**Profesionālās izglītības kompetences centrs   
„RĪGAS VALSTS TEHNIKUMS”**

DATORIKAS NODAĻA

Izglītības programma: Programmēšana

**KVALIFIKĀCIJAS DARBS**

**Attēlu ģenerēšana ar neironu tīkliem**

Paskaidrojošais raksts 63 lpp.

Audzēknis: Daniils Sredņakovs

Vadītājs: Igors Litvjakovs

Normu kontrole:

**2022./2023. m.g.**

**Rīga**

**ANOTĀCIJA**

Kvalifikācijas darba ietvaros tika izstrādāts neironu tīkls, kas spēj ģenerēt attēlus. Kvalifikācijas darba autors ir Rīgas Valsts tehnikuma audzēknis Daniils Sredņakovs. Vadītājs ir Rīgas Valsts tehnikuma skolotājs Igors Ļitvjakovs.

Kvalifikācijas darbs ir veltīts ģeneratīvi konkurējoši tīkli (GAN – Generative Adversarial Networks), konkrēti dziļi konvolūcijas ģeneratīvie konkurējoši tīkli (DCGAN – Deep Convolutional GAN), izpētei attēlu ģenerēšanai. Darbs ietver DCGAN arhitektūras aprakstu, mācīšanās procesa ar izvēlētu datu kopu un analizēti ģenerēto attēlu rezultāti. Iegūtie rezultāti liecina par neironu tīklu izmantošanas potenciālu radošo darbu radīšanā, kā arī par potenciālu turpmākai pilnveidošanai. Programma tika uzrakstīta C++ valodā.

Kvalifikācijas darbā ir uzdevuma nostādne apraksts, kurā sīki aprakstīts noteiktais uzdevums, ar kuru jāsaskaras šim darbam. Prasību specifikācijas nodaļā ir sīki aprakstītas programmatūras ievades un izejas dati, funkcionālās prasības un nefunkcionālās prasības. Tālāk tiek sniegts skaidrojums par problēmas risināšanas rīku izvēli. Šajā sadaļā ir aprakstīts katrs izmantotais rīks, kā arī katra rīka izvēles pamatojums un tā pielietojamība konkrētajam uzdevumam. Nākamā darba nodaļa ir veltīta programmatūras produktu modelēšanai un projektēšanai. Šeit ir aprakstīts sistēmas arhitektūras modelis, kas ietver sistēmas arhitektūru un klases diagrammu. Tiek parādīts arī funkcionālās struktūras modelis, izmantojot datu plūsmas diagrammu. Atsevišķa darba sadaļa ir veltīta programmā izmantotās datu struktūras aprakstam. Šeit norādītas galvenās datu struktūras, to īpašības un izmantošanas veidi risināmo uzdevumu kontekstā. Lietotāja ceļvežā ir iekļautas sistēmas prasības, instrukcijas programmas instalēšanai, detalizēts programmas funkcionalitātes apraksts un programmas darbības piemēri, kas lietotājiem palīdzēs apgūt programmatūru. Darbā ir arī teorētiskā daļa, kurā ir izskaidrots neironu tīklu darbības princips kopumā, kā arī DCGAN lietotie principi, ir apskatīti un detalizēti izskaidroti. Avotu sadaļā ir uzskaitīti visi darbā izmantotie informācijas avoti, tostarp literatūra, raksti un tiešsaistes resursi. Ir arī saites uz izmantotajiem rīkiem, kā arī norādītas saites uz repozitorijiem, kur var iegūt šī kvalifikācijas darba ietvaros izstrādātā programmatūras koda kopiju. Darbā iekļauts arī pielikums, kurā sniegti papildu materiāli.

Kopumā kvalifikācijas darba apjoms ir 144lpp., kurā ietilpst 47 attēli, 17 vienādojumi un 2 pielikumi.

**ANNOTATION**

As part of the qualification work, a neural network capable of generating images was developed. The author of the qualification paper is Daniils Sredņakovs, a student of the Riga State Technical School. The manager is Igors Litvyakovs, a teacher at the Riga State Technical School.

The qualification work is devoted to the study of generative adversarial networks (GAN - Generative Adversarial Networks), specifically deep convolutional generative adversarial networks (DCGAN - Deep Convolutional GAN), for image generation. The work includes a description of the DCGAN architecture, the learning process with a selected data set, and analyzed the results of the generated images. The obtained results show the potential of using neural networks in the creation of creative works, as well as the potential for further improvement. The program was written in C++.

The qualification thesis contains a statement of the task, which describes in detail the specific task to be faced by this thesis. The requirements specification chapter details the software inputs and outputs, functional requirements and non-functional requirements. The following is an explanation of the choice of problem-solving tools. This section describes each tool used, as well as the rationale for selecting each tool and its applicability to the specific task. The next chapter of the work is devoted to the modeling and design of software products. A system architecture model that includes a system architecture and a class diagram is described here. A model of the functional structure is also shown using a data flow diagram. A separate section of the work is devoted to the description of the data structure used in the program. Here are the main data structures, their properties and ways of use in the context of the tasks to be solved. The user guide includes system requirements, instructions for installing the program, a detailed description of the program's functionality, and examples of how the program works to help users learn the software. The work also has a theoretical part, which explains the principle of neural networks in general, as well as the principles used in DCGAN, are discussed and explained in detail. The sources section lists all sources of information used in the thesis, including literature, articles, and online resources. There are also links to the tools used, as well as links to repositories where you can obtain a copy of the software code developed as part of this qualification. The work also includes an appendix that provides additional materials.

In general, the scope of the qualification work is 144 pages, which includes 90 images, 13 tables and 3 appendices.

SATURS

[IEVADS 5](#_Toc136178901)

[1. UZDEVUMA NOSTĀDNE 6](#_Toc136178902)

[2. PRASĪBU SPECIFIKĀCIJA 7](#_Toc136178903)

[2.1. Ieejas un izejas informācijas apraksts 7](#_Toc136178904)

[2.1.1. Ieejas informācijas apraksts 7](#_Toc136178905)

[2.1.2. Izejas informācijas apraksts 7](#_Toc136178906)

[2.2. Funkcionalās prasības 8](#_Toc136178907)

[2.3. Nefunkcionālās prasības 10](#_Toc136178908)

[3. UZDEVUMA RISINĀŠANAS LĪDZEKĻU IZVĒLES PAMATOJUMS 13](#_Toc136178909)

[4. PROGRAMMATŪRAS PRODUKTA MODELĒŠANA UN PROJEKTĒŠANA 14](#_Toc136178910)

[4.1. Sistēmas struktūras modelis 14](#_Toc136178911)

[4.1.1. Sistēmas arhitektūra 14](#_Toc136178912)

[4.1.2. Sistēmas klases modelis 18](#_Toc136178913)

[4.2. Funkcionālais sistēmas modelis 19](#_Toc136178914)

[4.2.1. Funkcionālais modelis 19](#_Toc136178915)

[4.2.2. Datu plūsmu modelis 20](#_Toc136178916)

[5. DATU STRUKTŪRU APRAKSTS 25](#_Toc136178917)

[6. LIETOTĀJA CEĻVEDIS 29](#_Toc136178918)

[6.1. Sistēmas prasības aparatūrai un programmatūrai 29](#_Toc136178919)

[6.2. Sistēmas instalācija un palaišana 30](#_Toc136178920)

[6.3. Programmas apraksts 34](#_Toc136178921)

[6.4. Testa piemērs 36](#_Toc136178922)

[7. DETALIZĒTS NEIRONU TĪKLA DARBĪBAS PRINCIPA APRAKSTS 40](#_Toc136178923)

[7.1. Lineārie neironu tīkli 40](#_Toc136178924)

[7.2. Detalizēts DCGAN metožu darbības apraksts 47](#_Toc136178925)

[7.2.1. Konvolucionālie neironu tīkli 47](#_Toc136178926)

[7.2.2. Citas kvalifikācijas darbā izmantotās metodes 53](#_Toc136178927)

[NOBEIGUMS 58](#_Toc136178928)

[INFORMĀCIJAS AVOTI 59](#_Toc136178929)

[PIELIKUMI 60](#_Toc136178930)

[1. pielikums 61](#_Toc136178931)

[2. pielikums 62](#_Toc136178932)

[3. pielikums 63](#_Toc136178933)

# IEVADS

Neironu tīkli ir spēcīgs rīks mākslīgā intelekta jomā, kas ļauj apstrādāt un analizēt datus, noteikt modeļus un pieņemt lēmumus, pamatojoties uz šo informāciju. Neironu tīklu ideja radās jau pirms datoru parādīšanās, kad pētnieki meklēja veidus, kā matemātiski modelēt cilvēka smadzenēs notiekošos procesus.

Neironu tīkli tiek izmantoti plašam lietojumu klāstam, sākot no attēlu un runas atpazīšanas līdz dabiskās valodas apstrādei un ieteikumu sistēmām. Tie ir īpaši noderīgi jomās, kurās tradicionālās plānošanas pieejas var būt grūti vai neiespējami izmantot, piemēram, lai atklātu modeļus lielās datu kopās vai prognozētu rezultātus, pamatojoties uz sarežģītām attiecībām starp mainīgajiem lielumiem. Veselības aprūpes jomā neironu tīkli tiek izmantoti, lai izstrādātu jaunas ārstēšanas metodes un terapijas visdažādākajām slimībām, sākot no vēža un diabēta līdz depresijai un trauksmei. Analizējot lielus medicīnisko datu apjomus, neironu tīkli var identificēt modeļus un attiecības, kas cilvēku pētniekiem var nebūt redzamas, kā rezultātā rodas jaunas atziņas un sasniegumi. Vēl viena joma, kurā neironu tīkliem varētu būt liela ietekme, ir autonomu sistēmu attīstība. Piemēram, autonomie transportlīdzekļi paļaujas uz neironu tīkliem, lai atpazītu un reaģētu uz objektiem savā vidē, padarot tos drošākus un efektīvākus. Tāpat droni un citas bezpilota sistēmas var izmantot neironu tīklus, lai pārvietotos un veiktu uzdevumus bez cilvēka iejaukšanās.

Tomēr viena no jomām, kas ar neironu tīklu palīdzību ir spējusi ievērojami progresēt, ir problēmas risinājums radošu produktu radīšanai ar datoru, piemēram, attēlu, tekstu, mūzikas un pat video ģenerēšanai. Šodien mēs varam redzēt pārsteidzošos rezultātus, kas sasniegti ar neironu tīkliem datorredzes un attēlu apstrādes jomā, kur tos izmanto, lai izveidotu fotoreālistiskus attēlus, pamatojoties uz tekstuālu aprakstu. Tādā veidā neironu tīkli paver mums jaunas iespējas radošuma jomā un palīdz paplašināt robežas tam, ko var izveidot ar datoru.

# UZDEVUMA NOSTĀDNE

Kvalifikācijas darba galvenais mērķis ir izveidot programmu, kas, izmantojot DCGAN (Deep Convolutional Generative Adversarial Networks) neironu tīkla arhitektūru, spēs ģenerēt jaunus attēlus, pamatojoties uz datu kopumu, atlasītu attēlu veidā par vienu konkrētu tēmu. DCGAN ir īpaša veida neironu tīkls, kas ir izstrādāts, lai ģenerētu jaunus attēlus, izmantojot dziļās mācīšanās metodes. Lai sasniegtu šo mērķi, ir veikts apjomīgs darbs, lai izpētītu DCGAN neironu tīkla arhitektūru, tā īpašības un darbības principus. Tika pētītas dažādas pirmapstrādes un datu apstrādes metodes, kā arī neironu tīkla optimizācijas un apmācības metodes. Balstoties uz šo darbu, tika izstrādāta programma, kurā tiek ievadīta attēlu datu kopa par konkrētu tēmu un tā tiek izmantota, lai apmācītu DCGAN neironu tīklu. Pēc tam apmācīto neironu tīklu var izmantot, lai ģenerētu jaunus attēlus, kas atbilst dotajam tematam.

Programmā jaunu attēlu ģenerēšanai, izmantojot DCGAN arhitektūru, jāiekļauj šādas funkcionalitātes:

* attēlu datu kopas ielādēšana un pirmapstrāde.
* jaunu attēlu ģenerēšana, pārveidojot nejaušus trokšņus attēlos ar 3 krāsu kanāliem RGB (sarkans, zaļš, zils).
* DCGAN neironu tīkla apmācība no iepriekš aprēķinātiem datiem.
* Galīgo rezultātu sniegšana lietotājam attēlu veidā, kā arī iespēja tos saglabāt JPEG un citu formātā.
* jau apmācīta neironu tīkla parametru saglabāšana turpmākai lietošanai, lai būtu iespēja ģenerēt attēlus par dažādām tēmām, kas atbilst apmācībai izmantotajam datu kopumam.
* parādīt informāciju par neironu tīkla mācību procesu

# PRASĪBU SPECIFIKĀCIJA

## Ieejas un izejas informācijas apraksts

### Ieejas informācijas apraksts

Programmā tiks nodrošināta šāda ieejas informācijas apstrāde.

* Uzglabāšanas ceļš - absolūtais ceļš uz mapi lietotāja atmiņas ierīcē, kurā tiks saglabāti jaunģenerētie attēli
* Faila formāts – lietotājs var izvēlēties, kādā formātā saglabāt attēlu
* Neironu tīkls - ja lietotājs vēlas ģenerēt attēlu no jau esoša neironu tīkla, viņš to var atlasīt no saraksta
* Attēlu skaits - norāda ģenerējamo attēlu skaitu
* Datu kopas ceļš - absolūts ceļš uz datu kopu lietotāja atmiņas ierīcē, kas ir mape ar attēliem.
* Neironu tīkla nosaukums - jaunā neironu tīkla nosaukums, kas apmācīts citā datu kopā
* Neironu tīklu parametri:
  + pakas izmērs - vesels skaitlis
  + mācīšanās ātrums - peldošā komata skaitlis
  + moments (impulss) - peldošā komata skaitlis
  + mācību laikmetu skaits - peldošā komata skaitlis
  + novirze - peldošā komata skaitlis

### Izejas informācijas apraksts

Programmā tiks nodrošināta šāda izejas informācijas apstrāde.

1. Neironu tīkla stāvoklis - Aplikācijas logā apmācības laikā būs teksta lauks, kurā tiks parādīts pašreizējās epohas numurs, iterācijas numurs, kā arī neironu tīkla zuduma funkcija (kļūda).
2. Attēli - Pēc neironu tīkla apmācības nejauši radītais troksnis tiek pārvērsts un normalizēts attēla vērtībās, pēc tam pikseļu vērtības tiek kodētas JPEG vai citu formātā un saglabātas datnē lietotāja norādītajā glabāšanas vietā.
3. Neironu tīkla apmācības stāvoklis – Neironu tīkla apmācības stāvoklis ir modeļa apmācības pašreizējais stāvoklis un norise. Programmai jāspēj parādīt informāciju par dažādiem mācīšanās aspektiem, tostarp:
   1. pašreizējā epoha vai iterācija, kurā notiek apmācība;
   2. zaudējumu vai kļūdas funkcijas vērtības katrā epohā/iterācijā;
   3. precizitāte jeb modeļa veiktspējas mērvienība attiecībā uz mācību un testa datiem;
   4. grafiki un vizualizācijas, kas parāda zaudējumu funkcijas un precizitātes izmaiņas apmācības laikā;
   5. mācīšanās rezultāts attiecīgajā laikmetā - Sākumā tiek ģenerētas 10 dažādas trokšņa vērtības, bet ar katru iterāciju tās nemaina savu vērtību. Tādējādi, apstrādājot nemainīgās vērtības un parādot tās uz ekrāna, būs iespējams izsekot neironu tīkla apmācības rezultātiem katrā epohā.
   6. cik ilgs laiks varētu būt nepieciešams, lai pilnībā apmācītu neironu tīklu

## Funkcionalās prasības

1. DCGAN neironu tīkla apmācība, izmantojot noteiktu datu kopu, kas sastāv no attēliem par noteiktu tēmu.
2. Iespēja norādīt vietu atmiņas ierīcē, kur atrodas treniņu datu kopa
   1. Ja norādītajā direktorijā nav attēlu, to nevar izmantot
   2. Ja norādītā direktorija neeksistē, to nevar izmantot
3. Iespēja izmantot attēlus no datu kopas apmācībai
   1. Attēla kodējums atbilst attēla nosaukumā norādītajam faila formātam
   2. Iespēja izmantot attēlu JPEG formātā
   3. Iespēja izmantot attēlu PNG formātā
   4. Iespēja izmantot attēlu BMP formātā
4. Iespēja norādīt vietu atmiņas ierīcē, kur tiks glabāti jaunizveidotie attēli
   1. Ja norādītā direktorija neeksistē, to nevar izmantot.
5. Iespēja pielāgot neironu tīkla nosaukumu.
   1. Ja ievadītais nosaukums ir vienāds ar jau esoša neironu tīkla nosaukumu, tad to nevar izmantot
6. Iespēja izvēlēties esošu neironu tīklu no saglabāto tīklu saraksta
7. Iespēja pielāgot neironu tīkla pakas izmērs.
   1. Ja ievadītais skaitlis nav vesels skaitlis, to nevar izmantot.
   2. Ja ievadītais skaitlis ir nulle, tad to nevar izmantot.
   3. Ja ievadītais skaitlis ir lielāks par datu kopas lielumu, tad to nevar izmantot
8. Iespēja pielāgot neironu tīkla mācīšanās ātrums.
   1. Ja ievadītais skaitlis nav peldošā komata skaitlis, to nevar izmantot
   2. Ja ievadītais skaitlis ir nulle, tad to nevar izmantot
   3. Ja ievadītais skaitlis ir vienāds vai lielāks par vienu (1), to nevar izmantot
9. Iespēja pielāgot neironu tīkla momentum.
   1. Ja ievadītais skaitlis nav peldošā komata skaitlis, to nevar izmantot
   2. Ja ievadītais skaitlis ir nulle, tad to nevar izmantot
   3. Ja ievadītais skaitlis ir vienāds vai lielāks par vienu (1), to nevar izmantot
10. Iespēja pielāgot neironu tīkla novirze.
    1. Ja ievadītais skaitlis nav peldošā komata skaitlis, to nevar izmantot.
    2. Ja ievadītais skaitlis ir nulle, tad to nevar izmantot.
11. Iespēja pielāgot neironu tīkla mācību laikmetu skaits.
    1. Ja ievadītais skaitlis nav vesels skaitlis, to nevar izmantot
    2. Ja ievadītais skaitlis ir nulle, tad to nevar izmantot
12. Neironu tīkla apmācības statuss ir jāparāda lietotājam teksta laukā
    1. Lietotājs nevar mainīt teksta lodziņā izejas vērtības.
    2. Informācijas logā ir jāparāda pašreizējā epoha.
    3. Informācijas logā ir jāparāda pašreizējā iterācija.
    4. Informācijas logā ir jāparāda pašreizējā diskriminatora kļūda.
    5. Informācijas logā ir jāparāda pašreizējā ģeneratora kļūda.
    6. Informācijas logā ir jāparāda laiku, kad beidzās iterācija.
    7. Informācijas logā ir jāparāda procentu skala, kas norāda pabeigto apmācību procentuālo daudzumu.
    8. Informācijas logā ir jāparāda laiku, cik ilgs laiks nepieciešams, lai apmācītu neironu tīklu.
13. Jaunu attēlu izveide, izmantojot apmācīto modeli.
    1. Iespēja saglabāt attēlu JPEG formātā
    2. Iespēja saglabāt attēlu PNG formātā
    3. Iespēja saglabāt attēlu BMP formātā
14. Iespēja saglabāt apmācītā neironu tīkla parametrus turpmākai izmantošanai
    1. Ģeneratora un diskriminatora katra slāņa neironu tīkla parametri tiek saglabāti secībā CSV failā
    2. Ja jau ir saglabāti neironu tīkla parametri ar tādu pašu nosaukumu, tad saglabātie parametri tiek pārrakstīti ar jauniem.
15. Apmācīšanas procesa un radīto attēlu vizualizācija rezultātu analīzei.
    1. Jāizveido diagramma, kurā parādīta diskriminatora un ģeneratora kļūdu vērtību pārmaiņu progresēšana.
    2. Parāda attēlu paraugus katram epoham.

## Nefunkcionālās prasības

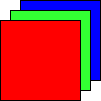
Sistēmas daļas prasības.

1. Programma ir jānodrošina pārnēsājamā formā (nav nepieciešamas instalēšanas lietojumprogrammas).
2. Programmai jābūt realizētai C++ programmēšanas valodā.
3. Liels apmācības un attēlu ģenerēšanas ātrums, lai nodrošinātu ātru programmas darbību. Īstenot daudzpavedienu programmā.
4. Tā kā ir nepieciešams strādāt ar lieliem datu masīviem, ir nepieciešams īstenot efektīvu darbu ar operatīvo atmiņu.

Izskata prasības.

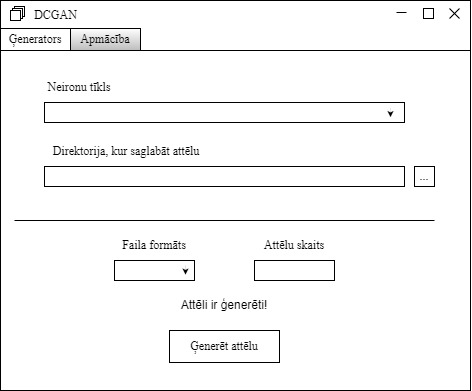
1. Lietotājam draudzīga saskarne datu ievadei un izvadei.
2. Lietojumprogrammas saskarnei jābūt latviešu valodā.
3. Programmai jābūt izstrādātai minimālisma veidā.
4. Nepieciešams izveidot lietojumprogrammas logotipa ikonu.

Tālāk ir parādīta lietotnes logotipa skice (skat. 2.1 att.). Logotips attēlo trīs attēla krāsu kanālus (sarkanu, zaļu un zilu), kas sadalīti trīs atbilstošos slāņos.



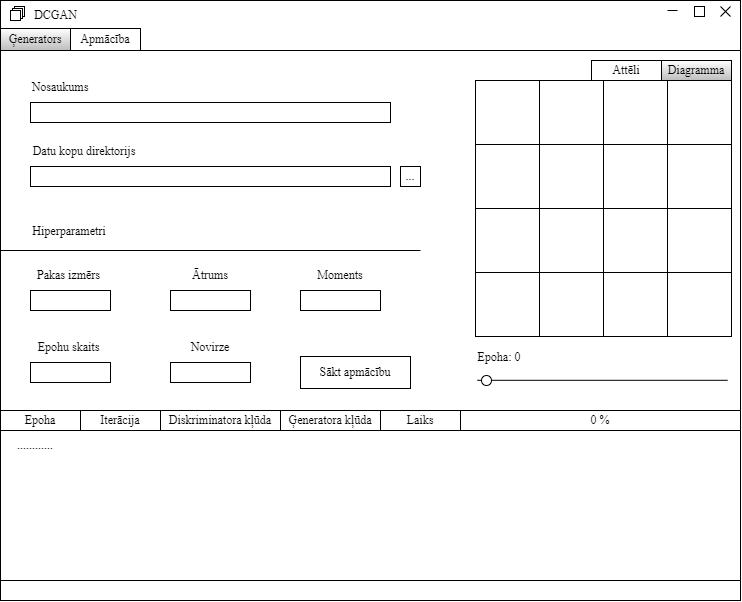
2.1. att. Logo skice

Zemāk ir programmas skices, uz kuru pamata tiks izstrādāts programmas interfeiss. Skices nodrošina vispārīgu interfeisa izkārtojuma un vizuālā dizaina koncepciju, tostarp komponentus, pogas, ievades laukus un citus lietotāja interfeisa elementus. Apakšējā skice ir attēlota attēlu ģenerēšanas kontroles cline (skat. 2.2 att.). Šajā cilnē lietotājs var kontrolēt ģenerēšanas procesu un mijiedarboties ar dažādiem parametriem.



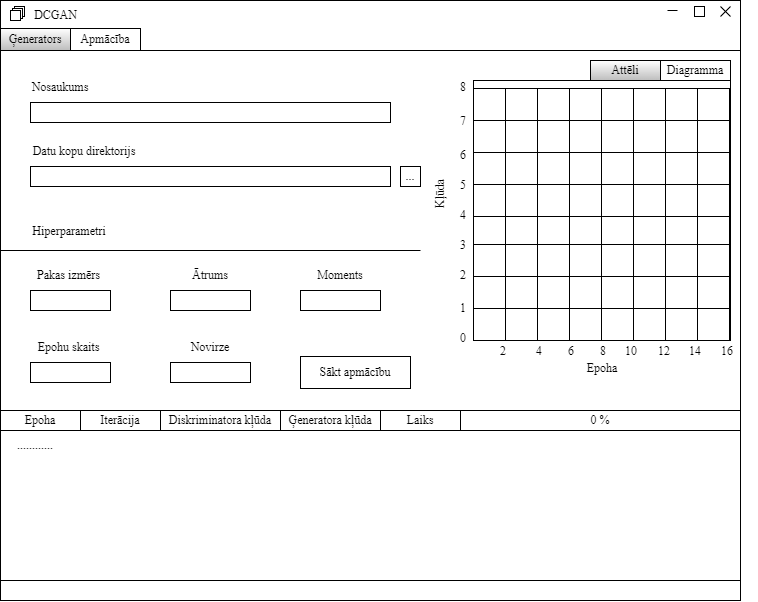
2.2. att. Attēlu ģenerēšanas kontroles cilnes skice

Tālāk ir sniegts interfeisa cilnes skicē, kas paredzēta darbam ar neironu tīklu apmācību. Augšējā labajā stūrī ir arī cilnes skice, kurā tiek parādīti attēla ģenerēšanas provizoriskie rezultāti (skat. 2.3 att.).



2.3. att. Neironu tīklu apmācības vadības cilnes skice

Tālāk ir parādīta augšējā labā loga otrās cilnes skice, kas ir diagramma, kas parāda izmaiņas neironu tīkla kļūdu vērtībās atkarībā no pagātnes laikmeta (skat. 2.4 att.).



2.4. att. Augšējā labā loga otrās cilnes skice

# UZDEVUMA RISINĀŠANAS LĪDZEKĻU IZVĒLES PAMATOJUMS

C++ programmēšanas valoda tika izvēlēta kā galvenais rīks neironu tīkla izveidei, jo C++ ir sistēmas programmēšanas valoda, kas nodrošina augstu veiktspēju un efektivitāti. Turklāt C++ nodrošina zemāka līmeņa piekļuvi aparatūrai, kas ļauj optimizēt neironu tīkla veiktspēju un arī strādāt tieši ar operatīvo atmiņu, kas ir svarīgi, strādājot ar lieliem datu masīviem. Turklāt C++ ir plaši izmantota programmēšanas valoda, tāpēc ir daudz rīku un bibliotēku, ko var izmantot, lai izveidotu lietojumprogrammu. Visbeidzot, C++ tika izvēlēts personīgās intereses dēļ apgūt šo programmēšanas valodu.

Visi izmantotie rīki ir uzskaitīti zemāk. Vispirms tiek rakstīts rīka nosaukums, bet iekavās tiek ierakstīta rīka versija.

1. С++ (20) - objektu orientēta programmēšanas valoda, kas tika izveidota kā uzlabota versija C. C++ apvieno gan augsta līmeņa, gan zema līmeņa valodu iespējas, padarot to par elastīgu un jaudīgu rīku programmētājiem.
2. Visual Studio 2022 (17.5.5) - Visual Studio ir Microsoft integrēta izstrādes vide (IDE), kas ļauj izstrādātājiem izveidot bagātīgas lietojumprogrammas daudzām platformām. Visual Studio ietver rīku kopumu, kas ļauj izstrādāt dažādās programmēšanas valodās, tostarp C++. Vide nodrošina arī dažādus rīkus atkļūdošanai, testēšanai, profilēšanai un lietojumprogrammu veidošanai.
3. STB - programmēšanas valodas C publiskā domēna bibliotēku kopa, ko izveidojis programmētājs Šons Barets (Sean Barrett). STB bibliotēkās ir funkcijas darbam ar attēliem, skaņu, ievadi/izvadi, matemātiskām operācijām un daudzām citām jomām. Izplatīts saskaņā ar MIT licenci.
4. stb\_image.h (2.28) - bibliotēka nodrošina vienkāršu un ērtu saskarni dažādu formātu (BMP, JPEG, PNG, TGA u.c.) attēlu ielādei un to tālākai apstrādei programmā
5. stb\_image\_write.h (1.16) - bibliotēka nodrošina vienkāršu un ērtu saskarni dažādu formātu attēlu ierakstīšanai failā vai atmiņā
6. stb\_image\_resize.h (0.97) - bibliotēka nodrošina vienkāršu un lietotājam draudzīgu interfeisu attēlu izmēru maiņai

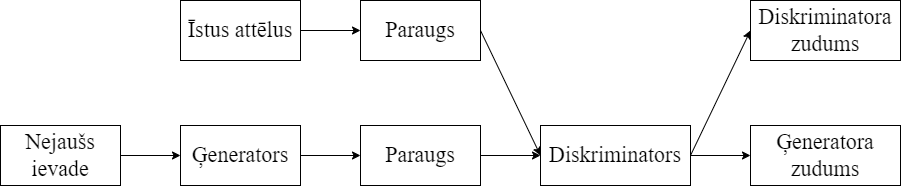
# PROGRAMMATŪRAS PRODUKTA MODELĒŠANA UN PROJEKTĒŠANA

Šajā nodaļā ir parādīta programmas galvenā strukturālā daļa, kurā apskatītas visas programmas ietvaros veiktās galvenās sastāvdaļas un procesi. Aprakstā ir sniegta shematiska informācija par katru procesu, neiedziļinoties tehniskās detaļās. Detalizētākai programmas izpratnei ieteicams skatīt nodaļu " DETALIZĒTS NEIRONU TĪKLA DARBĪBAS PRINCIPA APRAKSTS ", 42. lpp., kur ir sniegts detalizēts neironu tīkla un tā darbības apraksts, kas ļaus jums iegūt dziļu izpratni par programmas un tās algoritmu darbības principiem.

## Sistēmas struktūras modelis

### Sistēmas arhitektūra

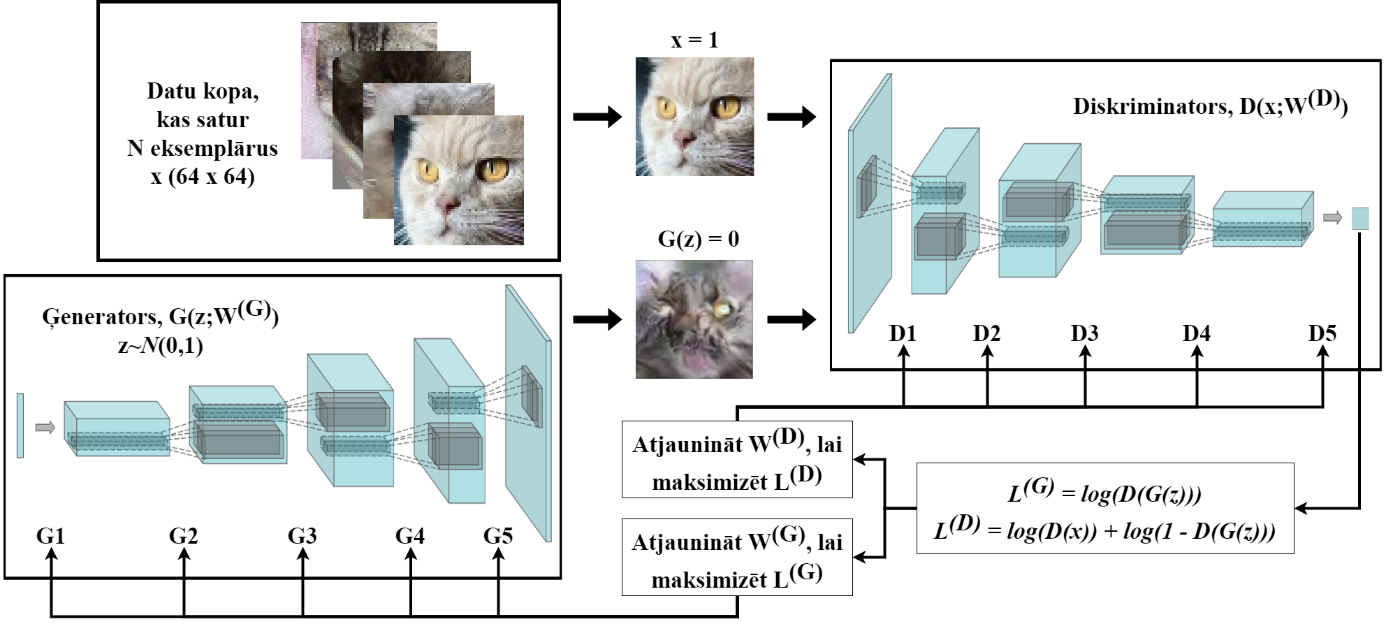
DCGAN (Deep Convolutional Generative Adversarial Network) ir neironu tīkla arhitektūra, kas izmanto konvolucionālos slāņus ģeneratora un diskriminatora modeļiem ģeneratīvajā pretinieku tīklā (GAN). Pirmo reizi tas tika ierosināts 2015. gada dokumentā ar nosaukumu " Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks ", ko veica Alec Radford, Luke Metz un Soumith Chintala.



4.1. att. GAN modeļa shēma

DCGAN motivācija bija novērst tradicionālo GAN ierobežojumus, kuri bieži radīja zemas kvalitātes attēlus. DCGANs izmanto dziļus konvolucionālus neironu tīklus, lai apgūtu attēlu hierarhiskus attēlojumus, ļaujot tiem ģenerēt attēlus ar lielāku detalizāciju un reālismu.

DCGAN arhitektūra ir līdzīga parastajai GAN arhitektūrai, pievienojot konvolucionālos slāņus gan ģeneratora, gan diskriminatoru tīklos. Ģeneratora tīkls kā ievadi ņem nejaušu trokšņa vektoru un ģenerē attēlu, savukārt diskriminētāju tīkls uzņem attēlu un klasificē to kā reālu vai viltotu.



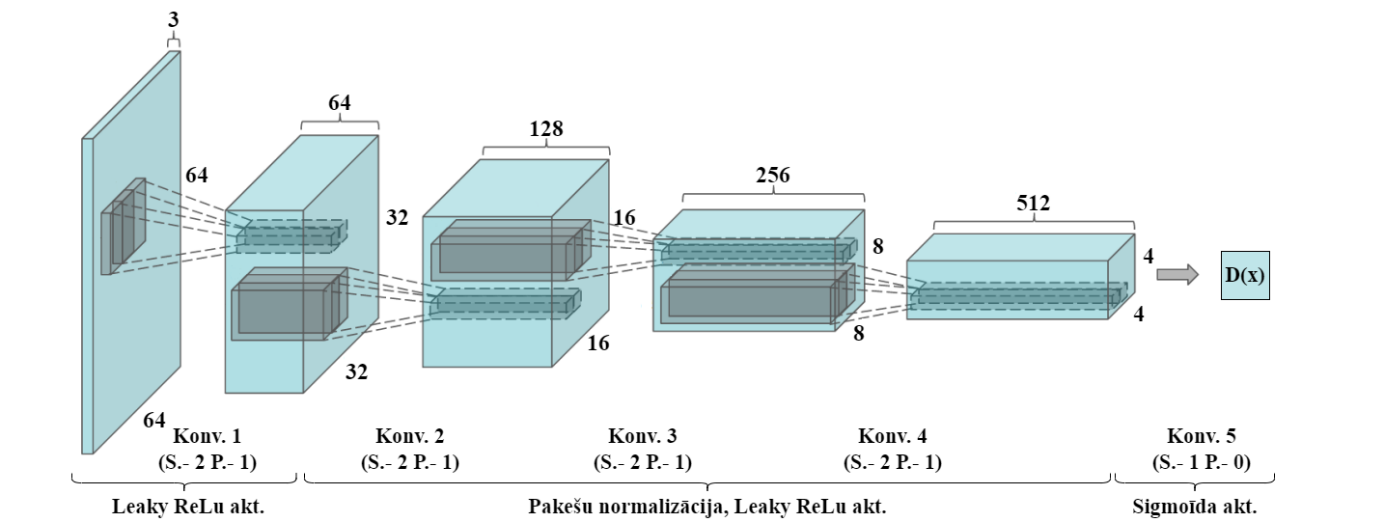
4.2. att. DCGAN modeļa shēma

***Diskriminators***

DCGAN diskriminators veic klasifikācijas uzdevumu un ir atbildīgs par ģenerēto un reālo attēlu nošķiršanu. Diskriminators ņem attēlu kā ievadi un izvada vērtību, kas norāda, cik iespējams, ka attēls ir reāls (vērtība vienāda ar 1) vai ģenerēta (vērtība vienāda ar 0). DCGAN diskriminators sastāv no vairākiem konvolūcijas slāņiem ar pakešu normalizāciju, LeakyReLU aktivizēšanas funkciju un sigmoīdu aktivizēšanu diskriminatora pēdējā slānī.

Diskriminatora slāņos izmantotās metodes (arī skat. 4.3 att.).

1. Konvolūcijas slānis: Konvolūcijas slāņi spēj noteikt lokālas atkarības un šablonus attēlos, izceļot dažādas iezīmes, piemēram, robežas, stūrus un tekstūras. Konvolucionālo neironu tīklu slāņi veido hierarhiju, kur katrs nākamais slānis no iepriekšējā slāņa iegūst abstraktākas un sarežģītākas iezīmes. Konvolūcija ļauj arī samazināt parametru skaitu un modeļa skaitļošanas sarežģītību, vienlaikus saglabājot galvenās pazīmes.
2. Pakešu normalizācija: Tas normalizē katras datu paketes vērtības, centrējot un normalizējot, pamatojoties uz vidējo vērtību un dispersiju. Tas palīdz paātrināt apmācību un stabilizēt slīpumus.
3. LeakyReLU aktivizācijas funkcija: pēc pakešu normalizācijas tiek lietota LeakyReLU aktivizācijas funkcija, kas negatīvajām vērtībām pievieno nedaudz negatīvu slīpumu, lai izvairītos no "mirušā neirona" problēmas.
4. Sigmoidālā aktivizācijas funkcija: tiek izmantota diskriminatora izvadei, lai iegūtu varbūtību, ka ievades attēls ir reāls vai ģenerēts.

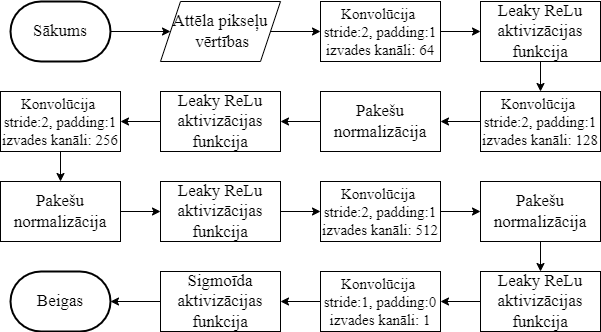


4.3. att. Diskriminatora modeļa shēma

Tālāk ir sniegts slāņa parametru apraksts.

* Diskriminators izmanto konvolūciju ar filtru (kodolu), kura izmērs ir 4x4, ar noteiktu soli (stride), kas vienāds ar 2, izņemot izvades slāni, kur tas ir 1. Tas izmanto arī padding, kas ir vienāds ar 1, izņemot izvades slānis, kur tas ir 0.
* Pakešu normalizācija tiek izmantota visos slāņos, izņemot ievades slāni un izvades slāni.
* LeakyReLu aktivizācijas funkcija tiek izmantota visos slāņos, izņemot izvades slāni, kurā tiek izmantota sigmoidālā aktivizācijas funkcija. LeakyReLu slīpums ir vienāds ar vertibu 0,2.
* Bias neirons netiek izmantots.
* Apvienošanas (pooling) slānis netiek izmantots.

Diskriminatora slāņu darbību attēlojumu var redzēt arī shematiskajā blokshēmā (skat. 4.4 att.).



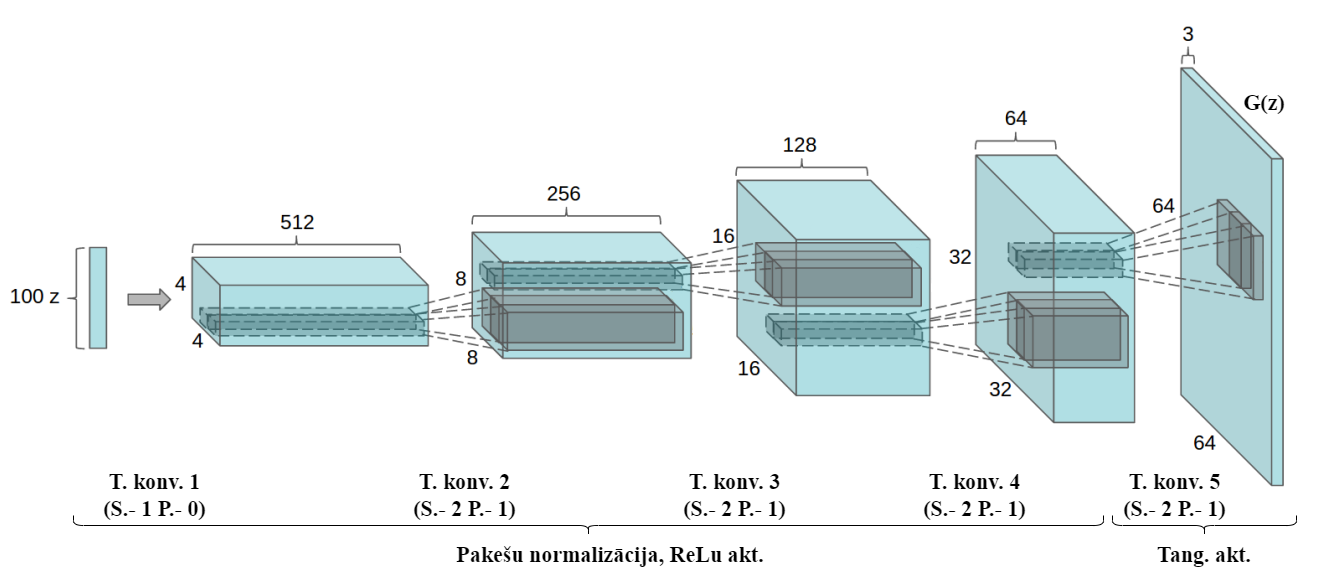
4.4. att. Diskriminatora shematiska blokshēma

***Ģenerators***

DCGAN ģenerators ir neironu tīkls, kas kā ievadi izmanto nejaušu vektoru (ko parasti dēvē par "troksni") un ģenerē sintezētu attēlu. Ģeneratora galvenais mērķis ir radīt attēlus, kas izskatās kā īsti un kurus diskriminētājs var pieņemt. Ģenerators izmanto transponēšanas konvolucionālos slāņus (transpose convolutional layers), kurus dažreiz kļūdaini sauc par dekonvolūcijas slāņiem, lai palielinātu dimensiju un pārveidotu troksni vēlamajā attēlā. DCGAN diskriminators sastāv no vairākiem transponētiem konvolūcijas slāņiem ar pakešu normalizāciju, ReLU aktivizācijas funkciju un tangenciālu aktivizācijas funkciju pēdējā ģeneratora slānī.

Ģeneratora slāņos izmantotās metodes (arī skat. 4.5 att.).

1. Transponētie konvolucionālie slāņi: ļauj palielināt ievades trokšņa dimensiju un pārveidot to par sarežģītākās iezīmēs. Tie ir pretēji parastajiem konvolūcijas slāņiem un ļauj ģeneratoram "izvērst" konvolūcijas operācijas.
2. Pakešu normalizācija: pakešu normalizācijas darbības princips ģeneratorā ir absolūti identisks darbības principam diskriminatorā.
3. ReLU aktivizācijas funkcija: galvenais uzdevums ir aktivizēt un pārraidīt pozitīvās vērtības bez izmaiņām un visas negatīvās vērtības aizstāt ar nulli.
4. Tangenciālās aktivizācijas funkcija: normalizē ģenerētā attēla pikseļu vērtības no -1 līdz 1.

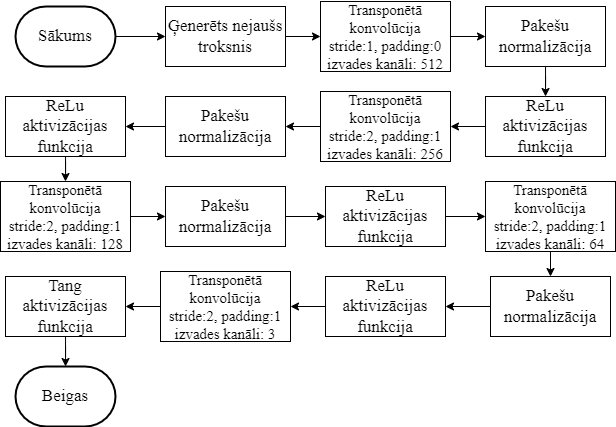
****

4.5. att. Ģeneratora modeļa shēma

Tālāk ir sniegts slāņa parametru apraksts.

* Ģenerators izmanto transponētu konvolūciju ar filtru (kodolu), kura izmērs ir 4x4, ar noteiktu soli (stride), kas vienāds ar 2, izņemot ievades slāni, kur tas ir 1. Tas izmanto arī padding, kas ir vienāds ar 1, izņemot ievades slānis, kur tas ir 0.
* Pakešu normalizācija tiek izmantota visos slāņos, izņemot izvades slāni.
* ReLu aktivizācijas funkcija tiek izmantota visos slāņos, izņemot izvades slāni, kurā tiek izmantota tangenciālā aktivizācijas funkcija.
* Bias neirons netiek izmantots.
* Apvienošanas (pooling) slānis netiek izmantots.

Ģeneratora slāņu darbību attēlojumu var redzēt arī shematiskajā blokshēmā (skat. 4.6 att.).

****

4.6. att. Ģeneratora shematiska blokshēma

### Sistēmas klases modelis

C++ programmēšanas valodai nav ērta rīka darbam ar daudzdimensiju masīviem, ir jārealizē klases, kas veic masīvu darbu.

1. Tensor3d: Šī klase realizē trīsdimensiju masīvu, ko izmanto attēlu vai citu trīsdimensiju datu attēlošanai.
2. Tensor4d: Šī klase realizē četrdimensiju masīvu, ko izmanto neironu tīkla svaru vai citu četrdimensiju datu glabāšanai.

Tālāk ir norādītas klases, kas tiek izmantotas paša neironu tīkla realizācijai.

1. nn (Neural Network): Šī ir abstrakta klase, kurā tiek glabāti parametri un metodes, kas saistītas ar neironu tīkliem. Tā nodrošina pamata saskarni neironu tīklu izveidei un apmācībai.
2. Layer: ir ielikta kompozīta klase, kas glabā parametrus un metodes katram neironu tīkla slānim. Tā var saturēt slāņus, piemēram, konvolūcijas slāņus, transponētus konvolūcijas slāņus, partijas normalizāciju un citus, kas nepieciešami DCGAN ģeneratoram un diskriminatoram.
3. Ģenerators: ģeneratora klase ir iedzimta klase, kas manto no nn un realizē DCGAN ģeneratoru.
4. Diskriminators: diskriminatora klase ir iedzimta klase, kas manto no nn un realizē DCGAN diskriminatoru.
5. Image Processing: klase satur metodes attēlu apstrādei, ko var izmantot datu pirmapstrādei, vizualizācijai vai ģeneratora rezultātu glabāšanai.

Klases diagrammas realizācija ir redzama 1. pielikumā.

## Funkcionālais sistēmas modelis

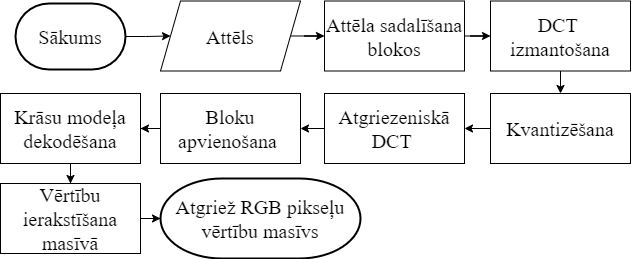
### Funkcionālais modelis

***Attēla dekodēšana***

Lai veiktu savu darbu, diskriminatoram ir jāievada attēla pikseļu vērtības, taču jūs nevarat vienkārši ņemt vērtības no faila, jo visas vērtības tiek kodētas, izmantojot grafisko datu glabāšanas algoritmu. Tāpēc, lai dekodētu vērtības, piemēram, JPEG formātā, jāveic šādi soļi.

1. Attēla sadalīšana blokos: dekodētājs sadala attēlu 8x8 pikseļu blokos.
2. Piemērot DCT: dekodētājs katram pikseļu blokam piemēro diskrēto kosīntransformāciju (DCT).
3. Kvantēšana: dekodētājs katram pikseļu blokam piemēro kvantizāciju.
4. Atgriezeniskā DCT: dekodētājs katram pikseļu blokam piemēro atgriezenisko diskrēto kosinusa transformāciju (IDCT).
5. Bloku apvienošana: dekodētājs apvieno visus pikseļu blokus atpakaļ sākotnējā attēlā.
6. Krāsu modeļa dekodēšana: dekodētājs piemēro faila galvenē norādīto krāsu modeli, lai pārveidotu attēlu tiešās RGB pikseļu vērtībās.

Attēla dekodēšana darbību attēlojumu var redzēt arī shematiskajā blokshēmā (skat. 4.7 att.).

****

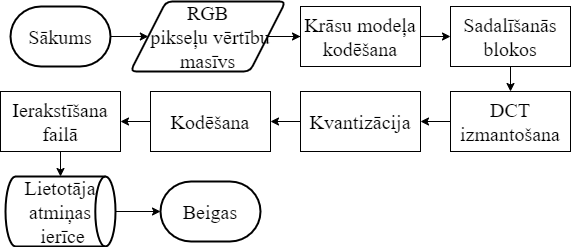
4.7. att. Attēlu dekodēšanas shematiskā blokshēma

***Attēla kodēšana***

Tāpat kā no attēla nevar uzreiz iegūt pikseļu vērtības, arī pikseļu vērtības nevajadzētu ierakstīt atpakaļ failā bez kodēšanas. Nekodētas pikseļu vērtības aizņem pārāk daudz vietas. Tāpēc, lai kodētu vērtības, piemēram, JPEG formātā, jāveic šādi soļi.

1. Attēla sagatavošana: attēlu konvertē YCbCr krāsu modeļa telpā, kurā tiek atdalītas spilgtuma (Y) un krāsu (Cb, Cr) sastāvdaļas.
2. Sadalīšana blokos: attēlu sadala 8x8 pikseļu blokos.
3. DCT piemērošana: diskrētā kosīntransformācija (DCT) tiek piemērota katram blokam, lai saspiestu informāciju katrā blokā.
4. Kvantēšana: DCT rezultāts tiek kvantizēts, lai vēl vairāk saspiestu datus un samazinātu faila lielumu.
5. Kodēšana: Kvantētie koeficienti tiek kodēti, izmantojot datu saspiešanas algoritmu - Hafmena algoritmu.
6. Faila izveide: JPEG fails tiek izveidots ar galveni, kurā ir informācija par attēla izmēru, krāsu modeļa parametriem un citiem parametriem.
7. Ierakstīšana failā: kodētie dati tiek ierakstīti JPEG failā.
8. Dekodēšana: Kad fails ir atvērts, JPEG dekodētājs dekodē datus, atjaunojot sākotnējo attēlu, izmantojot apgrieztus soļus: dekvantifikācija, apgrieztā DCT, bloku apvienošana, krāsu modeļa dekodēšana.

Attēla dekodēšana darbību attēlojumu var redzēt arī shematiskajā blokshēmā (skat. 4.7 att.).



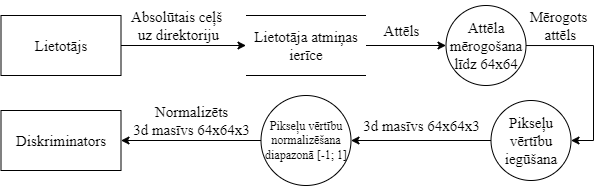
4.8. att. Attēlu kodēšanas shematiskā blokshēma

### Datu plūsmu modelis

***Datu kopa***

Lai apmācītu neironu tīklu, ir nepieciešami datu piemēri, no kuriem sistēma mācīsies. Tie tiek ņemti no lietotāja sniegtās datu kopas. Datu kopa ir direktorija, kurā tiek glabāti attēli. Lai tos izmantotu diskriminatora apmācībai, no attēla ir jāizņem pikseļu vērtības un jāieved tās normalizētā formā, lai nodrošinātu stabilu diskriminatora darbību (skat. 4.9 att.).

Pirmkārt, lietotājs norāda absolūto ceļu uz lietotāja krātuves sistēmas direktoriju, kurā tiek glabāta datu kopa. Pēc ievadītās absolūtā ceļa vērtības izmantošanas no direktorija tiek izgūts nejaušs attēls. Pēc tam attēla izmērs tiek mainīts, garuma un platuma vērtības kļūst par 64 (64x64). Pēc tam RGB pikseļu vērtības tiek iegūtas no attēla un ierakstītas trīsdimensiju masīvā, kur pirmie divi masīva indeksi ir atbildīgi par pikseļa pozīciju attēlā (X un Y), bet trešais indekss ir atbildīgs par pikseļa krāsu kanāliem. Nākamais solis ir normalizēt attēla vērtības no diapazona [0, 255] līdz vērtībām [-1,1], tas tiek darīts, jo lielas skaitliskās vērtības izraisa nestabilu diskriminatora darbību. Visbeidzot, trīsdimensiju masīvs ar normalizētām vērtībām tiek ievadīts diskriminatora ievadē.



4.9. att. Datu kopas datu plūsmas diagramma

***Ģenerators***

Ģenerators ir neironu tīkls, kas izmanto nejauši ģenerētu troksni, lai to pārveidotu attēlā. Lai to izdarītu, troksnis iziet cauri daudziem transponētas konvolūcijas slāņiem, kas paplašina troksni, palielinot tā lielumu , bet pēc tam samazinot kanālu skaitu. Daudzos slāņos troksnim tiek piemērota pakešu normalizācija, kas stabilizē ģeneratora darbību. Arī katrā slānī tiek izmantota aktivizācijas funkcija, kas arī normalizē vērtības un pārraida neirona aktivizāciju uz slāņa izvadi (skat. 4.10 att.).

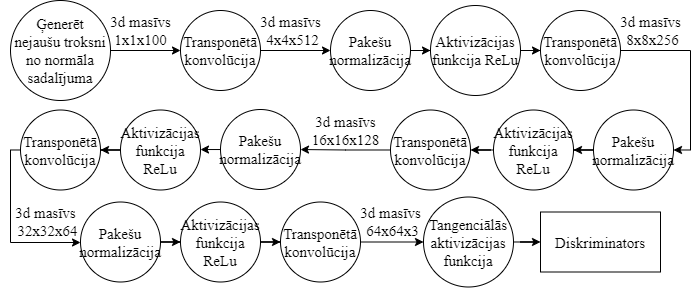
Ģeneratora pašā sākumā, izmantojot normālo sadalījumu, tiek ģenerēti 100 nejauši peldošā komata skaitļi. Šos 100 nejaušos skaitļus var uzskatīt par trīsdimensiju masīvu, kas satur 1 pikseli, kuram ir 100 kanāli (1x1x100). Šis ģenerētā trokšņa trīsdimensiju masīvs tiek ievadīts ģeneratora pirmajā slānī, kas veic transponētu konvolūcijas darbību ar parametriem, kur padding ir iestatīts uz 0, soli uz 1, izvades kanālu skaitu uz 512. Šī darbība maina pazīmju telpas izmēru, citiem vārdiem sakot, maina masīva lielumu līdz vērtībām 4x4x512. Pēc šīs darbības masīva vērtībām tiek piemērota pakešu normalizācija. Šī darbība aprēķina masīva vērtību vidējo un dispersiju, pēc tam centrē, mērogo un pēc tam reizina vērtības ar mērogošanas un nobīdes parametriem. Beidzot tiek piemērota ReLu aktivizācijas funkcija, kas normalizē masīva vērtības, visas negatīvās vērtības aizstājot ar 0.

Masīvs tiek nodots neironu tīkla otrajam slānim, kas veic transponētu konvolūcijas darbību ar parametriem, kur padding ir iestatīts uz 1, soli uz 2, izvades kanālu skaitu uz 256. Šī darbība maina masīva lielumu līdz vērtībām 8x8x256. Pēc šīs darbības masīva vērtībām tiek piemērota pakešu normalizācija un tiek piemērota ReLu aktivizācijas funkcija.

Masīvs tiek nodots neironu tīkla trešajam slānim, kas veic transponētu konvolūcijas darbību ar parametriem, kur padding ir iestatīts uz 1, soli uz 2, izvades kanālu skaitu uz 128. Šī darbība maina masīva lielumu līdz vērtībām 16x16x128. Pēc šīs darbības masīva vērtībām tiek piemērota pakešu normalizācija un tiek piemērota ReLu aktivizācijas funkcija.

Masīvs tiek nodots neironu tīkla ceturtajam slānim, kas veic transponētu konvolūcijas darbību ar parametriem, kur padding ir iestatīts uz 1, soli uz 2, izvades kanālu skaitu uz 256. Šī darbība maina masīva lielumu līdz vērtībām 32x32x64. Pēc šīs darbības masīva vērtībām tiek piemērota pakešu normalizācija un tiek piemērota ReLu.

Masīvs tiek nodots neironu tīkla piektajam slānim, kas veic transponētu konvolūcijas darbību ar parametriem, kur padding ir iestatīts uz 1, soli uz 2, izvades kanālu skaitu uz 3. Šī darbība maina masīva lielumu līdz galīgajai vērtībai 64x64x3. Pēc šīs darbības masīva vērtībām tiek piemērota partijas normalizācija un tiek piemērota ReLu. Tomēr šajā slānī netiek piemērota pakešu normalizācija, un ReLu aktivizācijas funkcijas vietā tiek izmantota tangenciālās aktivizācijas funkcija, kas normalizē masīva vērtības diapazonā [-1, 1].  
Pēc visām šīm darbībām tika iegūts trīsdimensiju masīvs, kas ir ģenerēts attēls, kuru var nodot diskriminatora ievadei apmācībai.

****

4.10. att. Ģeneratora datu plūsmas diagramma

***Diskriminators***

Darbības princips un diskriminatora izmantotie procesi ir ļoti līdzīgi ģeneratora darbībai. Tomēr tā vietā, lai paplašinātu ievades attēlu (masīvu), diskriminators saloka attēlu, lai iegūtu pazīmes, lai tālāk iegūtu varbūtības vērtības - attēls ir reāls vai ģenerēts. Katrā diskriminatora slānī tiek izmantota konvolūcijas darbība, kas samazina masīva izmēru, bet palielina kanālu skaitu attēla pazīmju iegūšanai. Tāpat kā ģeneratorā, diskriminants izmanto pakešu normalizāciju, bet tiek izmantotas citas aktivizācijas funkcijas (skat. 4.11 att.).

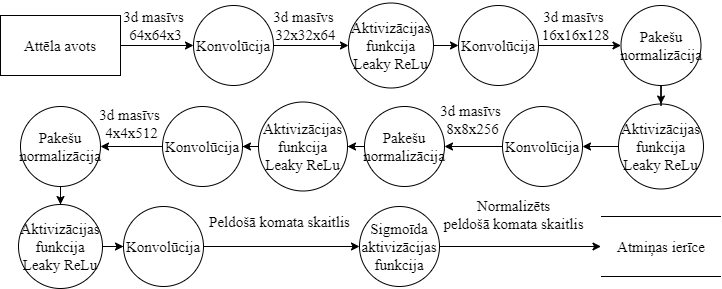
Sava darba sākumā ģenerators kā ievadi saņem vai nu ģenerētu attēlu no ģeneratora, vai reālu attēlu no datu kopas , kas abi ir trīsdimensiju masīvs ar izmēru 64x64x3. Šis masīvs tiek padots uz pirmo diskriminatora slāni, kas veic konvolūcijas darbību ar parametriem, kur padding ir iestatīts uz 1, soli uz 2, izvades kanālu skaitu uz 64. Šī darbība maina masīva lielumu līdz vērtībām 32x32x64. Pēc šīs darbības masīva vērtībām tiek piemērota Leaky ReLu aktivizācijas funkcija, kas normalizē masīva vērtības, visas negatīvās vērtības reizinot ar 0,2.

Masīvs tiek nodots neironu tīkla otrajam slānim, kas veic konvolūcijas darbību ar parametriem, kur padding ir iestatīts uz 1, soli uz 2, izvades kanālu skaitu uz 128. Šī darbība maina masīva lielumu līdz vērtībām 16x16x128. Pēc šīs darbības masīva vērtībām tiek piemērota pakešu normalizācija un tiek piemērota Leaky ReLu aktivizācijas funkcija.

Masīvs tiek nodots neironu tīkla trešajam slānim, kas veic konvolūcijas darbību ar parametriem, kur padding ir iestatīts uz 1, soli uz 2, izvades kanālu skaitu uz 256. Šī darbība maina masīva lielumu līdz vērtībām 8x8x256. Pēc šīs darbības masīva vērtībām tiek piemērota pakešu normalizācija un tiek piemērota Leaky ReLu aktivizācijas funkcija.

Masīvs tiek nodots neironu tīkla ceturtajam slānim, kas veic konvolūcijas darbību ar parametriem, kur padding ir iestatīts uz 1, soli uz 2, izvades kanālu skaitu uz 512. Šī darbība maina masīva lielumu līdz vērtībām 4x4x512. Pēc šīs darbības masīva vērtībām tiek piemērota pakešu normalizācija un tiek piemērota Leaky ReLu aktivizācijas funkcija.

Masīvs tiek nodots neironu tīkla piektajam slānim, kas veic konvolūcijas darbību ar parametriem, kur padding ir iestatīts uz 0, soli uz 1, izvades kanālu skaitu uz 1. Šī darbība maina masīva lielumu līdz vērtībām 1x1x1. Tas nozīmē, ka izejā mēs iegūstam peldošā komata skaitli. Pēc šīs darbības masīva vērtībām tiek piemērota sigmoīda aktivizācijas funkcija, kas normalizē vērtību diapazonā [0, 1]. Rezultātā mēs iegūstam vajadzīgo varbūtību, kas norāda, vai attēls ir vai nav patiess. Turklāt šī vērtība tiek pārsūtīta uz lietotāja operatīvo atmiņu, lai to izmantotu apmācībai un neironu tīkla svaru parametru atjaunināšanai.

****

4.11. att. Diskriminatora datu plūsmas diagramma

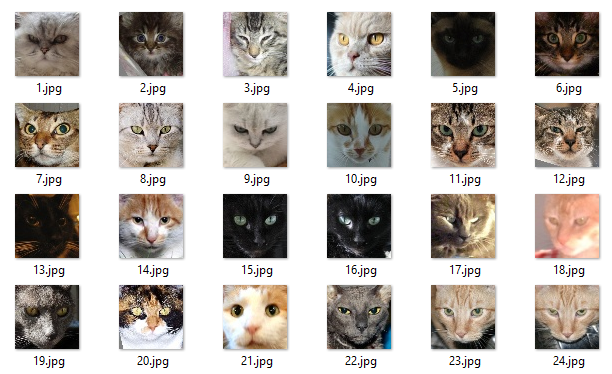
# DATU STRUKTŪRU APRAKSTS

***Datu kopa***

Datu kopa ir apmācību piemēru kolekcija, ko izmanto neironu tīkla apmācīšanai. DCGAN datu kopa sastāv no attēliem, kas ir galvenais modeļa ievades veids. Šie attēli var būt, piemēram, seju, ainavu vai objektu fotogrāfijas. Datu kopā var būt tūkstošiem vai pat vairāk attēlu atkarībā no pieejamajiem resursiem un risināmās problēmas. Labi sagatavota un daudzveidīga datu kopa ir būtiska veiksmīgai DCGAN apmācībai. Tam jābūt pietiekami reprezentatīvam, lai ģenerators varētu uzzināt sākotnējo datu īpašības un struktūru un radīt reālistiskus attēlus. Ir ārkārtīgi svarīgi, lai attēli atbilstu tikai vienai tēmai un tajos nesaturētu citus objektus, jo tādējādi tiek pārkāptas ar to pašu objektu saistīto atbilstošo pazīmju noteikšana.

Pirms datu kopas izveides ir jāatlasa tēma, kas pārstāv nepieciešamo mērķa apgabalu. Kad tēma ir definēta, ir jāsaliek piemēru kopa, kas atspoguļo mērķa apgabalu. Seju attēlu gadījumā tas var ietvert fotoattēlu vākšanu no tādiem avotiem kā sociālie mediji, aktieru sejas no “imdb” vai citiem avotiem, kur var atrast seju attēlus. Ir arī vietnes, kas nodrošina bezmaksas piekļuvi gatavām datu kopām neironu tīklu apmācībai, piemēram, “Hugginface” un “Kaggle”.

Datu kopa tiek saglabāta lietotāja atmiņas ierīcē vienā mapē. Failu nosaukumiem nav jābūt sakārtotiem saskaņā ar kādiem noteikumiem, bet parasti tie ir numurēti (1.jpg, 2.jpg ...). Attēla formātam jābūt JPEG, PNG vai BMP. Citi attēlu formāti netiks apstrādāti.

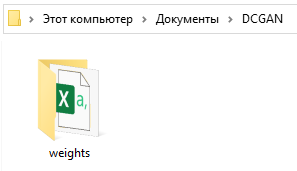


5.1. att. Datu kopas piemērs failu sistēmā

***Neironu tīkla parametri***

Pēc neironu tīkla apmācīšanas nepieciešams saglabāt parametrus (svarus), lai, ja lietotājs vēlas ģenerēt attēlu, izmantojot jau apmācīto tīklu, viņš to varētu atlasīt no saraksta. Neironu tīkla parametru saglabāšana tiek veikta, modeli serializējot failā. Katra slāņa svari tiek saglabāti kopā ar tīkla arhitektūru, ļaujot modeli un tā stāvokli atjaunot nākotnē.

Lai saglabātu svaru datus, lietotāja atmiņas ierīcē direktorijā "dokumenti" tiek izveidota mape "weights", kurā atrodas visi apmācītie neironu tīkli (skat. 5.2 att.). Mapes iekšpusē, saglabājot, izsaucot parametru saglabāšanas funkciju, tiek izveidots fails, kurā tiek ierakstītas neironu tīkla svaru vērtības. Faila nosaukums ir "nosaukums", ko lietotājs norādījis, apmācot tīklu, un CSV faila nosaukuma paplašinājums. CSV faila formāts tiek izmantots, lai ērti parādītu parametru vērtības tabulas formātā, ko var atvērt, izmantojot, piemēram, Microsoft Excel.



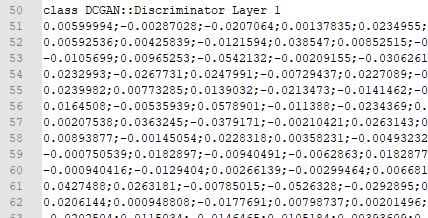
5.2. att. Saglabāto parametru piemērs failu sistēmā

Neironu tīkla svaru vērtības tiek ierakstītas failā, kas ir galvenais elements neironu tīkla darbībā un kas norāda, ar kādām vērtībām ievades vērtības jāreizina, lai iegūtu vēlamo rezultāts. Svari konvolucionālajos tīklos ir kodols, ko var attēlot kā 4-dimensiju masīvu, kur pirmie 2 indeksi norāda pikseļa pozīciju telpā (X, Y), trešais indekss apzīmē ienākošos kanālus, bet ceturtais indekss norāda izejošos kanālus.

Zemāk ir aprakstīts, kā parametri tiek ierakstīti failā.

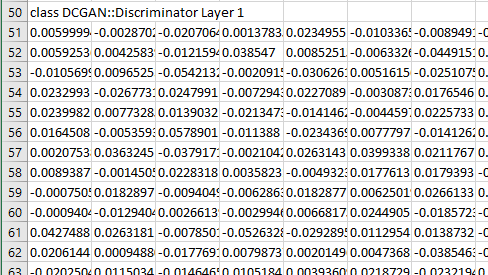
Pirmkārt tiek uzrakstīts apzīmējums - tiek norādīta klase, kurai pieder šādi parametri - diskriminators vai ģenerators, un pēc tam tiek norādīts tā slāņa numurs, kuram parametri pieder.

Sākas masīva vērtību ierakstīšanas darbība failā. Sākotnēji visas ceturtā masīva indeksa [0][0][0] vērtības tiek ierakstītas atsevišķā faila rindā. Pēc tam, kad šajā rindā ir ierakstītas visas izvades kanālu vērtības, ieraksts tiek pāriet uz jaunu rindu. Šajā jaunajā rindā ir visas masīva [0][0][1] ceturtā indeksa vērtības. Šo procesu atkārto, līdz ir ierakstītas visas trešā masīva indeksa vērtības. Kad visas masīva trešā indeksa vērtības ir aizpildītas, nākamajā rindā sāk ierakstīt visas masīva ceturtā indeksa vērtības [0][1][0]. Šo darbību atkārto visām iespējamām trešā un otrā indeksa kombinācijām, līdz visas masīva vērtības ir pilnībā ierakstītas failā. Tādējādi katrā faila rindā būs norādītā masīva ceturtā indeksa vērtības, un rakstīšana turpināsies, līdz visas masīva vērtības būs ierakstītas un saglabātas failā (skat. 5.3 att.).



5.3. att. Parametru ierakstīšanas veids failā

Starp katru ierakstīto masīva vērtību tiek rakstīts arī semikolu, kas CSV formātā nozīmē vērtības rakstīšanu jaunā tabulas šūnā (skat. 5.4 att.).



5.4. att. Saglabāto parametru tabulas skats

***Attēlu formāti***

BMP (Bitmap): ir Microsoft izstrādāts attēla formāts. Tas attēlo attēlu kā nesaspiestu bitu straumi, kur katrs pikselis ir attēlots ar savu krāsu un pozīciju uz audekla. BMP formāts atbalsta dažādus krāsu dziļumus, tostarp 1, 4, 8, 16, 24 un 32 bitus. BMP katrs pikselis tiek kodēts, izmantojot noteiktu bitu skaitu atkarībā no krāsu dziļuma. Piemēram, 24 bitu BMP formātā katrs pikselis ir kodēts ar 24 bitiem, kur katram no RGB (sarkans, zaļš, zils) kanāliem tiek piešķirti 8 biti. Tādējādi katram pikselim ir iespēja attēlot vairāk nekā 16 miljonus krāsu.

JPEG (Apvienotā fotogrāfiju ekspertu grupa): ir zudumu saspiešanas formāts, ko parasti izmanto fotogrāfisku attēlu glabāšanai un pārsūtīšanai. Tas ir optimizēts attēlu saspiešanai ar nepārtrauktiem toņiem un gradientiem. JPEG formātā tiek izmantota saspiešanas metode, kuras pamatā ir kosinusa transformācija (DCT) un kvantēšana. Kodējot JPEG, attēls tiek sadalīts 8x8 pikseļu blokos. Katram blokam tiek pielietota kosinusa transformācija, kas telpisko informāciju pārvērš frekvences informācijā. Pēc tam transformācijas koeficienti tiek kvantēti un kodēti, izmantojot mainīga garuma kodēšanu (VLC). Kvantizācija ļauj samazināt koeficientu vērtību precizitāti, ieviešot attēlā zudumus. Tas ļauj ievērojami saspiest JPEG failus, taču var tikt ieviesti daži saspiešanas artefakti.

PNG (Portable Network Graphics): ir attēla formāts, kas izstrādāts, lai aizstātu novecojušo GIF formātu. Tas izmanto bezzudumu saspiešanu un atbalsta indeksētus, gradētus un caurspīdīgus attēlus. PNG formātā datu saspiešanai tiek izmantots saspiešanas algoritms Deflate. PNG formātā katrs pikselis tiek kodēts, izmantojot noteiktu bitu skaitu atkarībā no krāsu dziļuma. PNG atbalsta dažādus krāsu dziļumus, tostarp 1, 2, 4, 8, 16 bitus katrā kanālā. Tā atbalsta arī alfa kanālu caurspīdīgumu, kas ļauj saglabāt attēlus ar caurspīdīgu fonu.

Galvenās atšķirības starp šiem formātiem ir saspiešanas veids un tā sekas. BMP nesaspiež attēlus un nodrošina augstu kvalitāti, bet aizņem vairāk vietas diskā. JPEG nodrošina labu saspiešanas pakāpi, taču var rasties kvalitātes zudums. PNG nodrošina bezzudumu saspiešanu un atbalsta caurspīdīgumu, taču var aizņemt vairāk vietas nekā JPEG.

# LIETOTĀJA CEĻVEDIS

## Sistēmas prasības aparatūrai un programmatūrai

***Kompilētā programma***

Ierīces vienkāršības dēļ programmai gandrīz nav nopietnu sistēmas prasību, kas nepieciešamas pareizai programmas darbībai, un kuru neievērošanas gadījumā varētu rasties fatālas kļūdas.

Viens no šiem nosacījumiem ir Windows operētājsistēmas klātbūtne, jo programma tika izstrādāta un pārbaudīta tieši šai platformai. Citu operētājsistēmu atbalsts var nebūt pieejams vai ierobežots. Turklāt, lai nodrošinātu nevainojamu programmas darbību, vēlams, lai būtu vismaz 200 MB brīvas operatīvās atmiņas. Tā kā programmatūra izmanto lielu datu apjomu, operatīvās atmiņas trūkums var izraisīt programmas darbības traucējumus.

Neskatoties uz to, ka minimālās prasības programmas veiktspējai ir nenozīmīgas, ir svarīgi atzīmēt, ka programma joprojām ir ļoti atkarīga no procesora apstrādes jaudas, lai panāktu ātru savu uzdevumu izpildi. Ātrai datu analīzei augstas veiktspējas procesors ir galvenais faktors, lai nodrošinātu programmas ātru un vienmērīgu darbību.

***Programmas pirmkods***

Lai kompilētu programmas kodu, nepieciešama lietojumprogramma Visual Studio. Microsoft Visual Studio ir Microsoft integrēta izstrādes vide dažādu lietojumprogrammu izveidei. Sistēmas prasībās jaunākajai Microsoft Visual Studio 2022 versijai var būt ietverti šādi līdzekļi un specifikācijas:

* Operētājsistēma - Windows 10 versija 1909 vai jaunāka.
* Procesors - Daudzkodolu procesors ar takts frekvenci vismaz 1,8 GHz. Lai nodrošinātu vislabāko veiktspēju, ieteicams izmantot procesoru ar frekvenci 2,8 GHz vai augstāku.
* Operatīvā atmiņa - ieteicamais operatīvās atmiņas apjoms ir 8 GB vai vairāk. Tomēr Visual Studio instalēšanai un pamatdarbībai var būt nepieciešami vismaz 2 GB RAM.
* Cietā diska vieta — minimālā nepieciešamā vieta cietajā diskā ir aptuveni 30 GB. Tomēr ir ieteicams, lai jums būtu vismaz 50 GB brīvas vietas, lai instalētu Visual Studio un papildu komponentus.
* Grafikas adapteris — lai strādātu ar Visual Studio, ieteicams izmantot grafikas adapteri, kas atbalsta DirectX 11 vai jaunāku versiju.
* Citas prasības — ir jāinstalē .NET Framework versija 4.5 vai jaunāka versija.

## Sistēmas instalācija un palaišana

***Kompilētā programma***

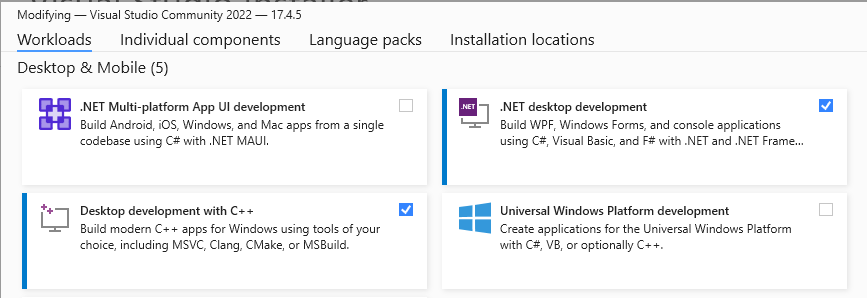
Lai instalētu jau kompilētu programmu, rīkojieties šādi.

1. Klonējiet GitHub repozitoriju, kurā ir apkopotā programma. Saite uz repozitoriju ir norādīta avotos.
2. Datorā atveriet komandu uzvedni vai termināli.
3. Pārejiet uz mapi, kurā vēlaties klonēt repozitoriju. Piemēram, varat izmantot komandu cd <ceļš\_uz\_mapi>, lai pārietu uz pareizo mapi.
4. Komandu uzvednē palaidiet git clone <saite\_uz\_repozitoriju>. Nospiediet Enter, lai izpildītu komandu.
5. Kad klonēšana ir pabeigta, programmas .exe failu atradīsit mapē "Release".

***Programmas pirmkods***

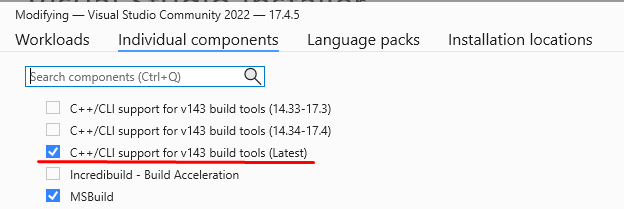
Nepieciešamās aparatūras instalēšanas un programmatūras koda konfigurēšanas process ietver šādus soļus.

1. Sāciet ar .NET Framework instalēšanu, jo .NET Framework ir nepieciešams, lai darbinātu Microsoft Visual Studio. Varat lejupielādēt un instalēt .NET Framework no Microsoft ražotāja oficiālās vietnes.
2. Pēc tam lejupielādējiet Visual Studio 2022 instalatoru no oficiālās Microsoft tīmekļa vietnes. Visual Studio 2022 instalētājs ļauj datorā instalēt Visual Studio izstrādes vidi.
3. Palaidiet Visual Studio 2022 instalēšanas programmu un izpildiet instalēšanas norādījumus. Jums tiks piedāvātas opcijas un iestatījumi, lai izvēlētos instalējamās sastāvdaļas.
4. Logā “Darba slodzes” (Workloads) pārliecinieties, ka ir atlasītas komponentes “Darbvirsmas izstrāde ar .NET” (Desktop development with .NET) un “Darbvirsmas izstrāde ar C++” (Desktop development with C++). Šīs sastāvdaļas ir nepieciešamas darbam ar .NET un C++ (skat. 6.1 att.).



6.1. att. Visual Studio uzstādītāja logs

1. Logā "Individuālie komponenti" (Individual components) izvēlieties komponentu "C++/CLI atbalsts" (C++/CLI support). Šī komponente ļaus izmantot “kopīgo valodas infrastruktūru” (Common Language Infrastructure) C++ izstrādē (skat. 6.2 att.).
2. Turpiniet instalēšanu, sekojot instalēšanas programmas norādījumiem, un pagaidiet, līdz instalēšana ir pabeigta.

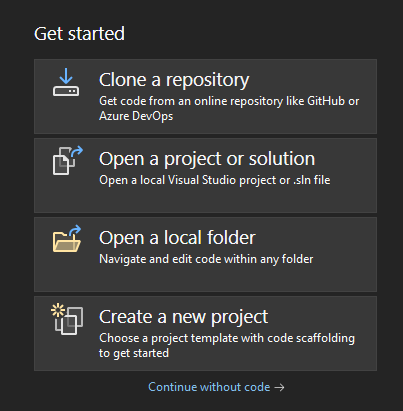


6.2. att. Visual Studio komponentu instalēšanas logs

Kad visi nepieciešamie elementi ir atlasīti, noklikšķiniet uz "instalēt", un instalētājs lejupielādēs un instalēs visus nepieciešamos komponentus, un Visual Studio instalēšana būs pabeigta.

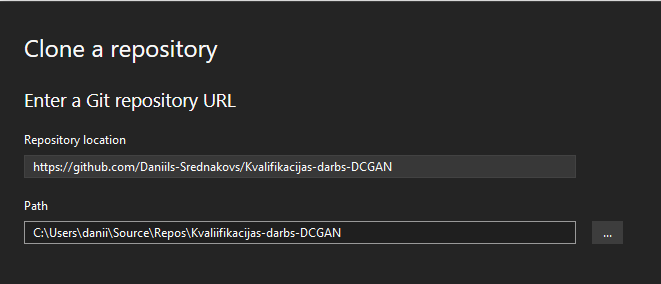
Lai instalētu projektu, izmantojot Git repozitoriju lietojumprogrammā Visual Studio 2022, veiciet šīs darbības:

1. Atveriet lietojumprogrammu Visual Studio 2022.
2. Galvenajā izvēlnē atlasiet "Klonēt repozitoriju" (Clone a repository) (skat. 6.3 att.).



6.3. att. Visual Studio galvenais logs

1. Atvērtajā logā "Klonēt repozitoriju " ievadiet hipersaiti uz Git repozitoriju, kuru vēlaties klonēt. Repozitorija hipersaitei ir jābūt pieejamai (skat. 6.4 att.).
2. Izvēlieties vietu, kur vēlaties saglabāt klonēto repozitoriju savā datorā.
3. Noklikšķiniet uz pogas "Klonēt" (Clone), lai sāktu klonēšanas procesu. Visual Studio lejupielādēs visus failus no repozitorija un izveidos projektu, pamatojoties uz datiem.
4. Kad repozitorija klons ir pabeigts, projekts "DCGAN" tiks instalēts jūsu datorā un automātiski atvērts programmā Visual Studio.

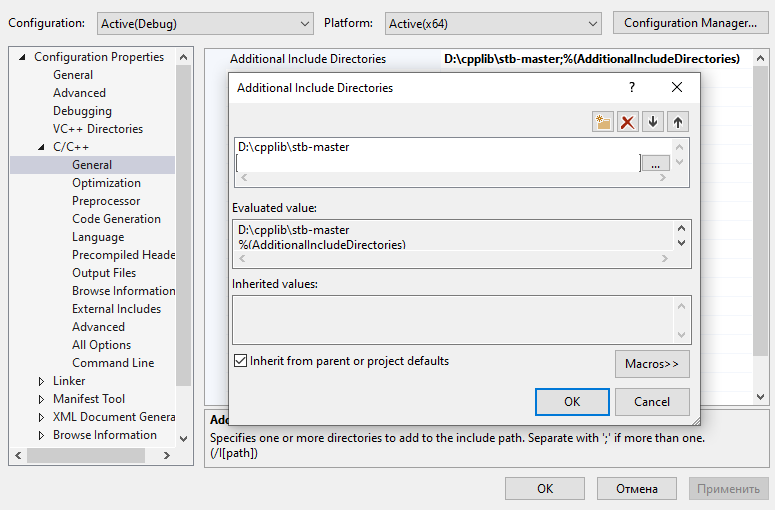


6.4. att. Visual Studio klonu repozitorija logs

Nākamais solis ir brīvi pieejamās STB bibliotēkas instalēšana. Šī bibliotēka nodrošina rīkus darbam ar attēliem C programmēšanas valodā. Lai instalētu bibliotēku, jādodas uz Github vietni (avotos ir norādīta saite uz repozitoriju) un jālejupielādē repozitoriju.

Lai informētu Visual Studio, ka projektā tiek izmantotas trešo pušu bibliotēkas, projekta iestatījumos ir jānorāda ceļš uz bibliotēkām. To var izdarīt, veicot šādas darbības (arī skat. 6.5 att.).

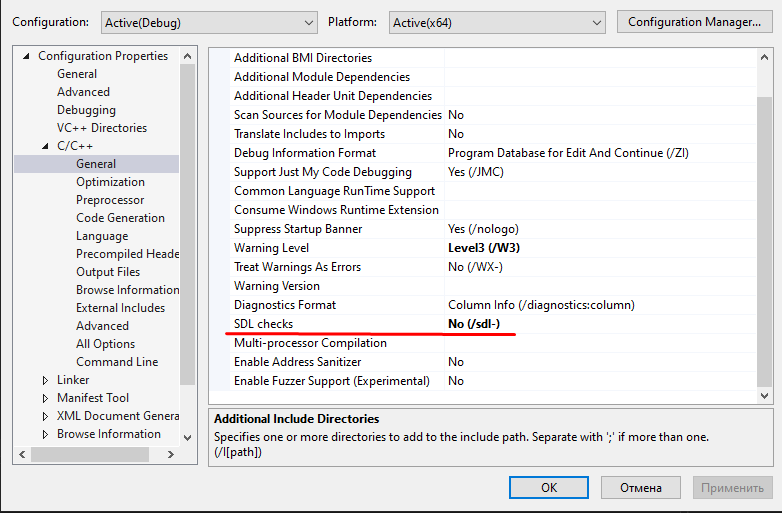
1. Ar peles labo pogu noklikšķiniet uz projekta “Risinājumu pārlūks” (Solution Explorer) un atlasiet “Īpašības” (Properties).
2. Projekta rekvizītu logā atlasiet sadaļu "C/C++".
3. Sadaļā "Vispārīgi" (General) atrodiet iestatījumu "Papildu iekļaušana direktorijās" (Additional Include Directories) un noklikšķiniet uz tā. Tiks atvērts logs iestatījumu rediģēšanai.
4. Iestatījumu rediģēšanas logā pievienojiet ceļu uz mapi ar trešās puses bibliotēkas galvenes failiem. Lai to izdarītu, ievades lauka labajā pusē noklikšķiniet uz pogas ar elipses formas ikonu (...) un atlasiet vajadzīgo mapi, kurā saglabājāt galvenes failus.
5. Noklikšķiniet uz pogas “Lietot” (Apply) un pēc tam uz “ok”, lai aizvērtu projekta rekvizītu logu.



6.5. att. Visual Studio projekta iestatījumu logs

Tā kā STB bibliotēkā tiek izmantotas novecojušas (nedrošas) C programmēšanas valodas metodes, ir nepieciešams izslēgt SDL pārbaudi Visual Studio iestatījumos. To var izdarīt šādi (arī skat. 6.6 att.).

1. Ar peles labo pogu noklikšķiniet uz projekta “Risinājumu pārlūks” (Solution Explorer) un atlasiet “Īpašības” (Properties).
2. Projekta rekvizītu logā atlasiet sadaļu "C/C++".
3. Sadaļā "Vispārīgi" (General) atrodiet iestatījumu “SDL pārbaudes” (SDL checks) un mainiet vērtību uz "Nē" (No).

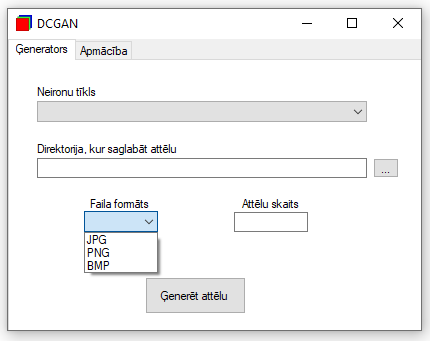


6.6. att. Visual Studio projekta iestatījumu logs

## Programmas apraksts

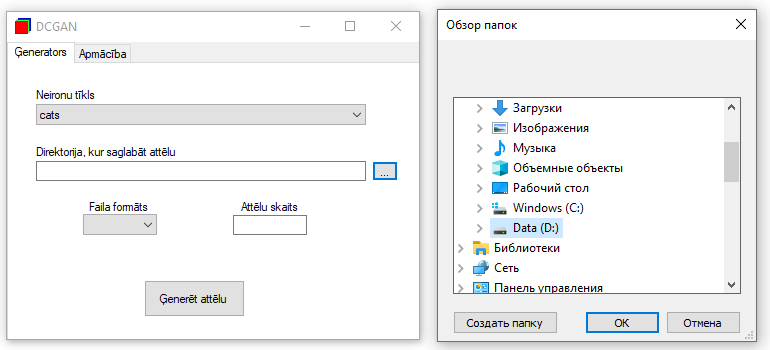
Kad tiek palaists programmas izpildāmais fails (DCGAN.exe), tiek atvērta loga lietojumprogramma, kas ir saskarne darbam ar neironu tīklu. Tūlīt pēc ieslēgšanas lietotājam tiks parādīts neliels logs. Augšējā panelī ir 2 cilnes, lai pārslēgtos starp attēla ģenerēšanu un neironu tīkla apmācību. Pirmajā cilnē, kas ir atbildīgs par attēlu ģenerēšanu, varat novērot attēlu ģenerēšanas pārvaldības elementus (skat. 6.7 att.).

1. Atlases izvēlne ar nolaižamo sarakstu, kurā lietotājam no pieejamajām opcijām jāizvēlas iepriekš apmācīts neironu tīkls.
2. Teksta lauks, kurā lietotājs norāda ceļu uz direktoriju, kurā viņš vēlas saglabāt ģenerētos attēlus.
3. Pa labi no teksta lauka atrodas poga, uz kuras noklikšķinot atveras mapes izvēles logs. Lietotājs var izvēlēties mapi, nevis manuāli ievadīt ceļu uz failu.
4. Zemāk kreisajā pusē ir atlases izvēlne ar nolaižamo sarakstu, kurā lietotājs izvēlas formātu, kādā tiks saglabāti ģenerētie attēli.
5. Pa labi no iepriekšējā elementa ir teksta lauks, kurā jāievada ģenerējamo attēlu skaits.
6. Poga, kas sāk attēla ģenerēšanas procesu, izmantojot visus lietotāja norādītos parametrus.



6.7. att. Attēlu izveides kontroles cilnes logs

Nospiežot pogu ar daudzpunktu, tiek atvērts mapju atlases logs (skat. 6.8 att.).



6.8. att. Mapes atlases logs

Otrajā cilnē, kas ir atbildīga par neironu tīkla apmācību, tiek atvērts lielāks logs ar ievades elementiem apmācības parametriem un informācijas logiem (arī skat. 6.9 att.).

Šajā cilnē ir šādi vienumi:

1. teksta lauks, kurā lietotājs ievada neironu tīkla nosaukumu;
2. teksta lauks, kurā lietotājs norāda direktoriju, kurā atrodas neironu tīkla apmācīšanai izmantotā datu kopa.

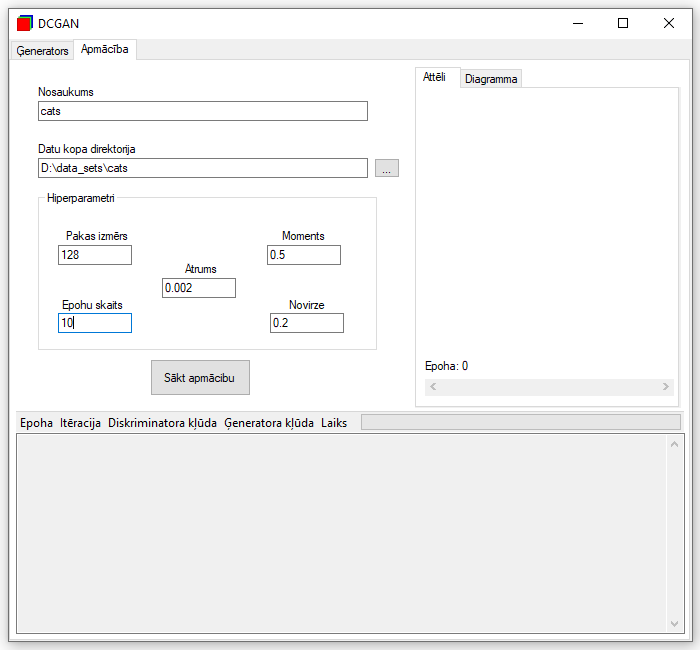
Tam seko neliels logs, kurā lietotājs norāda tīkla apmācības hiperparametrus:

1. mini-pakešu lielums, kas tiks izmantots, lai sadalītu datus mini paketā;
2. epohu skaits, ko neironu tīkls apmācīs;
3. gradienta nolaišanās mācību ātrums;
4. gradienta nolaišanās moments;
5. normālajam sadalījumam izmantotā novirze;
6. poga, kas sāk neironu tīkla mācību procesu, kā arī var pārtraukt mācīšanos, ja tā jau ir sākusies.

Loga apakšā ir liels teksta lauks, kas parāda informāciju par neironu tīkla apmācības procesu. Tas parāda pašreizējo laikmetu, iterāciju, diskriminatora kļūdu, ģeneratora kļūdu un darbības laiku. Ir arī neliela ielādes josla, kas parāda programmas norisi.

Augšējā labajā stūrī ir neliels logs ar divām cilnēm. Pirmā cilne ir attēlu ģenerēšanas provizorisko rezultātu logs, izmantojot ģeneratoru. Tas parāda tabulu ar 4 rindām un 4 kolonnām, kur katrā šūnā ir ģenerēts attēls. Izmantojot apakšējo slīdni šajā logā, varat skatīt izmaiņas attēlos atkarībā no apmācības laikmeta.

Otrā cilne ir grafisks attēlojums diagrammas veidā, kas ļauj izsekot ģeneratora kļūdas izmaiņām dažādos apmācības periodos. Diagramma tiek uzzīmēta, pamatojoties uz saņemtajām ģeneratora kļūdu vērtībām, kuras tiek parādītas uz vertikālās ass, un apmācības laikmetu skaitļi tiek parādīti uz horizontālās ass. Tādējādi lietotājs var vizuāli analizēt, kā mainās ģeneratora kļūda neironu tīkla apmācības laikā.



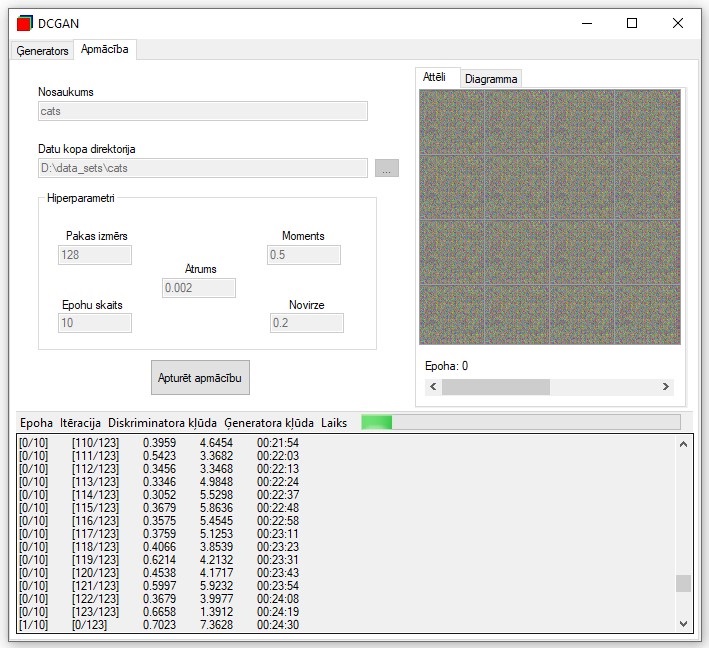
6.9. att. Neironu tīklu apmācības vadības cilnes logs

## Testa piemērs

Testa piemērs parāda un apraksta programmas procesu, ieskaitot apmācību un ģenerēšanu.

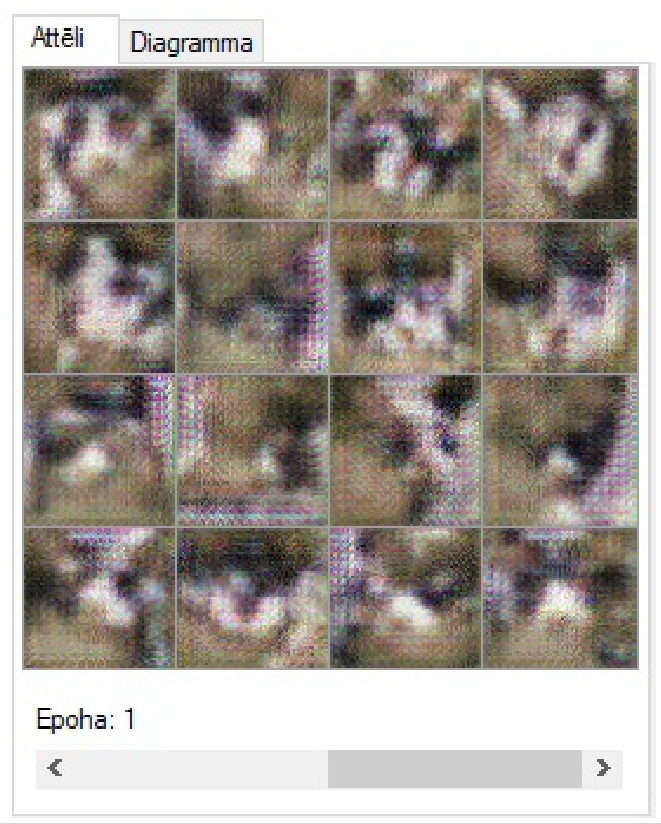
Programmas sākotnējā posmā, kad vēl nebija apmācītu neironu tīklu, lietotājam jādodas uz apmācības cilni (skat. 6.10 att.). Šajā cilnē nepieciešams ievadīt visus apmācībai parametrus un nospiest pogu, lai sāktu apmācības procesu. Pēc mācīšanās sākuma pogas nospiešanas visas ievadītās parametru vērtības kļūst nepieejamas izmaiņām, un mācīšanās sākuma poga mainās uz mācīšanās pārtraukšanas pogu. Lietotājs vairs nevar mainīt mācību parametrus un var tikai pārtraukt pašreizējo mācību procesu, ja nepieciešams.

Šajā laikā augšējā labajā logā, kas iepriekš bija bloķēts, tiek parādīta grafiska informācija par mācību procesu. Arī apakšējā teksta laukā sāk parādīties informācija par mācību procesu. Šeit lietotājs var skatīt datus, piemēram, pašreizējo laikmetu, pašreizējo iterāciju, diskriminatora un ģeneratora kļūdu vērtības un darbības laikus. Tādējādi lietotājs var uzraudzīt neironu tīkla mācību procesu.



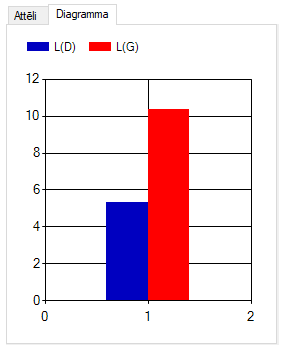
6.10. att. Neironu tīkla apmācības vadības cilnes logs apmācības laikā

Pārvietojot slīdni, lietotājs var sekot līdzi ģeneratora rezultātu izmaiņām atkarībā no laikmeta (skat. 6.11 att.).



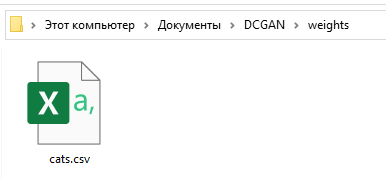
6.11. att. "Attēli" cilne apmācības laikā

Diagramma tiek parādīta loga augšējā labajā stūrī otrajā cilnē. Horizontālā koordināta ir atbildīga par laikmeta vērtību, vertikālā līnija - par zaudējumu funkcijas vērtībām. Sarkanā krāsa parāda ģeneratora zuduma funkciju, zilā krāsa parāda diskriminatora zuduma funkciju(skat. 6.12 att.).



6.12. att. Zaudējumu funkcijas diagramma

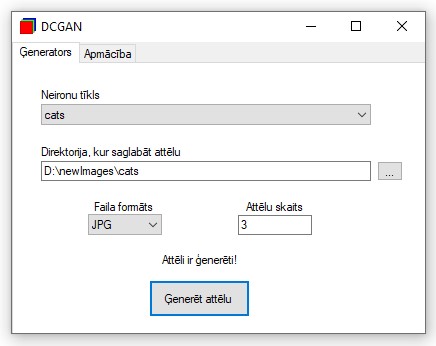
Pēc apmācības pabeigšanas neironu tīkla parametri tiek saglabāti dokumentu direktorijā failā ar nosaukumu, ko lietotājs piešķīris neironu tīklam (skat. 6.13 att.).



6.13. att. Saglabāts fails ar parametriem

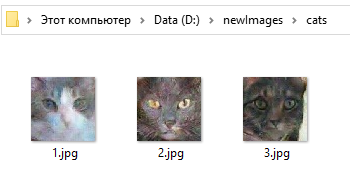
Pēc apmācītā neironu tīkla saņemšanas lietotājs var pāriet uz cilni darbam ar ģeneratoru. Šajā cilnē pirmais solis ir atlasīt neironu tīklu, kuru vēlaties izmantot attēlu ģenerēšanai. Pēc tam norādiet ceļu, kur tiks saglabāti ģenerētie attēli, atlasiet attēla formātu un norādiet nepieciešamo attēlu skaitu.

Pēc visu nepieciešamo parametru ievadīšanas noklikšķiniet uz pogas, kas sāks attēla ģenerēšanas procesu (arī skat. 6.14 att.).



6.14. att. Parametru ievadīšana un attēlu ģenerēšana

Kad ģenerēšanas process ir pabeigts, ekrānā tiks parādīta īsziņa par ģenerēšanas pabeigšanu, un ģenerētie attēli tiks saglabāti norādītajā direktorijā (skat. 6.15 att.).



6.15. att. Ģenerēti attēli failu sistēmā

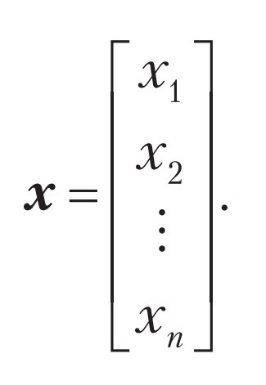
# DETALIZĒTS NEIRONU TĪKLA DARBĪBAS PRINCIPA APRAKSTS

## Lineārie neironu tīkli

***Matemātikas vienības***

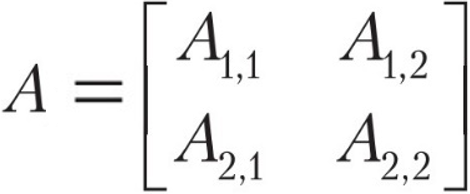
Lineārā algebra ir matemātiska joma, kas atrod plašu pielietojumu zinātnē un tehnoloģijā. Lai gan datorzinātnieki, iespējams, ar to bieži nesaskaras, jo tas attiecas uz nepārtrauktām, nevis diskrētām vērtībām, laba izpratne par lineāro algebru ir būtiska, lai izprastu daudzus mašīnmācīšanās algoritmus, jo īpaši tos, kas iesaistīti dziļajā mācībā. Tālāk ir aprakstītas matemātiskās vienības, kas tiek izmantotas mašīnmācīšanā.

* Skalārs - Skalāri attiecas uz atsevišķām skaitliskām vērtībām.
* Vektors - vektors ir skaitļu kolekcija, kas sakārtota noteiktā secībā, un katrs vektora elements tiek identificēts ar indeksu. Programmēšanā vektoru var attēlot kā masīvu (skat. 7.1 att.).



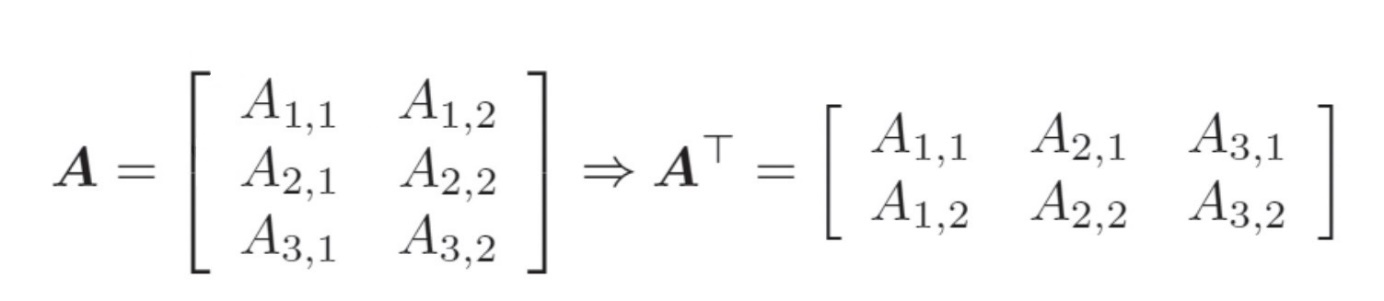
7.1. att. Vektors

* Matrica - matrica ir skaitļu kolekcija, kur katru elementu var identificēt, izmantojot divus indeksus, nevis vienu. Programmēšanā matricu var attēlot kā divdimensiju masīvu (skat. 7.2 att.).



7.2. att. Matrica

* Tenzors - Tenzors ir masīvs, kura dimensija ir lielāka par diviem. Izmanto kā attēla attēlojumu, kur attēla augstums, platums un kanāli tiek attēloti kā masīva izmēri.
* Matricas rotācija - matemātiska darbība, kas ietver matricas pagriešanu noteiktā leņķī ap tās centrālo punktu. Matricas elementi tiek pārvietoti uz jaunām pozīcijām, pamatojoties uz rotācijas leņķi un sākotnējo matricas elementu koordinātām.
* Transponēšana - attiecas uz darbību, kas pārveido matricu, apvēršot tās rindas un kolonnas (skat. 7.3 att.). Matricas X transponēšanas rezultātu apzīmē ar X⏉.



7.3. att. Transponēšanas operācija

***Neironi, svari un slāņi***

Neironu tīkla darbības pamatprincips ir salīdzinoši vienkāršs. Neironu tīkla pamatelements ir neirons, kas ņem ievadi no citiem neironiem vai no no ārpuses un rada izejas signālu. Neironi ir savienoti viens ar otru slāņos, un katrs slānis veic noteiktu ievades datu apstrādes veidu (skat. 7.4 att.). Parasti neirons darbojas ar skaitliskām vērtībām diapazonā [0,1] vai [-1,1]

Neironu tīkla algoritmā savienojumus starp neironiem sauc par sinapsēm jeb svariem. Svari ir saziņas punkti starp neironiem, un tie ļauj pārraidīt informāciju no viena neirona uz citu. Tie nosaka, cik spēcīgi ieejas signāls ietekmē tā neirona aktivāciju, ar kuru tas ir savienots, vai, vienkārši sakot, ar svaru palīdzību neironu tīklā, ievades informācija mainās, pārejot no viena neirona uz otru. Kad neirons saņem ievadi, tas reizina to ar savu svaru un nodod rezultātu aktivizēšanas funkcijai. Svari ļauj tīklam pielāgot intensitāti un mijiedarbību starp neironiem, lai sasniegtu vēlamo rezultātu.

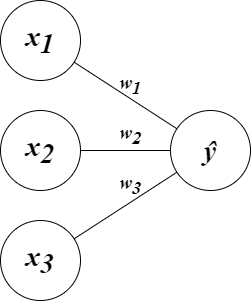
Matemātiski to var realizēt ar šādu formulu:

kur w - neironu svars;

x – neirons;

ŷ - izejas neirons;

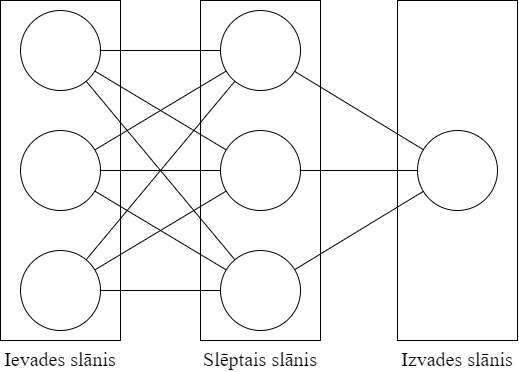
d - ar neironiem saistītā svara un neirona indekss.



7.4. att. Neironi, kas savienoti ar svariem

Neironu tīkls sastāv no trim galvenajiem slāņu veidiem: ievades slāņiem, slēptajiem slāņiem un izejas slāņiem (skat. 7.5 att.). Ievades slāņi saņem neapstrādātus datus un nodod tos pirmajam slēptajam slānim, kas veic matemātisku darbību kopumu ar ievades datiem. Pēc tam pirmā slēptā slāņa izvade tiek nodota nākamajam slēptajam slānim un tā tālāk, līdz tiek sasniegts izejas slānis.

“Pavairošana uz priekšu” (forward propagation) ir process, kurā neironu tīkli apstrādā un pārraida ievades datus caur virkni slāņu, lai iegūtu izvadi. Tas ir pirmais solis neironu tīkla mācību procesā un ir galvenā sastāvdaļa tīkla darbībā. Slēptajos slāņos katrs neirons ņem iepriekšējā slāņa ieeju svērto summu un izmanto aktivizācijas funkciju, lai iegūtu izejas vērtību. Pēc tam slēptā slāņa neironu izeja tiek nodota nākamajam slānim.



7.5. att. Neironu tīklu slāņi

Neironu tīklā ievades vērtību reizināšanu ar neirona svariem var attēlot kā matricas skalārā reizinājumu. Katram neironu tīkla neironam ir svaru kopums, ko izmanto, lai reizinātu ievades vērtības, pirms tās tiek nodotas aktivizācijas funkcijai. Šos svarus var attēlot kā vektoru, kur katrs vektora elements atbilst svaram (arī skat. 7.6 att.).

Līdzīgi ievades vērtības var attēlot arī kā vektoru, kur katrs vektora elements atbilst ievades vērtībai. Pēc tam ievades vērtību un svaru reizināšanu var attēlot kā šo vektoru skalāru reizinājumu, kur katrs ievades vektora elements tiek reizināts ar atbilstošo svaru svara vektorā, un rezultāti tiek summēti, lai iegūtu skalāru vērtību.

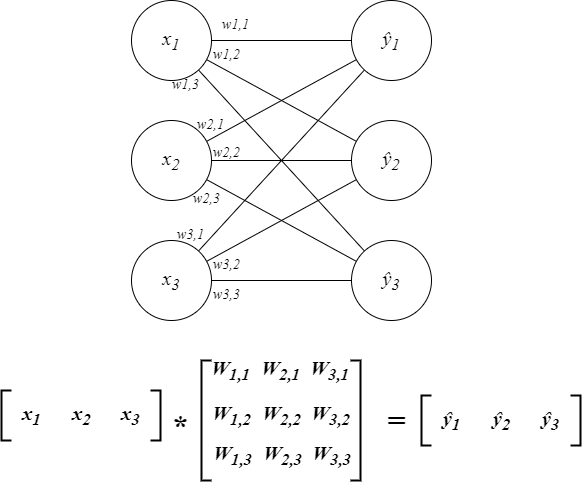
Matemātiski to var realizēt ar šādu formulu:

kur - transponētā neironu svara matrica;

x – neironu vektors;

n – ieejas neironu skaits;

ŷ - izejas neironu vektors.



7.6. att. Matricas reizināšana

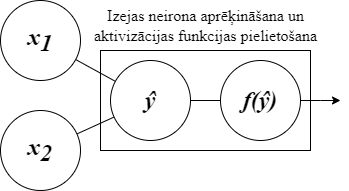
Kopumā neironu tīkla darbības pamatprincips ietver ievades datu apstrādi, izmantojot virkni savstarpēji saistītu neironu slāņu. Lai gan katras neironu tīkla arhitektūras specifika var atšķirties, šis vispārējais process veido pamatu visu veidu neironu tīkliem.

***Aktivizācijas funkcija (activation function)***

Aktivizācijas funkcija nosaka, kad un cik lielā mērā neirons tiek aktivizēts. Tas ņem ievadi no iepriekšējiem slāņiem vai neironiem un ģenerē izvadi, kas tiek nodota nākamajam slānim vai neironam (skat. 7.7 att.). Tā kā neirons darbojas tikai noteiktā vērtību diapazonā, piemēram, [0,1], aktivizācijas funkcija veic arī uzdevumu normalizēt ienākošo vērtību reizinājumu vajadzīgajā diapazonā. Aktivizācijas funkcija vienmēr tiek piemērota neironam, neskaitot ievades slāņa neironus.

Aktivizācijas funkcijas izvēle var būtiski ietekmēt neironu tīkla darbību. Ir vairāki dažādi aktivizācijas funkciju veidi, piemēram:

* tangenciālas aktivizācijas funkcija, kas normalizē vērtības diapazonā [-1,1];
* sigmoidālā aktivizācijas funkcija , kas normalizē vērtības diapazonā [-1,1].



7.7. att. Aktivizācijas funkcija neironu tīklā

Aktivizācijas funkcijas īstenošanas piemēru var redzēt apakšnodaļas "Detalizēts DCGAN metožu darbības apraksts " punktā "Citas kvalifikācijas darbā izmantotās metodes".

***Zaudējumu funkcija (loss function)***

Neironu tīklā zaudējumu funkcija, kas pazīstama arī kā izmaksu funkcija, ir matemātiska funkcija, kas mēra starpību starp prognozēto izvadi un faktisko izvadi. Zaudējumu funkcija tiek izmantota apmācības procesa laikā, lai pielāgotu neironu tīkla svarus un novirzes, lai samazinātu atšķirību starp prognozētajiem un faktiskajiem rezultātiem.

Neironu tīkla apmācības mērķis ir samazināt zaudējumu funkciju. Zaudējumu funkcijas izvēle ir atkarīga no konkrētās risināmās problēmas. Piemēram, binārās klasifikācijas problēmai var izmantot šķērsentropijas zuduma (binary cross-entropy) funkciju, savukārt regresijas problēmai piemērotāka var būt vidējās kvadrātiskās kļūdas (MSE) zuduma funkcija.

Zaudējumu funkcija īstenošanas piemēru var redzēt apakšnodaļas "Detalizēts DCGAN metožu darbības apraksts " punktā "Citas kvalifikācijas darbā izmantotās metodes".

***Atpakaļpavairošana (backpropagation)***

Atpakaļpavairošana ir algoritms, kas izmanto ķēdes likumu jeb salikto funkciju diferenciācijas noteikumu, lai aprēķinātu zaudējumu funkcijas gradientus attiecībā pret tīkla parametriem. Aprēķinot šos gradientus, mēs varam atjaunināt svarus apmācības laikā, lai samazinātu zaudējumu funkciju. Galvenais backpropagation mērķis ir atrast visus zaudējuma funkcijas gradientus attiecībā pret svariem

Pēc priekšējās izplatīšanās fāzes atpakaļpavairošana algoritms sākas, aprēķinot zuduma funkcijas gradientu attiecībā pret tīkla pēdējā slāņa izvadi. Pēc tam zudumu funkcijas gradientu attiecībā pret galīgā slāņa izvadi izmanto, lai aprēķinātu zuduma funkcijas gradientus attiecībā pret iepriekšējā slāņa svariem, izmantojot to pašu ķēdes likumu.

Pēc tam atpakaļpavairošana algoritms pāriet uz tīkla nākamo slāni un aprēķina zuduma funkcijas gradientus attiecībā pret katra neirona izvadi šajā slānī. Šie gradienti atkal tiek aprēķināti, izmantojot ķēdes likumu un gradientus no nākamā slāņa tīklā.

Šis process tiek atkārtots katram tīkla slānim, gradientus pavairojot no izvades slāņa uz ievades slāni. Visbeidzot, zaudējumu funkcijas gradienti attiecībā uz tīkla svariem un novirzēm tiek izmantoti, lai atjauninātu šo parametru vērtības, izmantojot optimizācijas algoritmu.

Atpakaļpavairošanas darbības shematisku blokshēmu var redzēt 2. pielikumā

***Datu kopa, iterācijas un epohas***

Pirms turpināt izskaidrot neironu tīkla apmācības principu, ir nepieciešams izskaidrot šādus terminus.

Datu kopa — apmācību piemēru kolekcija, kas tiek izmantota neironu tīkla apmācīšanai. Katrs apmācības piemērs sastāv no ievades (piemēram, attēli tiek izmantoti, lai apmācītu DCGAN) un atbilstošas ​​izvades vērtības (piemēram, DCGAN apmācībai šī ir vērtība, kas nosaka, vai attēls ir ģenerēts neironu tīkls vai nav). Datu kopā var būt tūkstošiem vai pat miljoniem piemēru, kas atspoguļo dažādas ievades situācijas vai objektus, uz kuriem modelis tiks apmācīts.

Iterācija — iterācija ir cikls, kurā modelis tiek apmācīts par katru apmācības piemēru un atjaunina parametrus, lai uzlabotu tā prognozēšanas spējas. Tādā veidā jūs varat iedomāties iterācijas skaitītāju, kas tiks atjaunināts katru reizi, kad neironu tīkls atjaunina savus svarus.

Epoha (laikmets) — nozīmē vienu reizi iziet cauri visai apmācības datu kopai. Lai gan iterācija aptver katra apmācības piemēra apstrādi, laikmets aptver vairākas iterācijas, kurās visi apmācības piemēri tiek izmantoti, lai vienu reizi apmācītu modeli. Tādā veidā jūs varat iedomāties epoha skaitītāju, kas tiks atjaunināts katru reizi, kad neironu tīkls savā apmācībā izmantos pilnīgi visus piemērus no datu kopas.

***Gradienta nolaišanās (Gradient descent)***

Gradienta nolaišanās, vienkāršāk sakot - ir pirmās kārtas optimizācijas algoritms, kas izmanto zudumu funkcijas gradientu attiecībā pret parametriem, lai noteiktu parametru atjauninājumu virzienu un lielumu.

Tas ir balstīts uz zaudējumu funkcijas gradienta aprēķināšanu, kas norāda ātrākās kļūdu samazināšanas virzienu. Algoritms secīgi atjaunina modeļa parametrus, kas pārvietojas pretējā gradienta virzienā, lai tuvotos optimālajām parametru vērtībām, kas nodrošina vislabāko prognozēšanas kvalitāti. Gradienta nolaišanās process turpinās, līdz ir pabeigts iepriekš noteikts epohu skaits.

Gradienta nolaišanās formulu var izteikt šādi:

kur – atjaunināts parametrs (svars);

– mācīšanās ātrums;

– svara zaudēšanas funkcijas gradients;

– moments;

– iepriekšējās svara izmaiņas;

Neironu tīkla apmācīšanai var izmantot vairākas dažādas gradienta nolaišanās metodes.

Stohastiskā gradienta nolaišanās (SGD - stochastic gradient descent) — optimizācijas paņēmiens, kas atjaunina modeļa svarus, pamatojoties uz zudumu funkcijas gradientu, kas aprēķināts vienā nejauši izvēlētā apmācības piemērā no visas datu kopas.

Mini-pakešu gradienta nolaišanās (Mini-batch GD) — optimizācijas paņēmiens, kas atjaunina modeļa svarus, pamatojoties uz gradientu, kas aprēķināts nelielai apmācības piemēru apakškopai no visas datu kopas, ko sauc par mini pakešu.

Izmantojot mini partijas, datu kopa tiek sadalīta mazās apakškopās, kas pazīstamas kā mini paketes. Katrā mini iepakojumā ir vairākas apmācības. Pēc tam katrai mini partijai piemērus apstrādā neironu tīkls saskaņā ar iepriekš aprakstīto darbplūsmu. Tomēr tā vietā, lai nekavējoties atjauninātu neironu tīkla svarus, visi iegūtie gradienti tiek saglabāti apmācības procesā. Pēc tam neironu tīkls iet cauri nākamajiem minipartijas datu paraugiem. Šis process tiek atkārtots katram mini partijas paraugam. Pēc visu minipartijas piemēru apstrādes, izmantojot saglabātos gradientus, aprēķina gradientu vidējo vērtību katram svaram. Šīs gradientu vidējās vērtības pēc tam tiek izmantotas, lai atjauninātu neironu tīkla svarus. Tādējādi minipakešu metode efektīvi izmanto skaitļošanas resursus, atjauninot svarus pēc katras minipartijas apstrādes, nevis pēc katra atsevišķa datu parauga.

***Hiperparametri***

Neironu tīkla hiperparametri ir parametri, kas tiek iestatīti pirms neironu tīkla apmācības un netiek apgūti apmācības laikā. Šie hiperparametri regulē neironu tīkla arhitektūru, tā apmācības procedūru un veiktspēju. Hiperparametrus var iestatīt uz dažādām vērtībām atkarībā no risināmās problēmas un izmantotā neironu tīkla veida.

Šeit ir daži neironu tīkla hiperparametri:

* Slēpto slāņu skaits: Šis hiperparametrs nosaka neironu tīkla dziļumu, t.i., slāņu skaitu starp ieejas un izejas slāņiem. Neironu tīklam var būt nulle vai vairāk slēptu slāņu.
* Neironu skaits uz slēpto slāni: Šis hiperparametrs nosaka neironu tīkla platumu, t.i., neironu skaitu katrā slēptajā slānī. Neironu skaita palielināšana uz slēpto slāni var padarīt neironu tīklu izteiksmīgāku, bet var arī palielināt pārmērīgas piemērotības risku.
* Aktivizācijas funkcija: Šis hiperparametrs nosaka aktivācijas funkciju, ko piemēro katra neirona izejai.
* Mācīšanās ātrums (learning rate): Šis hiperparametrs nosaka soļa lielumu, kas veikts gradienta nolaišanās optimizācijas algoritma laikā, ko izmanto, lai apmācītu neironu tīklu. Augstāks mācīšanās līmenis var nodrošināt ātrāku konverģenci, bet var arī izraisīt to, ka algoritms pārsniedz optimālo risinājumu.
* Moments (momentum) - gradienta nolaišanās moments ir metode, ko izmanto, lai paātrinātu konverģenci un izvairītos no iestrēgšanas lokālos minimumos, apmācot mašīnmācīšanās modeļus. Tas var palīdzēt pārvarēt tādas problēmas kā lēna konverģence vai svārstības apmācības procesā.
* Pakešu lielums (batch size): Šis hiperparametrs nosaka apmācības piemēru skaitu, kas tiek izmantoti, lai atjauninātu neironu tīkla svarus katras optimizācijas algoritma iterācijas laikā. Lielāks partijas lielums var nodrošināt stabilāku optimizācijas procesu, taču tam var būt nepieciešama arī lielāka atmiņa un aprēķini.

Hiperparametru izvēle parasti tiek veikta, izmantojot izmēģinājumu un kļūdu procesu, kurā tiek izmēģinātas dažādas hiperparametru kombinācijas un neironu tīkla veiktspēja tiek novērtēta validācijas kopā.

## Detalizēts DCGAN metožu darbības apraksts

### Konvolucionālie neironu tīkli

***Konvolūcija (convolution)***

Konvolūcijas (convolution) operācija ir matemātiska operācija, kas apvieno divas funkcijas, lai radītu trešo funkciju, kas izsaka, kā viena no sākotnējām funkcijām tiek modificēta ar otru. Neironu tīklu kontekstā konvolūcijas darbību izmanto, lai iegūtu funkcijas no ievades datiem, piemēram, attēliem.

Konvolucionālā neironu tīklā (CNN – Convolutional Neural Network) konvolūcijas operācija tiek piemērota ievades datiem, bīdot nelielu matricu, ko sauc par kodolu vai filtru virs ievades datiem. Kodols satur apgūstamu parametru kopumu, kas tiek optimizēts apmācības procesa laikā. Kodola mērķis ir iegūt konkrētu funkciju no ievades datiem.

Konvolūcijas operācijas laikā kodols tiek lietots katrai ievades datu daļai, un starp kodolu un ievades datiem, kas pārklājas, tiek aprēķināts punktveida produkts. Šī aprēķina rezultāts ir viena vērtība, kas tiek ievietota jaunā līdzekļu kartē (skat. 7.8 att.).

Citiem vārdiem sakot, mēs varam arī teikt, ka ievades vērtības ir neironi, bet kodola vērtības ir svari. Matemātiski konvolūcijas operāciju var izteikt ar šādu formulu:

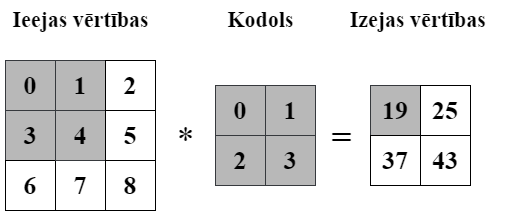
kur H – izejas vērtības matrica;

h – kodola platums;

w – kodola garums;

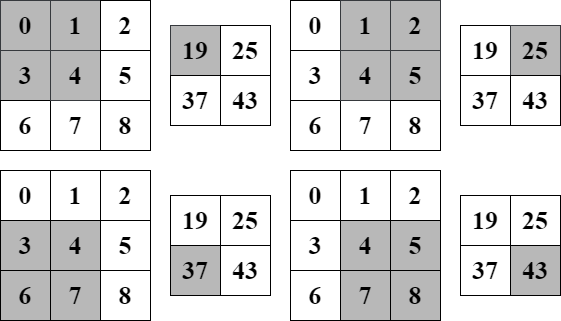
V – kodola matrica;

X – ieejas vērtības matrica.



7.8. att. Konvolūcijas operācija

Šis process tiek atkārtots katrai ievades datu atrašanās vietai, kurai var lietot kodolu, kā rezultātā tiek izveidota jauna līdzekļu karte (skat. 7.9 att.).



7.9. att. Konvolūcijas darbība katrai vērtībai

Konvolūcijas operācijā ir arī divi parametri, kas ietekmē darbību - tie ir solis (stride) un paddings.

Paddings attiecas uz papildu nulles pievienošanas procesu ap ievades attēlu, lai nodrošinātu, ka izvades vērtība pēc konvolūcijas paliek tāda pati kā ievades vērtība. Bez paddinga izvadītā attēla izmērs samazinās, filtram pārvietojoties pa ievades attēlu. Paddings ir svarīgs, ja ievades attēla izmērs nav saderīgs ar filtra izmēru vai ja mums ir jāsaglabā ievades attēla telpiskie izmēri izvadē.

No otras puses, solis attiecas uz pikseļu skaitu, ko filtrs pārvieto pa ievades attēlu katrā solī. Citiem vārdiem sakot, solis nosaka, cik daudz filtra pārvietojas pa ievades attēlu. Lielākas soļu vērtības rada mazāku izvades izmēru, un mazākas soļu vērtības rada lielāku izvades izmēru (skat. 7.10 att.).

Formula izvades izmēra aprēķināšanai pēc konvolucionālā slāņa ar padding un soli ir šāda:

.

*.*

kur h – apzīmē platums;

w – apzīmē garums;

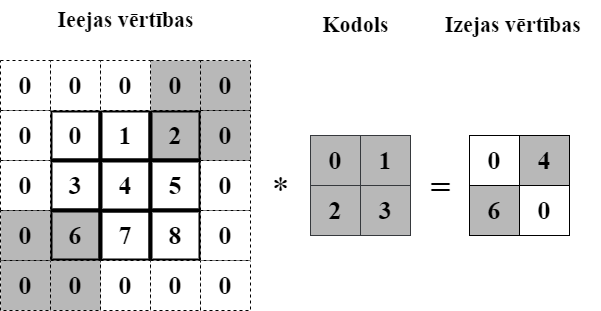
o – izejas vērtības matrica;

n – ieejas vērtības matrica;

k – kodols;

p – padding;

s – solis.

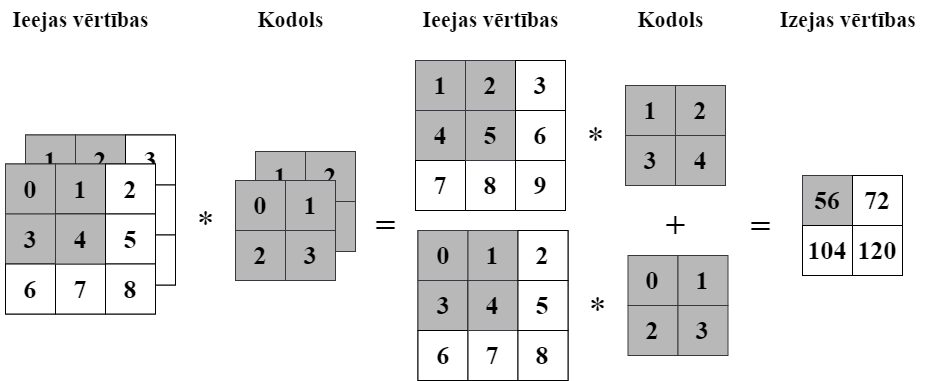


7.10. att. Konvolūcijas darbība ar soli = 3 un padding = 1

***Kanāli***

Iepriekšējie aprēķini, lai gan tie bija pareizi, attiecās tikai uz bezkrāsainiem attēliem. Realitātē attēliem parasti ir vairāki kanāli, kas satur informāciju par attiecīgās krāsas spilgtumu katrā attēla pikselī. Piemēram, ievades RGB krāsu attēls sastāv no trim kanāliem: sarkanā, zaļā un zilā.

Strādājot ar vairākiem kanālu attēliem, konvolūcijas operācija tiek veikta katrā kanālā atsevišķi. Piemēram, ja mums ir RGB attēls ar 3 kanāliem, mēs veiktu konvolūcijas darbību katrā no R, G un B kanāliem atsevišķi, izmantojot atšķirīgu filtru komplektu katram kanālam. Pēc tam katras konvolūcijas operācijas rezultātus apvieno vienā izejas tenzorā (skat. 7.11 att.).



7.11. att. Kanālu summēšana

Tomēr ir iespējams pievienot arī daudzkanālu kanālus katram kodolam. Veicot konvolūcijas operāciju neironu tīklos, izejas kanālu skaitu var iestatīt tā, lai iegūtu vajadzīgo pazīmju karšu skaitu. Tas tiek darīts vairākiem mērķiem:

1. modeļa sarežģītības palielināšana: izejas kanālu skaita palielināšana ļauj modelim apgūt sarežģītākus un abstraktākus datu elementus.
2. izšķirtspējas uzlabošana: palielinot izejas kanālu skaitu, var palīdzēt modelim labāk atšķirt dažādas pazīmes klases vai kategorijas.
3. lielāka elastība un pielāgojamība: vairāk izejas kanālu ļauj neironu tīkliem pielāgoties dažādiem datu un uzdevumu veidiem.

Konvolūcijas neironu tīklos attēlus parasti attēlo kā trīsdimensiju tenzorus, kur attēla platumu un garumu pārstāv pirmās divas dimensijas, bet kanālus - trešā dimensija. Tad konvolūcijas kodolu var attēlot kā četrdimensiju tensoru, kurā pirmie divi indeksi apzīmē kodola platumu un garumu, trešais indekss nosaka ievades kanālus, bet ceturtais indekss definē izvades kanālus.

Matemātiski konvolūcijas darbību, ņemot vērā kanālus un soļus, var izteikt ar šādu formulu:

kur H – izejas vērtības matrica;

h – kodola platums;

w – kodola garums;

nc - ievades kanālu skaits;

V – kodola matrica;

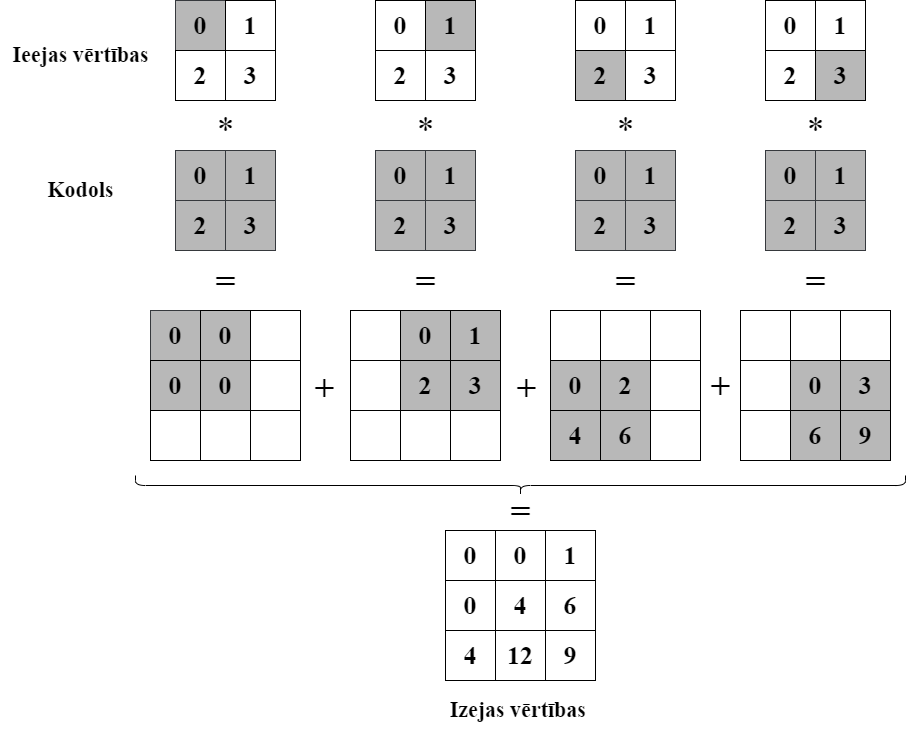
X – ieejas vērtības matrica;

s - soļa izmērs

***Transponētā konvolūcija***

Konvolūcijas neironu tīklos transponētā konvolūcija ir operācija, ko izmanto, lai palielinātu pazīmju karšu izšķirtspēju. Atšķirībā no parastās konvolūcijas operācijas, kas samazina ieejas pazīmju kartes lielumu, transponētās konvolūcijas operācija palielina ieejas pazīmju karti. Transponētās konvolūcijas operācija būtībā ir apgriezta konvolūcijas operācijai.

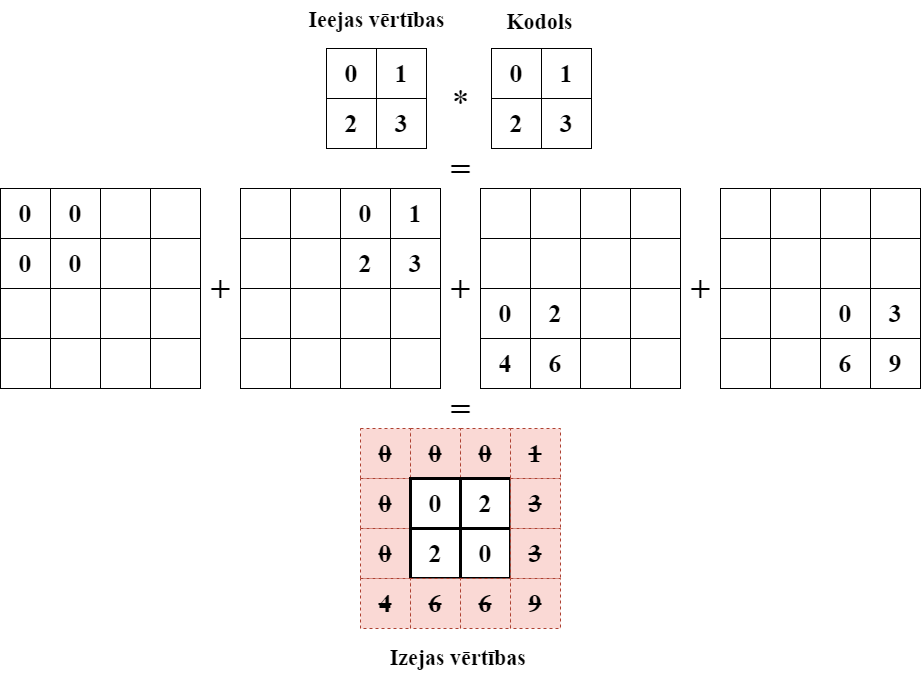
Transponētās konvolūcijas operācijas laikā filtrs tā vietā, lai pārvietotos pa ieejas pazīmju karti, pārvietojas pa izejas pazīmju karti. Lai aprēķinātu katru starpposma tenzoru, katru ieejas tenzora elementu reizina ar kodolu, lai iegūtais tenzors aizstātu daļu katrā starpposma tenzorā. Ņemiet vērā, ka aizstātās daļas pozīcija katrā starpposma tenzorā atbilst aprēķinā izmantotā elementa pozīcijai ieejas tenzorā. Visbeidzot visus starprezultātus summē, lai iegūtu rezultātu (skat. 7.12 att.).



7.12. att. Transponētās konvolūcijas operācija

Transponētajā konvolūcijā padding tiek lietots izvades funkciju kartei, nevis ievadei. Tomēr, nevis pievienojot padding objektu kartē, objektu karte tiek samazināta par margines izmēru.

No otras puses, solis attiecas uz soļa lielumu, ko izmanto, lai slīdētu konvolucionālo kodolu virs izvades funkcijas kartes. Lielāka soļa vērtība palielinaizvades līdzekļa kartes izmēru, savukārt mazāka soļa vērtība to samazina (skat. 7.13 att.).



7.13. att. Transponētā konvolūcijas darbība ar soli = 2 un padding = 1

Polsterējumu un soli var kombinēt, lai kontrolētu izvades funkciju kartes telpisko izšķirtspēju. Formula izvades izmēra aprēķināšanai pēc konvolucionālā slāņa ar polsterējumu un soli ir šāda:

kur h – apzīmē platums;

w – apzīmē garums;

o – izejas vērtības matrica;

n – ieejas vērtības matrica;

k – kodols;

p – padding;

s – solis.

### Citas kvalifikācijas darbā izmantotās metodes

***Binārā krusta entropija***

Binārā krustentropijas zuduma funkcija ir plaši izmantota binārās klasifikācijas problēmās, kur mērķis ir paredzēt vienu no diviem iespējamiem iznākumiem konkrētai ievadei. Tas mēra atšķirību starp prognozēto varbūtības sadalījumu un mērķa mainīgā lieluma faktisko varbūtības sadalījumu.

Bināro šķērsentropijas zudumu funkciju aprēķina, izmantojot Bernoulli sadalījuma negatīvo logaritmisko varbūtību. Citiem vārdiem sakot, tas mēra, cik labi modeļa prognozētais varbūtības sadalījums atbilst faktiskajam binārajam iznākumam. Funkciju izsaka šādi:

.

kur y – sagaidāmais rezultāts;

p – faktiskā varbūtība.

***Diskriminatora zuduma funkcija***

Diskriminatora zuduma funkcija tiek izmantota, lai aprēķinātu kļūdu vai zaudējumus, padodot īstu un ģenerētu (viltus) paraugu partiju un mēģinot noteikt, vai paraugs ir īsts vai viltots.

Diskriminatora zudumu funkcija ir bināra šķērentropijas funkcija, kas mēra starpību starp diskriminatora prognozēto varbūtības sadalījumu un paraugu patieso varbūtības sadalījumu.

Diskriminatora binārā šķērsentropijas zuduma funkcija ir definēta kā:

kur – diskriminatora zuduma funkcija;

n – mini pakešu izmērs;

;

;

***Ģeneratora zuduma funkcija***

Ģeneratora zuduma funkcija ir paredzēta, lai maksimāli palielinātu varbūtību, ka diskriminētājs kļūdīsies, vienlaikus klasificējot ģenerētos paraugus kā reālus. Citiem vārdiem sakot, ģenerators cenšas radīt paraugus, kas ir līdzīgi reāliem datu paraugiem, lai samazinātu diskriminētāja spēju atšķirt reālos un ģenerētos datus.

Matemātiskā izteiksmē ģeneratora zuduma funkciju var rakstīt šādi:

kur – ģeneratora zuduma funkcija;

n – mini pakešu izmērs;

.

Iepriekš minētā formula ir modificēta standarta formulas versija, un tā ir izstrādāta, lai maksimizētu zaudējumu funkciju. Tā ir sīkāk aprakstīta nākamajā nodaļā.

***Minimax spēle***

Termins "minimax" nāk no fakta, ka ģenerators cenšas samazināt diskriminētāja spēju pareizi klasificēt paraugus (tādējādi samazinot maksimālos zaudējumus), savukārt diskriminētājs cenšas maksimāli palielināt savu spēju pareizi klasificēt paraugus (tādējādi maksimāli palielinot minimālos zaudējumus).

Minimax spēli GANs var matemātiski formulēt kā:

kur G – ģenerators;

D – diskriminators;

n – mini pakešu izmērs;

– reāla attēla datu kopa

– viltus attēlu datu kopa

;

Bet pastāv problēma - diskriminatora apmācības sākumposmā ir viegli atšķirt oriģinālos paraugus no viltojumiem, jo ​​ģenerators vēl nav apmācīts. Tāpēc tā vietā, lai samazinātu ģeneratora zudumu funkciju, labāk ir maksimāli palielināt ģeneratora zudumu funkciju.

Minimax spēle starp ģeneratoru un diskriminētāju turpinās, līdz tiek sasniegts apturēšanas kritērijs, piemēram, ģenerators ražo paraugus, kas nav atšķirami no reālajiem datiem, vai kad apmācība sasniedz iepriekš noteiktu laikmetu skaitu.

***Normāls sadalījums***

Normāls sadalījumsir nepārtraukts varbūtības sadalījums ar zvanveida līkni, kurā lielākā daļa datu ir sagrupēti ap vidējo vai vidējo vērtību un kļūst retāki, jo vērtības attālinās no vidējā.

Mašīnmācīšanā normālo sadalījumu izmanto, lai inicializētu neironu tīklu svarus. Svaru inicializēšana ar vērtībām, kas iegūtas no normāla sadalījuma, var palīdzēt novērst gradientu izzušanu un eksplodēšanu apmācības laikā. Turklāt ienākošais troksnis ģeneratoram GAN bieži tiek ņemts no normāla sadalījuma.

Normālo sadalījumu raksturo divi parametri: vidējais, kas nosaka līknes pīķa atrašanās vietu, un standartnovirze, kas nosaka līknes izplatību. Standartnovirze ir mērs, cik daudz datu ir izkliedēti ap vidējo. Augstāka standartnovirze norāda, ka dati ir vairāk izkliedēti, savukārt zemāka standartnovirze norāda, ka dati ir ciešāk sagrupēti ap vidējo.

Piemērs nejauša skaitļa iegūšanai no normāla sadalījuma, izmantojot C++ programmēšanas valodu, ir redzams 3. pielikumā

***Attēla normalizēšana***

Normalizācija ir paņēmiens, ko izmanto, lai pikseļu vērtības ievietotu vēlamajā vērtību diapazonā. Tas ir svarīgi, ņemot vērā, ka diskriminators darbojas ar vērtībām diapazonā [-1;1].

Attēlu apstrādē normalizēšana ietver attēla pikseļu vērtību mērogošanu noteiktā diapazonā, parasti no 0 līdz 1 vai -1 un 1. To panāk, no katra pikseļa atņemot pikseļu intensitātes vidējo vērtību un dalot rezultātu ar standartnovirzi.

Attēla vērtību normalizēšanu kā matemātisku formulu var izteikt šādi:

kur - normalizēta pikseļu vērtība;

I – pikseļu vērtība;

Min - minimālā pikseļu vērtība;

newMin - jauna minimālā pikseļu vērtība;

Max – maksimālā pikseļu vērtība;

newMax – jauna maksimālā pikseļu vērtība.

***Aktivizācijas funkcijas***

Deep Convolutional Generative Adversarial Networks (DCGAN) ģeneratoru un diskriminētāju tīklos tiek izmantotas dažāda veida aktivizācijas funkcijas.

Rektificēta lineārā vienība (Rectified Linear Unit) - ReLU aktivizēšanas funkcija vienkārši aizstāj visas negatīvās ievades argumentu vērtības ar nulli, atstājot pozitīvās vērtības nemainīgas. Tas nozīmē, ka, ja ievades vērtība ir lielāka par nulli, tad funkcijas izvade būs vienāda ar šo vērtību, pretējā gadījumā izvade būs nulle.. Matemātiski formulu var attēlot šādi:

*()*

Leaky Rectified Linear Unit (LeakyReLU) - ReLU aktivizācijas funkcijas paplašinājums, kas funkcijai pievieno nelielu negatīvu slīpumu negatīvām ieejām. Funkcijas slīpuma vērtības var mainīt, DCGAN šī vērtība ir 0,2. Matemātiski formulu var attēlot šādi:

Sigmoida aktivācijas funkcija - Tas kartē jebkuru ievades vērtību uz vērtību no 0 līdz 1. Sigmoida funkcija tiek izmantota diskriminatora izejas slānī, lai izvadi pārveidotu par varbūtības vērtību, ko var interpretēt kā iespējamību, ka ievades attēls ir reāls. Matemātiski formulu var attēlot šādi:

Tanh aktivizācijas funkcija: Tanh aktivizācijas funkcija ir līdzīga sigmoida funkcijai, bet kartē jebkuru ievades vērtību uz vērtību no -1 līdz 1. To izmanto ģeneratora izejas slānī, lai mērogotu izejas pikseļu vērtības līdz ievades datu diapazonam. Matemātiski formulu var attēlot šādi:

***Pakešu normalizācija***

Pakešu normalizācija ir metode, ko izmanto dziļā mācīšanās procesā, lai normalizētu katra neironu tīkla slāņa ievadi. To ieviesa Sergejs Ioffe un Christian Szegedy savā 2015. gada rakstā "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift".

Apmācības laikā katra slāņa ievades vērtību sadalījums mainās, atjauninot tīkla svarus. Tas var novest pie parādības, kas pazīstama kā "iekšējā kovariācijas maiņa", kur ievades sadalījums slānī laika gaitā ievērojami mainās. Tas var palēnināt apmācības procesu, jo tīklam ir pastāvīgi jāpielāgojas mainīgajiem ievades sadalījumiem.

Pakešu normalizēšana risina šo problēmu, normalizējot ievades katram tīkla slānim. Konkrēti, tas atņem izejvielu partijas vidējo lielumu un dala ar izejvielu partijas standartnovirzi. Tas centrē un mērogo ievades, padarot tās konsekventākas visās partijās un ļaujot tīklam mācīties ātrāk un precīzāk. Partijas normalizāciju parasti piemēro pēc lineārās transformācijas slānī, bet pirms aktivizācijas funkcijas.

Pakešu normalizāciju matemātiskas formulas veidā var attēlot šādi:

kur - normalizējamā vērtība no iepakojuma

- pakešu paredzamā vērtība;

m – pakešu izmērs;

- pakešu dispersija;

– normalizētā vērtība;

maza pozitīva konstante,

- normalizētā vērtība ar saspiešanu un nobīdi;

- apmācāms parametrs;

– apmācāms parametrs;

# NOBEIGUMS

Šī kvalifikācijas darba gaitā tika izstrādāta un ieviesta attēlu ģenerēšanas programma, kas balstīta uz dziļa konvolucionālā ģeneratīvā-pretrunīgā neironu tīkla modeļa modeli. Darba mērķis tika veiksmīgi sasniegts, un pētījuma rezultāti apliecina neironu tīklu izmantošanas potenciālu radošajā procesā, kā arī to tālākas pilnveidošanas iespējas.

Programmas, kas ievieš neironu tīklu, programmas kods veiksmīgi pilda savu uzdevumu ģenerēt attēlus. Tomēr ir vairākas jomas, kurās kodu var uzlabot. Pirmkārt, apmācības un ģenerēšanas algoritmus var optimizēt, lai paātrinātu procesu un uzlabotu ģenerēto attēlu kvalitāti. Otrkārt, uzlabot programmas struktūru un klašu attiecības, lai samazinātu datora slodzi, paātrinātu programmas izpildi un novērstu fatālu kļūdu iespējamību.

Viens no šī darba galvenajiem secinājumiem ir tāds, ka neironu tīkli ir liels potenciāls jaunu radošu darbu radīšanai un ģenerēšanai ar datoru. Pilnveidojot neironu tīklu modeļus, nākotnē būs iespējams ātri izveidot unikālus attēlus ar daudzveidību un estētisku kvalitāti.

Noslēgumā jāsaka, ka šis darbs veiksmīgi ieviesa DCGAN neironu tīklu un demonstrēja neironu tīkla attēlu ģenerēšanas potenciālu. Tas ļauj radīt jaunus radošus darbus un paver perspektīvas turpmākiem uzlabojumiem un paplašināšanai. Darba rezultāti ir nozīmīgs ieguldījums mākslīgā intelekta un datorredzes jomā, un tos var izmantot dažādās jomās, piemēram, dizainā, mākslā, izklaidē un citās. Kopumā neironu tīkli ir daudzsološa tehnoloģija, kas turpinās veidot programmēšanas un mākslīgā intelekta nākotni. Tā kā šī joma turpina attīstīties, mēs varam sagaidīt jaunus un novatoriskus neironu tīklu lietojumus, kas vēl vairāk uzlabos to lietderību un ietekmi uz sabiedrību.

# INFORMĀCIJAS AVOTI

***Izmantotie avoti***

"Ceļvedis konvolūcijas aritmētikā dziļai apguvei" Vincent Dumoulin, Francesco Visin - https://arxiv.org/abs/1603.07285 – (Resurss apskatīts 27.05.2023.).

"Ģeneratīvie pretinieku tīkli" Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio - https://arxiv.org/abs/1406.2661 – (Resurss apskatīts 27.05.2023.).

"Neuzraudzīta reprezentācijas mācīšanās ar dziļiem konvolucionāliem ģeneratīviem pretinieku tīkliem" Alec Radford, Luke Metz, Soumith Chintala - https://arxiv.org/abs/1511.06434 – (Resurss apskatīts 27.05.2023.).

"Partijas normalizācija: dziļas tīkla apmācības paātrināšana, samazinot iekšējo kovariācijas nobīdi" - https://arxiv.org/abs/1502.03167 – (Resurss apskatīts 27.05.2023.).

"Dziļā mācīšanās" Ian Goodfellow, Yoshua Bengio, Aaron Courville - https://www.deeplearningbook.org/ – (Resurss apskatīts 27.05.2023.).

"Ienirt dziļā mācībā" - Aston Zhang, Zack C. Lipton, Mu Li, Alex J. Smola - https://d2l.ai/index.html – (Resurss apskatīts 27.05.2023.).

***Izmantotie instrumenti***

“Visual studio 2022” Microsoft Corporation - https://visualstudio.microsoft.com – (Resurss apskatīts 27.05.2023.).

“STB” Sean Barrett - https://github.com/nothings/stb – (Resurss apskatīts 27.05.2023.).

"Kaķu sejas" - https://www.kaggle.com/datasets/spandan2/cats-faces-64x64-for-generative-models – (Resurss apskatīts 27.05.2023.).

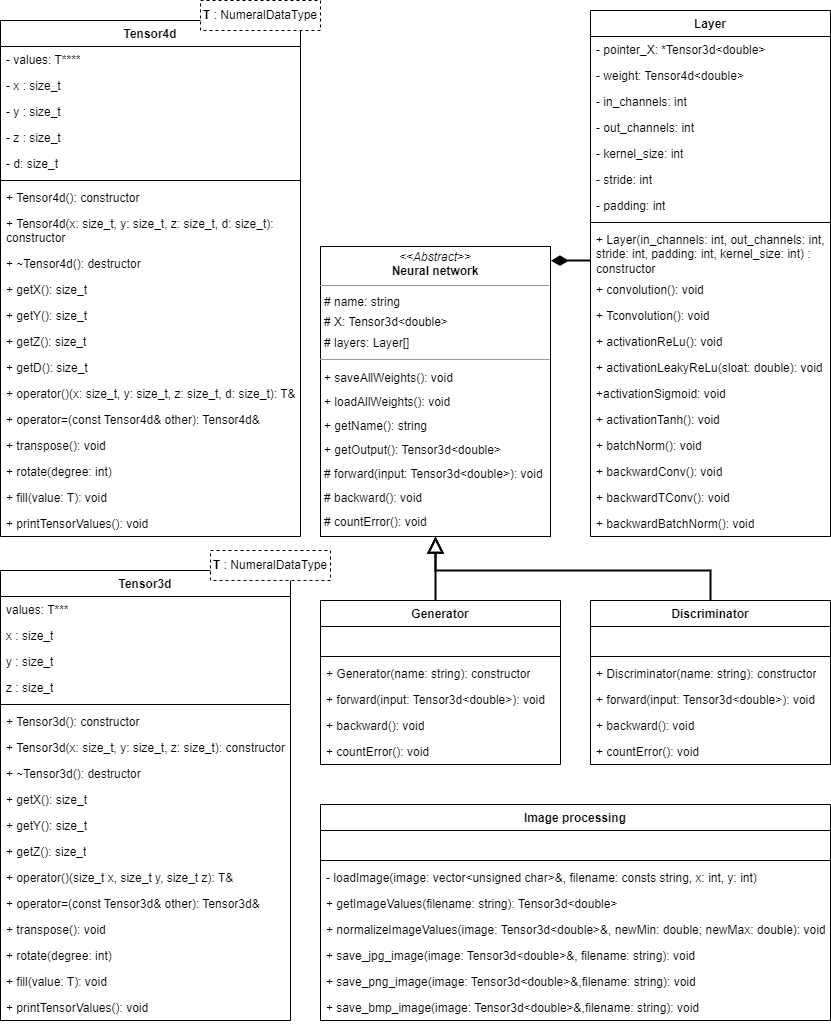
***Saite uz programmas avota koda repozitoriju***

“DCGAN” Daniils Srednakovs - https://github.com/Daniils-Srednakovs/Kvalifikacijas-darbs-DCGAN – (Resurss apskatīts 27.05.2023.).

# PIELIKUMI

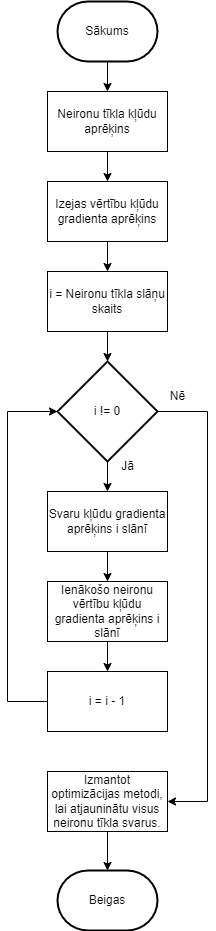
## 1. pielikums

**Sistēmas klases modelis**



## 2. pielikums

**Atpakaļpavairošanas shematiskā blokshēma**



## 3. pielikums

**Normāls sadalījums c++ valodā**

double normalDistribution(double mju, double sigma)

{

std::random\_device rd;

std::default\_random\_engine e(rd());

std::normal\_distribution<double> distN(mju, sigma);

return distN(e);

}