balstoties uz iepriekšējā slāņa neironu izvades datiem. Ja iegūtie dati neatbilst vēlamajiem rezultātiem, svara vērtība tiek pamainīta. Šo procesu sauc par mācīšanos, un neironi paši optimizējas mācoties. Katrs neirons saņem ievades datus un veic kādu operāciju, piemēram, skalāro reizināšanu ar svaru vai nelineāru funkciju. (skatīt 2. attēlu).[6.]



(2. attēls: Neironu tīkla shēma, kas sastāv no ievades slāņa, slēptā slāņa un izvades slāņa. [6.])

KNT sastāv no trīs veidu slāņiem: Ievades slānis – saglabā attēla pikseļu vērtības; Konvolūcijas slānis – galvenais KNT slānis, kur notiek lielākoties visas skaitļošanas darbības, šeit tiek iegūti iezīmju vektori, no tiem tiek izveidota iezīmju karte; Apvienošanas *(Pooling)*

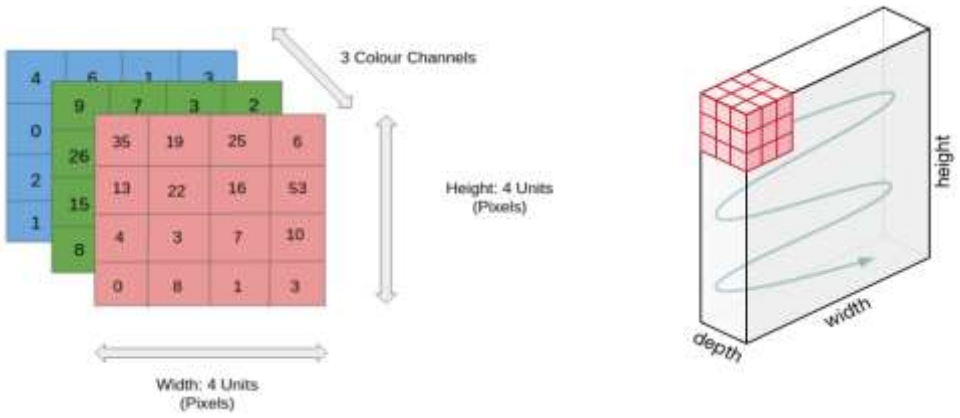
slānis – apkopo iezīmes un samazina iezīmju kartes dimensijas, samazinot parametru daudzumu priekš sistēmas apmācības; Pilnībā savienotais *(fully-connected)* slānis – pārveido iepriekšējo slāņu izvadi vienā vektorā, kas kalpo kā nākamajiem posmiem kā ievade vai kā gala rezultāts (skatīt 3. attēlu). KNT struktūra ļauj tajā iekodēt attēlam raksturīgus līdzekļus, padarot tīklu piemērotāku uz attēlu vērstiem uzdevumiem nekā citi mākslīgie neironu tīkli, vienlaikus arī samazinot modeļa izveidei nepieciešamos parametrus. KNT var sastāvēt no vairākiem katra tipa slāņiem, atkarībā no nepieciešamības. [6.]



(3. attēls: KNT arhitektūras piemērs ar dažādu tipu slāņiem.[5.])

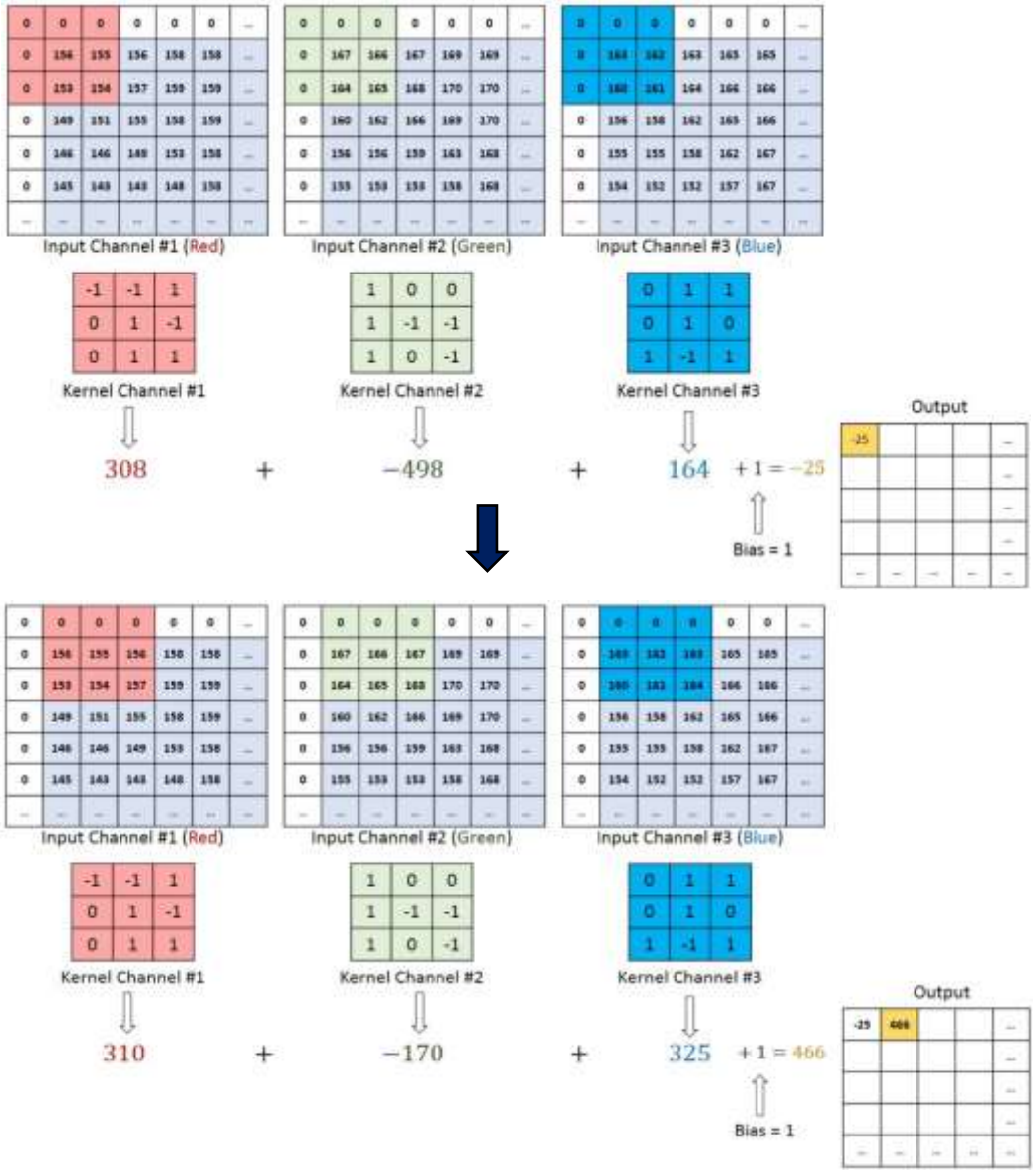
1.3. Iezīmju vektoru iegūšana

Lai no RGB attēla iegūtu iezīmes, attēlu sadala 3 plāksnēs katrai krāsu vērtībai – sarkanā, zaļā un zilā, iegūstot konkrēta pikseļa vērtību tajā krāsu plaknē (skatīt 4. attēlu). Tālāk izmanto filtrus, kurus sauc par "kerneļiem" – matrica, kas virzās pāri ievades attēlu pikseļiem (skatīt 4. attēlu) un nolasa to vērtības, sareizina, iegūstot rezultātā jaunu punktu matricu. Šī matrica apzīmē kopējā attēla pikseļu vērtības, un var būt arī 3D, kā šajā gadījumā, attēla vienam pikselim atbilst 3 vērtības – pa vērtībai no katras krāsas.[5.]



(4. attēls: 4x4 pikseļu liela attēla sadalījums pa RGB plaknēm/kanāliem ar pikseļu vērtībām (pa kreisi). Kerneļa virzība pāri ievades attēlam (pa labi).[5.])

Kernelis pārvietojas no kreisās uz labo pusi ar noteiktu intervālu, piemēram, 1 pikselis, un pēc katra pārvietojuma reizes, veic elementu reizināšanu ar kerneļa matricas vērtībām un attēla vērtībām, kuras kernelis ir pārklājis. Tālāk filtrs turpina virzīties pa labi, līdz tas ir nomērojis visu attēla platumu. Tad tas atlec atpakaļ uz kreiso pusi pavirzās noteikto intervālu uz leju un veic tās pašas darbības atkal, līdz viss attēls tiek noskenēts tādā veidā. Gadījumā, ja attēlam ir vairāki kanāli, piemēram, RGB ar 3 krāsu kanāliem, kernelim ir tāds pats dziļums kā ievades attēlam, piemēram, ar RGB būtu 3, tiek veikta matricu reizināšana, rezultāti tiek sasummēti, pieskaita vērtību *(bias)*, lai rezultāti nebūtu sistemātiski "aizspriedumaini" kļūdu dēļ, kas var rasties sistēmas apmācībā, un rezultātā iegūst vienu kanāla dziļu iezīmes izvadi jeb iezīmju karti (skatīt 6. attēlu).[5.]



(6. attēls: Kerneļa virzība pāri RGB attēlam ar intervālu 1 un rezultātā iegūtā iezīmes karte.[5.])

Šīs konvolūcijas darbības mērķis ir no ievades attēla iegūt augsta līmeņa iezīmes. Parasti pirmais konvolucionālais slānis uztver zema līmeņa elementus, tādus kā malas, krāsas, gradientu orientācijas utt., savukārt nākamie mēģina atpazīt augstāka līmeņa iezīmes, piemēram, figūras, apgaismojums un formas. Tālāk apvienošanas slānis samazina objekta

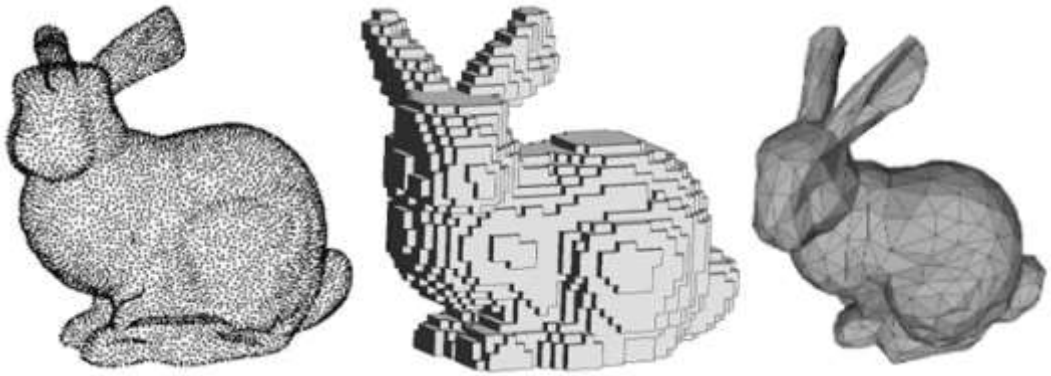
telpisko izmēru, samazinot skaitļošanas jaudu datu apstrādei, kā arī palīdz iegūt dominējošās pazīmes, kas pozicionāli un rotācijas nemainīgas, samazina trokšņu daudzumu, saglabājot efektīvu sistēmas apmācību. Atkarībā no attēlu sarežģītības, slāņu skaitu var palielināt, lai vēl vairāk atrastu zema līmeņa detaļas, taču uz lielākas skaitļošanas jaudas rēķina.[6.]

Tagad ir iegūta sistēmai saprotama datu forma. Iegūtā iezīmju karte tiek pārveidota kolonnas (viendimensionālā) vektorā. Šis vektors tiek ievadīts neironu tīklā, sistēma tiek apmācīta, un katrā apmācības iterācijā pielieto atpakaļpropagāciju, un laika gaitā sistēma spēj noteikt atšķirību starp dominējošajām un zema līmeņa iezīmēm un klasificēt tās.[5.]

1.4. 3D modeļa iegūšana

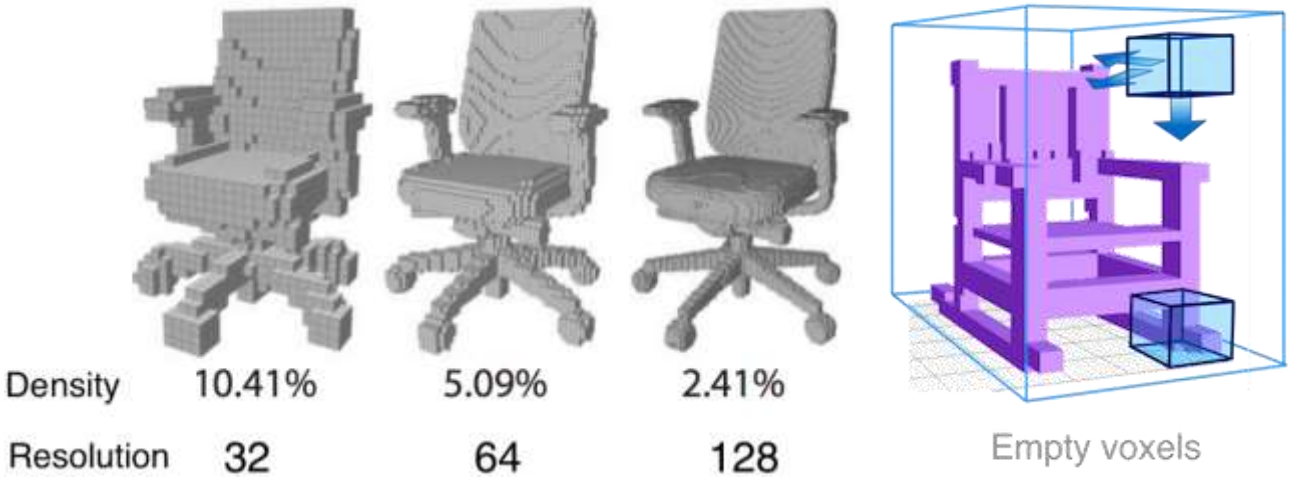
Attēls ir tikai 3D projekcija 2D plaknē, tādēļ, veidojot 3D modeli tikai no viena attēla, daļa informācijas tiek zaudēta attēlojot no augstākās dimensijas telpas objektu zemākā dimensijā. Tāpēc tikai no viena skatapunkta attēla nevar iegūt pietiekami daudz datu, lai izveidotu precīzu objekta 3D modeli. Vienīgais veids, kā izveidot 3D uztveri no 2D attēla, ir ar iepriekšējām zināšanām par 3D formu, kas ir attēlota attēlā. Uz konvolucionālā neironu tīkla balstīta sistēma spēj apgūt šīs nepieciešamās zināšanas, lai atpazītu attēlā rādīto figūru vai formu un kā tā izskatītos 3D telpā.[7.]

Atšķirībā no 2D attēla, kam ir tikai viens attēlošanas vieds digitālā formātā (pikseļi), 3D informāciju digitāli var attēlot dažādos veidos – galvenie ir vokseļi (telpisks pikselis), punktu mākonis un daudzstūru "siets" *(polygonal mesh)* (skatīt 7. attēlu).



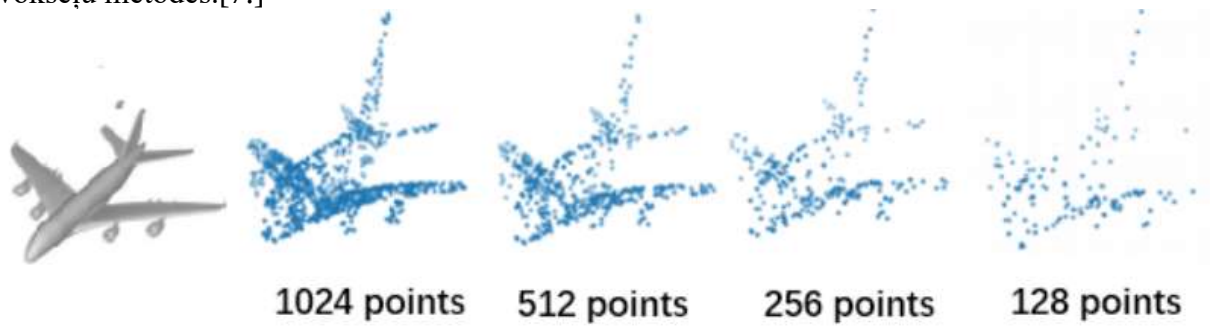
(7.attēls: zaķa modeļa reprezentācija punktu mākonī (pa kreisi), vokseļos (vidū), un daudzstūru sietā (pa labi). [8.])

Lai izveidotu modeli no vokseļiem, KNT ir jānosaka vokseļu atrašanās vietas telpā, klasificējot uzzinātās iezīmes un konkrētu pikseļu atrašanās vietu plaknē un attiecīgi izveidot vokseli katram pikselim un visi vokseļi kopā veido modeļa struktūru. Šī ir vienkāršākā metode, savukārt tai ir lieli trūkumi. Pirmkārt, ja 3D modelim ir maza rezolūcija, vairāki vokseļi tiks apvienoti vieni lielākā, zaudējot lielus daudzumus informācijas – modelis būs šķautņains un daudzas detaļas tiks zaudētas, tāpēc ir nepieciešama augstāka rezolūcija modeļiem, kas prasa eksponenciāli vairāk resursu datu apstrādāšanai un renderēšanai (skatīt 8. attēlu). Otrkārt, šī metode ir ļoti izšķērdīga, jo noderīgu vokseļu blīvums samazinās palielinoties izšķirtspējai un sistēmai aprēķinos ir jāizmanto tukši vokseļi, kas tiek izmantoti, lai noteiktu tikai lietderīgo vokseļu atrašanās vietu telpā, nelietderīgi patērējot skaitļošanas resursus (skatīt 8. attēlu). [9.]



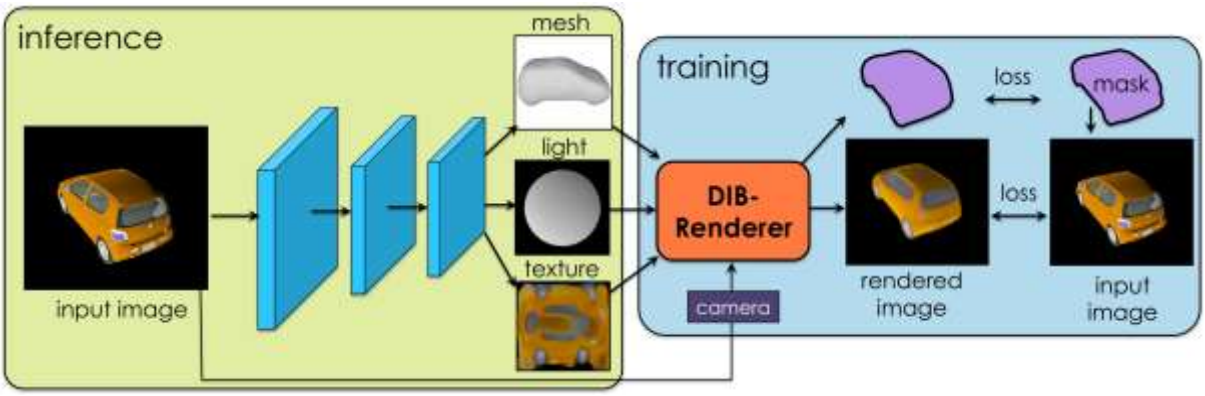
(8. attēls: Noderīgo vokseļu blīvums atkarībā no rezolūcijas (pa kreisi), tukšie vokseļi attēloti ar zilām kastēm – vokseļi, kas nesatur nekādu modeļa struktūras informāciju (pa labi).[10.])

Punktu mākonis ir kopa ar punktiem telpā, kur katram punktam ir savas koordinātas Dekarta koordinātu sistēmā (x,y,z) un citas īpašības, piemēram, krāsu vērtība. 3D modeļa detalizācija ir atkarīga no punktu skaita un blīvuma mākonī – jo vairāk punktu, jo precīzāks objekta attēlojums (skatīt 9. attēlu). Savienojot šos punktus, iegūst objekta struktūru, bet, lai pareizi šos punktus savienotu, ir nepieciešamas zināšanas par formām un figūrām telpā no to 2D attēlojumiem – sistēma ir jāapmāca atpazīt formas, kā arī apmācāmo 3D formu paraugu izmēri var atšķirties, sarežģījot mācību metožu pielāgošanu, kā arī nav iespējams tieši pielietot tradicionālus konvolūcijas algoritmus. Savukārt punktu mākonis ir ļoti kompakta reprezentācija, jo saglabā tikai lietderīgo punktu koordinātas un neko vairāk, atšķirībā no vokseļu metodes.[7.]



(9. attēls: 3D modeļa attēlojums atkarībā no punktu daudzuma.[11.])

Daudzstūru siets ir virsotņu *(vertices)*, malu *(edges)* un virsmu *(faces)* apkopojums, kas veido objekta virsmu trīs dimensijās. Šis modeļa attēlošanas veids ir vispopulārākais, jo, pirmkārt, tas ir lietotājam draudzīgs – šo metodi izmanto lielākoties visās manuālajās modelēšanas programmatūrās, populārākās ir "blender", "Unity", "Cinema 4D", otrkārt, daudzstūra sieta 3D modeļi neprasa daudz atmiņas un modeļa veidošanā ir iespējams tieši izmantot augsta līmeņa iezīmes, tādas kā apgaismojums, tekstūra un forma - ar šo metodi var uztvert un kompakti attēlot sīkas detaļas bez salīdzinoši lieliem resursa patēriņiem (skatīt 10. attēlu).[12.] Trūkums ir, ka daudzstūru sieta kvalitāte arī ir atkarīga no tā elementu skaita, piemēram, lai apaļas un gludas virsmas sastāv no daudziem maziem daudzstūriem, un daudzstūru sieti var atšķirties ar virsotņu un malu skaitu un to topoloģiju, kamēr to forma ir gandrīz vienāda.[13.]



(10. attēls: KNT shēma, kur 3D modelis tiek iegūts ar daudzstūra sieta reprezentācijas metodi. Sistēma atlasa augsta līmeņa iezīmes un izvadē uzrenderē 3D modeli.[12.])

Lai iegūtu 3D modeli, sistēmu apmāca ar dažādiem objektiem, dažādām formām meklēt korelācijas starp attēlā attēlotā objekta iezīmēm un datiem, kas iegūti mašīnmācīšanas rezultātā, pēc tam no ievades datiem, kas ir kāda objekta attēls, iegūt datus kādā no 3D informācijas digitālajiem attēlošanas veidiem un uzrenderēt 3D modeli, kas sistēmai šķiet atbilstošs attēlā attēlotajam objektam.

3D modeļu ieguvē var arī izmantot kombinācijas ar vairākām 3D informācijas reprezentācijas metodēm, lai kompensētu par kādiem specifiskiem trūkumiem, ja nav pietiekami jaudīgi vai konkrētam darbam pielāgoti datori. Metodes izvēle ir arī atkarīga no tā, kāds ir darba mērķis – spēļu un filmu industrijās vismazāk izmantos vokseļu metodi un visvairāk izmantos daudzstūru sieta metodi, jo to ir ērtāk izmantot un nav nepieciešami daudz resursu, lai glabātu datus, ja ir vajadzīgi ir lieli, detalizēti modeļi kā ēkas. Vokseļu metodi izmanto, kad nav nepieciešamība pēc augstas rezolūcijas modeļiem un vajag vienkāršu sistēmu, vai nav pieejamas vismodernākās tehnoloģijas, lai veiktu ātri un efektīvi lielāka mēroga datu apstrādi, piemēram vides digitālā plānojumā vai atveidē, lai robots mijiedarbotos ar vidi ar vienkāršām darbībām. Punktu mākoņus, savukārt, var vairāk pielietot rūpniecībā kvalitātes pārbaudē, meklējot defektus un citas problēmas, jo punktu mākonis sastāv no precīzām punktu koordinātām, un labi apmācīta sistēma spētu precīzi noteikt vietas ar bojājumiem. 3D modeļa ieguvei no viena attēla visas metodes der vienlīdz labi – neatkarīgi kādā veidā un kādas iezīmes tiek nolasītas, sistēmai tāpat ir jāveic minēšanas process dziļuma vērtībai. No viena attēla ir iespējams tikai nolasīt augstumu un platumu, formas dziļums ir atkarīgs no tā, cik labi sistēma spēj noteikt attēlā attēloto formu un spēt to pārveidot 3D formātā.

2. Punktu mākoņa ģenerēšana 3D objekta rekonstrukcijai

Punktu mākoņa ģenerēšana 3D objekta rekonstrukcijai ir process, kurā ar mašīnmācīšanās palīdzību tiek iegūts punktu mākonis, no kura var iegūt objektam atbilstošu 3D modeli. Process tiks veikts izmantojot atvērto pirmkodu (skatīt 1. pielikumu), kas sastāv no skriptiem, lai uztrenētu sistēmu veidot precīzus punktu mākoņus no ievades un lai norenderētu 3D modeli no iegūtā punktu mākoņa. Atvērtais kods apmāca sistēmu atpazīt un veidot krēslu 3D modeļus. Eksperimenta veikšanai tika izmantots WSL terminālis ar Ubuntu distribūciju, atvērtais pirmkods punkta mākoņa ģenerēšanai (skatīt 1. pielikumu), "Python" programmēšanas valodas 3.8 versija ar nepieciešamajām bibliotēkām, lai varētu palaist kodu. Eksperimenta mērķis ir uztrenēt sistēmu un ar to izveidot krēsla 3D modeli.

2.1. Prasības un ierobežojumi

Izmantotais kods tika izstrādāts palaišanai uz UNIX bāzētas operētājsistēmas. Lai šo prasību piepildītu tika izveidota operētājsistēmas "Linux" subsistēma uz "Windows" (WSL) – tiek izveidota atsevišķa miniatūras sistēma ar atbilstošo operētājsistēmu iekš pamata operētājsistēmas, kas funkcionē kā atsevišķa digitāla vide.

Lai uztrenētu sistēmu punktu mākoņa ģenerēšanai, nepieciešams ieinstalēt "Python" (versija 3.8) programmēšanas valodu ar bibliotēkām "numpy" (versija 1.23.1), "scipy", "termcolor", "tensorflow". Svarīgi ir lejupielādet tieši norādītās versijas, jo jaunākām versijām mainās funkcijas, kas neļauj palaist kodu. Sistēma, uz kuras tika veikts eksperiments, nebija saderīga ar noklusējuma bibliotēkas "tensorflow" versijām, jo šī sistēma neatbalsta AVX instrukcijas, kuras šai bibliotēkai ir nepieciešamas, tāpēc bija jāieinstalē speciāli pielāgota bibliotēkas "tensorflow" versija (skatīt 2. pielikumu). Šī instalācija ļauj ignorēt AVX nepieciešamību, bet limitē datu apstrādi tikai uz CPU nevis uz GPU, kā būtu ar noklusējuma bibliotēkas "tensorflow" versijām, krasi palielinot nepieciešamo laiku, lai pilnībā apstrādātu datus. Ieteicamo iterāciju daudzums bija jāsamazina par 100 reizēm, jo citādāk viens sistēmas apmācības cikls ilgtu 20 diennaktis.

Ja ir lejupielādēta bibliotēka "tensorflow" ar versiju 2.0 vai augstāku, nepieciešams visos "Python" skriptos (galā ".py") pārrakstīt funkcijas "import tensorflow as tf" uz "import tensorflow.compat.v1 as tf ", "tf.disable_v2_behavior()" (skatīt 11. attēlu).

(11. attēls: funkcijas "import tensorflow as tf" pārrakstīšana)

Tā kā datu apstrādes laiks ir krasi palielinājies, nepieciešams samazināt iterāciju daudzums, cik daudz sistēma tiks trenēta. Mapē "scripts" atrodas visi skripti, ar kuriem tiks palaistas sistēmas trenēšanas darbības. Failā "run-pretrain.sh" "--toIt" vērtība tika samazināta uz 5000, failā "run-finetune.sh" tika nomainītas vērtības: "{1..25}" uz "{1..4}"; "--load=orig-pre_it50000" uz "--load=orig-pre_it5000"; "--fromIt" un "toIt" vērtības 4000 uz 1000. Failos "run-evaluate.sh" un "run-compute-error.sh" vērtība "--load=orig-ft_it100000" nomainīta uz "--load=orig-ft_it4000".

Samazinot iterāciju skaitu, nepieciešams ir arī iestatīt citu vērtību, pie kuras iterācijas tiek saglabāts kontrolpunkts *(chekpoint)* – fails, kurā tiek saglabāta apgūtā informācija. Sākotnēji kontrolpunkts tiek saglabāts ar katru 5000. iterāciju. Tā kā kopējais iterāciju skaits

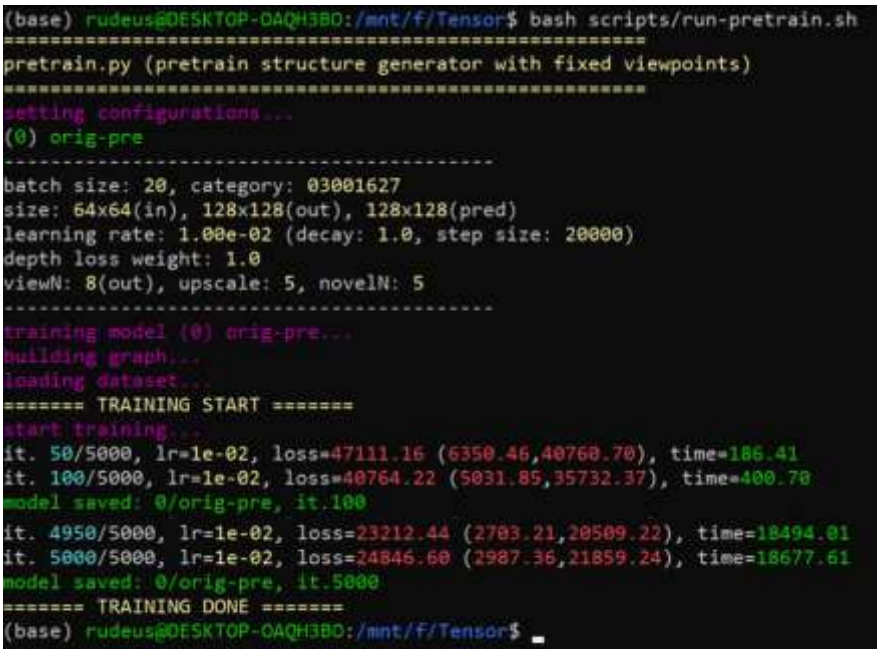
tika samazināts, tika iestatīts, ka kontrolpunkti saglabāsies ik pēc 100 iterācijām. Failos "train.py" un "pretrain.py" funkcijās "if (i+1)%5000==0:" vērtība tiek nomainīta no 5000 uz 100.

3D modeļa renderēšanai tika ieinstalēta programmatūras "blender" versija 2.78, un "Python" programmēšanas valodas OpenEXR versija, kas ir attēlu glabāšanas formāts. OpenEXR tiek plaši izmantots lielas precizitātes prasošām operācijām, piemēram, fotoreālistiskai renderēšanai, piekļuve tekstūrai, attēlu kompozīcionēšanai. Arī tika lejupielādēta "ShapenetCore v2" datu kopa (skatīt 3. pielikumu), kas sastāv no dažādu objektu 3D modeļiem, precīzai punktu mākoņa renderēšanai.

2.2. Sistēmas apmācīšana

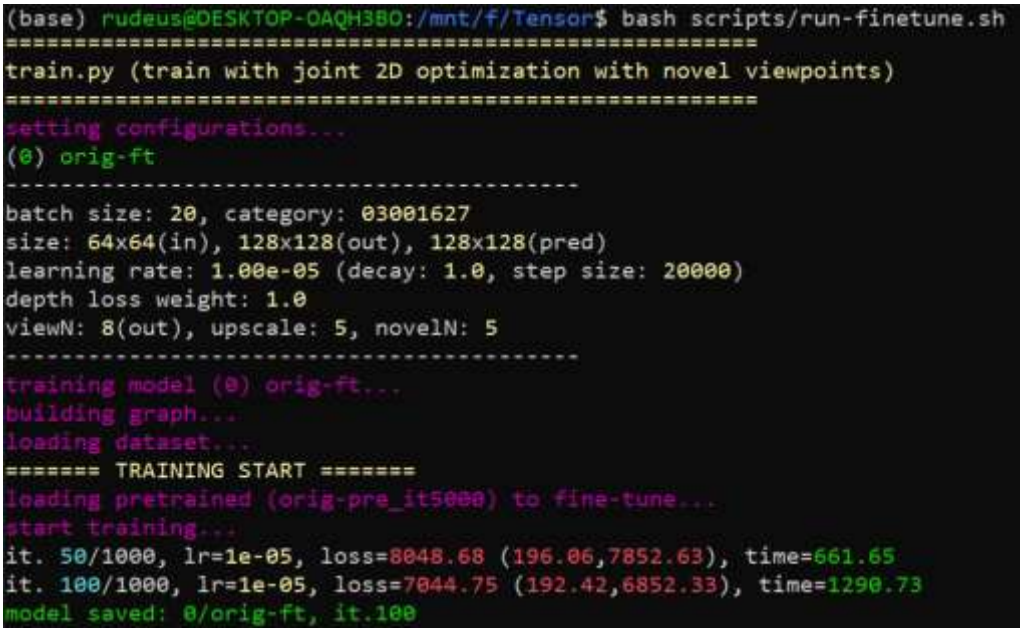
Lai uzsāktu sistēmas trenēšanu, ir nepieciešams lejupielādēt datu kopu, kas sastāv no ievades dažādu krēslu RGB attēliem, iepriekš renderētiem dziļuma attēliem un pamatpatiesības punktu mākoņiem sistēmas apmācībai. To izdara tverot failus ar operētājsistēmas "Linux" iekšējām funkcijām no saites, kas norādīta atvērtajā pirmkodā (skatīt 4. pielikumu).

Pēc datu sagatavošanas, sistēma ir gatava apmācībai. Tiek palaists skripts "run-pretrain.sh", kas sāk sistēmas apmācību. Sistēma ik pēc 100 iterācijām saglabā kontrolpunktu un beidz apmācību 5000. iterācijā (skatīt 12. attēlu). Trenēšana aizņēma aptuveni 5 stundas.



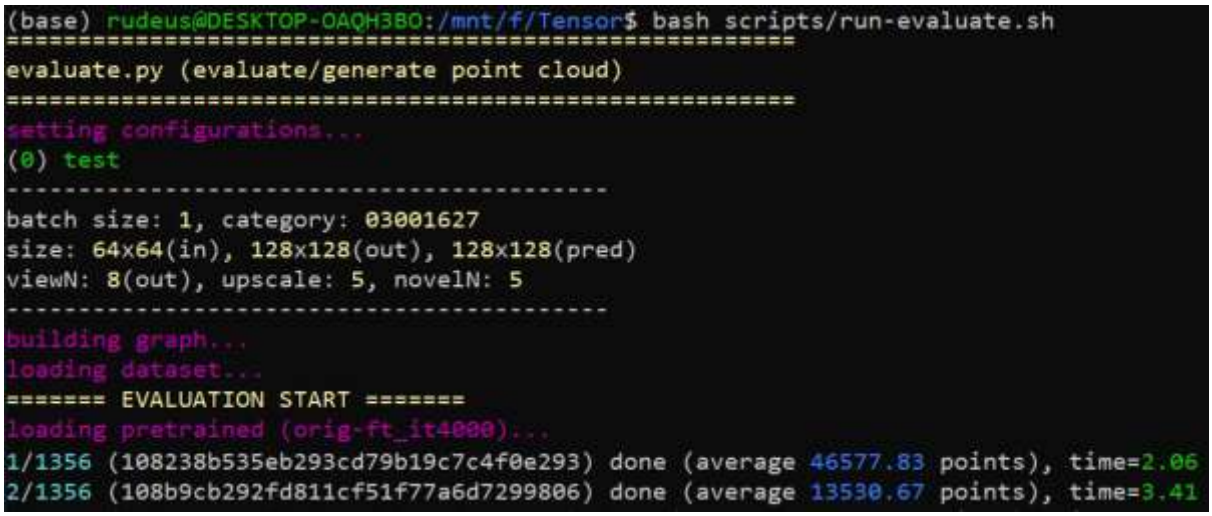
(12. attēls: Apmācības process. Sistēma uzrāda ik pēc 50 iterācijām laiku sekundēs, cik ilgi prasīja tās izpildīt un ik pēc 100 saglabā kontrolpunktu. Attēlā nav uzrādītas iterācijas 150 – 4900 atkļūdošanas teksti, citādāk attēla izmērs ir pārāk liels.)

Tālāk tiek palaists skripts "run-finetune.sh". Sistēma uzlabo un precizē apmācības procesā iegūtos kontrolpunktus (skatīt 13. attēlu). Šis process tiek atkārtots 4 reizes. Katrs cikls bija ~ 3.5 stundas ilgs, kopā ~ 14 stundas.



(13. attēls: Datu precizēšanas un uzlabošanas process. Attēlā parādīts 1. cikla sākums)

Pēc tam seko novērtēšana. Tiek palaists skripts "run-evaluate.sh". Tiek ģenerēti punktu mākoņi no datiem, kas iegūti apmācībā (skat. 14. attēlu).



(14. attēls: Punktu mākoņu veidošana no apmācībā iegūtiem datiem. Punktu daudzums katrā mākonī variē – no 10 tūkstošiem līdz 60 tūkstošiem punktu.)

Pēdējais solis šīs sistēmas apmācībā ir kļūdu aprēķināšana, salīdzināšana ar pamatpatiesību. Tiek palaists skripts "run-compute-error.sh". Iegūtie punktu mākoņi tiek salīdzināti ar trenēšanas procesā izmantotajiem datiem, ar pamatpatiesībām, tiek aprēķināta kļūda apmācīšanas procesā, šeit tiek noteikts, cik pilnīga ir bijusi apmācība (skatīt 15. attēlu). Process ilga ~22 stundas. Iegūtās vērtības atšķiras no oriģinālajām pamatpatiesībām (skatīt 16. attēlu) ar variējošu starpību. Sistēmu ir izdevies apmācīt, bet sistēma nav precīza. Lai iegūtu labākus rezultātus, ir nepieciešams izmantot lielāku iterāciju skaitu. Lai iegūtu norādītos rezultātus (skatīt 16. attēlu), ir nepieciešams krietni vairāk iterāciju un ciklu apmācīšanas un datu uzlabošanas procesā. Sākotnēji, apmācīšanas procesā bija 500 000 iterāciju un datu uzlabošanas procesā 100 000 iterāciju, bet sistēmas ierobežojumu dēļ bija nepieciešams

samazināt iterāciju skaitu, kas arī samazina rezultātu precizitāti un kvalitāti. Mape, kas satur visu datus sistēmas trenēšanai un iegūtos rezultātus, sver 53 GB.

```
(base) rudeus@DESKTOP-OAQH3BO:/mnt/f/Tensor$ bash scripts/run-compute-error.sh
======================================================
evaluate_dist.py (evaluate average distance of generated point cloud)
======================================================
setting configurations...
(0) test
----------------------------------------
batch size: 1, category: 03001627
size: 64x64(in), 128x128(out), 128x128(pred)
viewN: 8(out), upscale: 5, novelN: 5
----------------------------------------
loading dataset...
======= EVALUATION START =======
1/1356 108238b535eb293cd79b19c7c4f0e293: 2.3758(pred->GT),1.5736(GT->pred), time=64.74
2/1356 108b9cb292fd811cf51f77a6d7299806: 6.3632(pred->GT),9.2346(GT->pred), time=114.62
3/1356 10dc303144fe5d668d1b9a1d97e2846: 2.2634(pred->GT),1.0551(GT->pred), time=170.88
4/1356 11040f463a3895019fb4103277a6b93: 2.0526(pred->GT),1.6833(GT->pred), time=236.55
5/1356 111cb08c8121b8411749672386e0b711: 6.4024(pred->GT),5.5647(GT->pred), time=291.25

1355/1356 u45c7b89f-d996-4c29-aecf-4b760d1fb2b6: 5.8171(pred->GT),5.8377(GT->pred), time=77907.28
1356/1356 u6028f63e-4111-4412-9098-fe5f4f0c7c83: 5.6061(pred->GT),6.2792(GT->pred), time=77959.40
======= EVALUATION DONE =======
(base) rudeus@DESKTOP-OAQH3BO:/mnt/f/Tensor$
```

(15. attēls: apmācītās sistēmas kvalitātes noteikšana. Sarkanajā krāsā ir iegūto rezultātu atšķirības no pamatpatiesībām *(GT)*.)

	pred→GT	GT→pred
original	1.7342	1.8371

(16. attēls: Pirmkoda autoru iegūtie rezultāti attiecībā ar pamatpatiesībām *(GT)*.)

2.3. 3D modeļa renderēšana

Lai iegūtu 3D modeli no iegūtajiem datiem nepieciešama ir papildus programmatūra "blender". Kad šī programmatūra ir ieinstalēta, tiek palaists skripts "run.sh", kas atrodas mapē "render" ar papildus nosacījumiem, kā norādīts pirmkoda instruktāžā. Zem mapes "render" tiek palaista komanda "./run.sh 03001627 8", kur "03001627" ir "ShapeNet" datu kopas objekta kategorija un "8" ir skatu punktu jeb perspektīvu skaits – objekta projekcijas no dažādām pusēm. "03001627" kategorija sastāv no krēslu un krēsliem līdzīgu objektu modeļiem. Šī darbība neizdodas, jo WSL nespēj lietot skriptā esošo komandu "blender", kas izsauc un izpilda programmatūrā "blender" iebūvētās funkcijas (skatīt 17. attēlu).

```
(base) rudeus@DESKTOP-OAQH3BO:/mnt/f/tensor/render$ ./run.sh 03001627 8
./run.sh: line 1: blender: command not found
./run.sh: line 2: blender: command not found
```

(17. attēls: Kļūdas ziņojums, ka nevar izpildīt skriptā "run.sh" ierakstītu instrukciju)

Šo kļūdu rada programmatūras "blender" instalācijas lokācija jeb programmatūras failu glabāšanās vieta. Programmatūra ir uzinstalēta ārpus WSL vides, kas nozīmē, ka iekš WSL konkrētās programmatūras funkcijas nevar lietot. WSL ir arī atsevišķa "PATH" vērtība, kas norāda, kur atrodas programmatūras faili atrodas, lai spētu izsaukt un lietot konkrētās programmatūras komandas. Tas nozīmē, ka programmatūra, kas ir uzinstalēta ārpus WSL nav

lietojama iekš WSL. Lai šo problēmu atrisinātu, ir nepieciešams instalēt programmatūru "blender" iekš WSL. Šādā gadījumā ar WSL termināli nepietiek, jo nepieciešams programmatūru pilnībā uzinstalēt uz WSL, kam ir nepieciešams pilna operētājsistēmas "Linux" versija ar grafisku lietotāja saskarni *(GUI)*, kur var darboties vidē ne tikai ar komandām, bet arī mijiedarboties ar darbvirsmu. Lai šo panāktu, nepieciešams specializēts GPU draiveris. [14.] Sistēmai, ar kuru tika veikts eksperiments, nav piedāvāti specializēti GPU draiveri.

Uzrenderēt 3D krēsla uz mājas apstākļos apmācītās sistēmas no iegūtajiem datiem nav iespējams, ar ko arī noslēdzas eksperiments. Veiksmīgai eksperimenta veikšanai nepieciešams uzdevumam piemērotāka sistēma.

Secinājumi

1. Mašīnmācīšanās pielietojumu ir daudz un visdažādākajās nozarēs, un tehnoloģijām attīstoties, tās aktualitāte pieaug.

2. KNT pamatprincipi balstās uz bioloģisku nervu sistēmu darbību principiem, piemēram kā cilvēka smadzenes.

3. KNT algoritms ir visatbilstošākais attēlu apstrādei, jo tā struktūra samazina datu izmērus, nezaudējot svarīgu informāciju, vienlaicīgi efektīvi iegūstot dažādus attēlu parametrus un iezīmes.

4. Lai gan attēlus var attēlot digitāli tikai vienā veidā, 3D informācijas attēlošanai ir vairākas metodes, galvenokārt, vokseļu, punktu mākoņa un daudzstūra sieta metodes.

5. Nav vienas konkrētas metodes, kā vislabāk attēlot 3D modeli – tas ir atkarīgs no mērķa un pieejamajiem resursiem.

6. Efektīvai attēlu apstrādei priekš mašīnmācīšanas nepieciešams GPU un daudz vietas datu glabāšanai. Mašīnmācīšanās tieši attēlu apstrādē ir resursu prasošs process, mājas apstākļos grūti izpildāms.

Darba mērķi izdevās daļēji sasniegt, eksperimentā neizdevās iegūt krēsla 3D modeli no uztrenētās sistēmas ar atvērto pirmkodu. Lai to panāktu, nepieciešama uzdevumam piemērotāka sistēma vai, šī darba gadījumā, dziļas zināšanas programmēšanā un datorzinātnēs, lai spētu pielāgot kodu attiecīgajai operētājsistēmai, lai nebūtu jāizmanto atsevišķas darba vides, kas var apgrūtināt eksperimenta veikšanu.

Izmantotie literatūras avoti

1. MIT, Grimson, E., (2016) Introduction to Machine Learning. *Introduction To Computational Thinking And Data Science, Lecture 11.* Pieejams: https://ocw.mit.edu/courses/6-0002-introduction-to-computational-thinking-and-data-science-fall-2016/resources/lecture-11-introduction-to-machine-learning/.
2. MathWorks. What Is Machine Learning? How it works, why it matters, and getting started. Pieejams: https://ch.mathworks.com/discovery/machine-learning.html. Skatīts 2022. gada 4. decembrī.
3. Burns, E., & Brush, K. (2021). Deep Learning. *Techtarget AI* Pieejams: https://www.techtarget.com/searchenterpriseai/definition/deep-learning-deep-neural-network. Skatīts 2023. gada 8. janvārī.
4. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature, 521(7553), 436-444*.
5. Saha, S., (2018) A Comprehensive Guide to Convolutional Neural Networks. *towards data science* Pieejams: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. Skatīts 2023. gada 10. janvārī
6. O'Shea, K., & Nash, R. (2015). An introduction to Convolutional Neural Networks. *arXiv preprint arXiv:1511.08458.*
7. Lin, C. H., Kong, C., & Lucey, S. (2017). Learning efficient point cloud generation for dense 3D object reconstruction. *arXiv preprint arXiv:1706.07036*
8. Hoang, L., Lee, S. H., Kwon, O. H., & Kwon, K. R. (2019). A deep learning method for 3D object classification using the wave kernel signature and a center point of the 3D-triangle mesh. *Electronics, 8(10), 1196*.
9. Liu, Z., Tang, H., Lin, Y., & Han, S. (2019). Point-voxel CNN for efficient 3D deep learning. *Advances in Neural Information Processing Systems, 32.*
10. Lê, P. (2018) Create 3D model from a single 2D image in PyTorch. *Vitalify Asia* Pieejams: https://medium.com/vitalify-asia/create-3d-model-from-a-single-2d-image-in-pytorch-917aca00bb07. Skatīts 2023. gada 10. janvārī.
11. Cao, W., Liu, Q., & He, Z. (2020). Review of pavement defect detection methods. Ieee Access, 8, 14531-14544.
12. Chen, W., Ling, H., Gao, J., Smith, E., Lehtinen, J., Jacobson, A., & Fidler, S. (2019). Learning to predict 3d objects with an interpolation-based differentiable renderer. *Advances in Neural Information Processing Systems, 32.*
13. Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W. Y., Johnson, J., & Gkioxari, G. (2020). Accelerating 3D deep learning with Pytorch3D. *arXiv preprint arXiv:2007.08501.*
14. Microsoft. (2022) *Run Linux GUI apps on the Windows Subsystem for Linux.* Pieejams: https://learn.microsoft.com/en-us/windows/wsl/tutorials/gui-apps. Skatīts 2022. gada 8. janvārī

Pielikumi

1. pielikums
Mašīnmācīšanās atvērtais pirmkods punktu mākoņa ģenerēšanai 3D objekta rekonstrukcijai

Praktiskajā daļā izmantotais sistēmas apmācīšanas atvērtais pirmkods punktu mākoņa ģenerēšanai un 3D objekta izveidei ir pieejams GitHub repozitorijā: https://github.com/chenhsuanlin/3D-point-cloud-generation. Šis kods ir papildinājums [7.] avotam un ir licencēts zem MIT licences (MIT License). Skatīts 2022. gada 8. decembrī.

2. pielikums
Speciāli pielāgota programmēšanas valodas "Python" bibliotēkas "tensorflow" versija

Sistēma, uz kuras tika veikts eksperiments, nebija saderīga ar noklusējuma "tensorflow" versijām, tāpēc tika ieinstalēta pielāgota bibliotēkas versija no šīs saites: https://tf.novaal.de/barcelona/tensorflow-2.7.0-cp38-cp38-linux_x86_64.whl Atverot saiti uzreiz notiks faila lejupielāde, jo fails tika tverts ar "Linux" iekšējām komandām. Skatīts 2022. gada 14. decembrī.

3. pielikums

Datu kopa ar dažādu 3D objektu modeļiem

Eksperimentā lietotā "ShapeNetCore v2" 3D modeļu datu kopa, kas ir izmantojama datorgrafikas, robotikas, mašīnmācīšanas un citu nozaru pētījumos. Datu kopa tiek nepārtraukti papildināta un ir pieejama: https://shapenet.org/download/shapenetcore. Šo kopu izstrādāja Prinstonas, Stanfordas un TTIC universitāšu pētnieki. Skatīts 2022. gada 8. janvārī.

4. pielikums

Datu kopa sistēmas apmācībai

Eksperimentā lietotā dažādu attēlu, modeļu, punktu mākoņu saturoša datu kopa sistēmas apmācībai. Saite tiek norādīta koda lietošanas nosacījumos 1. pielikuma failā "README.md". Pieejama:
https://cmu.box.com/shared/static/s4lkm5ej7sh4px72vesr17b1gxam4hgy. Atverot saiti uzreiz notiks faila lejupielāde, jo fails tika tverts ar "Linux" iekšējām komandām. Skatīts 2022. gada 14. decembrī.