LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

OBJEKTU ATPAZĪŠANA ATTĒLOS AR MĀKSLĪGAJIEM NEIRONU TĪKLIEM, IZMANTOJOT VĀJI PĀRRAUDZĪTO MĀCĪŠANĀS METODI

BAKALAURA DARBS

Autors: **Sergejs Ivanovs**
Studenta apliecības Nr.: si11020
Darba vadītājs: Dr.sc.ing. Andrejs Zujevs

RĪGA 2016

Anotācija

Šajā darbā tiek apskatīti konvolūciju neironu tīkli, kas tiek apmācīti ar vāji pārraudzīto mācīšanās metodi.

Darba mērķis ir izstrādāt neironu tīkla struktūru, ko būtu iespējams apmācīt ar attēliem, kuri tiek aprakstīti, norādot sarakstu ar objektiem attēlā, taču nenorādot šo objektu atrašanās vietu.

Darbā ir apskatīti neironu tīklu pamatprincipi un to attīstības vēsture. Īpaši detalizēti tika apskatīts konvolūciju neironu tīklu darbības princips, kā arī tika definētas vairākas vadlīnijas neironu tīklu struktūras definēšanai. Tika izstrādāts un veiksmīgi apmācīts konvolūciju neironu tīkls, kā arī uz šī konvolūciju tīkla bāzes izstrādāts neironu tīkla prototips, kas ir paredzēts vāji pārraudzītas mācīšanās metodei.

Atslēgvārdi: neironu tīkli, konvolūciju tīkli, vāji pārraudzīta mācīšanās metode, neironu tīklu apmācība.

Abstract

Object Detection in Images Using Artificial Neural Networks with Weakly Supervised Learning Method

In this work the convolutional neural networks that are trained using weakly supervised learning method are being studied.

The aim of this work is to develop a neural network structure, which would make it possible to train it with the images, which are described with a list of the objects in the picture, but without defining the object location.

It this work the basic principles of neuronal networks and their history are observed. In particular, convolutional neural networks were studied, as well as the neural network structure development guidelines were defined. Convolutional neuronal network was developed and successfully trained, as well as on the convolution network base was developed neuronal network prototype suitable for weakly supervised learning method.

Keywords: neural network, convolutional neural network, weakly supervised learning, neural network training.

SATURA RĀDĪTĀJS

APZIMĒJUMU SARAKSTS

Apzīmējums/termins	**Definīcija**
Neironu tīkls	informācijas apstrādes sistēma, kas sastāv no relatīvi liela skaita skaitļošanas elementu (sauktu par neironiem) un savu spēju veikt noteiktu uzdevumu ieguvusi apmācības ceļā [1].
Neironu tīkla slānis	ir neironu kopums, kas topoloģiski izvietoti kopā un kuru darbināšana un apmācība parasti notiek pēc viena algoritma [1].
Mākslīgais neirons	matemātiskā funkcija, kas modelē bioloģiskā neirona darbību.
Dziļā mācīšanās (no angl. v. *deep learning*)	mašīnmācīšanās nozare, kas modelē augstā līmeņa datu abstrakcijas izmantojot kompleksās datu apstrādes struktūras un nelineāras transformācijas.
Tenzors	daudzdimensiju matrica.
Vāji aprakstīts attēls	attēls kuram ir aprakstīts tajā redzamo objektu saraksts, bet nav noteiktas objektu atrašanas vietas un izmēri.

IEVADS

Datorredzes joma vienmēr ir bijusi izaicinoša. Tās attīstība pavisam nesen vēl bija ierobežota ar skaitļošanas resursiem. Šodien, kad pat galda datoram ir pietiekama jauda un resursi ietilpīgo datorredzes metožu pielietošanai, paveras plašas iespējas dažādu uzdevumu risināšanai. Piemēram, uz mūsdienu vidējās klases videokartes var apmācīt sarežģītu konvolūciju neironu tīklu, izmantojot lielu apmācības kopu, tādu kā ImageNet [2]. Skaitļošanas resursu pieejamība šodien dod iespēju risināt objektu atpazīšanas un lokalizēšanas uzdevumus, izmantojot mākslīgos neironu tīklus.

Pēdējā laikā strauji attīstās ne tikai skaitļošanas resursi, bet arī neironu tīklu arhitektūras un datu apstrādes pieejas. Attēlu apstrādei un objektu atpazīšanai īpaši ir piemēroti konvolūciju neironu tīkli. To attīstība deva iespēju efektīvi risināt objektu atpazīšanas uzdevumus, jo to izmantošana objektu atpazīšanā dod iespēju ņemt vērā objektu specifiskās pazīmes, ar to pazeminot skaitļošanas resursu izmantošanu un paaugstinot neironu tīkla darbības efektivitāti un precizitāti.

Neironu tīklu darbībai ir nepieciešams to apmācīt, izmantojot paraugu kopu. Šodienas neironu tīklu risinājumi pieprasa, lai apmācības kopa būtu specifiski sagatavota. Objektu atpazīšanas uzdevumam ir nepieciešams, lai neironu tīkls tiek apmācīts ar attēliem, kur katrā attēlā ir viens precīzi aprakstīts objekts, kas ir cieši izgriezts no attēla. Tas dod iespēju apmācīt neironu tīklu ar konkrētu objektu, un atpazīt to tālākos neironu tīkla pielietojumos.

Ņemot vērā, ka liela mākslīgā neironu tīkla apmācīšanai ir nepieciešams ļoti liels datu apjoms (tūkstoši attēlu uz vienu objektu klasi), tad, lai iegūtu pietiekošu apmācības kopas apjomu, ir nepieciešams daudz laika un naudas resursu, jo lielas kopas izveidošanai ir nepieciešams algot cilvēkus, kuri atlasīs attēlus, apstrādās tos un veiks attēla objektu aprakstīšanu [3].

No šīs problēmas seko nākamā: dažreiz apmācības kopas nepietiekamais apjoms ierobežo neironu tīkla precizitāti. Dažādās neironu tīklu struktūrās lielāks apmācību piemēru skaits dod iespēju veiksmīgāk apmācīt neironu tīklu [4]. Tas palielinās neironu tīkla precizitāti, un tas spēs atpazīt vairākus objektu paveidus, formas un izskatus, kā arī tas būs mazāk atkarīgs no objekta pozīcijas un skatupunkta uz to.

Ar pieejamām apmācīšanas datu kopām var apmācīt neironu tīklu līdz pietiekoši lielai precizitātei, bet dažām neironu tīklu topoloģijām pieejamu datu nepietiek, lai precīzi apstrādātu attēlus, kas nav bijuši apmācības kopā, kā arī ne vienmēr ir pieejamas attēlu datubāzes ar nepieciešamām objektu klasēm. Ņemot vērā to, ka dažas attēlu datubāzes ir

pieejamas tikai akadēmiskiem nolūkiem, tad komerciālos risinājumos vajadzīgo attēlu datubāzes atrašana vai izveide tiek papildus apgrūtināta [5].

Ņemot vērā apmācības datu sagatavošanas grūtību, ir vērts šo procesu atvieglot. Viens no veidiem ir izstrādāt neironu tīklu struktūru, kas spētu apmācīties izmantojot attēlus, kur attēlotiem objektiem nav norādīta precīza atrašanās vieta un izmērs, bet ir norādīts objektu saraksts attēlā, un ierobežot pilnībā aprakstītu datu izmantošanu. Ja neironu tīkls spētu apmācīties uz vāji aprakstītiem attēliem, tad nepieciešamās datu kopas sagatavošana būtu daudz vieglāka.

Darba mērķis: iepazīties ar vāji pārraudzītās mācīšanās metodes pielietošanu konvolūciju neironu tīklu uzbūvē un to apmācībā objektu atpazīšanai attēlos.

Mērķa sasniegšanai tika izvirzīti sekojošie uzdevumi:

1) izpētīt neironu tīklu darbības principu un attīstīšanas vēsturi;
2) iepazīties ar konvolūciju neironu tīklu uzbūvi un darbības principiem;
3) apskatīt šobrīd pieejamas mašīnmācīšanās bibliotēkas un izvēlēties atbilstošu bibliotēku praktiskā darba realizācijai;
4) izstrādāt konvolūciju neironu tīkla struktūru vāji pārraudzītas mācīšanās metodes pielietošanai;
5) apmācīt un testēt izstrādāto neironu tīklu.

Neironu tīkla struktūras aprobēšanai, tiks izstrādāts mākslīgais konvolūciju neironu tīkls un apmācīts ar vāji pārraudzīto mācīšanās metodi. Tās realizēšanai tiks apskatītas un salīdzinātas vairākas mašīnmācīšanās bibliotēkas un izvēlēta viena, kas visvairāk atbilst izvēlētās problēmas risināšanai.

Lai definētu strādājošu neironu tīkla struktūru, tiks apskatīti vairāki pētījumi un materiāli, kā arī tiks apskatītas metodoloģijas apmācības datu sagatavošanai un neironu tīkla struktūras definēšanai. Šis pētījums dos ieskatu, kā nodefinēt vajadzīgo konvolūciju neironu tīkla struktūru un sasniegt uzstādīto mērķi.

Darba sākumā īsi tiks aprakstīta mākslīgo neironu tīklu attīstītības vēsture un darbības princips. Daudz detalizētāk tiks apskatīti konvolūciju neironu tīkli un to darbības princips. Tiks aprakstīta neironu tīkla struktūra, kā arī apmācības un testēšanas kopu sagatavošanas princips, kas ļaus pareizi apmācīt neironu tīklu. Tiks salīdzinātas pieejamās mašīnmācīšanās bibliotēkas un izvēlēta atbilstošā bibliotēka objektu atpazīšanas uzdevuma risināšanai ar vāji pārraudzīto metodi. Darba praktiskā daļā tiks aprakstīta izveidotā neironu tīkla struktūra, tās apmācīšanas un testēšanas process. Darba beigās tiks apkopoti rezultāti un aprakstīti darba secinājumi.

1. MĀKSLĪGO NEIRONU TĪKLU APSKATS

Šīs daļas nolūks ir virspusēji aprakstīt mākslīgos neironu tīklus, to uzbūvi un vēsturisko attīstību. Tiks aprakstīti neironu tīklu izmantošanas piemēri un attīstīšanas iespējas.

Mākslīgais neironu tīkls ir skaitļošanas rīks, kas balstās uz bioloģisko neironu sistēmu īpašībām. Tas tiek uzskatīts par nelineāru statistisko datu modelējošu instrumentu, kur tiek modelēta atkarība starp ieejas un izejas datiem.

Neironu tīkli tiek izmantoti, lai simulētu vai tuvinātu tādu funkciju darbību, kuru darbība ir nezināma un ir atkarīga no liela ieejas datu skaita. Mākslīgais neironu tīkls sastāv no savstarpēji saistītiem mākslīgiem neironiem, kuri veic savstarpēju informācijas apmaiņu. Galvenā mākslīgo neironu tīklu īpašība ir neirona tīkla saišu svaru maiņa, atkarībā no tai padotās informācijas, tāpēc mākslīgais neironu tīkls ir spējīgs apmācīties, balstoties uz ieejošo informāciju un sagaidāmo rezultātu.

Neironu tīklu priekšrocības ir to spēja apmācīties, apskatot paraugu datu kopas. Šādā veidā, mākslīgie neironu tīkli tiek izmantoti, kā neprognozējamo funkciju tuvināšanas rīki. Lai neironu tīkls varētu prognozēt funkcijas rezultātu, tā apmācīšanai nav nepieciešama visa datu kopa, bet tikai šīs datu kopas paraugi. Tas dod iespēju apmācīt neironu tīklus, kad nav pieejama pilna datu kopa vai nav zināma precīza atkarība starp ieejas datiem un sagaidāmo rezultātu. Šī īpašība dod iespēju pielietot neironu tīklus datorredzē un objektu atpazīšanā, jo tiek iegūts tīkls, kuru var apmācīt ar izvelēto objektu attēlu piemēriem, gadījumos, kad ir skaidri redzams, ka nav iespējams iegūt visus iespējamos kāda objekta attēlus [6].

1.1. Mākslīgo neironu tīklu attīstības vēsture

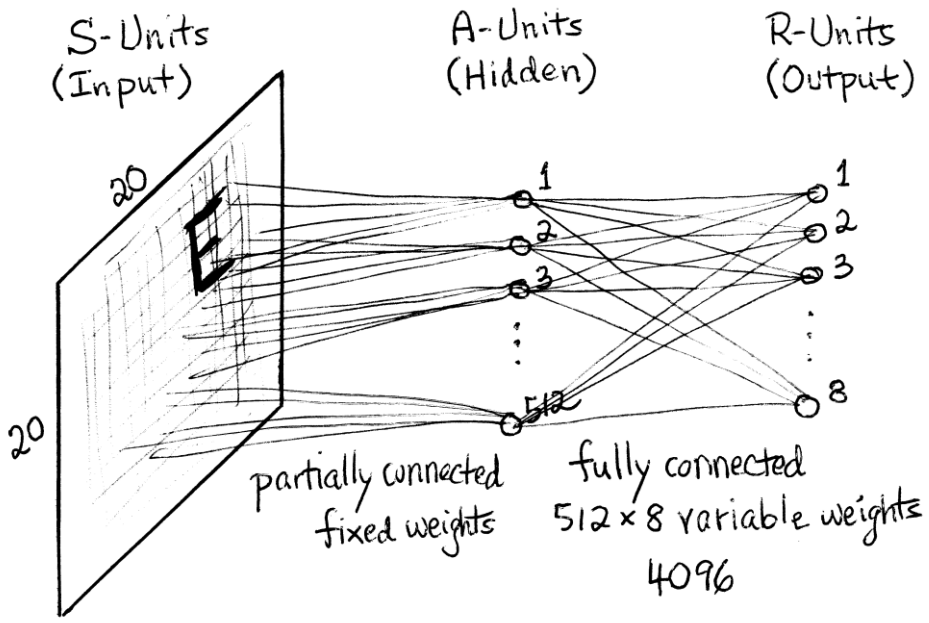
1940. gadu beigās psihologs un fiziologs Donalds Hebbs izteica ideju, ka mācīšanās balstās uz to, ka saites starp smadzeņu neironiem, jeb sinapsēm var pastiprināties vai vājināties laika gaitā atkarībā no to izmantošanas biežuma. Šī ideja tika izmantota pirmo neironu tīklu izstrādē.

Mākslīgo neironu tīklu attīstība sākās 1943. gadā, kad Makkaloks un Pitts izveidoja skaitļošanas modeli neironu tīkliem, balstoties uz matemātiskiem algoritmiem un smadzeņu darbības teoriju. Viņi izvirzīja pieņēmumu, ka neironus var uzskatīt par skaitļošanas ierīcēm, kas darbojas ar bināriem skaitļiem. Šo modeli nosauca par sliekšņa loģiku. Šim modelim pētnieki piedāvāja elektronisko neironu konstrukciju un parādīja, ka līdzīgs tīkls var izpildīt gandrīz visas ciparu un loģiskas operācijas**.** Makkaloks un Pitts ierosināja, ka šāds tīkls var

tikt apmācīts, tas var atpazīt attēlus un vispārināt informāciju, kas pēc būtības nozīmē, ka šādam tīklam ir visas inteliģences iezīmes [7].

Šis modelis lika pamatus divām dažādām neironu tīklu pētniecības pieejām. Viena pieeja bija vērsta uz smadzeņu bioloģisko procesu pētniecību, bet otra uz neironu tīklu izmantošanu, kā mākslīgā intelekta metodi dažādu uzdevumu risināšanai.

Franks Rozenblatts 1957. gadā izstrādāja matemātisku datora modeli smadzeņu informācijas uztverei, jeb perceptronu, kas apmācīšanas laikā izmantoja saskaitīšanu un atņemšanu. 1958. gadā tika piedāvāts elektroniskās ierīces modelis, kurš pēc idejas varēja imitēt cilvēka domāšanas procesus [8]. Pēc diviem gadiem tika demonstrēta strādājošā ierīce, kas varēja atpazīt dažādus burtus.



1.1. att. **Pirmā perceptrona shēma** [9]

Interese priekš mākslīgiem neironu tīkliem samazinājās, kad Minskijs un Peiperts 1969. gadā publicēja savu pētījumu. Viņi atrada galvenās skaitļošanas problēmas, kas parādās mākslīgo neironu tīklu izstrādes gaitā. Pirmā problēma bija tāda, ka vienslāņa neironu tīkls nevarēja izpildīt "izslēdzošā vai" operāciju. Otrā svarīgā problēma bija tā laika datori. Tie nebija pietiekoši jaudīgi, lai efektīvi apstrādātu milzīgu skaitļošanas apjomu, kas bija nepieciešams lieliem neironu tīkliem.

Neironu tīklu pētniecība palēninājās līdz brīdim, kad datori sasniedza pietiekoši lielu skaitļošanas jaudu. Tālāko neironu tīklu attīstību stimulēja kļūdu atgriezeniskās izplatīšanās metodes izstrāde 1975. gadā, kas palīdzēja efektīvi risināt daudzslāņu neironu tīklu apmācīšanās uzdevumu un ļāva atrisināt "izslēdzošu vai" problēmu [10].

1975. gadā Fukušimā izstrādāja “kognitronu”, kas kļuva par vienu no pirmajiem daudzslāņu neironu tīkliem.

Paralēli sadalītas apstrādes algoritms 1980. gadu vidū tika izmantots, lai modelētu neironu procesus.

Neskatoties uz lielo entuziasmu zinātnieku vidū par kļūdu atgriezeniskās izplatīšanās metodes izstrādi, tas radīja daudz strīdu par to, vai šī metode var tikt realizēta smadzenēs. Nav līdz galam skaidrs vai modelis, kas izmantots mākslīgo neironu tīklos, ir saistīts ar smadzeņu bioloģisko struktūru.

Laika gaitā neirona tīklu mašīnmācīšanās pielietojumu aizstāja daudz vieglākas metodes, piemēram, lineārie klasifikatori, tomēr, sākoties 2000. gadam, atjaunojās interese par dziļās mācīšanās attīstītību neironu tīklos.

No 2009. līdz 2012. gadam tika izstrādāts rekursīvais neironu tīkls un dziļais vienvirziena neironu tīkls.

Pēc 2012. gada sakās neironu tīklu strauja attīstība, kas tika veicināta ar vairākiem faktoriem. Pirmkārt, skaitļošanas ierīču veiktspēja palielinājās līdz tādam līmenim, ka tie varēja būt izmantoti neironu tīklu apmācīšanai un darbināšanai. Turklāt lielās veiktspējas skaitļošanas ierīces bija pieejamas lielam cilvēku lokam. Pēdējā laika neironu tīklu attīstību veicināja iespēja izmantot grafiskos procesorus apmācīšanas procesos, kas būtiski paātrināja neironu tīklu apmācības procesu un deva iespēju apmācīt neironu tīklus ar sarežģītām struktūrām.

Pētniecības rezultātos tika izstrādātas vairākas konvolūciju neironu tīklu struktūras, kur tika pielietotas jaunas metodoloģijas un dziļās mācīšanās principi. Tas deva iespēju palielināt neironu tīklu pielietošanas iespējas un precizitāti [2], [4], [7], [11]–[15].

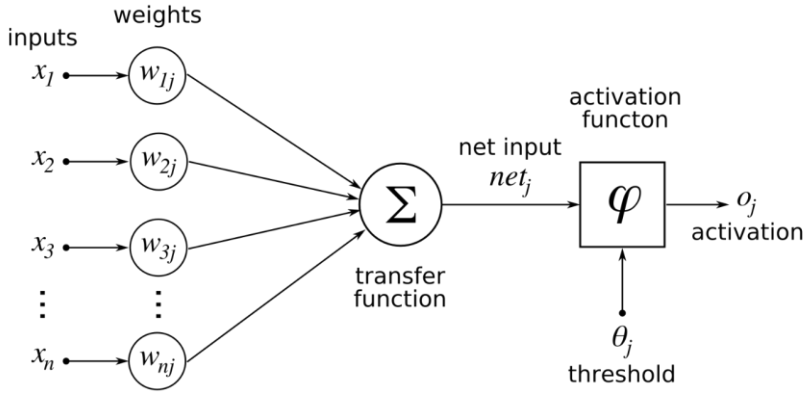
Neironu tīklu apmācīšanai tika sagatavotas vairākas apmācīšanas datu kopas, kuras ir pieejamas publiskai lietošanai. Šo datu kopu pieejamība veicināja neironu tīklu pētīšanu un jaunu risinājumu veidošanu [5], [16], [17].

Šobrīd mākslīgos neironu tīklus izmanto ļoti plašā jomu spektrā. Tos izmanto dažādās zinātņu jomās, mašīnmācīšanās, kognitīvās zinātnēs, balss atpazīšanā, attēlu aprakstīšanā, neiroloģijā, gēnu inženierijā, izmanto dažādu tirgu analizēšanā un svarīgo lēmumu pieņemšanā.

1.2. Neironu tīkla uzbūve un topoloģija

Mākslīgā neirona tīkla pamata sastāvdaļa ir neirons, kas ir salīdzinoši vienkāršs skaitļošanas elements un kas ir līdzīgs bioloģiskam neironam. Neironam var būt vairākas ieejas un izejas, kas atgādina daudzargumentu funkcijas darbību.

Skaitļošanas neirons sastāv no vairākām daļām: svariem (kas raksturo saišu "stiprumu" starp neironiem), summēšanas funkcijas, aktivizācijas funkcijas un zuduma funkcijas.



1.2. att. **Neironu tīkla summēšanas un aktivizācijas funkcijas darbības shēma** [18]

Neirona darbības shēma ir redzama attēlā 1.2, kur x_1, x_2, x_3, … , x_n ir neirona ieejas vērtības, w_{1j}, w_{2j}, w_{3j}, .. , w_{nj} ir neirona svari. Ar "Σ" simbolu ir apzīmēta summēšanas funkcija, ar "φ" simbolu ir apzīmēta aktivācijas funkcija, kur net_j ir summēšanas funkcijas rezultāts, O_j ir aktivācijas funkcijas rezultāts un Θ_j ir aktivizācijas funkcijas sliekšņa parametrs.

1.2.1. Svari un summēšanas funkcija

Svari ir skaitliskās vērtības, kas tiek definētas apmācības sākumā un tiek izmainītas neironu tīkla apmācības procesā. Šī svaru maiņa nodrošina to, ka apmācītais neironu tīkls var risināt vēlamo problēmu. Katrai neirona ieejai atbilst viens svars.

Lai no visām ieejas vērtībām iegūtu vienu skaitli, ņemot vērā svarus, tiek pielietota summēšanas funkcija. Tipiskākā summēšanas funkcija izskatās šādi:

$$net = \sum_{i=1}^{n} w_i x_i + b, \quad (1)$$

kur w_i ir svari, x_i ir ieejas vērtības un b ir nobīde (angl. val. *bias*).

Nobīde dod iespēju palielināt precizitāti un ņemt vērā svarus pat ja ieejas vērtība ir nulle, kā arī palīdz turēt neironu svarus diapazonā, kas var dot sagaidāmo rezultātu.

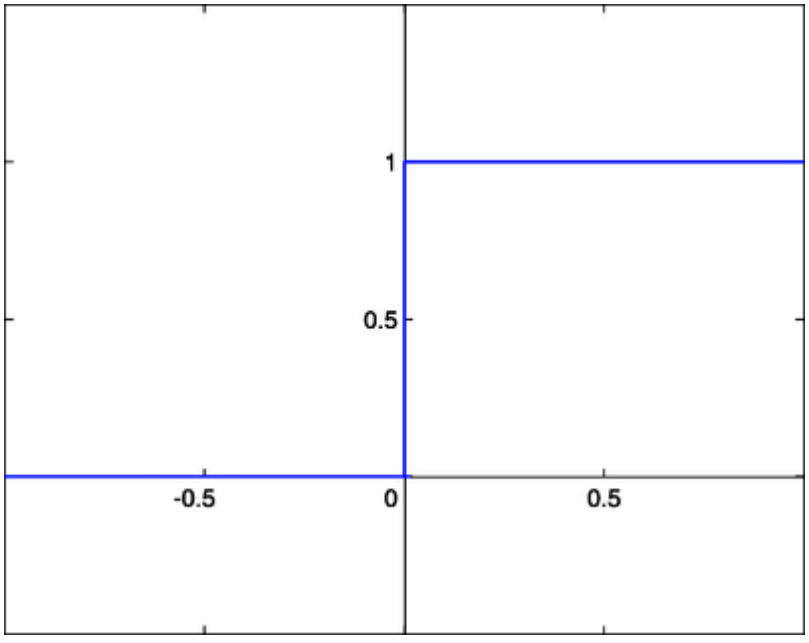
1.2.2. Aktivizācijas funkcija

Ar aktivizācijas funkcijas palīdzību, izmantojot summēšanas funkcijas rezultātu, tiek iegūta izejas vērtība. Aktivizācijas funkcija normalizē summēšanas rezultātu, un tās vērtības ir sadalītas noteiktā diapazonā. Visbiežāk šis intervāls ir no -1 līdz 1. Šī normalizācija dod iespēju novērtēt svaru pareizību un padot neirona izejas vērtību nākamā slāņa neironam.

Aktivizācijas funkcijas var būt ļoti dažādas, tā var būt sliekšņa funkcija, lineārā funkcija, sigmoidālā funkcija, vai daudzas citas. Funkciju izvēle var būtiski ietekmēt neironu tīkla precizitāti un apmācības ātrumu. Apskatīsim visizplatītākos.

Viena no vieglākām aktivizācijas funkcijām ir sliekšņa funkcija, kuras grafiks ir apskatāms attēlā *1.3*. Šī funkcija tika izmantota oriģinālā perceptronā. Šīs funkcijas darbības princips ir salīdzināt ieejas vērtību ar kaut kādu slieksni. Ja ieejas vērtība ir mazāka par slieksni, tad izvadā iegūst vienu vērtību A_0 un, ja ir lielāka, tad izejā iegūst citu rezultātu A_1. Visbiežāk šo rezultātu vērtības ir 0 un 1.

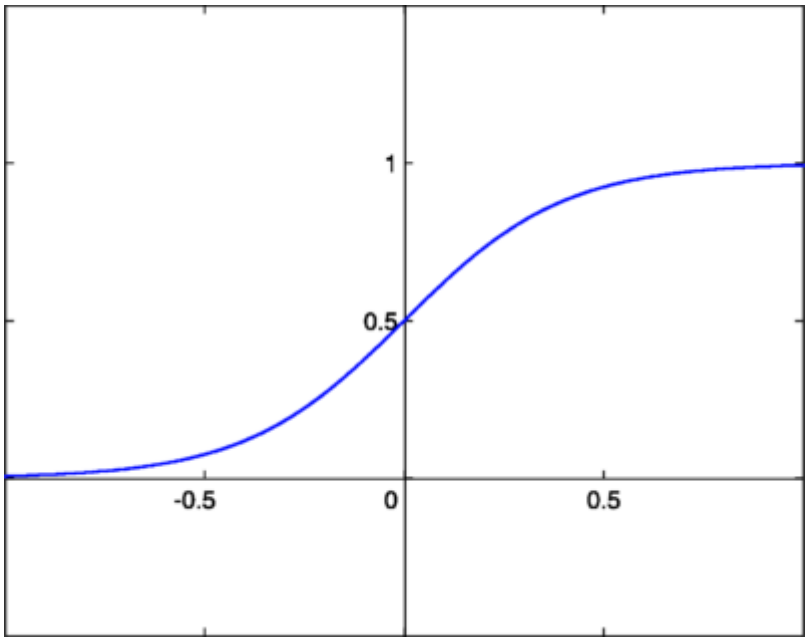
Sliekšņa aktivizācijas funkciju visbiežāk pielieto binārās klasifikācijas uzdevumos, kad ir nepieciešams sadalīt ieejas datus divās grupās. Cits veids, kur var izmantot sliekšņa aktivizācijas funkciju, ir pazīmju atpazīšanā, kur funkcija tiek pielietota neliela neironu tīkla beigās un izvada 0, ja pazīme ir atrasta un 1, ja pazīme nav atrasta.



1.4. att. **Sliekšna aktivizācijas funkcijas grafiks** [18]

Nepārtraukta sigmoidālā funkcija ir ļoti izplatīta neironu tīklu aktivācijas funkcija, un tiek izteikta ar formulu: $\sigma(t) = \frac{1}{1+e^{-\beta t}}$, kur β ir funkcijas slīpuma parametrs. Šī funkcija ir ļoti līdzīga sliekšņa funkcijai, bet tai ir nenoteiktības reģions. Tā kā sigmoidālai funkcijai ir

nenoteiktības reģions, tad to var pielietot jau daudz grūtākos uzdevumos, jo ar to var precīzāk novērtēt summēšanas funkcijas vērtību. Sigmoidālai funkcijai ir savas priekšrocības, jo tai ir viegli aprēķināt atvasinājumu, kas ir ļoti vērtīgi svaru izmaiņu aprēķināšanai dažādos apmācības algoritmos. Sigmoidālās funkcijas grafiks ir apskatāms attēlā *1.5.*



1.6. att. **Sigmoidālās aktivizācijas funkcijas grafiks** [18]

Eksistē vēl daudz citu aktivācijas funkciju, kur katra funkcija var būt vispiemērotākā kaut kādam uzdevumam.

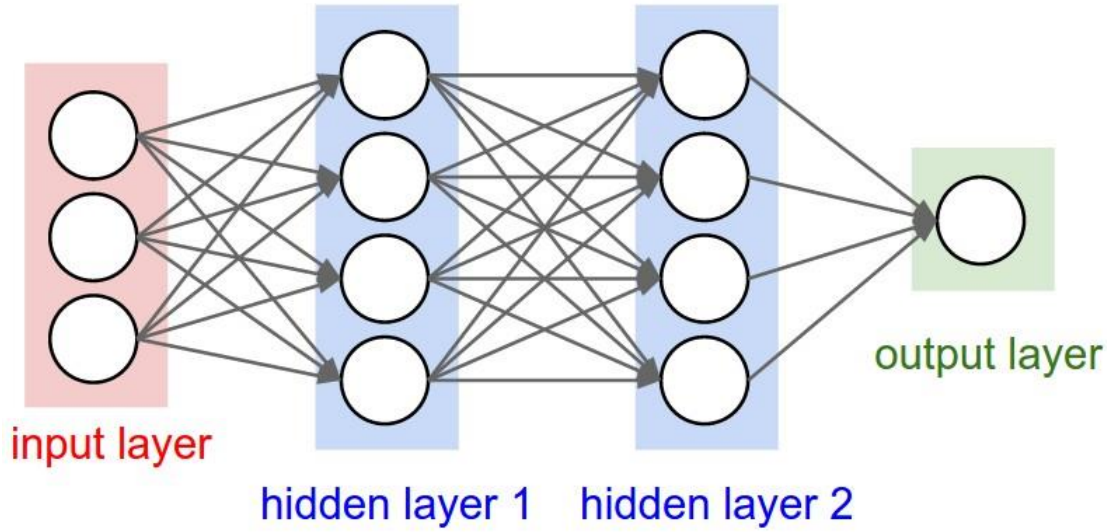
1.2.3. Zuduma funkcija

Ar zuduma funkcijas palīdzību neironu tīkla izejas informācija tiek salīdzināta ar sagaidāmo rezultātu. Ar zuduma funkcijas rezultātu var novērtēt, cik precīzi strādā apmācāmais neironu tīkls un, balstoties uz to, optimizēt neironu tīklu un to svaru vērtības. Ir zināmas vairākas standarta zuduma funkcijas un ir svarīgi izvēlēties atbilstošu uzdevumam funkciju vai definēt to pašam [19]. Neironu tīkla apmācības mērķis ir samazināt zuduma funkcijas rezultātu, tāpēc ir svarīgi izvēlēties tādu zuduma funkciju, kas atbilst izvēlētajai problēmai un to vērtība pareizi raksturotu neironu tīkla darbības precizitāti.

1.3. Neironu tīklu veidi

Neironu tīkli tiek sadalīti pēc to struktūru īpašībām, un ir zināmi vairāki neironu tīklu veidi. Visizplatītākie ir vienvirziena un rekursīvie neironu tīkli. Šajā darbā tiek apskatīti vienvirziena konvolūciju neironu tīkli, tāpēc apskatīsim tos.

Vienvirziena neironu tīkls sastāv no ieejas slāņa, slēptajiem slāņiem un izejas slāņiem. Pilnsaistes neironu tīkls ir vienkāršākais vienvirziena neironu tīkls, kurš sastāv no vairākiem slāņiem un kur katra iepriekšēja slāņa neirona izeja tiek padota katram neironam no nākamā slāņa, kas ir redzams attēlā *1.7.*



1.7. att. **Pilnsaistes neironu tīkla struktūra** [20]

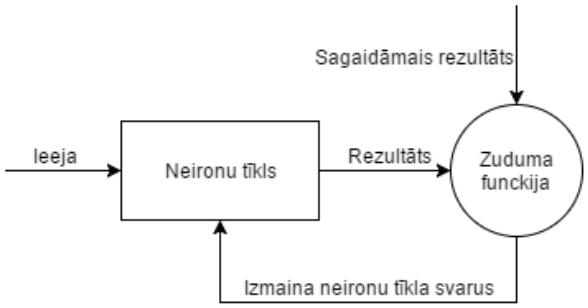
Ieejas slānī tiek padotas visa neirona tīkla ieejas. Piemēram, ja neironu tīkls apstrādās 10x10 pikseļu lielu melnbaltu attēlu, tad šim neironu tīklam ieejas slānī būs 100 neironi, kur katram tiks padotas atbilstošas pikseļu vērtības. Slēptie slāņi sastāv no vairākiem neironiem, kur katram tiek padotas ieejas vērtības no iepriekšējā slāņa. Ja tas ir pirmais slēptais slānis, tad ieejas vērtības tiek padotas no ieejas slāņa. Izejas slānis ir neironu slānis, kas izvada galējo neirona tīkla darbības rezultātu. Informācija tiek padota no viena slāņa uz nākamo, tikai vienā virzienā.

1.4. Neironu tīklu apmācības stratēģijas

Neironu tīklu ļoti svarīga sastāvdaļa ir to apmācīšanas stratēģija. Izvēlētās apmācības stratēģijas jeb apmācības likumi ietekmē visa neironu tīkla struktūras definēšanu, jo, lai apmācītu tīklu ar vienu no stratēģijām, ir nepieciešams to ņemt vērā tīkla struktūras definēšanā. Šajā nodaļā tiks apskatītas visas klasiskās neironu tīklu apmācības stratēģijas, kā arī šajā darbā apskatāmo vāji pārraudzīto mācīšanās metodi.

1.4.1. Pārraudzīta metode.

Galvenā pārraudzītas neironu tīkla mācīšanās metodes īpašība ir tāda, ka tiek definēts "mācītājs", kurš ir "gudrāks" par neironu tīklu. Neironu tīkls tiek apmācīts ar datu kopu, kur katram mācīšanās eksemplāram ir zināms sagaidāmais rezultāts. Piemēram apskatot objektu atpazīšanas problēmu, tad, lai apmācītu neironu tīklu, tam vajag padot attēlus ar vēlamajiem objektiem un pārraudzīt tā rezultātus salīdzinot ar sagaidāmo rezultātu [13], [21].



1.8. att. **Pārraudzītas mācīšanās metodes darbības shēma**

Pārraudzītas mācīšanās metodes darbību var skaidri redzēt shēmā, kas ir apskatāma attēlā *1.8*. Neironu tīkls ieejā saņem datus, kuriem ir zināms sagaidāmais rezultāts. Pēc tam, kad neironu tīkls izvada ieejas datu rezultātu, tas tiek salīdzināts ar sagaidāmo rezultātu, izmantojot zuduma funkciju. Bastoties uz zuduma funkcijas rezultātu, tiek izmainīti neironu tīkla svari, lai samazinātu zuduma funkcijas vērtību.

1.4.2. Nepārraudzīta mācīšanās metode

Nepārraudzīta neironu tīklu mācīšanās metode tiek pielietota, kad nav pieejama apmācības kopa ar pareiziem rezultātiem [22]. Nepārraudzīta mācīšanās metode var būt pielietota, lai sadalītu objektu kopu, balstoties uz objektu īpašībām, dažreiz pat nezinot šīs īpašības. Šīs metodes pielietošana ir perspektīva, jo tas dod iespēju neironu tīklam apmācīties ar datiem, kurus nav nepieciešams iepriekš sagatavot, tomēr šobrīd šī metode netiek plaši pielietota un attīstīta, jo tās darbība ir lēnāka nekā pārraudzītas metodes darbība [13], [15].

1.4.3. Vāji pārraudzīta mācīšanās metode

Vāji pārraudzīta metode izmanto pārraudzīto mācīšanās metodi, turklāt apmācībai tiek izmantota arī neaprakstīta vai vāji aprakstītā informācija [13].

Kā piemēru apskatīsim objektu atpazīšanas uzdevumu. Lai apmācītu neironu tīklu, šī uzdevuma risināšanai vajag izmantot parastu pārraudzīto mācīšanās metodi. Lai to apmācītu, ir nepieciešama ļoti liela apmācības datu kopa, jo neironu tīklam ir jāapmācās tā, lai tas atpazītu objektu visdažādākajos veidos. Taču galvenā problēma ir tāda, ka katram objektam, kas ir padots apmācīšanai, ir jābūt izgrieztam no attēla, lai neironu tīkls apmācītos tieši uz šo objektu, neņemot vērā visu, kas ir objektam apkārt. Iegūt vajadzīgo pamācības kopu dažreiz ir problemātiski. Ir pieejamas dažādas bibliotēkas ar aprakstītiem attēliem, ko var izmantot neironu tīklu apmācīšanai. Kā piemērs ir ImageNet bibliotēka [23]. Tomēr šī bibliotēka neatspoguļo visus iespējamos reālās pasaules objektu izskatus [15]. Tāpēc uz reālās dzīves bildēm ar noteiktu apmācības kopu apmācīts neironu tīkls mēdz kļūdīties, un dažādos uzdevumos rodas nepieciešamība papildināt apmācīšanas datu kopu, lai palielinātu precizitāti konkrētās situācijās. Šeit rodas divas iespējas. Pirmā ir sagatavot papildus kopu ar aprakstītiem attēliem neironu tīkla apmācīšanai. Tā kā objektu atpazīšanas problēmai ir nepieciešami tūkstoši attēlu, tad var secināt, ka jauno datu kopu sagatavošana ir ļoti laikietilpīgs un dārgs process, jo objektus vajag izgriezt un aprakstīt. Lai izveidotu ImageNet datubāzi, bija nepieciešams liels darba resursu apjoms: ImageNet izveides straujākā periodā pie tā strādāja 50000 cilvēku no 167 valstīm. Divu gadu laikā tika izveidota anotēta datubāze ar 15 miljonu attēliem un 22 tūkstošiem klasēm [3].

Otrs risinājums ir izmantot jau apmācīto neironu tīklu datu aprakstīšanai [12]. Šos datus varēs tālāk izmantot citu un lielāku neironu tīklu apmācīšanai. Turklāt šai pieejai nepieciešams aprakstīt tikai to, kādi objekti ir attēlā, bet nav nepieciešams tos izgriezt. Tas dod iespēju daudz vieglāk sagatavot apmācīšanas kopu, jo var izmantot jau pieejamus aprakstītus attēlus, kas plaši pieejami publiskai lietošanai internetā.

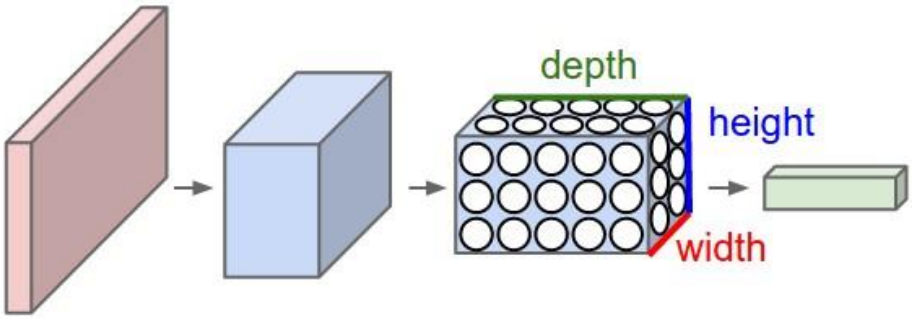
Šīs pieejas izmantošanu var nosaukt par vāji pārraudzīto metodi, jo sākuma tīkls tiks apmācīts ar parastu pārraudzīto apmācīšanas metodi un tālākā daļa tiek apmācīta ar vāji aprakstītiem attēliem, tā, ka tiem ir aprakstīti kādi objekti ir attēlā, bet nav aprakstīts kur šie objekti atrodas un kādi ir to objektu izmēri [11].

2. KONVOLŪCIJU NEIRONU TĪKLI

Konvolūciju neironu tīkli ir ļoti līdzīgi parastiem vienvirziena pilnsaistes neironu tīkliem. Tie ir izveidoti no neironiem, kuriem ir svari un nobīdes. Tāpat kā parasts neironu tīkls, konvolūciju neironu tīkli sastāv no vairākiem slāņiem un dod izejā rezultātu. Taču, ar ko konvolūciju neironu tīkli atšķiras no parastiem neironu tīkliem? Konvolūciju neironu tīkls ieejā saņem attēlus, kas dod iespēju apstrādāt specifiskās attēlu īpašības, un izmantot tās neironu tīklu darbības principos un to arhitektūrā, lai precīzāk spētu noteikt attēla objektus.

Pilnsaistes neironu tīkliem ir ierobežojums: apstrādājot pat nelielu krāsaino attēlu ar 200x200 pikseļu izmēru, ieejas neironu skaits būtu 200x200x3 = 120000, kas ir vienāds ar svaru skaitu. Nākamos slāņos svaru skaits var palielināties, tāpēc attēlu apstrādei parasti pilnsaistes neironu tīkli nav piemēroti, jo to struktūra ir pārāk liela un tik liels svaru skaits negarantē pareizu rezultātu un nozīmē, ka ir nepieciešams liels atmiņas resursu apjoms.

Savukārt konvolūciju neironu tīkli izmanto attēlu īpašības, un uzskata tos kā trīsdimensiju masīvus jeb tenzorus. Atšķirībā no parasta neironu tīkla, konvolūciju neironu tīkla slāņi tiek sakārtoti trīs dimensijās: platums, augstums un dziļums, kā var redzēt attēlā *2.1*. Šī īpašība dod iespēju apstrādāt iepriekšējā slāņa neironus ne visus kopā, bet pa nelielām grupām. Konvolūciju neironu tīkla slāņa neirons tiek saistīts ar nelielu iepriekšējā slāņa neironu kopu, kas ļauj būtiski samazināt neirona tīkla svaru skaitu.



2.1. att. **Konvolūciju neirona tīkla darbības principa shēma** [20]

Pēc būtības konvolūciju neironu tīkla slānis transformē vienu neironu apjomu citā. Visbiežāk ar konvolūciju slāņa palīdzību neironu svaru apjoms tiek samazināts, izdalot augstākā līmeņa attēla objekta pazīmes.

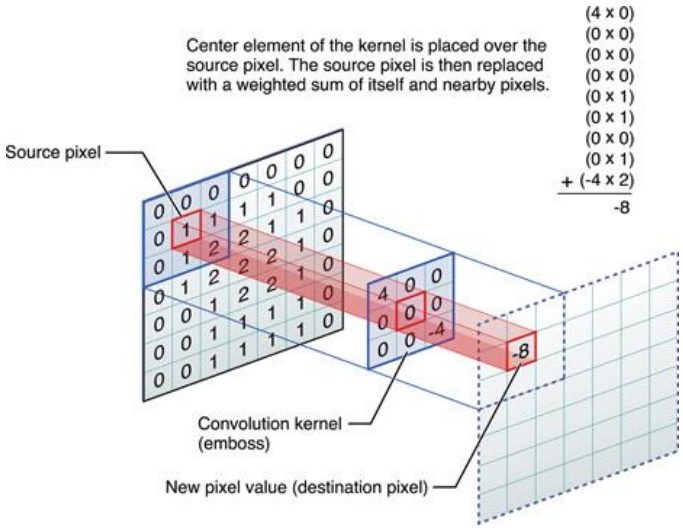
Konvolūcijas neironu tīkls sastāv no trīs slāņu tipiem: konvolūciju slānis (angl. val. *convolutional layer*), apvienošanas slānis (angl. val. *pooling layer*) un pilnsaistes slānis (angl. val. *fully connected layer*). Dažreiz pielieto vēl dažādus transformāciju slāņus, tādus kā normalizācijas slānis, izmešanas slānis (angl. val. *dropout*).

Šo slāņu kombināciju izmanto dažādu konvolūciju neironu tīklu arhitektūru definēšanai. Apskatīsim katru no tiem.

2.1. Konvolūciju slānis

Konvolūciju slānis ir galvenā konvolūciju neirona tīkla sastāvdaļa. Tas tiek izmantots, lai izdalītu objektu atpazīšanai nepieciešamās pazīmes. Pazīmju izdalīšana dod iespēju palielināt neironu tīklu darbības precizitāti salīdzinot ar neironu tīklu, kurš neizmanto konvolūciju slāņus [20]. Turklāt, izmantojot konvolūciju slāni, var samazināt vai palielināt neironu svaru skaitu, kā rezultātā palielinot neironu tīkla ātrdarbību.

Konvolūciju slāņa parametri, jeb svari, sastāv no apmācāmiem filtriem. Konvolūciju filtrs pēc būtības ir neliela kodola matrica, kas tiek izmantota attēla objektu pazīmju izdalīšanai. Filtra darbība ir līdzīga matricu reizināšanai, sākuma attēla reģioni tiek reizināti ar filtra matricu, rezultātā iegūstot vienu ciparu par katru sākuma attēla pozīciju. Konvolūcijas darbības princips ir redzams attēlā 2.2., kur tiek pielietota matricas reizināšana ar kodola matricu, kuras izmērs ir 3x3. Reizināšanas rezultātā tiek iegūts viens cipars. Konvolūcijas process ietver sevī kodola matricas reizināšanu ar attēla pozīcijām, kas tiek iegūtas ar noteiktu soli.



2.2. att. **Kodola matricas konvolūcijas pielietošana** [24]

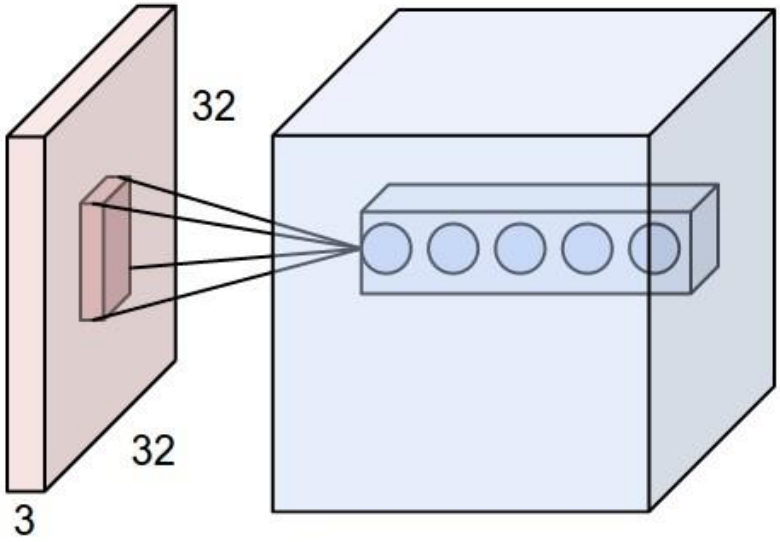
Tā kā konvolūcija var būt pielietota ne tikai ieejas attēlam, bet arī jebkuru citu slāņu rezultātiem, tāpēc tiek izmantoti trīsdimensiju konvolūciju filtri, kur tās dziļums sakrīt ar ieejas datu dziļumu. Piemēram, ja tiek apstrādāts krāsains attēls, tad lai pielietot tam konvolūciju ir nepieciešams filtrs ar dziļumu trīs, jo ieejas attēlam ir trīs kanāli, kas raksturo katra pikseļa krāsu.

Ir zināms, ka nav efektīvi katru neironu saistīt ar visiem iepriekšējā slāņa neironiem, tāpēc konvolūciju slāņa filtra telpisks izmērs (augstums un platums) visbiežāk ir daudz

mazāks nekā ieejas datu izmērs, lai varētu izdalīt objektu pazīmes no vairākiem ieejas datu reģioniem. Filtra augstumu un platumu sauc par konvolūciju slāņa uztveres lauku.

Ieejas datu apstrādei katrs filtrs tiek bīdīts pa attēlu ar noteiktu soli. Konvolūciju filtra pielietošanas rezultāts ir viens skaitlis, tāpēc apstrādājot trīs dimensiju ieejas matricu ar vienu filtru, rezultātā saņem matricu ar dziļumu viens. Lai izdalītu ieejas datu vairākas pazīmes, konvolūciju slānim tiek definēti vairāki filtri. Pielietojot vairākus filtrus, to rezultāti tiek sagrupēti vairākos slāņos, izejā saņemot trīsdimensiju matricu, jeb tenzoru ar dziļumu vienādu ar konvolūciju slāņa filtru skaitu. Visu konvolūciju slāņa filtru izmēri ir vienādi, jo filtru darbības rezultātā vajag iegūt vienādas divdimensiju matricas, lai tos sagrupētu vienā lielā trīsdimensiju izejas tenzorā.

Konvolūciju slāņa filtru vērtības reprezentē neironu tīkla svarus, un, apmācot konvolūciju neironu tīklu, tiek mainītas konvolūciju filtru vērtības. Apmācīšanas sākumā filtru vērtības tiek definētas nejaušā kārtībā un apmācības gaitā filtru vērtības tiek mainītas tā, lai izdalītu objektu specifiskās pazīmes. Sākuma vērtību nejaušība nodrošina to, ka apmācības laikā filtri tiek apmācīti dažādu pazīmju izdalīšanai.



2.3. att. **Konvolūciju slāņa darbības piemērs** [20]

Apskatīsim piemēru, kur tiek apstrādāts attēls ar izmēriem 32x32x3, kur 3 nozīmē, ka attēls ir krāsains un katram pikselim ir trīs krāsu kanāli. Konvolūcijas pielietošanas darbība ir redzama attēlā 2.3. Pieņemsim, ka konvolūciju slānis ir definēts tā, ka izejas dziļums ir 5. Ja tiek pieņemts, ka neironam ir 5x5 liels uztveres lauks, sanāk, ka šim konvolūciju neironu slānim ir 3x5x5x5 = 375 svari. Ir svarīgi, ka svaru skaits nav atkarīgs no ieejas attēla platuma un augstuma. Tā var skaidri redzēt, ka, izmantojot konvolūciju slāni, tiek būtiski samazināts neironu tīkla svaru skaits.

Izejas matricas izmēru ietekmē vairāki parametri: dziļums, solis (angl. *stride*), nulles papildināšana (angl. zero-padding), un neirona uztveres lauka izmērs.

Konvolūciju slāņa darbības rezultāta dziļumu nosaka konvolūciju tīkla struktūras definēšanas laikā, un tas ir vienāds ar filtru skaitu.

Solis nosaka, par cik soļiem tiek bīdīts filtrs jeb uztveres lauks. Kad solis ir vienāds ar 2, tad filtri tiek bīdīti par divām vienībām, samazinot pārklāšanos starp soļiem, arī samazinot izejas matricas izmēru.

Nulles papildināšana tiek izmantota, lai mākslīgi palielinātu ieejas matricas izmērus, pieliekot tās malās nulles vai neitrālās vērtības. Tas dod iespēju kontrolēt izejas matricas izmēru. Ar nulles papildināšanas izmantošanu ir iespējams palielināt izejas matricas izmēru, vai izejas matricai saglabāt ieejas matricas izmērus, kas var būt nepieciešams dažādām konvolūciju neironu tīklu struktūrām.

Izejas trīsdimensiju matricas izmēru var izrēķināt ar sekojošu formulu:

$$W_2 = (W_1\text{-}F+2P)/S+1,$$

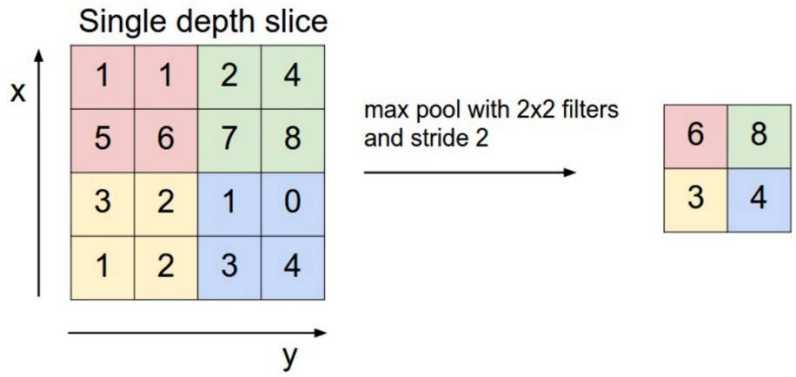
kur W_1 ir ieejas matricas platums, W_2 ir izejas matricas platums, F ir uztveres lauka izmērs, S ir solis, kas tiek pielietots filtru bīdīšanai un P ir nulles papildināšanas izmērs. Ir svarīgi, lai šis formulas aprēķināšanas rezultāts būtu vesels skaitlis. Lai to panāktu, var mainīt parametrus, kamēr tiek sasniegts vēlamais rezultāts. Dažreiz tieši tādā situācijā ir lietderīgi pielietot nulles papildināšanu.

Tādi paši aprēķini ir pielietojami izejas matricas augstumam. Savukārt izejas matricas dziļums sakrīt ar iepriekš noteikto dziļumu.

2.2. Apvienošanas slānis

Konvolūciju neironu tīklu struktūrās gandrīz vienmēr pielieto apvienošanas slāni pēc konvolūciju slāņa pielietošanas. Apvienošanas slāņa mērķis ir samazināt datu dimensijas, ar to paātrinot neirona tīkla darbību, kā arī samazinot iespēju, ka neironu tīkls tiks pārapmacīts(svari būs tik pielāgoti apmācības kopai, ka nevarēs korekti strādāt ar datiem, kuri nebija apmācības kopā). Ļoti bieži izmanto apvienošanu pēc maksimuma, kur no uztveres lauka tiek ņemts lielākais elements. Dažreiz apvienošanas slānis aprēķina vidējo vērtību elementiem, kas ir uztveres laukā.

Apvienošanas slāņa darbība ir savā ziņā līdzīga konvolūciju slāņa darbībai. Apvienošanas slānim ir uztveres lauks, uz kura tiek pielietota apvienošanas slāņa operācija. Apvienošanas pēc maksimuma darbības princips ir redzams attēlā 2.4., kur kreisajā pusē ir attēlota apvienošanas slāņa ieeja un labajā pusē ir attēlots apvienošanas slāņa rezultāts.



2.4. att. **Apvienošanas pēc maksimuma darbības princips** [25]

Visbiežāk tiek pielietots apvienošanas slānis ar filtra izmēru 2x2 un soli 2. Sanāk, ka no katriem četriem pikseļiem tiek izvēlēts viens - vislielākais, ar to samazinot informācijas daudzumu par 75%. Apvienošanas slānis darbojas uz katru ieejas datu dziļuma slāni atsevišķi, tāpēc pielietojot apvienošanas slāni, ieejas datu dziļums netiek samazināts, tiek samazināti tikai datu telpiskais izmērs.

2.3. Pilnsaistes slānis

Pilnsaistes slānis tiek pielietots konvolūciju neironu tīkla beigās, sagrupējot iegūto informāciju no konvolūciju un apvienošanas slāņiem. Šis slānis ir pēdējais un izvada ieejas attēla aprakstu. Pilnsaistes neironu slāņi ir ļoti līdzīgi konvolūciju slāņiem. Vienīgā atšķirība ir tāda, ka konvolūciju slānim neironi ir savienoti ar lokālu reģionu no ieejas datiem, savukārt pilnsaistes slāņa neirons ir savienots ar visiem ieejas datiem.

Zinot šīs atšķirības, pilnsaistes slāni var reprezentēt kā konvolūciju slāni, kuram uztveres lauka izmērs ir vienāds ar ieejas matricas izmēru.

2.4. Konvolūciju neironu tīkla apmācība

Apskatīsim neironu tīklu, kurš klasificē ievadattēlu, kura izejas vērtības parāda, kas atrodas uz attēla.

Sākumā neirona tīklam tiek definēti svari. Visbiežāk svaru vērtības tiek izvēlētas nejauši. Tad neironu tīklam tiek padoti apmācības dati, neironu tīkls aprēķina rezultātu balstoties uz tā svariem. Tālāk šis rezultāts tiek novērtēts ar zuduma funkciju, salīdzinot to ar sagaidāmo rezultātu. Zuduma funkcijas rezultāts būs mazāks, ja tīkla rezultāts ir tuvāks sagaidāmajam rezultātam.

Ir vairāki veidi, kā definēt zuduma funkciju. Viena no visbiežāk izmantojamām zuduma funkcijām ir vairākklašu atbalsta vektoru mašīnas zuduma funkcija (no angl.v. *Multiclass Support Vector Machine*), tālāk apzīmēsim to kā SVM zuduma funkciju. SVM zuduma funkciju var aprēķināt pēc formulas:

$$L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + \Delta),$$

kur s_j ir nepareizo klašu rezultāts, s_{yi} ir pareizās klases rezultāts un Δ ir atļauta kļūda [26].

Vēl viena bieži izmantojama zuduma funkcija klasifikācijas uzdevumiem ir Softmax klasifikators, kuram ir cita zuduma funkcija. Softmax klasifikators atšķirībā no SVM zuduma funkcijas izvada atbilstošu klašu varbūtības. Softmax klasifikatora zuduma funkcijas vērtību var aprēķināt pēc formulas:

$$L_i = -f_{y_i} + log \sum_j e^{f_j},$$

kur f_{y_i} ir sagaidāmā jeb pareizā rezultāta vērtība un $f_j$ir katras klasifikatora klases vērtība [26].

Izmantojot zuduma funkciju tiek aprēķināta neirona tīkla precizitāte konkrētam paraugam. Zinot šo precizitāti, var izmainīt neironu tīkla svarus tā, lai tas prognozētu rezultātu tuvāku realitātei.

Neironu tīkla optimizācijas mērķis ir samazināt zuduma funkcijas vērtību. Zinot pašreizējo neironu tīkla zuduma funkcijas vērtību, pieņemot, ka svaru izmaiņas ir tuvu nullei, var precīzi aprēķināt zuduma funkcijas gradientu, ar to arī uzzinot, kā mainīt neironu tīkla svarus, lai samazinātu zuduma funkcijas vērtību [27].

3. MAŠĪNMĀCĪŠANAS BIBLIOTĒKAS

Šodien ir pieejamas vairākas bibliotēkas mākslīgā intelekta un mašīnmācīšanās risinājumu izveidošanai. Šajā sadaļā tiek aprakstītas un salīdzinātas populārākās dziļās mācīšanās bibliotēkas. Šīs nodaļas mērķis ir izvēlēties atbilstošu bibliotēku darba problēmas risināšanai.

Tiks apskatītas piecas populārākās mašīnmācīšanās bibliotēkas uz doto brīdi: Caffe, CNTK, TensorFlow, Theano un Torch [28].

3.1. Caffe

Caffe ir viena no pirmajām bibliotēkām, kas piedāvāja plaši pielietojamu, augstas kvalitātes neironu tīklu instrumentu kopu. Šī bibliotēka parādījās 2013. gadā un uz to brīdi šajā bibliotēkā konvolūciju tīklu risinājumi bija vērtējami kā ļoti labi funkcionējoši. Tā kā šī bibliotēka ir viena no pirmajām, kas parādījās tirgū, tad tā joprojām ir viena no plašāk pielietojamākajām bibliotēkām konvolūciju tīklu risinājumos. Tomēr šī bibliotēka piedāvā diezgan sliktu atbalstu rekursīviem neironu tīkliem un valodas modelēšanai. Caffe bibliotēka piedāvā tikai "pycaffe" saskarni, bet tā nav labāka kā komandrindas saskarne. Neironu tīklu modeli definē parastā teksta redaktorā un tālāk to palaiž komandrindā [29]. No tā var secināt, ka Caffe bibliotēka ir grūti pielietojama, tomēr tā kā tā ir balstīta uz C++ programmēšanas valodu, šī bibliotēka var būt nokompilēta uz visdažādākajām ierīcēm un var tikt pielietota dažādās platformās. Viena no Caffe bibliotēkas priekšrocībām ir tās darbības ātrums. Viss ko Caffe bibliotēka spēj veikt, notiek ātri un efektīvi izmantojot pieejamus resursus [30].

Caffe bibliotēkas arhitektūras kvalitāte šobrīd tiek uzskatīta par vidējo. Šīs bibliotēkas pamata struktūrvienība ir slānis, kas ievieš vairākus ierobežojumus, jo šos slāņus ir grūti definēt un izmantot. Šī arhitektūra ierobežo bibliotēkas uzlabošanu un pielāgošanu mūsdienīgām vajadzībām [28].

Kopumā var teikt, ka Caffe bibliotēka ir pielietojama šaura loka uzdevumiem. Ņemot vērā tās arhitektūras ierobežojumus, var pieņemt, ka bibliotēkas būtiska uzlabošana nenotiks. Neskatoties uz to, bibliotēka piedāvā lielisku ātrdarbību, stabilitāti un iespēju to izmantot uz vairākām ierīcēm un platformām.

3.2. CNTK

CNKT ir mašīnmācīšanās bibliotēka no Microsoft. Tā visbiežāk tiek pielietota runas atpazīšanai un modelēšanai [31]. Šīs bibliotēkas struktūra tiek definēta kā vektoru operāciju simboliskā diagramma. Operācijas var būt tādas kā matricu reizināšana vai konvolūcija. Izmantojot šīs operācijas, lietotājs var viegli definēt jaunus slāņu tipus. CNTK bibliotēkas saskarne ir ļoti līdzīga Caffe saskarnei - tiek definēts konfigurācijas fails, kas tālāk tiek iedarbināts caur komandrindu. Situāciju pasliktina tas, ka CNTK bibliotēkai nav pieejams augsta līmeņa programmēšanas valodu interfeiss, tāds kā Python [28].

CNTK ir balstīts uz C++ un ir pielietojams dažādās ierīcēs un platformās. CNTK bibliotēka piedāvā ļoti labu veiktspēju uzdevumiem, kurus tā var risināt. SNTK bibliotēkas pozitīvais aspekts ir tās spēja dot vieglu izmantojamību vairāku video procesoru paralēlai datu apstrādei [32].

Kopā var pateikt kā CNTK bibliotēkai ir daudz priekšrocību, bet tās pielietošanas loks šobrīd ir šaurs un tā saskarne ierobežo efektīvu risinājumu izstrādi.

3.3. Tensorflow

Tensorflow ir saskarne mašīnmācīšanās algoritmu definēšanai un šo algoritmu realizācijas kopums [33]. Tensorflow bibliotēka piedāvā ļoti labas modelēšanas iespējas. Tā izmanto simbolisko vektoru operāciju grafa pieeju informācijas apstrādes procesa definēšanai, kas salīdzinoši viegli dod iespēju aprakstīt jaunu neironu tīklu pat ar ļoti sarežģītu struktūru..

Tensorflow bibliotēka atbalsta divas programmēšanas valodas, tā ir uzrakstīta ar Python API, kas balstās uz C/C++ kodolu. Tas padara bibliotēkas darbību ātru. Divu valodu pieejamība dod iespēju ātri un efektīvi modelēt neironu tīklus izmantojot augsta līmeņa vidi, un, ja ir nepieciešama integrācija ar citām sistēmām, vai īpaša ātrdarbība, tad var izmantot C++ saskarni [34].

Tā kā bibliotēka atbalsta C++ saskarni, tad to var optimizēt un kompilēt jebkurām ierīcēm. To ir iespējams darbināt uz jebkura mēroga ierīcēm sākot ar ARM procesoru ierīcēm un beidzot liela mēroga sistēmām ar simtiem ierīču un tūkstošiem videokaršu. Tensorflow sistēma ir viegli pielāgojama un var būt pielietota vairāku dziļās mācīšanās algoritmu definēšanai, iekļaujot dziļo neironu tīklu modeļu apmācības un darbošanās algoritmus. Pagaidām TensorFlow neatbalsta Windows operētājsistēmu.

Šobrīd Tensorflow neatbalsta tiešas matricu operācijas, bet prasa tās kopēt, lai tās apstrādātu. Šī problēma ļoti palēnina bibliotēkas darbību un padara to četras reizes lēnāku salīdzinot ar citām bibliotēkām, ja apstrādē tiek izmantotas matricu operācijas [28], [35].

3.4. Torch

Torch ir skaitļošanas ietvars, kas ir uzrakstīts uz Lua programmēšanas valodas un kurš atbalsta mašīnmācīšanās algoritmus. Šīs bibliotēkas versijas savos risinājumos izmanto vairākas lielas kompānijas, tādas kā Google DeepMind un Facebook [36].

Lai definētu neironu tīklu, Torch bibliotēkā tiek izmantots slāņu grafs. Tas palīdz ātri un viegli definēt tīklu ar jau eksistējošiem slāņiem, bet sakarā ar rupju slāņu grafa detalizācijas pakāpi, ir diezgan grūti pievienot jaunus slāņu tipus. Tomēr jauno slāņu definēšana ir daudz vieglāka nekā Caffe bibliotēkā, jo tam tiek izmantota Lua, nevis C++ programmēšanas valoda. Torch bibliotēka tiek uzskatīta kā izcils variants konvolūciju neironu tīklu implementēšanai [28].

Tā kā Torch bibliotēka balstās uz LuaJIT, tā ātrdarbība ir ļoti liela (piemēram, salīdzinot ar C++). Lua programmēšanas valoda dod iespēju veikt skaitļošanu nedomājot par iespējamu ātrdarbības samazināšanu, jo par to rūpējas LuaJIT kompilators. Lua programmēšanas valoda dod daudz iespēju un palielina ātrdarbību, bet tomēr pagaidām tā nav ļoti izplatīta, tāpēc Torch bibliotēkas pielietošanas izplatīšana var būt ierobežota.

Tā kā Torch bibliotēka nepiedāvā citas valodas izņemot Lua, to pielietošana un integrēšana lielākos projektos ir ļoti sarežģīta un dažādos gadījumos nav iespējama, kas ir liels mīnuss šai bibliotēkai, jo neironu tīklu darbība gandrīz vienmēr ir saistīta ar integrāciju ar citām sistēmām.

Vēl Torch bibliotēka no citām bibliotēkām izceļas ar savu ātrdarbību [35]. Tās darbība ir ātra un tai nav ātrdarbības ierobežojumu, ar kuriem saskaramies, piemēram, Tensorflow bibliotēkā. Turklāt bibliotēkai ir pieejami ļoti daudz jau apmācītu neironu tīklu modeļi.

Kopumā var teikt, ka Torch bibliotēka ir ļoti laba salīdzinoši ar citām apskatītajām bibliotēkām, tomēr, tā kā tā balstās uz Lua programmēšanas valodu, ir ierobežojumi ar šīs bibliotēkas integrāciju citās sistēmās.

4. PRAKTISKAIS DARBS

Darba praktiskai daļai tika definēti vairāki uzdevumi. Sākumā tiks apskatīti esošie konvolūciju neironu tīklu risinājumi, kas balstās uz vāji pārraudzīto mācīšanās metodi. Balstoties uz tiem, tiek definēts vāji pārraudzītas mācīšanās metodes darbības princips, kā arī tiks izstrādāta neliela konvolūciju neironu tīkla struktūra, kas spētu mācīties, izmantojot vāji pārraudzīto mācīšanās metodi, un klasificēt objektus starp 20 lielām klasēm, kuri ietver sevī lielu objektu klašu skaitu.

Vēlamais rezultāts tiks sasniegts, mēģinot apmācīt nelielu konvolūciju neironu tīklu ar neliela izmēra attēliem. Ņemot vērā konvolūciju neironu tīklu struktūras izstrādes un apmācības datu sagatavošanu, process ir ļoti laikietilpīgs, kā arī to, ka lielā konvolūciju neironu tīkla apmācības process var aizņemt vairākās nedēļas, tika pieņemts lēmums darba ietvaros izstrādāt nelielu konvolūciju neironu tīklu struktūru. Tas dod iespēju veikt neironu tīkla apmācību pietiekoši ātrā laika periodā un saņemt gatavus rezultātus šī darba izstrādes laikā.

Neironu tīkla apmācīšanas veiksmes gadījumā, tas dos iespēju ļoti ātri klasificēt attēla objektus starp lielām klasēm, kas var tikt, izmantots citu neironu tīklu ātrdarbības uzlabošanai, jo, zinot objekta lielo klasi, tālāko atpazīšanu var veikt ar neironu tīklu, kas spēj atpazīt objektu klases, kuras ietver atpazīta lielā klase.

Autors apzinās, ka neliels konvolūciju neironu tīkla ieejas attēla izmērs var neļaut izmantot vāji pārraudzīto mācīšanās metodi un apmācīšana var beigties neveiksmīgi. Tomēr ņemot vērā to, ka šobrīd nav zināms neviens likums neironu tīkla struktūras definēšanai noteiktam uzdevumam, pārbaudīt vai neironu tīkla struktūra ir veiksmīga var tikai eksperimentālā veidā.

Šī darbā ietvaros tiks izstrādāta neironu tīkla struktūra ar 2 konvolūciju slāņiem un 2 adaptācijas slāņiem, kas spētu apmācīties, izmantojot vāji aprakstītus attēlus. Neironu tīkla izstrādes laikā tiks izmantotas divas datubāzes neironu tīkla apmācīšanai. Otrās datubāzes izmantošanai tiks izstrādāts datu sagatavošanas rīks, kas tiks izmantots apmācības datu sagatavošanai priekš vāji pārraudzītas mācīšanās metodes.

4.1. Neironu tīklu bibliotēkas izvēle

Apskatot vairākas mašīnmācīšanās bibliotēkas, tika atrasti to trūkumi un priekšrocības. Izvēloties bibliotēku tika ņemti sekojoši faktori:

1) Bibliotēkas attīstības iespējas, jo ir svarīgi, lai izstrādātu neironu tīklu varētu pilnveidot nākotnē.
2) Bibliotēkas arhitektūra – tai jābūt saprotamai un viegli izmantojamai
3) Risinājumu, kas ir izstrādāts ar izvēlēto bibliotēku, ir jāspēj integrēt citā lielākā sistēmā.
4) Izstrādājamas bibliotēkas veiktspēja

Caffe bibliotēka tika izslēgta no izvēles saraksta, jo tās attīstīšanās iespējas ir ierobežotas, ko ietekmē tās arhitektūra, turklāt tai ir neērta saskarne.

CNTK bibliotēka netika izvēlēta, jo tās pielietošanas loks ir šaurs un tā nav īsti piemērota konvolūciju neironu tīklu rādīšanai.

Torch bibliotēkai ir visvairāk priekšrocību, jo tā ir ātra, ar ērtu arhitektūru un viegli izmantojamu saskarni valodā Lua. Tomēr tās pielietošana citā sistēma ir apgrūtināta, jo tā nevar strādāt, kā daļa no lielākas sistēmas, un to nevar integrēt citos risinājumos. To izmantošana var būt pamatota akadēmiskos pētījumos, bet, tā kā darbā izstrādājams neironu tīkls tālāk tiks izmantot lielākā sistēmā, šī bibliotēka nav labākais izvēles variants.

Izņemot Caffe, Torch un CNTK bibliotēkas no izvēles saraksta paliek TensorFlow bibliotēka. Tai ir daudz priekšrocību: pirmkārt, tai ir dinamiska arhitektūra, ko var izmantot vairākos risinājumos. Otrkārt, tā ir izstrādāta c++ valodā un turklāt tai ir interfeisa valoda Python, kas ļauj veikt ātru izstrādi, kā arī tas dod iespēju, risinājumus, kas tika izveidoti uz Tensorflow bibliotēkas pamata, viegli integrēt lielākā sistēmā. Tensorflow bibliotēkas lielākais trūkums, ir tās ātrdarbība. Šī bibliotēka ir vislēnākā no apskatītajām, tomēr ņemot vērā to, cik strauji attīstās šī bibliotēka, tās veiktspējas problēmas laika gaitā tiks atrisinātas, par ko jau ir paziņots bibliotēkas oficiālajos resursos [37]. Neskatoties uz veiktspējas trūkumiem, šī bibliotēka atbilst visām izvirzītajām prasībām. Šī bibliotēka tiks izmantota šī darba konvolūciju neironu tīkla izstrādei. Neironu tīkls tiks definēts izmantojot Python valodu.

4.2. Neironu tīkla struktūra

Šajā nodaļā tiks aprakstīta izveidotā neironu tīkla struktūra objektu atpazīšanai neatkarīgi no objekta atrašanās vietas uz apmācības attēla. Neironu tīkla struktūras definēšana tiek balstīta uz vairākiem darbiem, kas risināja līdzīgu problēmu, vai kuru rezultātus var izmantot kā daļu no šī tīkla [11], [12], [14], [38]–[41].

4.2.1. Neironu tīkla struktūras izveidošanas principi

Neironu tīkla struktūras izveidei šobrīd nav definētu noteikumu, tomēr, apskatot vairāku pētnieku un neironu tīklu izstrādātāju darbus, var definēt vairākas vadlīnijas neironu tīklu izveidē.

Jebkura neironu tīkla izveidē pirmais, kas jāņem vērā, ir neironu tīkla slāņu skaits. Ir zināms un pierādīts, ka, ja neironu tīklam ir tikai viens slēptais slānis, tad tas spēj risināt tikai lineāras funkcijas, kas nav pietiekoši attēlu atpazīšanas uzdevumos. Teorētiski, ja neironu tīklam ir viens slēptais slānis, tad šo tīklu var jau izmantot gandrīz jebkuriem uzdevumiem, tajā skaitā arī attēlu atpazīšanas uzdevumos.

Ja neironu tīklam ir vairāk slēpto slāņu, tad tas teorētiski nemaina neironu tīkla iespējas, bet praktiski var palīdzēt optimizēt neironu tīkla ātrdarbību, precizitāti un izmantot vairākus paņēmienus, kurus var realizēt tikai ar vairākiem neironu tīklu slāņiem.

Konvolūciju neironu tīklu gadījumā slēpto slāņu skaits var būt daudz lielāks, jo tiek izmantotas slāņu grupas: konvolūciju slānis iet kopā ar apvienošanas pēc maksimuma slāni un dažreiz ar normalizācijas slāni. Tomēr ne visi šie slāņi ir apmācāmi, un tāpēc šo slāņu kopas var uzskatīt par vienu "loģisku slāni".

Konvolūciju slāņu skaitu izvēlas pēc apmācāmās datu kopas sarežģītības un apmācāmo objektu klašu atšķirības vienai no otras. Ja ir nepieciešams neironu tīkls, kas spētu atpazīt 10 objektu klases, tad pietiek ar 1-2 konvolūciju slāņiem, bet, ja ir nepieciešams, lai neironu tīkls varētu atpazīt 1000 klašu objektus, tad ir nepieciešama jau lielāka neironu tīkla struktūra. Tā piemēram GoogleNet dziļās mācīšanās neironu tīkls sastāv no 21 konvolūciju neironu līmeņiem, kur katrs līmenis sastāv no vairākiem paralēliem konvolūciju slāņiem [4].

Vienā no pēdējām publikācijām no Google pētniekiem par konvolūciju neironu tīkliem, ir definētas vairākas vadlīnijas, kuras var izmantot sava neironu tīkla struktūras izveidē [42]. Dažas no vērtīgajām vadlīnijām no šī darba ir:

1) Sastādot konvolūciju neironu tīkla struktūru, ir jāņem vērā, lai apstrādes laikā neveidotos vājās vietas, kur apstrāde ir daudz lēnāka. Ir jāpieturas pie tā, ka ar katru

slāni ir jāsamazina pazīmju skaits, kas dos iespēju efektīvi izmantot neironu tīkla struktūru.

2) Pēc iespējas izmantot mazākus uztveres laukus konvolūciju slāņiem. Labāk izmantot divus secīgus konvolūciju slāņus ar 3x3 uztveres lauku, nekā vienu konvolūciju slāni ar 6x6 uztveres lauku. Tas palīdz samazināt neironu skaitu, kas savukārt palielina neironu tīkla ātrdarbību.
3) Ja datu apjoms ir pietiekoši liels, tad neironu tīkla precizitāte var būt palielināta ar neironu tīkla dziļumu (secīgo konvolūciju slāņu skaita) un neironu tīkla plašumu (paralēlu konvolūciju slāņu skaita). Neironu tīklu dziļumu un plašumu ir nepieciešams nobalansēt, ņemot vērā skaitļošanas resursu pieejamību. Kaut gan šī optimizācija nav pielietojama visos gadījumos, un to vajag pielietot tikai nepieciešamības gadījumā kā arī pārbaudīt tās darbības efektivitāti.
4) Var izmantot konvolūciju sāņus ar 1x1 uztveres lauku, lai samazinātu dimensiju skaitu, ar to arī optimizēt neironu tīklu darbību. Turklāt šī pieeja dod iespēju samazināt dimensiju skaitu būtiski nezaudējot konvolūciju slāņu informācijas reprezentācijas precizitāti, kas palīdz neironu tīklam apmācīties ar mazāku iterāciju skaitu.

Praktiskā darba ietvaros neironu tīkla struktūras definēšanā tika ņemtas šīs vadlīnijas.

4.2.2. Neironu tīkla struktūras apraksts

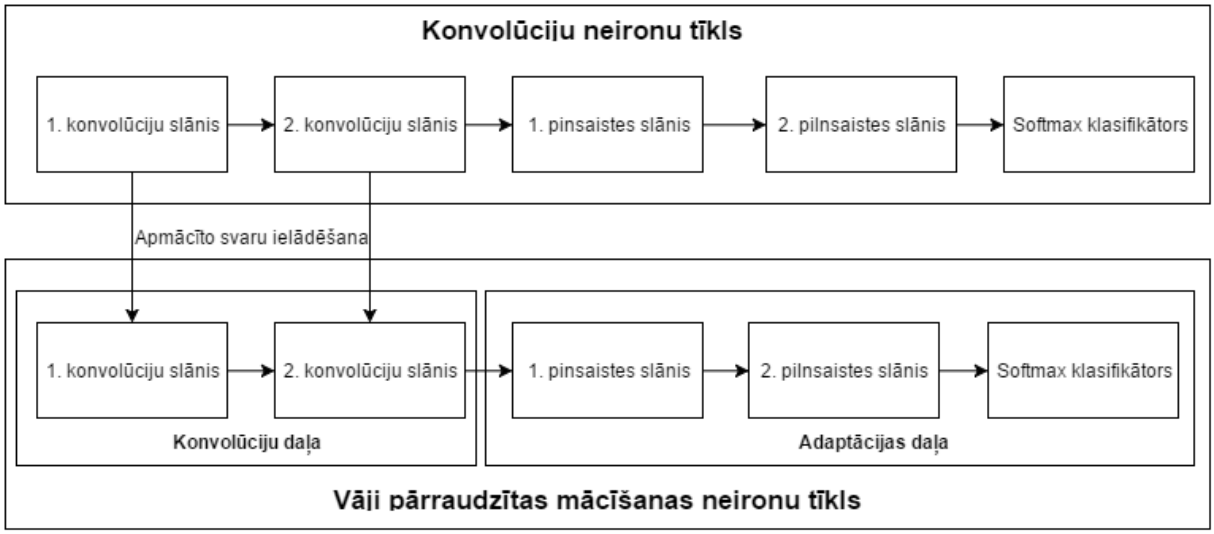
Objektu atpazīšanai šobrīd izmanto konvolūciju neironu tīklus. Tie tiek apmācīti ar objektu apmācīšanas kopām un katrs objekts tiek cieši izgriezts pirms padošanas apmācībai. Lai šis tīkls atpazītu objektu attēlā, kurā ir vairāki objekti, ar lielu precizitāti, tiek izmantota slīdošā loga metode. Attēlu, kuru nepieciešams apstrādāt un atpazīt tā objektus, padod neironu tīklam pa daļām. Tiek ņemts kāds "loga" izmērs, un ar šo loga izmēru no pamatattēlā tiek izgriezta mazāka attēla daļa. Šis mazākais attēls tiek padots neironu tīklam, un tās atpazīšanas rezultāts tiek pieglabāts. Tālāk šis logs tiek pabīdīts par noteiktu soli un jaunais attēls tiek apstrādāts ar neironu tīklu. Tādā veidā tiek apstaigāts viss sākuma attēls. Šis process tiek atkārtots vairākas reizes ar dažādiem "logu" izmēriem. Šī pieeja nodrošina, ka neironu tīkls varēs atpazīt objektus, kuri ir dažādos izmēros un atrodas dažādās attēla vietās.

Kad tiek apstrādāts viss attēls ar vairākiem logiem, tiek sakrāta informācija par iespējamiem objektiem katrā loga pozīcijā. Izmantojot šo informāciju, var noteikt visticamākos atpazītos objektus un to atrašanās vietu attēlā.

Šo pieeju var izmantot arī neironu tīkla apmācīšanā. Tradicionālā konvolūciju neironu tīkla struktūra to neatļauj – ir nepieciešams ieviest papildus slāņus, lai sasniegtu tīkla

apmācīšanu ar attēliem, kur var būt vairāki objekti, kā arī izmainīt apmācības metodiku. Vāji pārraudzīta metode nozīmē ierobežotu pilnībā aprakstītu datu izmantošanu.

Lai sasniegtu neironu tīkla veiksmīgo apmācīšanu, tas tiks sastādīts no divām daļām: konvolūciju daļu un adaptācijas daļu [11]. Konvolūciju daļa, tiek izmantota objektu pazīmju izdalīšanai, savukārt adaptācijas daļa tiek izmantota, lai no izdalītajām pazīmēm saņemtu objektu sarakstu uz attēla. Abu daļu shēma ir redzama attēlā *4.1.* Shēmas augšējā pusē ir attēlots konvolūciju neironu tīkls, kura konvolūciju slāņi tiek izmantoti vāji pārraudzītas mācīšanās neironu tīkla sākumā. Neironu tīkla otrā daļa sastāv no trijiem adaptācijas slāņiem, kuri tiek izmantoti konvolūciju slāņa izvada adaptācijai jauniem ievaddatiem.



4.1. att. **Neironu tīkla abu daļu shēma**

Lai veiksmīgi apmācītu neironu tīklu ar vāji pārraudzīto mācīšanās metodi, ir nepieciešams sadalīt apmācību: neironu tīkla daļas tiks apmācītas atsevišķi. Konvolūciju daļa tiek apmācīta neatkarīgi no adaptācijas daļas, bet otrā daļa tiks apmācīta, izmantojot pirmās daļas rezultātus.

Pirmā neironu tīkla daļa kalpo, kā objektu specifisko pazīmju izdalītājs. Šī daļa ir nepieciešama objektu atpazīšanai. Objektu specifisko pazīmju izdalīšanai vislabāk der standarta konvolūciju neironu tīkls.

Objektu pazīmju izdalīšanas neironu tīkla daļu nav iespējams apmācīt ar vāji aprakstītiem attēliem, jo objektu atrašanās vietas nav zināmas. Padodot apmācībai vāji aprakstītu attēlu ar vairākiem objektiem, nedos iespēju apmācīt neironu tīklu, kurš varētu izdalīt konkrēta objekta pazīmes, jo vāji aprakstīts attēls var saturēt vairākus objektus un konvolūciju neironu tīkls tos uztvers kā vienu objektu, kā rezultātā tiks sasniegti nepareizi rezultāti. Ņemot to vērā, neironu tīkla pirmo daļu ir nepieciešams apmācīt izmantojot

pārraudzīto mācīšanās metodi, kur uz katra apmācīšanas attēla ir tieši viens cieši apgriezts objekts. Šim nolūkam ir piemērotas vairākas attēlu datu bāzes, kur katrai bildei ir atzīmēts objektu atrašanās vieta. Šī darba neironu tīkla apmācībai tiks izmantota CIFAR 100 attēlu datu bāze, un tiks apmācīti 20 klašu objekti to atpazīšanai.

Lai apmācītu objektu pazīmju izdalīšanas neironu tīkla daļu ar konkrētu objektu attēliem ir nepieciešams zināt, ko šis tīkls atpazīst, tāpēc konvolūciju slāņiem tiek pielikti pilnsaistes slāņi, kuri izdod neirona tīkla atpazīšanas rezultātus, nodrošina iespēju pielietot zuduma funkciju un apmācīt visu neironu tīkla pirmo daļu. Veiksmīgi apmācot neironu tīkla pirmo daļu to, kopā ar pilnsaistes slāņiem, var izmantot objektu atpazīšanā.

Pēc konvolūciju daļas apmācīšanas, to vairs neapmācīs. Tā tiks izmantota otrās neironu tīkla daļas apmācīšanai, bet apmācīšanas gaitā tās svari netiks mainīti, mainot tikai otrās neironu tīkla daļas svarus.

Otrās daļas apmācībai var izmantot vāji aprakstītus attēlus. Tas ļauj izmantot jebkurus attēlus, kuriem ir zināms konteksts. Darba ietvaros tiks izmantota Microsoft COCO attēlu datubāze, kas piedāvā neizgrieztus attēlus ar aprakstītiem objektiem uz tiem [43].

Neironu tīkla adaptācijas daļa ieejā saņem pirmā neironu tīkla daļas pēdējā konvolūciju slāņa izejas rezultātus. Sanāk, ka otrā neironu tīkla daļa saņem izdalītas objektu pazīmes no pirmās daļas. Tā kā objektu atrašanās vietas nav zināmas, tad pirmajam neironu tīklam, izmantojot slīdoša loga metodi, tiek padotas vairākas attēla daļas ar vairākiem mērogiem. Tas palīdz atrast objektus, kuriem nav zināma atrašanās vieta uz sākuma attēla, kā arī palīdz padarīt neironu tīklu neatkarīgu no objektu izmēra attēlā.

Apstrādājot sākuma attēlu pa daļām, viens no variantiem ir to apstrādāt ar pirmo neironu tīkla daļu visus sākuma attēla reģionus un zinot informāciju par tiem, jau apmācīt adaptācijas slāņus no otrās daļās. Tas palīdzētu izfiltrēt visus reģionus bez objektiem pirms otrās daļas apmācības. Tomēr tas nav iespējams, jo, nezinot objektu atrašanas vietas, nav zināms kādus objektus ir atpazinusi pirmā neironu tīkla daļa. Tas arī palielina atmiņas izmantošanu, jo pirms otrā slāņa apmācības, ir nepieciešams glābāt visus pirmās neironu tīkla daļas rezultātus par katru sākuma attēla apgabalu.

Gadījumā, kad pirmajai neironu tīkla daļai tiek padota attēla daļa, kur nav neviena objekta, pirmās daļas izejā saņems pazīmes ar zemu ticamību. Lai neņemtu tos vērā, tiek izmantots "apvienošana pēc maksimuma slānis" otrā neironu tīkla daļas sākumā. Tas palīdz neņemt vērā sākuma attēla reģionus bez objektiem un ierobežot otrā tīkla apmācību uz negatīviem paraugiem. Lai ekonomētu atmiņas izmantošanu un apmācītu otro neironu tīkla daļu uz konkrētiem objektiem, tiek izmantota secīga pieeja – otro neirona tīkla daļu apmāca ar visiem rezultātiem no pirmās daļas secīgi, neņemot vērā citu sākuma apgabalu rezultātus.

Hipotētiski pavienošanas slāņa izmantošana neļaus negatīviem piemēriem stipri ietekmēt neironu tīkla precizitāti, jo šī metode tika izmantota cita vāji pārraudzīta neironu tīkla apmācībā [11]. Tomēr šī metode var nenostrādāt šī darba ietvaros izstrādājamam tīklam, jo tā izmērs ir daudz mazāks un tas var novest pie citiem rezultātiem.

Ņemot vērā to, ka neirona tīkla otrās daļas rezultātam nav jābūt atkarīgam no objektu atrašanās vietas, tad adaptācijas slāņi var būt pilnsaistes slāņi, kas jau no pazīmju kopas, saņemtas no konvolūciju slāņiem, var izvadīt objektu sarakstu uz sākuma attēla.

Neironu tīkla izveidošana tika sadalīta divās daļās. Sākumā tika izveidots konvolūciju neironu tīkls pārraudzītas mācīšanās metodei. Vāji pārraudzītas mācīšanās neironu tīkla izveidei tika izmantoti konvolūciju slāņi no pirmā neironu tīkla, kuriem tika pielikti pilnsaistes slāņi, kas tika apmācīti ar vāji pārraudzīto mācīšanās metodi.

Pirmās neironu tīkla daļas izveide balstījās uz Tensorflow bibliotēkas konvolūciju neironu tīklu izveides pamācību un piemēru, kas spēj apmācīties izmantojot CIFAR 100 attēlu datubāzes [44].

4.1. tabula

Konvolūciju neironu tīkla struktūras parametri

	Dziļums	Platums	Uztveres lauks	Malu papildināšana	Solis	Rezultāta platums	Rezultāta dziļums
Pirmais konvolūciju slānis	3	24	5	2	1	24	64
Apvienošanas sānis	64	24	3	0	2	12	64
Normalizācija							
Otrais konvolūciju slānis	64	12	5	2	1	12	64
Normalizācija							
Apvienošanas slānis	64	12	3	0	2	6	64
Pirmais pilnsaistes slānis	2304	1				1	384
Otrais pilnsaistes slānis	384	1				1	192
Softmax	192	1				1	20

Pirmā neironu tīkla struktūra sastāv no diviem konvolūciju slāņiem, kuriem seko apvienošanas un normalizācijas slāņi. Tālāk seko divi pilnsaistes slāņi un pēdējais ir softmax klasifikators, kas izvada atrastās klases identifikatoru. Detalizētu konvolūciju neironu tīkla struktūru var redzēt tabulā *4.1*. Tabulā var redzēt neironu tīkla slāņu secību, kā arī konvolūciju

un apvienošanas slāņu parametrus. Tabulā nav minēts ieejas un rezultātu augstums, jo tas ir vienāds ar platumu. Detalizētāku neironu tīkla struktūru var apskatīt pielikumā 6.

Pirmais konvolūciju slānis saņem tenzoru ar dziļumu 3, jo neironu tīkls tiek apmācīts ar krāsainiem attēliem. Pirmā konvolūciju slāņa uztveres lauks ir 5x5 un filtru skaits ir 64, tāpēc izejas dziļums ir 64. Izmantojot malas papildināšanu un soli vienādu ar 1, pirmā konvolūciju slāņa rezultāta platums tika saglabāts nemainīgs. Tālāk pielietojot apvienošanas slāni, matricas platums tika samazināts divreiz. Pēc normalizācijas tenzors tiek padots otrajam konvolūciju slānim, kur tenzora dimensijas tiek saglabātas. Pēc normalizācijas ar tiek apvienošanas slāni, tenzora platums atkal samazināts divas reizēs un tālāk tiek pārveidots uz vienas dimensijas tenzoru, lai to izmantotu pilnsaistes slāņos. Trīs secīgos pilnsaistes slāņos viendimensiju matricas izmērs tiek pakāpeniski samazināts, saņemot 20 izejas.

Konvolūciju neironu tīkls tika apmācīts, izmantojot CIFAR100 datubāzi ar 100 objektu tipiem. Apmācībā tika izmantotas 20 šīs datubāzes superklases, kas ietver sevī visus 100 objektu tipus. CIFAR100 datubāzes objektu klases un to superklases ir apskatāmas pielikumā 4.

Otrais neironu tīkls sastāv no diviem iepriekš apmācītiem konvolūciju slāņiem, kas tiek paņemti no pirmā neironu tīkla. Konvolūciju slāņiem seko, divi pilnsaistes slāņi un softmax klasifikators, kuri tiek apmācīti ar vāji pārraudzīto mācīšanās metodi. Otrā neironu tīkla struktūra ir ļoti līdzīga pirmā neironu tīkla struktūrai – atšķiras tikai izeju skaits. Detalizētāku neironu tīkla struktūru var apskatīt pielikumā 7.

Otrā daļa tika apmācīta, izmantojot datus no Microsoft COCO datubāzes. Klasifikators tika apmācīts izmantojot 12 superklases, kuras ietver sevī visas Microsoft COCO klases. Microsoft COCO attēlu datubāzes objektu klašu sarakstu var apskatīt pielikumā 5.

4.3. Konvolūciju neironu tīkla definēšana iekš TensorFlow

Neironu tīklu realizācijai tika izmantota TensorFlow bibliotēka. Programmas kods tika rakstīts Python programmēšanas valodā un tika sadalīts astoņos failos. Šajā nodaļā tiks apskatīti svarīgākie programmas koda fragmenti.

```
with tf.variable_scope('conv1') as scope:
  kernel = _variable_with_weight_decay('weights', shape=[5, 5, 3, 64],
                                       stddev=1e-4, wd=0.0, trainable=trainable)
  conv = tf.nn.conv2d(images, kernel, [1, 1, 1, 1], padding='SAME')
  biases = _variable_on_cpu('biases', [64], tf.constant_initializer(0.0), trainable=trainable)
  bias = tf.nn.bias_add(conv, biases)
  conv1 = tf.nn.relu(bias, name="conv1")
  _activation_summary(conv1)
```

4.2. att. **Konvolūciju slāņa definēšana**

Pirmā konvolūciju slāņa definēšana ir redzama attēlā *4.2.* Mainīgais "kernel" definē konvolūciju slāņa filtrus, parametrs "shape" definē filtru uztveres lauku un filtru skaitu. Mainīgais "conv" definē pašu konvolūciju, kur tiek izmantota Tensorflow bibliotēkas funkcija tf.nn.conv2d. Tālāk tiek definētas apmācības nobīdes, kas tiek pievienotas konvolūciju slānim izmantojot funkcijas tf.nn.bias_add. Tālāk konvolūciju mainīgajam tiek pielietota taisngrieža (anlg. val. *rectifier*) aktivācijas funkcija tf.nn.relu, kura šobrīd ir viena no populārākajām aktivācijas funkcijām konvolūciju neironu tīkliem, jo to ir viegli skaitīt un tā nerada problēmas gradientu izplātīšanai. Aktivācijas funkcijas formula ir:

$f(x) = \max(0, x)$.

Pēc aktivizācijas funkcijas pielietošanas tiek pielietota statistikas rakstīšanas funkcija _activation_summary. Konvolūciju un to filtru mainīgajos tiek izmantots "trainable" mainīgais, kas definē vai vajag mainīt mainīgā svarus apmācīšanas laikā. Mūsu gadījumā konvolūciju neironu tīkla apmācīšanas laikā svaru maiņa ir atļauta, taču adaptācijas slāņu apmācībā svaru maiņa ir aizliegta.

```
pool1 = tf.nn.max_pool(conv1, ksize=[1, 3, 3, 1], strides=[1, 2, 2, 1],
                       padding='SAME', name='pool1')
norm1 = tf.nn.lrn(pool1, 4, bias=1.0, alpha=0.001 / 9.0, beta=0.75,
                  name='norm1')
```

4.3. att. **Apvienošanas un normalizācijas slāņu definēšana**

Pēc konvolūciju slāņa definēšanas tiek izveidoti apvienošanas un normalizācijas slāņi, tā kā ir redzams attēlā *4.3.* Otrā konvolūciju slāņa definēšana notiek līdzīgi pirmajam, tikai mainās filtru izmērs.

Pēc konvolūciju slāņiem seko pilnsaistes slāņi, to definēšana ir redzama attēlā *4.4*. Tam tiek definēts ieejas neironu skaits un izejas neironu skaits ar mainīgo "shape", kā arī tiek pieskaitītas nobīdes un pielietota aktivācijas funkcija. Pilnsaistes slāņa īpašības tiek sasniegtas izmantojot funkciju "tf.matmul", ar kuru ieejas matrica tiek reizināta ar pilnsaistes slāņa svariem.

```
with tf.variable_scope('local4') as scope:
  weights = _variable_with_weight_decay('weights', shape=[384, 192],
                                        stddev=0.04, wd=0.004)
  biases = _variable_on_cpu('biases', [192], tf.constant_initializer(0.1))
  local4 = tf.nn.relu(tf.matmul(local3, weights) + biases, name=scope.name)
```

4.4. att. **Pilnsaistes slāņa definēšana**

Pēc neironu tīkla struktūras definēšanas tiek aprakstīta zuduma funkcija, kas ir redzama attēlā *4.5*. Zuduma funkcija tiek aprēķināta izmantojot Tensorflow funkciju "tf.nn.sparse_softmax_cross_entropy_with_logits".

```
labels = tf.cast(labels, tf.int64)
cross_entropy = tf.nn.sparse_softmax_cross_entropy_with_logits(
    logits, labels, name='cross_entropy_per_example')
cross_entropy_mean = tf.reduce_mean(cross_entropy, name='cross_entropy')
tf.add_to_collection('losses', cross_entropy_mean)
```

4.5. att. **Zuduma funkcijas definēšana**

Pēc neironu tīkla struktūras definēšanas, tiek veikta apmācības grafika definēšana, kas ir apskatāms attēlā *4.6*. Sākumā no ieejas datu ģeneratora tiek saņemti attēli un to apraksti. Attēliem tiek aprēķinātas prognozes, kurus tiek salīdzinātas ar sagaidāmo rezultātu un tiek aprēķināta zuduma funkcijas vērtība. Balstoties uz zuduma funkcijas vērtību, neironu tīkla svari tiek apmācīti.

```
images, labels = cifar100.distorted_inputs()
logits = cifar100.inference(images)
loss = cifar100.loss(logits, labels)
train_op = cifar100.train(loss, global_step)
```

4.6. att. **Apmācības grafika definēšana**

Lai veiktu apmācību, tika izveidota Tensorflow sesija uz mainīgo "train_op" un kad tā tika veikta, Tensorflow bibliotēka aprēķina visus atkarīgos mainīgos un ar to sasniedz tīkla apmācību.

Adaptācijas slāņu struktūra un apmācības procesa definēšana ir ļoti līdzīga pirmajam konvolūciju neironu tīklam, tāpēc to neapskatīsim detalizētāk.

Tā kā otrās datubāzes attēli var saturēt vairākas objektu klases, tika pielietota cita zuduma funkcija, lai korektāk novērtētu neironu tīkla darbību. Šim nolūkam tika izmantota "sigmoid_cross_entropy_with_logits" Tensorflow funkcija, kura atbilst sekojošai formulai:

$$f(x,z) = \max(x,0) - x * z + \log(1 + e^{-|x|}),$$

kur x ir neironu tīkla prognozes un z ir sagaidāmais rezultāts.

Pirms adaptācijas slāņu apmācīšanas, konvolūciju slāņu vērtības tika ielādētas no iepriekš saglabāta konvolūciju neironu tīkla modeļa, tā kā tas ir redzams attēlā 4.7., kur mainīgajā "cifar_vars" glabājas konvolūciju slāņu mainīgie.

```
ckpt = tf.train.get_checkpoint_state(FLAGS.cifar100_train_dir)
if ckpt and ckpt.model_checkpoint_path:
  tf.train.Saver(cifar_vars).restore(sess, ckpt.model_checkpoint_path)
  global_step = ckpt.model_checkpoint_path.split('/')[-1].split('-')[-1]
```

4.7. att. **Konvolūciju slāņu svaru ielādēšana no modeļa**

Līdzīgā veidā tika ielādēti modeļi, ja bija nepieciešams turpināt apmācīt neironu tīklu pēc apmācīšanas pārtraukuma.

Detalizētāk konvolūciju neironu tīklu apmācīšanas struktūras definēšanas kodu, kā arī apmācības procesa definēšanu var apskatīt pielikumos 2. un 3.

4.4. Apmācīšanas un testēšanas datu sagatavošana

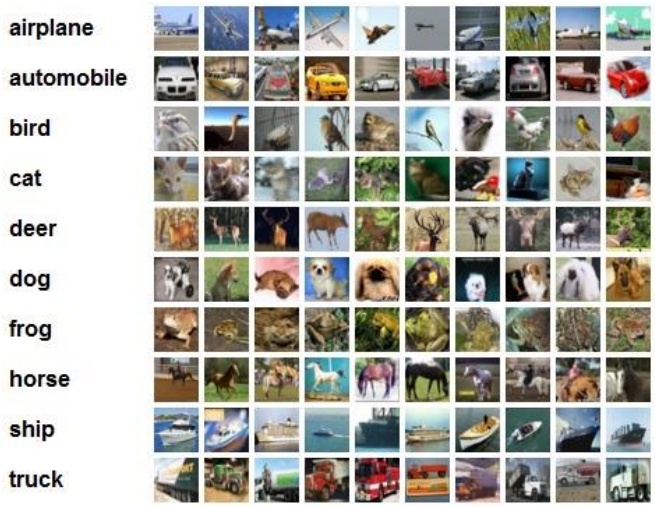
Lai apmācītu neironu tīkla abas daļas, ir nepieciešams sastādīt divas apmācības kopas. Viena ir pilnīgi aprakstīta attēlu kopa, kas tiek izmantota konvolūciju neironu tīkla daļas apmācīšanai. Otra ir vāji aprakstīta attēlu kopa, kas tiks izmantota neironu tīkla adaptācijas daļas apmācīšanai. Apmācības un testēšanas kopu attēli nepārklājas, lai precīzi novērtētu neironu tīkla precizitāti uz attēliem, kuri nebija izmantoti apmācīšanai.

4.4.1. Konvolūciju neironu tīkla daļas apmācīšanas kopa

Attēli tiek ņemti no CIFAR100 attēlu datubāzes [16]. Datubāze satur 100 objektu klases, kas ir sagrupētas pa 20 superklasēm. Katrai klasei ir pieejami 600 aprakstīti attēli, 500 no tiem ir apmācības attēli un 100 ir testēšanas attēli.

Attēlu datubāze sastāv no 60000 32x32 pikseļu krāsainiem attēliem. Katrai klasei ir pieejami 500 apmācīšanas attēli un 100 validācijas attēli. Attēlu piemērus, sagrupētus pa klasēm var redzēt att.2.

Attēli datubāzē ir jau izgriezti un ir pieejami vienādā izmērā, tāpēc neironu tīkla struktūra tiks balstīta uz šo attēlu izmēru. Tā kā apmācības kopas attēli ir salīdzinoši mazi, tad, lai saglabātu neironu tīkla zemu sarežģītību, pirmā tīkla daļa tiks apmācīta ar 20 superklasēm, neņemot vērā precīzas attēlu klases. Tas palīdzēs izdalīt objektu superklašu pazīmes, kas ir derīgas lielākiem objektu tipiem, kā arī palielinās pieejamo attēlu skaitu uz katru klasi. Ņemot vērā, ka apmācības klases otrās neironu tīkla daļai atšķirsies, tad pirmās neironu tīkla objektu klases satura plašums var palīdzēt vieglāk apmācīt otro neirona tīkla daļu.



4.8. att. **Attēlu piemēri no CIFAR100 datubāzes** [16]

Neironu tīkla apmācīšanai tika izmantoti iepriekš sagatavoti binārie faili, kas satur attēlu informāciju, katra attēla klasi un superklasi. Kopējais apmācības un testēšanas attēlu ar klašu anotācijām informācijas apjoms sastāda 175 megabaitus. Tas ļaus apmācīt neironu tīklu ātrākā laikā. Binārā faila struktūra ir redzama attēlā 4.9. attēlā. Sākuma ir viens baits, kas raksturo attēla superklasi, tālāk seko attēla mazās klases baits, beidzot ir 3072 pikseļu baiti, kas raksturo krāsaino attēlu ar izmēru 32x32. Pirmie 1024 baiti apraksta attēla sarkanas krāsas kanāla pikseļu, tālāk seko informācija par zaļo kanālu un zilo kanālu.

```
<1 x coarse label><1 x fine label><3072 x pixel>
...
<1 x coarse label><1 x fine label><3072 x pixel>
```

4.9. att. **CIFAR100 binārā faila struktūras shēma** [16]

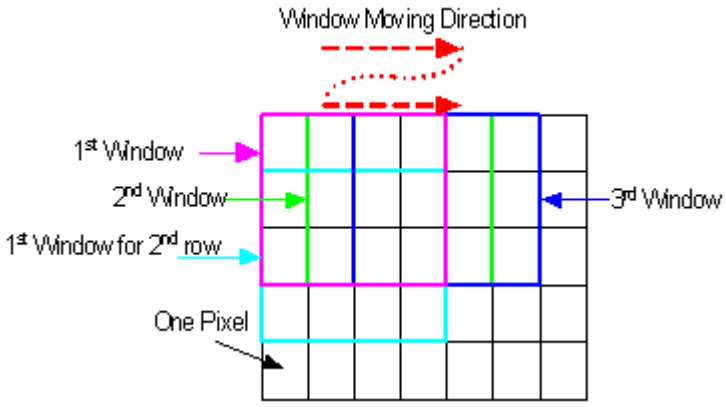
Attēli tiek sagrupēti pa paketēm (angl. v. *batch*), kuru izmērs ir 128 attēli. Šīs kopas izmērs tika izvēlēts, lai samazinātu atmiņas izmantošanu apmācības procesā, kā arī, lai optimizētu neironu tīkla apmācības ātrumu.

4.4.2. Otrās daļas apmācīšanas datu kopa.

Otrās daļas apmācīšanai ir nepieciešama datu kopa, kur ir pieejami vāji aprakstīti attēli, kuru objekti nav izgriezti. Šim nolūkam tika izvēlēta MC COCO attēlu datubāze [17]. Datubāzē ir pieejami attēli ar vairākiem iezīmētiem objektiem. Lai pārbaudītu vāji pārraudzītas mācīšanās metodi, objektu attēlu atrašanas vietas netiek izmantotas.

Tā kā pirmā neironu tīkla daļa spēj apstrādāt 32x32 krāsaino attēlu, tad arī no šīs datu kopas vajag iegūt apmācīšanas attēlus ar tādu pašu izmēru.

Datubāzē attēli ir daudz lielāki, tāpēc tos uzreiz padot neironu tīklam nevar. Lai sagatavotu apmācības attēlus, katram datubāzes attēlam tiek pielietota slīdošā loga metode.



4.10. att. **Slīdoša loga darbības principa ilustrācija** [45]

Slīdošā loga meklēšanas metode tiek izmantota, lai apstrādātu attēlu pa daļām. Lai apstrādātu visus attēla apgabalus, tas tiek sadalīts pēc principa, kas ir attēlots attēlā 4.10. Sākumā tiek izgriezts attēls no augšējā kreisā stūra ar izmēru, kas ir nepieciešams tālākai apstrādei. Nākamajā solī "logs" tiek pabīdīts par noteiktu soli pa labi. Solis tiek izvēlēts tādā veidā, lai iespējamo soļu skaits būtu vesels skaitlis un lai viss attēla platums būtu iekļauts vienā no izgrieztajiem attēliem.

Šī darba ietvaros tiks ņemts logs ar izmēru 32x32 pikseļi, jo šis ieejas attēlu izmērs ir nepieciešams konvolūciju neironu tīkla daļai, savukārt lai neironu tīkls varētu apmācīties ar dažādu objektu izmēriem, pirms slīdošā loga metodes pielietošanas, sākuma attēls tiek samazināts par vairākiem izmēriem ar noteiktu soli.

Lai optimizētu neironu tīkla apmācību, apmācības attēli tiks iepriekš sagatavoti. Tas palīdzēs izgriezt attēlus tikai vienreiz un jau uz tiem veikt neironu tīkla apmācības mēģinājumus un eksperimentus.

Rezultātā no sākuma attēla tika izdalīti vairāki attēlu fragmenti ar izmēru 32x32. Katram attēlu fragmentam tiek piesaistīta sākuma attēla esošo objektu klašu informācija. Tas nozīmē, ka visiem izgrieztajiem attēliem no viena sākuma attēla būs vienāda anotācija.

Datu ģenerēšanai tiek izmantota MSCOCO datubāzes piedāvāta bibliotēka priekš python vides [46]. Šī bibliotēka ļauj izfiltrēt vajadzīgos datus, sagrupēt attēlus ar to anotācijām. Šis interfeiss darba gaitā tika papildināts ar vairākām funkcijām, kas palīdz ātri iegūt attēlu aprakstus un objektu klases par attiecīgo objektu.

Izmantojot modificētu Microsoft COCO datubāzes interfeisu, tika izveidota neliela lietotne, kas spēj sagatavot binārus apmācīšanas failus priekš darbā izstrādātā neironu tīkla. Dati tika sagatavoti ņemot vērā MsCOCO datubāzes 12 superklases, jo tie tiks izmantoti vāji pārraudzītajā mācīšanā. Bināro failu struktūra sastāv no divām daļām. Sākuma tiek ielikti 12 baiti, kuri reprezentē kādas klases objekti atrodas uz sākuma attēla. Otrā daļā tiek ierakstīts informācija par attēlu, līdzīgā veidā kā CIFAR100 datubāzes bināros failos. Sagatavotu bināro failu struktūru var apskatīt attēlā 4.11.

```
<12 x coarse labels><3072 x pixel>
...
<12 x coarse labels><3072 x pixel>
```

4.11. att. **Sagatavotu no MSCOCO datubāzes attēlu binārо failu struktūra**

Apmācības datu sagatavošanas lietotni var izmantot apmācības failu izveidošanai priekš Tensorflow dažādām neironu tīklu konfigurācijām, tāpēc to var izmantot arī ārpus šī darba. Attēlu sagatavošanas kodu var apskatīt pielikumā 1.

4.5. Neironu tīkla apmācība

Šajā nodaļā tiks apskatīts neironu tīkla apmācīšanas process. Tiks atsevišķi apskatīta konvolūciju neironu tīkla apmācīšana ar pārraudzīto mācīšanās metodi un adaptācijas slāņu apmācīšana izmantojot vāji pārraudzīto mācīšanās metodi, balstoties uz apmācītiem konvolūciju slāņiem. Visas šajā darbā veiktas apmācības tika veiktas uz datora aprīkotu ar procesoru Intel i7-6700K un 16Gb operatīvu atmiņu. Videokarte neironu tīklu apmācīšanā netika izmantota.

4.5.1. Konvolūciju neironu tīkla apmācība

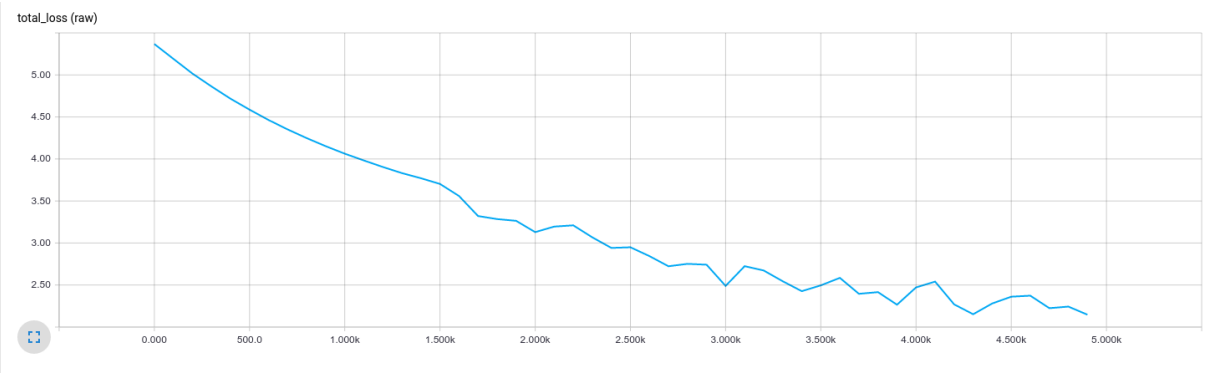
Konvolūciju neironu tīklu apmācība notika izmantojot 128 attēlu pakas no sagatavotas apmācības kopas, jo tas palīdz samazināt atmiņas izmantošanu un paātrināt neironu tīkla apmācību [47]. Lai neironu tīkls nepārapmācītos un spētu veiksmīgi minēt objektu klases attēliem, kuri netika izmantoti apmācīšanā, kā arī, lai palielinātu apmācības kopas dažādību, pirms padošanas apmācībai, attēli tika dažādos veidos deformēti, līdzīgā veidā, kā tas tika darīts CIFAR10 konvolūciju neironu tīkla apmācībā [44]. Apmācībai attēls tika apgriezts līdz izmēram 24x24, ar nejauši izvēlētu centru, kas palīdzētu neironu tīklam atpazīt attēlus ar daļēji aizsegtiem objektiem. Tālāk izgrieztie attēli tika nejauši balināti, lai neironu tīkls nebūtu atkarīgs no attēlu dinamiskā diapazona. Tālāk attēlam tika pielietota nejauša atspoguļošana, spilgtuma un kontrasta maiņa.

Neironu tīkls apmācības ieejā saņem deformētu attēlu ar izmēru 24x24 un izdod atpazīšanas rezultātu, vienu no atpazītām objektu klasēm.

Katru desmito iterāciju tika rakstīta statistikas informācija par neironu tīkla apmācību, kura ietver svaru izmaiņas, mācīšanās nobīdes izmaiņas, zuduma funkcijas rezultātus un informāciju par katra slāņa apmācību. Apmācības informācija tika vizualizēta izmantojot TensorBoard rīku [48]. Tas deva priekšstatu par neironu tīkla apmācības situāciju un ļāva aptuveni novērtēt neironu tīkla precizitāti. Zuduma funkcijas vērtību maiņu apmācības laikā var redzēt attēlā *4.12*. Var redzēt, ka apmācības laikā zuduma funkcijas vērtība samazinājās un beigās sāka svārstīties ap zemu vērtību. Pēc svaru izmaiņu rādītājiem varēja redzēt, kad neironu tīkls pārstāj apmācīties un svaru vērtības vairs netiek mainītas.

Katru simto iterāciju apmācāmais modelis tika pieglabāts, lai nepieciešamības gadījumā apmācību varētu turpināt pēc tās pārtraukšanas. Saglabātie modeļi tika izmantoti neironu tīkla testēšanas nolūkos, lai novērtētu tā precizitāti uz testēšanas kopas attēliem, kas tiks aprakstīts neironu tīkla testēšanas nodaļā.

Konvolūciju neironu tīkla apmācība tika pabeigta, kad tika sasniegta zema zuduma funkcijas vērtība un svaru izmaiņas kļuva par neievērojamām, balstoties uz testēšanas rezultātiem, kad tika sasniegta pietiekoši liela precizitāte.

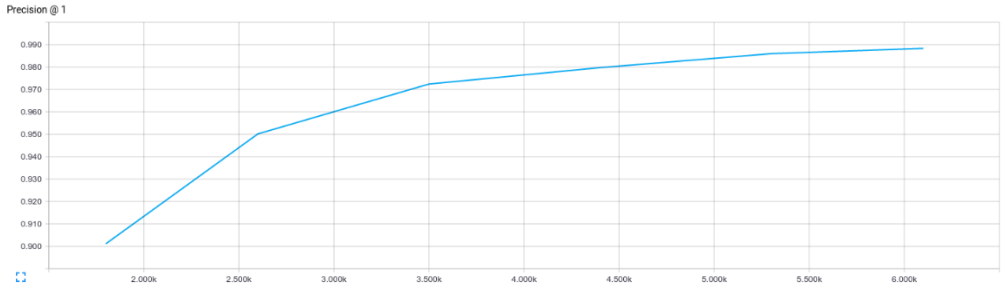


4.13. att. **Konvolūciju neironu tīkla zuduma funkcijas grafiks apmācības laikā**

Neironu tīkla apmācība tika pabeigta pēc 6200 iterācijām, kad tika sasniegta vislielākā precizitāte - tālāka apmācība nemainīja tīkla precizitāti. Konvolūciju neironu tīkla apmācības process aizņēma apmēram 5 stundas.

4.5.2. Konvolūciju neironu tīkla testēšana

Testēšanas process tika veikts paralēli apmācības procesam. Reizi piecās minūtēs tika iegūts pēdējais saglabātais konvolūciju neironu tīkla modelis, un, izmantojot to, tika pārbaudīts, cik precīzi šis modelis strādā.



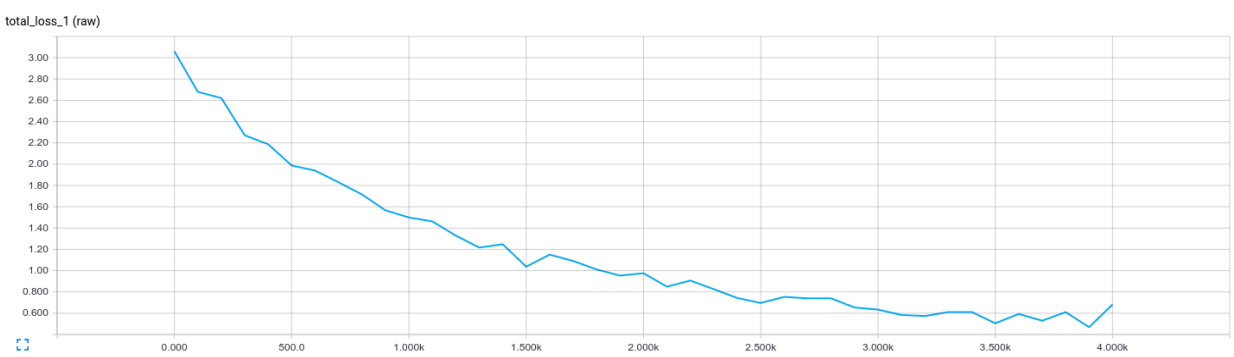
4.14. att. **Konvolūciju neironu tīkla precizitātes grafiks**

Precizitātes pārbaudei tika izmantota tf.nn.in_top_k funkcija, kas atgriež informāciju par to, vai neironu tīkla mērķa rezultāts ir starp prognozēm ar lielāko varbūtību. Rezultātu pareizība tika summēta un aprēķināta vidējā precizitātes vērtība. Ar šo vērtību tika definēta konvolūciju neironu tīkla rezultātu ticamība.

Neironu tīkla precizitātes maiņa apmācīšanas laikā ir redzama grafikā attēlā 4.14. Precizitāte sasniedza 98.8% un tālāk nemainījās, tāpēc tālāka neironu tīkla apmācība tika pārtraukta.

4.5.3. Adaptācijas daļas apmācība

Pēc veiksmīgas konvolūciju neironu tīkla apmācības, tika apmācīti adaptācijas slāņi ar vāji aprakstītiem attēliem, kas bija iepriekš sagatavoti no MsCOCO datubāzes attēliem. Pirms apmācīšanas sākuma konvolūciju slāņu svari tika iegūti no apmācīta konvolūciju slāņa modeļa. Konvolūciju slāņa rezultātu jau saņēma pilnsaistes neironu tīkla slāņi, kuri tika apmācīti.



4.15. att. **Neironu tīkla ar adaptācijas slāņiem zuduma funkcijas grafiks apmācības laikā**

Tika izvēlēts apmācīt adaptācijas slāņus ar 128 attēlu pakām. Ieejas attēli no apmācības kopas tika deformēti līdzīgi kā konvolūciju neironu tīkla apmācībā. Tas nodrošina neironu tīkla neatkarību no apmācības kopas īpašībām.

Zuduma funkcijas vērtību maiņa apmācības laikā ir apskatāma attēlā 4.15. Tās vērtība samazinājās ļoti strauji un jau pēc trīs tūkstošiem iterācijām sasniedza ļoti zemu līmeni. Tas var liecināt par to, ka adaptācijas slāņi veiksmīgi prognozē otrās datu kopas attēlu klases, balstoties uz iepriekš apmācītiem konvolūciju slāņiem.

Adaptācijas slāņu apmācībā, tāpat kā konvolūciju neironu tīkla apmācībā, katru simto iterāciju tika saglabāts neironu tīkla modelis, kas pēc tam tika izmantots neironu tīkla precizitātes testēšanā.

4.5.4. Adaptācijas daļas testēšana

Adaptācijas slāņu apmācības precizitāte tiek testēta līdzīgi kā konvolūciju neironu tīkla testēšanā. Attēli tika apstrādāti pa daļām, izmantojot slīdošā loga metodi. Precizitātes novērtēšanai tika ņemtas visticamākās prognozes no visām apskatītām attēla pozīcijām. Testēšanas laikā tika konstatēts, ka neskatoties uz to, ka apmācības laikā zuduma funkcijas vērtība samazinājās, neironu tīkla darbības precizitāte uz testēšanas kopas attēliem palika nepietiekoša.

4.6. Praktiskā darba rezultātu apkopojums

Praktiskā darba ietvaros tika izstrādāti divi konvolūciju neironu tīkli, kas tika apmācīti ar divām attēlu kopām. Pirmais konvolūciju neironu tīkls tika apmācīts ar pārraudzīto mācīšanās metodi un tālāk tas tika izmantots otrā neironu tīkla apmācībai.

Pirmā neironu tīkla apmācībai tika izmantota 50000 attēlu kopa, savukārt adaptācijas slāņu apmācībai tika izmantota 300000 attēlu kopa.

Otrā neironu tīkla apmācībai tika izstrādāts rīks, kas sagatavo apmācības un testēšanas datus. Pirmā neironu tīkla apmācīšana bija veiksmīga - tas tika apmācīts piecu stundu laikā un sasniedza precizitāti līdz 98.8%, detalizētāk apmācīšanas precizitāti var apskatīt sadaļā 4.5.2.

Otrā neironu tīkla apmācība aizņēma mazāk laika, tomēr testēšanas laikā tika secināts, ka neironu tīkla precizitāte nepieaug. Tas var būt saistīts ar to, ka pirmā konvolūciju neironu tīkla sarežģītības līmenis ir relatīvi neliels (slāņu skaits un to dziļums) un tāpēc konvolūciju slāņi neizdala pietiekoši daudz objektu pazīmes, lai adaptācijas slāņi spētu veiksmīgi apmācīties, kas tika uzskatīts par iespējamo variantu eksperimenta sākumā. Otrā iespēja ir saistīta ar to, ka zuduma funkcija neatspoguļoja reālo neironu tīkla precizitāti un tāpēc tīkls nespēja apmācīties. Otrā neironu tīkla iespējamās problēmas var pētīt tālāk un optimizēt neironu tīklu struktūras, lai otrais tīkls spētu apmācīties ar vāji pārraudzīto metodi.

5. REZULTĀTI UN DISKUSIJA

Šī darbā ietvaros tika apskatīti neironu tīklu darbības princips un to attīstības vēsture. Detalizēti tika apskatīti konvolūciju neironu tīkli, kas tika izmantoti praktiskā darba izveidei. Tika apskatītas un salīdzinātas vairākas šobrīd populārās mašīnmācīšanās bibliotēkas, kas palīdzēja izvēlēties vispiemērotāko bibliotēku neironu tīklu izstrādei.

Praktiskā darba ietvaros tika veiksmīgi izstrādāts un apmācīts konvolūciju neironu tīkls. Tam tika izmantota pārraudzīta mācīšanās metode un tas tika apmācīts, izmantojot CIFAR100 attēlu datubāzi. Rezultātā ar pietiekamu precizitāti neironu tīkls varēja minēt kurai klasei pieder uz attēla redzamais attēls. Ņemot vērā neironu tīkla nelielo izmēru, tā darbība ir salīdzinoši ļoti ātra. Tā kā tas tika apmācīts uz 20 objektu superklasēm, tad tā ātrdarbības priekšrocības var izmantot klasifikācijas uzdevumu priekšapstrādē, lai samazinātu ar lielu neironu tīklu meklējamo klašu skaitu.

Tika izstrādāta neironu tīkla struktūra, kurai vajadzētu apmācīties izmantojot vāji pārraudzīto mācīšanās metodi, tomēr veiksmīgi to apmācīt neizdevās. Apmācīšanas neveiksmes iemesls ir konvolūciju neironu tīkla mazs izmērs. Mazs izmērs tika definēts, jo ieejas attēla izmērs arī ir mazs. Izvirzīta hipotēze, ka maza konvolūciju neironu tīkla izmērs būs pietiekams, lai apmācītu adaptācijas slāņus, neapstiprinājās. Tomēr tā kā neviens vēl nemēģināja pielietot vāji pārraudzīto metodi uz tik maza neironu tīkla, šis apmācības mēģinājums bija interesants un vērtīgs. Ar šo eksperimentu tika parādīts, ka vāji pārraudzīta mācīšanās metode nestrādā tik labi, kā lielos konvolūciju neironu tīklos.

Piemēram, darbā [11], kur tika veiksmīgi izmantota vāji pārraudzīta mācīšanās metode, izmanto daudz lielāku konvolūciju neironu tīklu. Tā darba ietvaros konvolūciju neironu tīklam ir 5 konvolūciju slāņi un ieejas attēlu izmērs ir 224x244 pikseļi, tāpēc konvolūciju slāņi izdala daudz vairāk pazīmju, nekā šī darba ietvaros izstrādāts neironu tīkls.

Vēl viens iespējamais veiksmes faktors ir tas, ka līdzīga darba pētnieki apmācīja konvolūciju slāņus, izmantojot ļoti lielu attēlu datubāzi ImageNet, kas ietver sevī visas klases no adaptācijas slāņu apmācības izmantotās datubāzes MsCOCO. Šī darba ietvaros klases konvolūciju slāņu apmācībā un adaptācijas slāņu apmācībā pārsvarā pārklājas, tomēr ir daudz klāšu, kuras ir vienā attēlu datubāzē un to nav otrā datubāzē. Rezultātā tas varēja ietekmēt konvolūciju tīklu, kur tā slāņi neizdalīja vajadzīgās pazīmes, kas savukārt bija nepieciešamas adaptācijas slāņu apmācībai uz otrās datubāzes klasēm.

6. SECINĀJUMI

Mākslīgo neironu tīklu joma parādās jau sen, un no tā brīža attīstās. Neskatoties uz to, objektu atpazīšanas uzdevumos neironu tīklus sāka pielietot pavisam nesen, kad sāka strauji attīstīties konvolūciju neironu tīkli un tehnoloģijas nepieciešamas tiem. Konvolūciju neironu tīkli deva iespēju efektīvi izmantot datoru resursus neironu tīkla darbības laikā un palielināt tīkla prognozējamo rezultātu precizitāti.

Šodien neironu tīklu pielietošana kļuva pieejama plašam izstrādātāju lokam, jo ir pieejamas vairākas atvērta koda bibliotēkas priekš neironu tīklu izstrādes.

Šobrīd zināmās neironu tīklu apmācības datubāzes domātas konkrētām objektu klasēm, tomēr ja nepieciešams apmācīt neironu tīklu uz citas objektu klases, kura nav pieejama esošās attēlu datubāzēs, tā ir nopietna problēma, jo pirmkārt, ir jāsavāc pietiekoši liels attēlu skaits un, otrkārt, jāapraksta atrastus attēlus atbilstoši uzdevuma kontekstam, kas var prasīt milzīgu resursu apjomu. Šādas problēmas risināšanai ir iespējams izmantot vāji pārraudzīto mācīšanās metodi, kas atvieglotu apmācības un testēšanas datu kopu sagatavošanas procesu un samazinātu tā apjomu.

Vāji pārraudzīta mācīšanās metode ļauj izmantot jau esošos konvolūciju neironu tīklus (kas jau ir apmācīti) kopā ar adaptācijas slāņiem, kuri tiek apmācīti, izmantojot jauno apmācības kopu.

Adaptācijas slāņu veiksmīgas apmācības pamatā ir iepriekšējo konvolūciju slāņu pietiekošs dziļums, kas nodrošina nepieciešamu objektu noteikto pazīmju skaitu. Vāji pārraudzītās mācīšanās metodes izmantošanai praksē, nepieciešams izvēlēties tādus konvolūciju neironu tīklus, kuriem objektu raksturiezīmes ir kopīgas dotam uzdevumam. Tāpēc katras problēmas ietvaros ir jāmeklē savs risinājums un šī darba ietvaros definētais tīkls var būt izmantots, kā vadlīnija citas vāji pārraudzītas metodes neironu tīkla izstrādei.

Kopumā var teikt, ka vāji pārraudzītas mācīšanās metodes pielietošana ir labs risinājums specifisku objektu klašu atpazīšanas uzdevumiem. Šo mācīšanās metodi ir īpaši izdevīgi pielietot, kad jau ir pieejams apmācīts konvolūciju neironu tīkls, kas spēj izdalīt pietiekoši daudz pazīmes vajadzīgajam uzdevumam.

7. IZMANTOTĀ LITERATŪRA

[1] Jānis Zuters, "Neironu tīkli."

[2] L. Yao and J. Miller, "Tiny ImageNet Classification with Convolutional Neural Networks."

[3] Fei-Fei Li, "How we're teaching computers to understand pictures." [Tiešsaite]. Pieejams: https://www.ted.com/talks/fei_fei_li_how_we_re_teaching_computers_to_understand_pictures?language=en. [Apskatīts: 26.05.2016].

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions."

[5] "ImageNet Download FAQ." [Tiešsaite]. Pieejams: http://image-net.org/download-faq. [Apskatīts: 26.05.2016].

[6] "What does Artificial Neural Network (ANN) mean?" [Tiešsaite]. Pieejams: https://www.techopedia.com/definition/5967/artificial-neural-network-ann. [Apskatīts: 26.05.2016].

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "4824-Imagenet-Classification-With-Deep-Convolutional-Neural-Networks," pp. 1–9.

[8] R. Rojas, "Weighted Networks – The Perceptron."

[9] "Frank Rosenblatt." [Tiešsaite]. Pieejams: http://slidegur.com/doc/256305/rosenblatt. [Apskatīts: 26.05.2016].

[10] Jürgen Schmidhuber, "Who Invented Backpropagation?" [Tiešsaite]. Pieejams: http://people.idsia.ch/~juergen/who-invented-backpropagation.html.

[11] M. Oquab, F. L. Bottou, I. Laptev, and F. J. Sivic, "Is object localization for free ? – Weakly-supervised learning with convolutional neural networks," *Cvpr*, no. iii, pp. 685–694, 2015.

[12] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and Transferring Mid-level Image Representations Using Convolutional Neural Networks," *2014 IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 1717–1724, 2014.

[13] C. Biemann, "Unsupervised and Knowledge-free Natural Language Processing in the Structure Discovery Paradigm," *Doctor*, 2007.

[14] S.-Y. Bai, S. Agethen, T.-H. Chao, and W. Hsu, "Semi-supervised Learning for Convolutional Neural Networks via Online Graph Construction," *arXiv1511.06104 [cs]*, pp. 1–9, 2015.

[15] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative

Unsupervised Feature Learning with Convolutional Neural Networks," *Adv. Nueral Inf. Process. Syst. 27 (Proceedings NIPS)*, pp. 1–13, 2014.

[16] "The CIFAR-10 and CIFAR-100 datasets." [Tiešsaite]. Pieejams: https://www.cs.toronto.edu/~kriz/cifar.html. [Apskatīts: 26.05.2016].

[17] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dolí, "Microsoft COCO: Common Objects in Context."

[18] "Artificial Neural Networks/Activation Functions." [Tiešsaite]. Pieejams: https://en.wikibooks.org/wiki/Artificial_Neural_Networks/Activation_Functions. [Apskatīts: 26.05.2016].

[19] "Setting up the data and the model." [Tiešsaite]. Pieejams: http://cs231n.github.io/neural-networks-2/#losses. [Apskatīts: 26.05.2016].

[20] "Convolutional Neural Networks (CNNs / ConvNets)." [Tiešsaite]. Pieejams: http://cs231n.github.io/convolutional-networks/. [Apskatīts: 26.05.2016].

[21] "Supervised Learning with Neural Networks." [Tiešsaite]. Pieejams: https://www.cl.cam.ac.uk/teaching/2004/ArtInt1/learn.pdf. [Apskatīts: 26.05.2016].

[22] "Artificial Neural Networks: Introduction and Application." [Tiešsaite]. Pieejams: http://natureofcode.com/book/chapter-10-neural-networks/. [Apskatīts: 26.05.2016].

[23] "About ImageNet." [Tiešsaite]. Pieejams: http://image-net.org/about-overview. [Apskatīts: 26.05.2016].

[24] "Performing Convolution Operations." [Tiešsaite]. Pieejams: https://developer.apple.com/library/ios/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html. [Apskatīts: 26.05.2016].

[25] "UNDERSTANDING CONVOLUTIONAL NEURAL NETWORKS FOR NLP." [Tiešsaite]. Pieejams: http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/. [Apskatīts: 26.05.2016].

[26] "Linear Classification." [Tiešsaite]. Pieejams: http://cs231n.github.io/linear-classify/. [Apskatīts: 26.05.2016].

[27] "Loss function optimization." [Tiešsaite]. Pieejams: http://cs231n.github.io/optimization-1/. [Apskatīts: 26.05.2016].

[28] Kenneth Tran, "Evaluation of Deep Learning Toolkits." [Tiešsaite]. Pieejams: https://github.com/zer0n/deepframeworks/blob/master/README.md. [Apskatīts: 26.05.2016].

[29] "DIY Deep Learning for Vision: a Hands-On Tutorial with Caffe." [Tiešsaite]. Pieejams: https://docs.google.com/presentation/d/1UeKXVgRvvxg9OUdh_UiC5G71UMscNPlv

ArsWER41PsU/preview?pref=2&pli=1&slide=id.gc2fcdcce7_216_0. [Apskatīts: 26.05.2016].

[30] "Caffe." [Tiešsaite]. Pieejams: http://caffe.berkeleyvision.org/. [Apskatīts: 26.05.2016].

[31] "CNTK - Computational Network Toolkit." [Tiešsaite]. Pieejams: https://www.cntk.ai/. [Apskatīts: 26.05.2016].

[32] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-Bit Stochastic Gradient Descent and its Application to Data-Parallel Distributed Training of Speech DNNs."

[33] "TensorFlow." [Tiešsaite]. Pieejams: https://www.tensorflow.org/. [Apskatīts: 26.05.2016].

[34] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, and G. Research, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems."

[35] "Benchmark TensorFlow." [Tiešsaite]. Pieejams: https://github.com/soumith/convnet-benchmarks/issues/66.

[36] "torch - a Scientific Computing Framework for LUAJIT." [Tiešsaite]. Pieejams: http://torch.ch/. [Apskatīts: 26.05.2016].

[37] "TensorFlow - Roadmap." [Tiešsaite]. Pieejams: https://www.tensorflow.org/versions/r0.8/resources/roadmap.html. [Apskatīts: 26.05.2016].

[38] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[39] W. Liu, J. He, and S. Chang, "Large Graph Construction for Scalable Semi-Supervised Learning," *ICML*, pp. 679–689, 2010.

[40] A. Krizhevsky and G. E. Hinton, "4824-Imagenet-Classification-With-Deep-Convolutional-Neural-Networks," pp. 1–9.

[41] "What is weakly supervised learning (bootstrapping)?" [Tiešsaite]. Pieejams: http://stackoverflow.com/questions/18944805/what-is-weakly-supervised-learning-bootstrapping. [Apskatīts: 26.05.2016].

[42] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision."

[43] "COCO - Common Objects in Context." [Tiešsaite]. Pieejams: http://mscoco.org/.

[Apskatīts: 26.05.2016].

[44] “Convolutional Neural Networks.” [Tiešsaite]. Pieejams: https://www.tensorflow.org/versions/r0.8/tutorials/deep_cnn/index.html. [Apskatīts: 27.05.2016].

[45] “Computer Vision - Implementation.” [Tiešsaite]. Pieejams: https://sites.google.com/site/hsi2013logan99/computer-vision/implementation. [Apskatīts: 26.05.2016].

[46] “Interface for accessing the Microsoft COCO dataset.” [Tiešsaite]. Pieejams: https://github.com/pdollar/coco/blob/master/PythonAPI/pycocotools/coco.py. [Apskatīts: 27.05.2016].

[47] “What is batch size in neural network?” [Tiešsaite]. Pieejams: http://stats.stackexchange.com/questions/153531/what-is-batch-size-in-neural-network. [Apskatīts: 27.05.2016].

[48] “TensorBoard: Visualizing Learning.” [Tiešsaite]. Pieejams: https://www.tensorflow.org/versions/r0.8/how_tos/summaries_and_tensorboard/index.html. [Apskatīts: 26.05.2016].

8. PIELIKUMI

1. pielikums
Apmācības datu sagatavošanas rīka pamata pirmkods

```
def main():
    cocoInstance = coco.COCO("/home/sergey/src/coco/data/annotations/instances_train2014.json")

    bin_file_path = "/home/sergey/src/coco/data/bin"

    file_name = "train_small_1"
    imgIds = cocoInstance.getImgIds()
    idsWithCats = cocoInstance.getImageIdsWithSuperCategories(imgIds)
    img_dim = 32
    sliding_step = 30
    min_scale = 0.20
    max_scale = 0.5
    scale_step = 0.25
    images_to_process = 3000
    start = 0
    max_count = len(imgIds)

    print("Images to process for one file: ", images_to_process)

    number = 1
    finish = False
    while start+images_to_process <= max_count:
        output_path = get_output_path(bin_file_path, file_name, number)
        print("Writing data to ", output_path)
        for img_data, lab in idsWithCats[start:start+images_to_process]:

            path = get_mscoco_path(mscocoImagePath, img_data['file_name'])
            binaryLabel = bytearray(lab)

            image = Image.open(path)

            rescaled_images = get_rescaled_image_exact_size(image, img_dim)
            sliding_window_images = []

            for rez_img in rescaled_images:
                sliding_window_images+= get_sliding_window_images(rez_img, sliding_step, img_dim)

            output_file = open(output_path, 'ab+')
            for sliding_img in sliding_window_images:
                if sliding_img.mode == "RGB":
                    r, g, b = sliding_img.split()
                    r_data = np.array(r)
                    g_data = np.array(g)
                    b_data = np.array(b)

                    output_file.write(binaryLabel)
                    output_file.write(r_data)
                    output_file.write(g_data)
                    output_file.write(b_data)
        start+=images_to_process
        if (start + images_to_process) > max_count and not finish:
            images_to_process = max_count-start
            finish = True
        number+=1

    print("done")
```

Konvolūciju neironu tīkla izveduma funkcijas pirmkods

```
def inference(images):
  conv2 =  cifar100_conv(images)
  # norm2
  norm2 = tf.nn.lrn(conv2, 4, bias=1.0, alpha=0.001 / 9.0, beta=0.75,
                    name='norm2')
  print("norm2", norm2)
  # pool2
  pool2 = tf.nn.max_pool(norm2, ksize=[1, 3, 3, 1],
                         strides=[1, 2, 2, 1], padding='SAME', name='pool2')
  print("pool2", pool2)
  # local3
  with tf.variable_scope('local3') as scope:
    #Transforming shape of tensor to use it in fully connected layers
    dim = 1
    for d in pool2.get_shape()[1:].as_list():
      dim *= d
    reshape = tf.reshape(pool2, [FLAGS.batch_size, dim])

    weights = _variable_with_weight_decay('weights', shape=[dim, 384],
                                          stddev=0.04, wd=0.004)
    biases = _variable_on_cpu('biases', [384], tf.constant_initializer(0.1))
    local3 = tf.nn.relu(tf.matmul(reshape, weights) + biases, name=scope.name)
    _activation_summary(local3)
    print("local3", local3)

  # local4
  with tf.variable_scope('local4') as scope:
    weights = _variable_with_weight_decay('weights', shape=[384, 192],
                                          stddev=0.04, wd=0.004)
    biases = _variable_on_cpu('biases', [192], tf.constant_initializer(0.1))
    local4 = tf.nn.relu(tf.matmul(local3, weights) + biases, name=scope.name)
    _activation_summary(local4)
    print("local4", local4)

  # softmax,
  with tf.variable_scope('softmax_linear') as scope:
    weights = _variable_with_weight_decay('weights', [192, NUM_CLASSES],
                                          stddev=1/192.0, wd=0.0)
    biases = _variable_on_cpu('biases', [NUM_CLASSES],
                              tf.constant_initializer(0.0))
    softmax_linear = tf.add(tf.matmul(local4, weights), biases, name=scope.name)
    _activation_summary(softmax_linear)
    print("softmax", softmax_linear)

  return softmax_linear
```

```
def train():
    with tf.Graph().as_default():

        global_step = tf.Variable(0, trainable=False)
        images, labels = cifar100.mscoco_inputs(False)
        conv2 = cifar100.cifar100_conv(images, trainable = False)
        cifar_vars = tf.all_variables()
        logits = cifar100.mscoco_inference_from_conv(conv2)

        loss = cifar100.ms_coco_loss(logits, labels)
        train_op = cifar100.train(loss, global_step)

        # Create a saver.
        saver = tf.train.Saver(tf.all_variables())
        # Build the summary operation based on the TF collection of Summaries.
        summary_op = tf.merge_all_summaries()
        # Build an initialization operation to run below.
        init = tf.initialize_all_variables()
        # Start running operations on the Graph.
        sess = tf.Session(config=tf.ConfigProto(
            log_device_placement=FLAGS.log_device_placement))

        sess.run(init)
        ckpt = tf.train.get_checkpoint_state(FLAGS.cifar100_train_dir)
        print("CKPT", ckpt)
        #restoring cifar100 convolutional layers
        if ckpt and ckpt.model_checkpoint_path:

          tf.train.Saver(cifar_vars).restore(sess, ckpt.model_checkpoint_path)
          global_step = ckpt.model_checkpoint_path.split('/')[-1].split('-')[-1]
          print("global_step", global_step)

        tf.train.start_queue_runners(sess=sess)
        summary_writer = tf.train.SummaryWriter(FLAGS.train_dir, sess.graph)
        for step in xrange( FLAGS.max_steps):
          start_time = time.time()
          _, loss_value = sess.run([train_op, loss])
          duration = time.time() - start_time
          assert not np.isnan(loss_value), 'Model diverged with loss = NaN'

          if step % 10 == 0:
            num_examples_per_step = 128
            examples_per_sec = num_examples_per_step / duration
            sec_per_batch = float(duration)

            format_str = ('%s: step %d, loss = %.2f (%.1f examples/sec; %.3f '
                          'sec/batch)')
            print (format_str % (datetime.now(), step, loss_value,
                                 examples_per_sec, sec_per_batch))

          if step % 100 == 0:
            summary_str = sess.run(summary_op)
            summary_writer.add_summary(summary_str, step)
          # Save the model checkpoint periodically.
          if step % 100 == 0 or (step + 1) == FLAGS.max_steps:
            checkpoint_path = os.path.join(FLAGS.train_dir, 'model.ckpt')
            saver.save(sess, checkpoint_path, global_step=step)
```

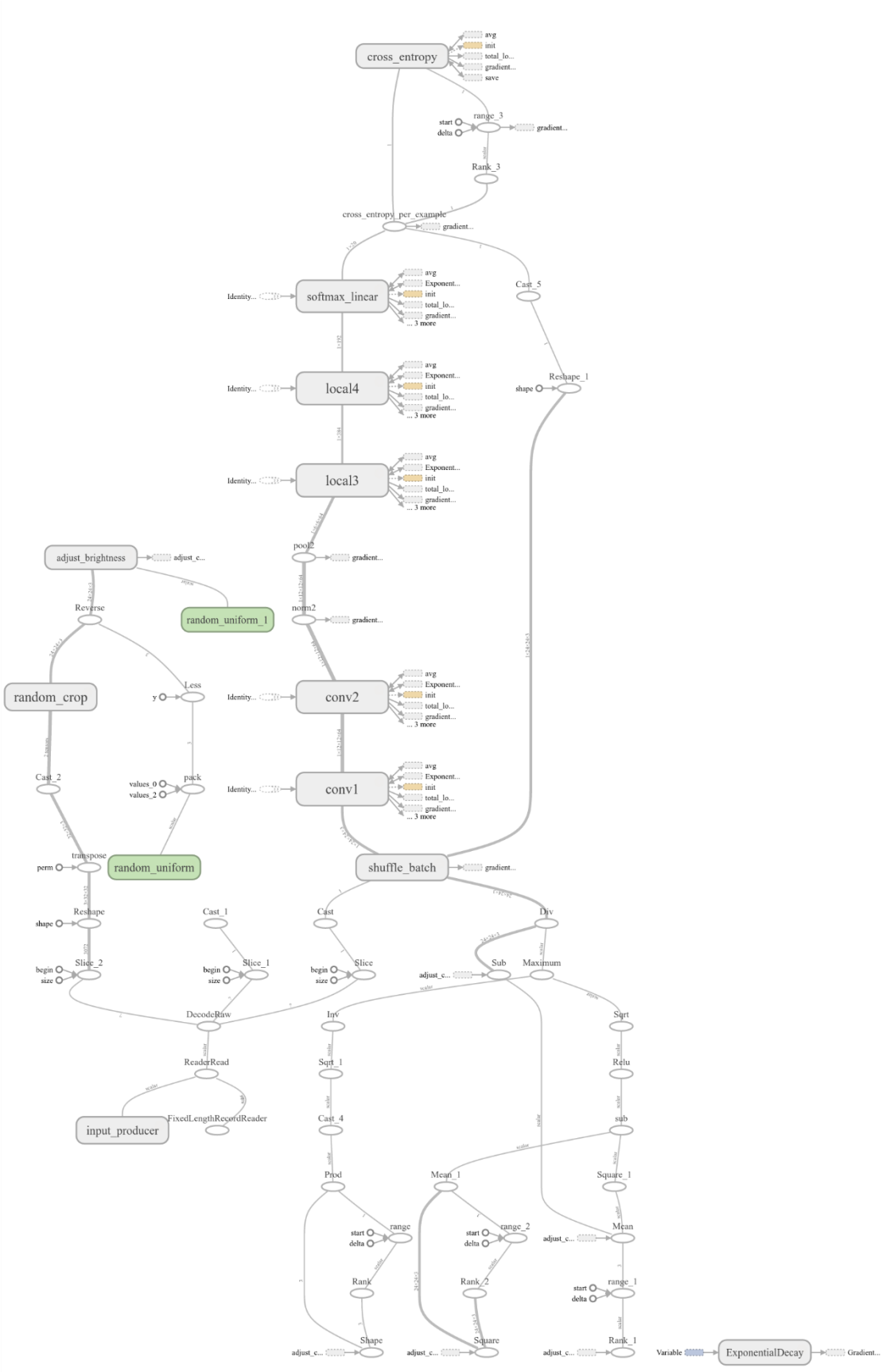
4. pielikums
CIFAR100 attēlu datubāzes klašu un superklašu sadalījums

Superclass	**Classes**
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

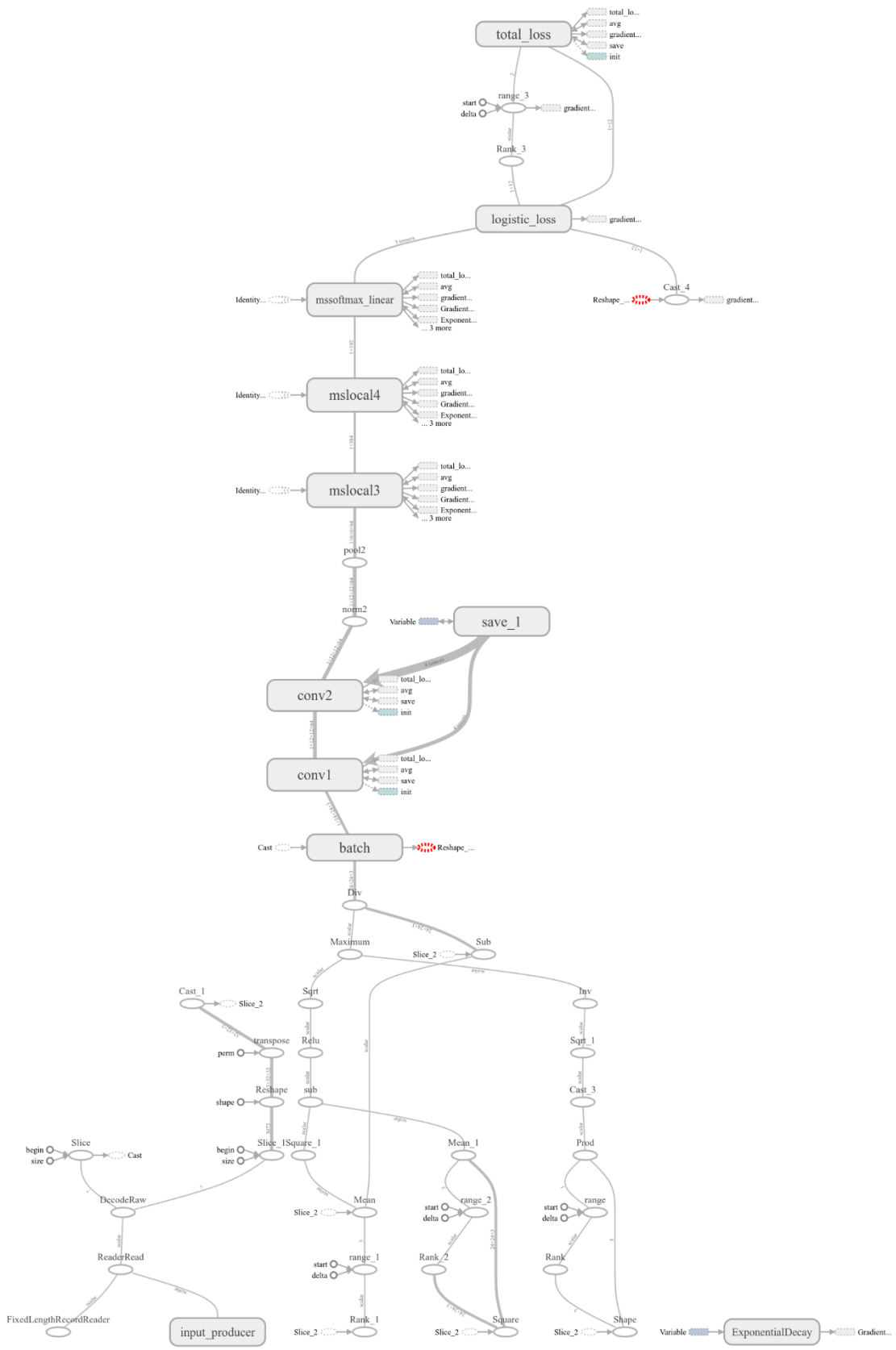
5. pielikums

MsCOCO attēlu datubāzes klašu un superklašu sadalījums

Superclass	**Classes**
person	person
vehicle	bicycle, car, motorcycle, airplane, bus, train, truck, boat
outdoor	traffic light, fire hydrant, stop sign, parking meter, bench
animal	bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe
accessory	backpack, umbrella, handbag, tie, suitcase
sports	frisbee, skis, snowboard, sports ball, kite, basebal bat, baseball glove, skateboard, surfboard, tennis racket
kitchen	bottle, wine glass, cup, fork, knife, spoon, bowl
food	banana, apple, sandwich, orange, broccoli, carrot, hot dog, pizza, donut, cake
furniture	chair, couch, potted plant, bed, dining table, toilet
electronic	tv, laptop, mouse, remote, keyboard, cell phone
appliance	microwave, oven, toaster, sink, refigerator
indoor	book, clock, vase, scissors, teddy bear, hair drier, toothbrush


cross_entropy
avg
init
total_lo...
gradient...
save
range_3
start
delta
gradient...
Rank_3
cross_entropy_per_example
gradient...
softmax_linear
Identity...
Exponent...
... 3 more
Cast_5
local4
Reshape_1
shape
local3
pool2
adjust_brightness
adjust_e...
random_uniform_1
Reverse
norm2
random_crop
Less
y
conv2
Cast_2
values_0
values_2
pack
conv1
transpose
perm
random_uniform
shuffle_batch
Reshape
Cast_1
Cast
Div
Slice_2
Slice_1
Slice
begin
size
Sub
Maximum
DecodeRaw
Inv
Sqrt
ReaderRead
Sqrt_1
Relu
input_producer
FixedLengthRecordReader
Cast_4
sub
Prod
Mean_1
Square_1
range
range_2
Mean
Rank
Rank_2
range_1
Shape
Square
Rank_1
Variable
ExponentialDecay
Gradient...

Adaptācijas neironu tīkla grafiks, ģenerētais izmantojot TensorBoard rīku



Bakalaura darbs “Objektu atpazīšana attēlos ar mākslīgajiem neironu tīkliem, izmantojot vāji pārraudzīto mācīšanās metodi” izstrādāts LU Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti.

Autors: _____ Sergejs Ivanovs

Rekomendēju darbu aizstāvēšanai

Vadītājs: Dr.sc.ing. Andrejs Zujevs _____ 29.05.2016.

Recenzents: asociētais profesors Jānis Zuters

Darbs iesniegts Datorikas fakultātē 30.05.2016.

Dekāna pilnvarotā persona:_____

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē

__.06.2016. prot. Nr. ___

Komisijas sekretār__ : _____