# IEVADS

Līdz ar strauju tehnoloģiju attīstību un skaitļošanas jaudas pieaugumu, ir plaši audzis arī trīsdimensiju datorgrafikas pielietojums. Tā ir kļuvusi pieejamāka un sastopama ne tikai datorspēlēs, virtuālās realitātes lietotnēs un citās tīri sintētiskās vidēs, bet arī nodrošinot specefektus filmās un videomateriālos, kā arī reālā laikā papildinātās realitātes lietotnēs. Neatkarīgi no pielietojuma, svarīga sastāvdaļa veiksmīga un vizuāli baudāma rezultāta iegūšanai ir ne tikai kvalitatīvi izstrādāti 3D modeļi, apgaismojums un renderēšanas metodes, bet arī šiem modeļiem izvēlētās tekstūras, kuru sagatavošana tradicionāli prasa daudz darba.

Tekstūru jeb, plašāk runājot, izmantoto materiālu pamatā parasti ir difūzā jeb krāsu tekstūra – attēls, kura pikseļu krāsu vērtības tiek attēlotas uz modeļa virsmas. Tieši šī tekstūru veida ģenerēšana ir darba fokuss. Praktiskais mērķis ir, ņemot par pamatu nelielu jau gatavas tekstūras gabaliņu, spēt algoritmiski ģenerēt praktiskiem nolūkiem neierobežotu apjomu līdzīgas tekstūras. Īpaši noderīgs šāds risinājums ir neregulārām un daļēji regulāram tekstūrām, piemēram, akmens vai koka virsmām, kur materiālā ir daudz dažādas neregularitātes un nepieciešams salīdzinoši liels tekstūras paraugs, lai izveidotu vizuāli pārliecinošu dotā materiāla virtuālo virsmu.

Pirmajā darba daļā tiek sīkāk apskatīta tekstūru ģenerēšanas problēma, eksistējošie risinājumi risinājumi un to sniegtās priekšrocības.

…

Pētījuma daļā tiek apskatīts izveidotais risinājums ar mākslīgo neironu tīklu, skaidrota tā uzbūve

# Tekstūru ģenerēšana

Teksts…

## Problēmas apraksts

Lietderīgam tekstūru ģenerēšanas algoritmam jāspēj pēc dota parauga ģenerēt

# Neironu tīklu metodes

Izvēlētā metode – neironu tīkls – ir nelineārs statistiskais skaitļošanas modelis, kas izstrādāts, iedvesmojoties no dabā sastopamajiem bioloģiskajiem neironiem, un kura nolūks ir tuvināti realizēt kādu nezināmu “melnās kastes” funkciju, zinot tikai ieejas un, iespējams, arī vēlamos izejas datus. Viena no nozares pamatlicēja Dr. Roberta Hehta-Nilsena vārdiem, neironu tīkls definējams kā “skaitļošanas sistēma, kas veidota no liela skaita vienkāršiem un savstarpēji blīvi savienotiem skaitļošanas elementiem, kas apstrādā datus, balstoties uz to dinamiskā stāvokļa reakciju uz ārēju ievadi” [1]. Šajā nodaļā īsumā izklāstīti neironu tīklu darbības pamatprincipi, kā arī darbā apskatītas problēmas risinājumam nepieciešamās neironu tīklu metodes.

## Neironu tīkla uzbūve

Neironu tīklu veido tā pamatvienības – neironi. Tie ir salīdzinoši vienkārši skaitļošanas elementi, kam var būt vairākas ieejas, taču tikai viena izeja. Ieejas datu apstrādi nosaka neirona svari un nobīdes. Tās ir skaitliskas vērtības, kas, atkarībā no tīkla arhitektūras, tiek kombinētas ar ieejas datu vērtībām [2]. Var apgalvot, ka neirons matemātiski realizē kādu daudzargumentu funkciju.

Kopējā neironu tīkla struktūrā neironi ir izvietoti slāņos, kas veic savstarpēju datu apmaiņu. Pirmais neironu slānis apstrādā ieejas datus, savukārt šī slāņa neironu izejas dati kalpo kā ieejas dati jau nākošajam neironu slānim, tā turpinoties, līdz tiek iegūtas vērtības tīkla izejas slānī. Tīkla iekšējie slāņi, kas ne no vienas puses nav saistīti ar apkārtējo vidi, tiek dēvēti par slēptajiem slāņiem [2]. Gan tīkla izejas slāņa forma un vērtību skaits, gan slēptajos slāņos izmantoto vērtību skaits no iepriekšējo slāņu rezultātiem var variēt atkarībā no izvēlētās tīkla arhitektūras. Klasiskas arhitektūras piemērs redzams attēlā 2.1.



2.1. att. **Daudzslāņu perceptrons** [3]

Piemēra arhitektūras gadījumā redzamais tīkls ir vienkāršs daudzslāņu perceptrons, kas nozīmē, ka tas sastāv no pilnsaistes slāņiem, t.i., katra nākošā slāņa neironi ieejā izmanto visu iepriekšējā slāņā rezultātu kopu. Šādi slāņi praksē tiek lietoti, taču ir gadījumi, kad tie nav piemēroti to radītā lielā datu apjoma dēļ, jo katrai neirona ieejai tiek pielietots svara mainīgais. Šāds gadījums ir arī darbā apskatītā tekstūru ģenerēšana, kas pēc būtības ir attēlu apstrādes problēma un strādā ar lielu datu apjomu. Vislabāk to var redzēt skaitliskos piemēros. Klasiskā attēlu apstrādes problēmā, piemēram, attēlu atpazīšanā, ieejas slānī būtu pilns attēls. Ja tas ir, piemēram, 320x320 pikseļu attēls ar 3 kanālu RGB krāsu telpas informāciju, kopā tās ir 307200 ieejas vērtības. Kā redzams attēlā 2.2, katrai ieejas vērtībai ***x1*** līdz ***xn*** (kur n = 307200), tiek piekārtots svara mainīgais ***wi***.

../neuron.pdf

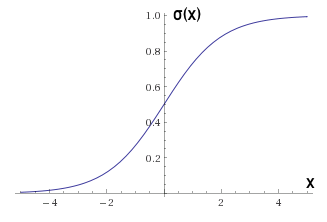
2.2. att. **Atsevišķa neirona uzbūve**

Tas nozīmē, kā sekojošajā slānī katram neironam būs vajadzīgi arī 307200 svaru mainīgie. Ņemot kopā visus slānī atrodamos neironus, mainīgo skaits būs vēl krietni lielāks. Tā kā šāds modelis ir neefektīvs gan no atmiņas, gan skaitļošanas viedokļa, šajā darbā apskatītājā un arī daudzās citās attēlu apstrādes problēmās parasti izmanto citu arhitektūras veidu – konvolūciju neironu tīklus. Tiesa, jāpiemin, ka arī šādās arhitektūrās, ja tiek risinātas klasifikācijas problēmas, pilnsaistes slāņi tiek lietoti, tomēr parasti tie ir pēdējie tīkla slāņi, kur mainīgo apjoms ir mazāks

Pašā neirona iekšienē ar ieejas vērtībām tiek veiktas kādas matemātiskas darbības. Atgriežoties pie daudzslāņu perceptrona gadījuma, parasti tā ir summa visām ieejām

kur ***b*** ir neirona nobīde – papildus svars, kas bieži šādās arhitektūrās tiek lietots, piemēram, lai būtu iespējams izejā iegūt nenulles vērtības, ja ieeja ir nulle vai tuvu tai.

Neironu izejas vērtības parasti tiek normalizētas izmantojot kādu aktivizācijas funkciju, tādējādi izvairoties no ekstrēmiem un iegūstot gan kvalitatīvāku tīkla darbību, gan lieki neapgrūtinot tīkla apmācības procesu. Tiek izmantotas daudz un dažādas aktivizācijas funkcijas, kuru izvēle atkal pamatojama tīkla arhitektūrā un pielietojumā. Klasiskais aktivizācijas funkcijas piemērs, kuru izmantotu daudzslāņu perceptrona arhitektūrā, ir loģistiskā jeb sigmoīdfunkcija , kas ierobežo izejas vērtības intervālā no 0 līdz 1 un nogludinā tās S-veida līknē, kā redzams attēlā 2.3. [4].



2.3. att. **Sigmoīdfunkcija**

Pēc aktivizācijas funkcijas pielietošanas tiek iegūta neirona rezultējošā vērtība, kas tālāk jau izmantojama kā ieejas dati nākošajos slāņos. Atsevišķu neirona uzvedība pati par sevi ir visai abstrakta un neironu tīkla rezultātu nosaka kopējā neironu mijiedarbība. Lai to ietekmētu, tiek regulētas tīkla mainīgo vērtības – neironu ieeju svari un to nobīdes.

## Neironu tīklu apmācība

Vēlamā tīkla funkcionalitāte tiek iegūta tā apmācības rezultātā, izmantojot kādu noteiktu apmācības algoritmu. Apmācības procesā tiek pakāpeniski mainīti tīkla neironu svari un nobīdes, kā vadlīniju izmantojot tīkla rezultātu izmaiņas katra apmācības perioda jeb epohas beigās [2].

Visizplatītākā pieeja neironu tīklu apmācīšanai ir pārraudzītā (no angļu val. *supervised learning*) apmācīšana, kur neironu tīkls tiek apmācīts no treniņdatu kopas, kam ir pieejami ne tikai ieejas dati, bet arī to pareizie izejas dati. Atkārtoti darbinot neironu tīklu uz piemēriem no treniņdatu kopas, salīdzinot rezultātus ar vēlamajiem un pakāpeniski regulējot tīkla mainīgo vērtības, tīkls tiek apmācīts modelēt sakarības starp ieejas un izejas datiem. Pēc veiksmīga apmācības procesa tīklam jāspēj iegūt kvalitatīvus rezultātus arī uz līdzīgiem ieejas datiem, kas nav bijuši sastopami treniņdatu kopā.

Tīkla apmācības procesā svarīga ir kļūdas funkcija (mēdz dēvēt arī par izmaksu funkciju no angļu val. *cost function*). Tīkla kļūda ir galvenā apmācības procesa metrika. Tas ir skaitlisks izmērāms lielums, kas raksturo starpību starp vēlamo un iegūto rezultātu. Vienkāršs, taču plaši izplatīts kļūdas funkcijas piemērs ir kvadrātiskā kļūdas funkcija.

Apmācības process nav nejaušs, bet gan tiek balstīts matemātiskās analīzes metodēs. Katra neironu ieeja uztverama kā vienargumenta funkcija, kam iegūstams parciālatvasinājums pret kļūdas funkciju, kas savukārt norāda uz tās pieaugumu atkarībā no dotās ieejas vērtības. No šiem parciālatvasinājumiem tīklam kā daudzargumentu funkcijai ir konstruējams funkcijas gradients – tās parciālatvasinājumu vektors, kas apraksta kopējo kļūdas funkcijas vērtības izmaiņu virzienu daudzdimensiju telpā. No matemātiskā skata punkta, neironu tīkla apmācība ir tā kļūdas funkcijas minimizācija, dodoties pretēji tās gradientam ar mērķi atrast tās globālo minimumu [3] [5].

## Konvolūciju neironu tīkli

Attēlu apstrādes problēmām vispopulārākā izvēle ir konvolūciju tīklu arhitektūras. Šis tīklu veids savu nosaukumu ieguvis no konvolūcijas operācijas matemātikā.

### Konvolūcijas operācija

Matemātiski konvolūcijas operācija divām funkcijām ***f*** un ***g*** ir integrālis, kas izsaka to pārklāšanos, veicot funkcijas ***g*** pārbīdi funkcijai ***f***. Tā var tikt definēta ierobežotā intervālā *[0, t]* ar noteikto integrāli

vai arī neierobežotā ar nenoteikto integrāli

[6].

Konvolūcija ir viena no svarīgākajām operācijām digitālo signālu apstrādē un tiek plaši lietota signālu kombinēšanai. Tā ir nozīmīga arī attēlu apstrādes gadījumos un divdimensiju konvolūcija tiek pielietota daudzos klasiskajos attēlu apstrādes algoritmos, kas radušies vēl pirms neironu tīklu metodēm. Tās darbības shematisks attēlojums redzams attēlā 2.4.



2.4. att. **Divdimensiju konvolūcijas darbība**

Jebkuru attēlu varam uztvert kā divdimensiju matricu no attēla pikseļu intensitātes vērtībām melnbaltam attēlam vai vienam krāsaina attēla krāsu kanālam. Visu 3 kanālu attēlošanai RGB telpā vajadzīgas 3 šādas matricas. Attēlam matricas formā var veikt konvolūciju ar citu matricu – konvolūcijas filtru. Tā parasti ir izmērā mazāka matrica, kas tiek “slidināta” pāri mērķa attēlam, katrā pozīcijā reizinot tās elementus ar pozīcijā atbilstošajiem attēla elementiem. Šī elementu reizinājumu summa tiek ņemta kā centrālā pikseļa vērtība filtrētajā attēlā. Aprēķina piemērs redzams arī attēlā 2.4, kur tiek aprēķinātas vērtības diviem pikseļiem, tātad aprēķinot divas dažādas filtra pozīcijas. Jāpiemin, ka, lai aprēķinātu vērtības attēla malējiem pikseļiem, uz kuriem nocentrējot filtru sanāk aprēķinā iekļaut neesošas vērtības ārpus definētās attēla matricas, parasti tās vai nu tiek aizstātas ar tuvākajām attēla pikseļu vērtībām (malas tiek “izstieptas”), vai arī ar neitrālām nulles vērtībām [7] [8].

Ar šo operāciju, izmantojot dažādas filtra matricas, iespējams iegūt daudzus noderīgus un plaši izmantotus attēlu apstrādes efektus. Tradicionāli tā tiek realizēti gan dažādi izpludināšanas filtri, gan attēlu asināšana, gan arī objektu malu izcelšana, kas nepieciešama datorredzes algoritmiem un attēlu atpazīšanas risinājumiem.



2.5. att. **Attēlu apstrādes filtru piemēri** [7]

Dotajā piemēra attēlā melnbaltajam mērķa attēlam tiek pielietoti trīs vienkārši 3x3 izmēra filtri. Šāds neliels filtra izmērs ir pilnībā pietiekams, lai iegūtu kvalitatīvus rezultātus. Tas norāda uz apkārtējo pikseļu skaitu, kas tiks ņemti vērā kombinējot centrālā mērķa pikseļa vērtību. Minēto 3x3 izmēra filtru gadījumā katrs aprēķina solis strādā ar deviņu pikseļu vērtībām, ieskaitot mērķa pikseli. Šāda pieeja ir arī intuitīvi loģiska, jo reālā attēlā, kas nav nejaušs, starp blakus pikseļiem pastāv sakarības, kas konvolucijā tiks ņemtas vērā. Palielinot filtra izmēru, pieaug arī attālāku un loģiski mazāk saistītu pikseļu savstarpējā ietekme.

### Pielietojums neironu tīklā

Konvolūcijas filtru izstrāde tradicionāli ir manuāls darbs, un, lai izstrādātu virkni filtru, kas galarezultātā sniedz kvalitatīvu problēmas risinājumu, ir jāvelta daudz laika un pūļu. Veiksmīga alternatīva ir konvolūcijas ienešana neironu tīkla arhitektūrā ar tā saukto konvolūciju neironu tīklu. Šo pieeju praksē pirmo reizi veiksmīgi izmantoja Jans LeKuns, viens no nozares spilgtākajiem zinātniekiem, kurš 1985. gadā izstrādāja arī pirmo kļūdu atgriezeniskas izplatīšanās metodes variantu. Viņš, kopā ar savu komandu *AT&T Bell Labs* paspārnē, 1990. gadā radīja pirmo konvolūciju neironu tīklu ar roku rakstītu ciparu atpazīšanai, kas ir klasiska un pietiekami netriviāla datorredzes problēma. Tīkla apmācībai tika izmantota datubāze ar pasta indeksu cipariem no reālām vēstulēm, kas sūtītas ASV teritorijā. Šos ciparus bija rakstījuši daudzi dažādi savā starpā nesaistīti cilvēki, nodrošinot, ka datu kopa ir daudzveidīga, ar dažādu izmēru un rakstības stilu piemēriem. LeKuna komandas izstrādātajam tīklam bija četri konvolūciju un viens pilnsaistes slānis, un tā bija pirmā konvolūciju tīkla arhitektūra, kuras visi slāņi tika pilnībā apmācīti caur kļūdas atgriezenisko izplatīšanos, neizmantojot cieti iekodētas konstantes. Apmācība ar tajā laikā pieejamiem skaitļošanas resursiem aizņēma aptuveni trīs dienas, kā rezultātā tīkls sasniedza 1,1% kļūdu treniņdatu kopai, kas sastāvēja no 9840 dažādiem ciparu attēliem, kā arī 3,4% kļūdu atsevišķai testa datu kopai. Minētie rādītāji nav izcili mūsdienu kontekstā, taču vērtējami kā ļoti veiksmīgi agrīno pētījumu ietvaros un pirmo reizi pavēra ceļu rezultātu reālai pielietošanai, kas arī vēlāk tika īstenots ar jau attīstītāku arhitektūru, pazīstamu kā *LeNet-5* [10][11].

### Tīkla darbība un priekšrocības

Vienā konvolūciju tīkla slānī, atšķirībā no pilnsaistes tīkla, katram attēla pikselim nav nepieciešams savs svara mainīgais. Neskaitot to, ka konvolūcija tīri loģiski strādā tikai ar savstarpēji tuviem un attēlā saistītiem pikseļiem, kas ļauj apstrādāt attēlu pa gabaliņiem, nevis ņemot vērā visu attēlu kā vienotu kopumu, šādā veidā arī nepieciešams daudz mazāks svaru apjoms, jo konvolūciju slānī mainīgo vērtības ir tieši filtra vērtības, kas arī kalpo kā slāņa svari.

Zinot ieejas attēlu un vēlamos izejas datus, tiek veikta konvolūcija ar ”melnās kastes” filtriem, t.i., neironu tīkls apmācības procesā piemeklē filtrus, kas dod nepieciešamo galarezultātu. Lai arī katrā epohā ir nepieciešams skaitļot visus konvolūciju rezultātus, tas tāpat ir daudz efektīvāk. Standarta 3x3 izmēra filtrs nozīmē, ka slānim ir jāpiemeklē 9 svari, kas ir daudz mazāks brīvo mainīgo skaits, nekā pilnsaistes slānī, un ir ievērojami ātrāk un vieglāk apmācāms. Arī rezultāti attēlu apstrādes problēmām ir labāki, jo konvolūciju filtri strādā kā iezīmju detektori (no angļu val. *feature detector*) un spēj atrast nepieciešamās iezīmes, piemēram, kādas noteiktas formas, jebkurā attēlā, kur tās ir sastopamas.

### Tīkla arhitektūras principi

Tīkla apmācība gan notiek noteiktas arhitektūras ietvaros un tās uzbūves lēmumi joprojām ir jāpieņem izstrādātājam. Tas ietver gan konvolūciju slāņu skaitu, gan pašu filtru izmēru un skaitu vienā slānī.

**Vairākus filtrus slānī**, līdzīgi kā ar attēlu matricām vairākiem krāsu kanāliem, varam uztvert kā vienu trīsdimensiju matricu, kuras dziļums ir slāņu skaits. Tādējādi katram ieejas attēla kanālam tiek atsevišķi rēķinātas konvolūcijas ar vairākiem filtriem, dodot vairākas rezultējošās matricas jeb iezīmju kartes (no angļu val. *feature maps*) un, salīdzinot ar sākotnējiem 3 krāsu kanāliem, palielinot kanālu skaitu attēla apstrādes rezultātā. Šāda pieeja nav netipiska un visai bieži konvolūciju tīklu slēptajos slāņos, lai varētu ņemt plašāku spektru attēla detaļas, tiek iegūts liels skaits dažādas iezīmju kartes un strādāts ar lielāku kanālu skaitu, ko pakāpeniski samazina atpakaļ uz attēlojamiem 3 krāsu kanāliem.

**Filtru izmērs** saistāms ar meklējamo iezīmju detalizāciju. Jo filtrs ir lielāks, jo ar lielākiem attēla gabaliem un lielāka mēroga, vispārīgākām iezīmēm konvolūcija operēs. Praksē gan reti tiek izmantoti filtri, kas lielāki par 5x5 pikseļiem, pamatā, vismaz pētījumā gaitā apskatītajās attēlu apstrādes arhitektūrās, pieturoties pie 3x3 filtru izmēra.



2.6. att. **Konvolūcija ar divu pikseļu soli** [12]

Konvolūcijas **solis** (no angļu val. *stride*) ir vēlviens svarīgs parametrs, kas ir aktuāls ne tikai neironu tīklu slāņos, bet arī tradicionālajos konvolūcijas pielietojumos. Tas norāda filtra pārbīdes attālumu, veicot konvolūciju. Ja filtru pārbīda pa vienam pikselim, soļa vērtība ir 1, un šādi, ja tiek izmantota attēla vērtību papildināšana malējiem pikseļiem, rezultējošā iezīmju karte ir tādā pašā izmērā kā ieejas attēls. Ja soļa vērtība ir lielāka, filtrs aprēķinu gaitā pārlec pa vairākiem pikseļiem, rezultātā iegūstot kontrastējošāku iezīmju karti, kuras pikseļiem ir mazāka sakarība vienam ar otru. Kā redzams attēlā 2.6, pie lielāka soļa iezmēra arī rezultējošās iezīmju kartes izmērs ir mazāks, nekā ieejas attēlam, jo tiek zaudēta daļa informācijas, tomēr arī šai tehnikai ir savi plusi un tā bieži tiek pielietota.

# PĒTĪJUMS

Teksts…

## Izvēlētais risinājums

Problēmas risinājumam tika izveidots ģeneratīvs neironu tīkls, kas tika trenēts, izmantojot ārēju tīklu attēlu analīzei. Iespaidojoties no Leona Gatisa pētījuma, arī šajā sistēmā tika izvēlēta viena no Oksfordas universitātes *Visual Geometry Group* izstrādātās VGG arhitektūras implementācijām [1]. Šī pieeja balstās principā, ka, lai veiksmīgi atdarinātu mērķa attēlu, tas vispirms ir precīzi formāli jāapraksta.

Implementācijai izvēlētā VGG16 arhitektūra ir tīkls ar 13 konvolūciju slāņiem, kas paredzēts plaša spektra attēlu atpazīšanai. Sākotnēji tas tika izstrādāts *ImageNet Large Scale Visual Recognition Challenge 2014* konkursam, kur, līdzās ar nedaudz sarežģītāko VGG19 arhitektūru, tās Oksfordas universitātes pētnieku grupai nodrošināja pirmās divas vietas, sekmīgi tiekot galā ar 1000 dažādajām objektu klasēm, kas konkursā iesniegtajiem risinājumiem bija jāspēj atpazīt [2].

Neviens materiāls nav formāli aprakstāms pilnībā precīzi, taču praktiskais mērķis ir panākt vizuāli pārliecinošu tuvinājumu, nevis absolūtu precizitāti. Izmantojot šādas attēlu atpazīšanas arhitektūras starprezultātus kā attēla analīzes rīku, ir iespējams iegūt sava veida tuvinātu attēla parametrisko modeli. Tas pamatojams spriedumā, ka kvalitatīvam attēlu atpazīšanas algoritmam būtu jāsniedz līdzīgi klasifikācijas rezultāti dažādiem attēliem, kuros redzams līdzīgs saturs, piemēram, viens un tas pats materiāls. VGG tīkla rezultāts ir vektors ar varbūtībām visām apskatāmajām objektu klasēm. Šīs varbūtības pašas par sevi gan nav lietderīgs lielums attēlu aprakstīšanas nolūkiem. Piedevām, identificējamo objektu klašu skaits ir galīgs un nejaušiem attēliem ar nevienā objektu klasē neietilpstošu saturu rezultāti ir neskaidri. Daudz lietderīgāki ir tīkla starprezultāti no konvolūciju slāņiem. Šie slāņi tiešā veidā strādā ar attēla pikseļu vērtībām un nodarbojas ar attēlu elementu izdalīšanu dažādos detalizācijas līmeņos. Attiecīgi, lai sniegtu līdzīgu galarezultātu, kvalitatīvi uzģenerētai tekstūrai arī šajos vidējos slāņos ir jāsniedz līdzīgi rezultāti kā oriģinālajam paraugam. Izvēloties noteiktu skaitu konvolūciju slāņu, no kuriem tiek ņemti to rezultāti, iegūstam netiešu tekstūru aprakstošo modeli. Tā kā tekstūra tiek aprakstīta caur tīkla rezultātiem, šī attēla statistikas kopa ir atkarīga no paša analizējošā tīkla arhitektūras. Lai arī ir bijuši eksperimenti ar nosacīti nejaušiem aprakstošajiem tīkliem (ievērojot specifiskus arhitektūras principus, taču izmantojot nejaušas filtru vērtības un tīklu netrenējot), svarīgi ir arī, lai tīkls būtu trenēts un konvolūciju slāņi veiktu savas noteiktās funkcijas. Konvolūcijas operācijas ir lineāras, tomēr tīkla arhitektūrā tiek izmantotas arī nelineāras aktivācijas, kas netrenētā vai vāji trenētā tīklā rezultējas daudz izteiktākā informācijas zudumā, dodot arī ievērojami sliktāku vai pat neizmantojamu rezultātu kā attēla aprakstošajai statistikai.

Ģeneratīvā tīkla trenēšana notiek uz konkrētu mērķa attēlu, kurš sākotnēji tiek padots VGG tīklam kā ieejas dati, lai iegūtu tā aprakstošo statistiku. Pats trenēšanas process ir iteratīvs, katrā iterācijā, izmantojot aktuālo tīkla stāvokli, ģenerējot attēlu un padodot to VGG tīklam statistikas iegūšanai tāpat, kā tas tika darīts ar mērķa attēlu. Atšķirība starp mērķa attēla statistiku un ģenerētā attēla statistiku ir ģeneratīvā tīkla kļūda – skaitlisks lielums, pēc kura vadoties tīkls tiek trenēts.

## Ģeneratīvā tīkla arhitektūra

# IZMANTOTĀ LITERATŪRA

|  |  |
| --- | --- |
| [1] | M. Caudill, “Neural Network Primer: Part I,” *AI Expert,* Feb. 1989. |
| [2] | J. Zuters, “Ar neironu tīkliem saistīto terminu vārdnīca,” [Online]. Available: http://home.lu.lv/~janiszu/courses/eanns/annsdictionary.pdf. [Accessed 15. Maijs 2017]. |
| [3] | M. A. Nielsen, “Neural Networks and Deep Learning,” Determination Press, 2015. |
| [4] | A. Karpathy, “Convolutional Neural Networks for Visual Recognition, Neural Network architectures,” Stanford University, [Online]. Available: http://cs231n.github.io/neural-networks-1/. [Accessed 15 Maijs 2017]. |
| [5] | Encyclopedia of Mathematics, “Gradient,” [Online]. Available: http://www.encyclopediaofmath.org/index.php?title=Gradient&oldid=28205. [Accessed 15 Maijs 2017]. |
| [6] | E. W. Weisstein, “Convolution,” MathWorld, [Online]. Available: http://mathworld.wolfram.com/Convolution.html. [Accessed 16 Maijs 2017]. |
| [7] | J. Ludwig, “Image Convolution,” Portland State University, [Online]. Available: http://web.pdx.edu/~jduh/courses/Archive/geog481w07/Students/Ludwig\_ImageConvolution.pdf. [Accessed 17 Maijs 2017]. |
| [8] | Department of Radio Engineering, “Padding of Borders,” Czech Technical University, [Online]. Available: http://radio.feld.cvut.cz/matlab/toolbox/images/linfilt4.html. [Accessed 17 Maijs 2017]. |
| [9] | F. Petitjean, “Deep Learning for Text Understanding from Scratch,” [Online]. Available: http://www.kdnuggets.com/2015/03/deep-learning-text-understanding-from-scratch.html. [Accessed 17 Maijs 2017]. |
| [10] | Y. LeCun, “Research and Projects,” [Online]. Available: http://yann.lecun.com/ex/research/. [Accessed 18 Maijs 2017]. |
| [11] | Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. Howard, W. Hubbard and L. Jackel, “Handwritten Digit Recognition with a Back-Propagation Network,” AT&T Bell Laboratories, 1990. [Online]. Available: http://yann.lecun.com/exdb/publis/pdf/lecun-90c.pdf. [Accessed 18 Maijs 2017]. |
| [12] | L. A. Gatys, A. S. Ecker and M. Bethge, “Texture Synthesis Using Convolutional Neural Networks,” University of Tubingen, 2015. |
| [13] | ImageNet, “Large Scale Visual Recognition Challenge 2014 (ILSVRC2014),” 2014. [Online]. Available: http://www.image-net.org/challenges/LSVRC/2014/. [Accessed 9 Maijs 2017]. |
| [14] | D. Ulyanov, V. Lebedev, A. Vedaldi and V. Lempitsky, “Texture Networks: Feed-forward Synthesis of Textures and Stylized Images,” Skolkovo Institute of Science and Technology, Yandex, University of Oxford, 2016. |