LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

ZĪMĒTU ATTĒLU KLASIFICĒŠANA AR NEIRONU TĪKLIEM

BAKALAURA DARBS

Autors: Edgars Beļevičs
Stud. apl. Nr.: eb12050
Darba vadītājs: Dr.sc.comp. Kārlis Freivalds

Rīga 2021

ANOTĀCIJA

Šī darba mērķis ir iepazīties un izvērtēt esošos zīmētu attēlu klasificēšanas rīkus uz sakropļotiem attēliem, kas ir raksturīgi pikseļgrafikā.

Darba gaitā tika izstrādāta metode pikseļattēlu atpazīšanai, izmantojot skiču datu kopu *QuickDraw*. Pētījuma ietvaros ir salīdzināta *Sketch-a-Net* neironu tīkla arhitektūra un tiek izveidoti vairāki patvaļīgi konvolucionālie neironu tīkli skiču atpazīšanai. Pikseļattēla pirmsapstrādes ietvaros tiek piedāvāts depikselizācijas un kontrastējošās krāsu sliekšņa algoritms, lai pielīdzinātu pikseļattēlu skices īpatnībām. Papildus darbā tiek salīdzināta *TU Berlin* datu kopas efektivitāte ar *QuickDraw* datu kopu.

Atslēgas vārdi: zīmēti attēli, pikseļgrafika, neironu tīkli, klasificēšana.

ABSTRACT

DRAWING CLASSIFICATION WITH NEURAL NETWORKS

The goal of this study was assessment of drawing classification tools with neural networks on corrupted images, specifically on pixel art images.

The paper offers solution for recognizing pixel art images using QuickDraw sketch dataset. For sketch recognition proposed Sketch-a-Net neural network architecture is compared with arbitrary created convolutional neural. In order to pixel art match sketch features depixelization and image thresholding algorithms are used as part of pixel art. In addition, the study also compares the efficiency of the TU Berlin dataset with QuickDraw dataset.

Keywords: drawing, pixel art, neural network, classification.

SATURS

Apzīmējumu saraksts 5
Ievads 6
1. Literatūras apraksts 7
1.1. Pikseļgrafikas pamatprincipi un lasāmība 7
1.2. Pikseļgrafikas depikselizācija 8
1.3. Datu kopa 10
1.3.1. Pikselizētas datu kopas ģenerēšana 10
1.3.2. Skiču datu kopas 11
1.4. Skiču atpazīšana 12
1.5. Skiču atpazīšanas neironu tīkli 14
1.5.1. *WaveNet* neironu tīkls 14
1.5.2. *ResNet* arhitektūras pielietojums 14
1.5.3. Fišera vektoru pielietojums 16
1.5.4. *Sketch2Tag* sistēma 16
1.5.5. *DeepSketch2* neironu tīkls 17
1.5.6. *Sketch-a-Net* neironu tīkls 17
1.6. Animācijas tēla klasificēšana 19
1.7. Attēlu klasificēšana, izmantojot siluetu 20
2. Materiāli un metodes 21
2.1. Datu kopa 21
2.2. Pētījuma gaita 22
2.3. Datu apstrāde 24
3. Rezultāti un diskusija 25
3.1. Pikseļgrafikas tēla depikselizācija, izmantojot gatavu algoritmu 25
3.2. Depikselizācijas algoritma implementācija 25
3.3. Sketch-a-Net neironu tīkls uz QuickDraw datu kopas 28
3.4. Ģenerēts neironu tīkls uz QuickDraw datu kopas 30
3.5. *Sketch-a-Net* un pašģenerēts neironu tīkls uz *TU Berlin* datu kopas 36
3.6. Pikeļgrafikas attēla atpazīšana, izmantojot *Sketch-a-Net* un pašģenerētu modeli 38
Secinājumi 42
Literatūras apskats 43
Pielikumi 45

APZĪMĒJUMU SARAKSTS

CNN – konvolucionālais neironu tīkls

LSTM – rekurentā mākslīgā neironu tīkla paveids “Long Short-Term Memory”

ReLU – iztaisnotā lineārā vienība

epohi – korekcijas soļi

dropout – izkritums

stride – soļi jeb temps

pooling – papildinājums

softmax cross-entropy loss – augstākā aktivācijas funkcijas vērtībā

batch – partija

maxpool – maksimuma izvēle

overfitting – pārmērīga pielāgošanās treniņdatu kopai

IEVADS

Pikseļgrafika par spīti vecmodīgam dizainam ir saglabājusi savu šarmu mūsdienās. Tā galvenokārt izpaužas video spēlēs, retāk videoklipos. Galvenais piesaistes faktors ir nostalģija, atsaucoties uz Nintendo kārtridžu laikiem. Tomēr laikmetīgums un vienkāršība ir arī efektīvs faktors jauno spēļu vai animāciju izstrādātājiem. Īpašu popularitāti pikseļgrafika iemanto viedierīču tirgū, ņemot vērā viedierīču mazos ekrānus [19].

Pašā pikseļgrafikas pamatā zīmētie tēli vadās pēc klasiskās zīmēšanas pamatprincipiem. Pikseļgrafikā zīmēto tēlu un to animāciju izveide parasti noris ierobežotā laukumā un krāsu spektrā. Vienkāršības dēļ uzzīmēto tēlu skaits var sasniegt lielus apjomus īsā laika posmā, tāpēc tradicionālā kategorizēšana lielos apjomos var ievērojami paildzināt projekta izstrādes laiku, ja netiek izmantoti tēlam raksturīgie atslēgvārdi. Taču speciāli pētījumi un tieši rīki pikseļgrafikas tēla kategorizēšanai vai atpazīšanai darba izstrādes laikā netika konstatēti, kā rezultātā tiek apsvērta neironu tīklu pielietošana. Lai neironu tīkla darbība būtu efektīva, ir nepieciešams liels apmācāmā datu apjoms, kas pikseļattēlu gadījumā brīvā publiskā vidē lielā apjomā nav pieejams. Šis faktors aprobežo mašīnmācīšanas iespējas, pieprasot izmantot pikseļattēlu ģenerējošus rīkus vai meklēt citas alternatīvas.

Darbā tiek apskatīti vairāki skiču atpazīšanas pētījumi, kas var būt labs vadlīniju avots pikseļgrafikas tēlu atpazīšanai, ņemot vērā, ka abi mākslas izpausmes veidi ietver līdzīgus zīmēšanas pamatprincipus. Tomēr, lai pikseļattēli būtu līdzvērtīgāki skicēm, darbā tiek arī apskatīti attēla pirmsapstrādes metodes, kas ietver pikseļattēla depikselizāciju kā vienu no variantiem. Tiek arī pieminētas citas pikseļgrafikas atpazīšanas alternatīvas, kā animācijas tēla klasificēšana un silueta izvešana klasificēšanas nolūkiem. Darba beigās šīs metodes tiek analizētas un tiek piedāvāts optimāls pikseļgrafikas atpazīšanas risinājums.

Hipotēze: Pikseļattēli var tikt atpazīti ar neironu tīklu, izmantojot pikseļattēlu ģenerējošus rīkus vai skiču datu kopas.

Darba mērķis: Izvērtēt esošos risinājumus pikseļgrafikas atpazīšanai un piedāvāt optimālus risinājuma metodes, izmantojot neironu tīklus.

Darba uzdevumi:

1. Iepazīties ar esošiem risinājumiem pikseļgrafikas atpazīšanai un izvērtēt to aktualitāti.
2. Izvērtēt un implementēt pikseļattēla pirmsapstrādes metodes.
3. Izveidot neironu tīklu pikseļattēlu klasificēšanai.

1. LITERATŪRAS APRAKSTS

1.1. Pikseļgrafikas pamatprincipi un lasāmība

Pikseļgrafikas pamatā attēls tiek veidots rastrgrafikā. Pamatelements ir katrs pikselis, kas tiek izkārtots laukumā jeb punktmatricās. Salīdzinot pikseļgrafiku ar citiem rastrgrafikas attēliem, pikseļgrafikā laukums un krāsu dziļums tiek būtiski ierobežots. Šī parādība 80. un 90. gados bija agrāko datoru nespēja tehniski nodrošināt plašu krāsu spektru un augstu atmiņas kapacitāti, kāds ir sastopams šodien. Tomēr par izšķirošo atpazīšanas faktoru pikseļgrafikā uzskata, ja attēlā katrs krāsu pikselis ir tīši definēts [19].

1.1.att. **Rastrgrafikas (kreisajā pusē) salīdzinājums ar pikseļgrafiku (labajā pusē).**

Figure 1.1 **Raster graphics (left) comparison with pixel art (right).**

Pikseļgrafika piekopj klasiskās zīmēšanas pamatprincipus, proti, tēlu pamatā veido līnijas. Piksekgrafikas kontekstā lielākā daļa zīmēto elementu līnijas iekļaujas viena pikseļa robežās. Attēlā tēla kontūra un siluets tiek tieši definēts kā viens no galvenajiem indikatoriem tēlu atpazīšanā. Piemēram, 1.2. attēlā ir aptuveni noteicams, kādā kategorijā atbilstošais tēls ir klasificējams, neizmantojot tēla ārpus kontūras informāciju [1]. Siluets turpretī izceļ tēla raksturīgākās īpašības, tādas kā galvu, locītavas, drēbes utml. [2]

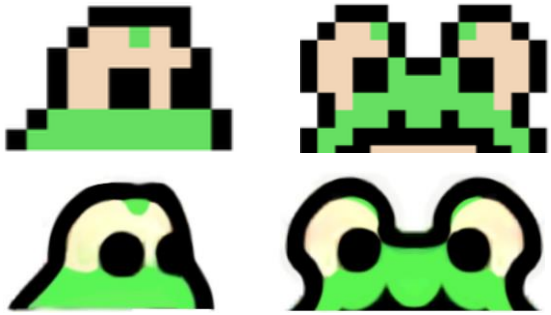
1.2.att. **Pikseļgrafikas tēla kontūras.**

Figure 1.2 **Pixel art sprite outlines.**

Samazinoties tēla izmēram, tēlus ir sarežģītāk atpazīt. Lasāmības palielināšanai maziem tēliem tiek izceltas dažas tēlam raksturīgākās īpašības. Piemēram, cilvēka rakstura tēlos galvenais

uzsvars ir uz sejas mīmiku. Palielinot galvas izmēru attiecībā pret ķermeņa izmēru, ir iespēja labāk raksturot tēla emocijas. Būtiska loma tēla atpazīšanai ir arī krāsām. Tās parasti ir 2 vai 3 krāsas, kas atšķir tēlu no citiem [1].

Pikseļgrafikas mazā izmēra dēļ, katram pikselim tēlā ir būtiski noteicoša loma līdz tādam līmenim, kad katrs nākamais papildu pikselis elementā var strauji mainīt tēla raksturojumu. Šādos gadījums viena no metodēm ir pārveidot pikseļgrafikas tēlu uz augstāku izšķirtspēju. 1.3. attēlā izmantojot šo metodi, ir vieglāk atpazīt, ka piemērā minētais attēls raksturo vardi [1].



1.3.att. **Pikseļgrafikas tēla (augšpusē) salīdzinājums ar augstākas izšķirtspējas tēlu (apakšā).**
Figure 1.3 **Pixel art sprite (left) comparison with high-definition sprite (right).**

1.2. Pikseļgrafikas depikselizācija

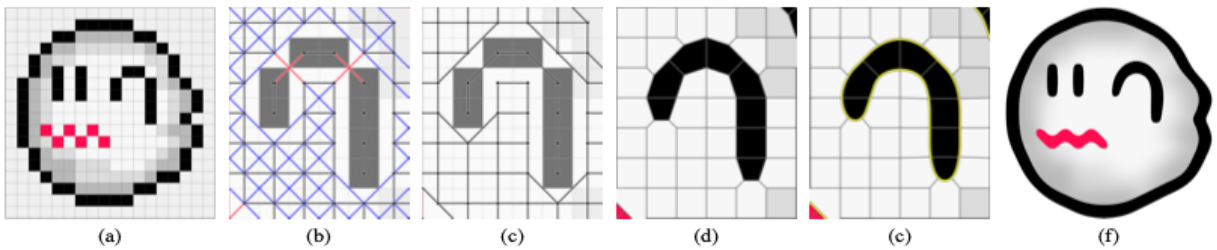
Depikselizācija ir metode, kas pārveido pikseļgrafikas attēlu uz vektorgrafiku, tādējādi padarot izšķirtspēju par nebūtisku faktoru. Tomēr depikselizācijā pikseļgrafikas savdabīgā daba izraisa šādus izaicinājumus salīdzinājumā ar dabīgām bildēm:

1. *Katrs pikselis ir svarīgs.* Katra pikseļa krāsa, kas krasi atšķiras no blakus esošiem pikseļiem, parasti reprezentē tēla elementu, piemēram, acis.
2. *Pikseļgrafikas līniju un līkņu savienojumi* konvertēšanas laikā var izskatīties kantaini, tādējādi zaudējot elementam raksturīgo formu.
3. *Neskaidri pikseļa definējumi mazākos reģionos.* Piemēram, uz 2x2 šaha galdiņa ir sarežģīti pateikt, kurai no abām diagonālēm jābūt savienotām atbilstoši pikseļattēla raksturojumam.
4. *Problēmas atšķirot līknes no kantainajām līnijām.* Pikseļgrafikā līniju kantainums piešķir konkrētam elementam vajadzīgo raksturu. Problēma sākas gadījumos, kad vajag izšķirties, vai šīs līnijas kantainums ir ar nodomu izveidots, vai tomēr tā ir līkne. [13]

Lai sasniegtu pilnīgu depikselizāciju, pikseļgrafikas attēla gala rezultātam jāatspoguļojas B-splainu līknēs, kas piedod attēlam gludās kontūrās. Pēc tam šīs līknes un informācija par krāsojumu

tiek izmantota, lai atveidotu depikselizētu bildi, izmantojot standarta renderēšanas rīkus. Tomēr pirms tam ir jāveic iepriekšēji sagatavošanas procesi, izveidojot pikseļattēlam atbilstošu grafu. Depikselizācijas algoritma pamatprincips norit sešos galvenos posmos (skat. 4. att.):

a) Tiek ievadīts pikseļgrafikas rakstura attēls.
b) Grafā katram krāsu pikselim tiek definēta virsotne. Iesākumā visas virsotnes tiek savienotas savā starpā ar šķautnēm. Pēc tam seko šķautņu izmešana no savstarpējiem krāsu pikseļiem. Ja krāsas būtiski atšķiras starp kaimiņu virsotnēm, šķautnes izmet. Šī procedūra ir salīdzinoši vienkārša, ņemot vērā pikseļgrafikas ierobežoto krāsu dziļumu.
c) Sarežģītākā daļa ir abu savstarpēju diagonāļu izšķiršana elementā, kas tiek panākts nosakot, kurai diagonālei ir lielāks svars. Šis svars tiek izrēķināts ar trim galvenajām metodēm:
    1) Vai dotā diagonāle reprezentē garāku savienojumu? Lielāks svars ir garākam savienojumam.
    2) Vai dotā diagonāle ir fons vai tomēr raksturo konkrētu elementu attēlā? Lielāks svars ir virsotnēm, kas ir vairāk izklīdinātas, reprezentējot atribūtu.
    3) Vai dotā diagonāle beidz savienojumu? Lielāks svars tiek piedots diagonālei, kas beidz savienojumu, lai neveidotu "saliņas" attēlā.
d) Ņemot vērā tikko izveidoto pikseļu savienojuma grafu, tiek atbilstoši izveidots jauns grafs, kas atspoguļo pikseļa formu attēlā, izmantojot Voronoja diagrammu, kur katra Voronoja šūna reprezentē konkrēto pikseli.
e) Lai padarītu kantainās pikseļu formas gludākas, konkrētās krāsas kontūras šķautnes tiek pārveidotas par B-splainu līknēm. Tā kā šī procedūra nenovērš pilnībā kantainumu, B-splainu līknes tiek vēl optimizētas, mainot B-splainu līkņu kontrolpunktu izvietojumu.
f) Un tiek iegūts gala rezultāts, renderējot vektorgrafiku [13].



1.4.att. **Pikseļgrafikas attēla depiksilizācija**

Figure 1.4 **Pixel art depixelizition**

Kopf et al. pētījumā minētais algoritms pārveidošanu veic vidēji 0,62 sekundēs. Ātrums galvenokārt bija atkarīgs no attēlā esošo līkņu izmēriem un skaita. Ir jāņem arī vērā, ka pētījumā minētais algoritms nav pietiekoši efektīvs, pārveidojot kropļojumnovērses pikseļgrafikas attēlos [13]. Kamēr kropļojumnovērse ir zināma un plaši pielietota prakse pikseļgrafikā 90. gadu vidū, lai iegūtu attēlam izteiktāku definējumu, tā ne vienmēr ir obligāta prakse, jo atsevišķās situācijās var padarīt mazus elementus attēlā miglainus [1].

1.3. Datu kopa

Pikseļgrafikai publiski nav plaši pieejamas datu kopas, kā tikai dažas interneta vietnes, no kurienes var ievākt atsevišķus brīvi pieejamus vai maksas darbus. Pati pikseļgrafikas attēlu veidošana ir arī laikietilpīgs un dārgs process, ņemot vērā pikseļgrafikas nišu. Serpa et al. pētījumā min, lai izveidotu pikseļgrafikas tēla kontūras un ēnojumu, tiek patērētas vidēji 40 minūtes un vairāk atkarībā no tēla sarežģītības pakāpes. Šim nolūkam Serpa et al. piedāvā neironu tīklu, kas ģenerē papildus variācijas no esošiem pikseļgrafikas tēliem, tomēr šīs variācijas pārsvarā izpaužas tikai ēnojumu un krāsu jeb ādaiņas ģenešanā kontekstā, turklāt noteikta pikseļgrafikas datu kopa ir tāpat vajadzīga, lai apmācītu modeli [18]. Lielā darba apjoma dēļ un pikseļgrafikas datu kopas trūkuma ir iespējami divi varianti, kā ielasīt lielu daudzumu attēlu pikseļgrafikas atpazīšanai:

1) Izmantot pikseļgrafikas ģenerēšanas metodi no plašāk pieejamas datu kopas, piemēram, no reālām bildēm:
2) Pārveidot pikseļgrafikas attēlu līdz tādai formai, kas atbilst kādai plašāk pieejamai datu kopai, piemēram, skicēm.

1.3.1. Pikselizētas datu kopas ģenerēšana

Pikselizācija ir pretējs process depikselizācijai, proti tiek ģenerēti pikseļgrafikas rakstura elementi, izmantojot animācijas filmām raksturīgus zīmējumus. Šī tehnika ir sevišķi lietderīga, ja trūkst pikseļgrafikas datu kopas. Tomēr esošās metodes kā bildes samazināšana vai renderēšana mazākā izmērā, var zaudēt oriģinālattēla kvalitāti, kamēr no-bildes-uz-bildi pārveidošanas metode, izmantojot neironu tīklus, pieprasa apmācības nolūkos pikseļgrafikai un animācijas tēlam līdzvērtīgu datu pāri, kuru sagatavošana lielā apjomā ir sarežģīts un laikietilpīgs process.

Ču Hans un Kiangs Vens piedāvā izmantot no interneta atlasītus treniņa datus, kas nav sapāroti, lai atvieglotu sagatavi. Pikselizācijai tiek izmantots *GridNet* un *PixelNet* apakštīkls, depikselizācijai tiek izmantots *DepixelNet* apakštīkls. *GridNet* uzdevums ir samazināt zīmējumu

līdz pikseļgrafikas izmēriem 1/4, 3/16 un 1/8 proporcijās. Tālāk *PixelNet* uzdevums ir producēt robainākas malas šiem trīs attēliem un paralēli saglabāt attēlam svarīgākās raksturīpašības. Depikselizācijai *DepixelNet* pēc tam veic atgriezeniskus procesus *PixelNet*, tikai šoreiz no ģenerētiem 3 attēliem tiek izvēlēts viens, lai izveidotu no pikseļgrafikas zīmētu animācijas tēlu.

Kopumā Ču Hana un Kiangs Vena risinājumam mācīšanas process 900 256 x 256 attēliem aizņēma 12 stundas, kamēr ģenerēšanas process vienam attēlam aizņēma 0,6 sekundes [10].

Gerstner et al. pētījumā *SLIC* (*Simple Linear Interative Clustering*) un *MCDA* (*Mass Constrained Deterministic Annealing*) piedāvātās metodes arī palīdz reālas bildes pārvērst pikseļgrafikas rakstura bildēs. *SLIC* algoritma princips balstās uz reālās bildes sadalīšanu reģionos jeb "superpikseļos", kur katrā iterācijā superpikseļiem tiek piešķirta reālās bildes pikseļu vidējā krāsa un pozīcija. *MCDA* ir globāla optimizēšanas metode, kura nosaka bildes izvēlēto krāsu paleti un piešķir no šīs paletes krāsu katram superpikselim jeb gala pikselim. Un šo abu algoritmu vairākkārtēja iterācija uzģenerē pikseļgrafikas bildi [8].

1.3.2. Skiču datu kopas

Tā kā skice ir pats sākumposms pikseļgrafikas tēla izveidei, alternatīva pikseļgrafikas rakstura datu kopām ir skiču datu kopa, kas pamatā piekopj ar roku zīmētu mākslas darbu paradigmas (skat. 1.5. att.) [18]. Skiču zīmēšana ir aizvēsturisks process jau no alu zīmēšanas laikiem. Tas ir daudz intuitīvāks process cilvēkiem, sevišķi bērniem, kas jau agrā vecumā piekopj prasmi zīmēt uz papīra kā veidu, lai komunicētu ar pieaugušajiem. Procesa gaitā bērns jau savlaicīgi iepazīstas ar konceptu par vairāku līniju savienošanu, lai atspoguļotu kopējo attēlu pilnvērtīgāk [11].

1.5.att. **No kreisās un labo pusi: raupja skice, definētas līnijas, definēts ēnojums, definētas atsevišķi krāsas, pikseļgrafikas tēla gala rezultāts**

Figure 1.5 **From left to right, the rough sketch, line sprite, gray sprite, color sprite and final sprite (Pārveidots no Towards Machine-Learning Assisted [18])**

Pēc līdzīga principa tika izveidota no šī laika lielākā skiču datu kopa *QuickDraw*. 2016. gada novembrī Google izlaida eksperimentālu spēli *"QuickDraw"*, lai apmācītu publiku, kā strādā mākslīgais intelekts. Spēle piedāvāja konkrētu vārdu, kas spēlētājam 20 sekundēs bija jāuzzīmē. Ja 20 sekunžu laikā spēlētājs var uzzīmēt attēlu, kas mākslīgajam intelektam ir atpazīstama, spēle beidzas un spēlētājs uzvar. Līdz katru mēģinājumu pavairojoties zīmējumu skaitam konkrētam vārdam, mākslīgais intelekts sāka mācīties un labāk atpazīt, kādu objektu spēlētājs zīmēja. Šādā veidā spēle ģenerēja vairāk kā vienu miljardu zīmējumu, no kuriem 50 miljoni tika atlasīti kā datu kopas sastāvs ar 345 kategorijām, kas ir publiski pieejams ikvienam interesentam. Datu kopa satur galvenokārt informāciju par x un y koordinātām masīvā, kas atspoguļo zīmuļa novietojumu katrai līnijai zīmēšanas laikā t [7].

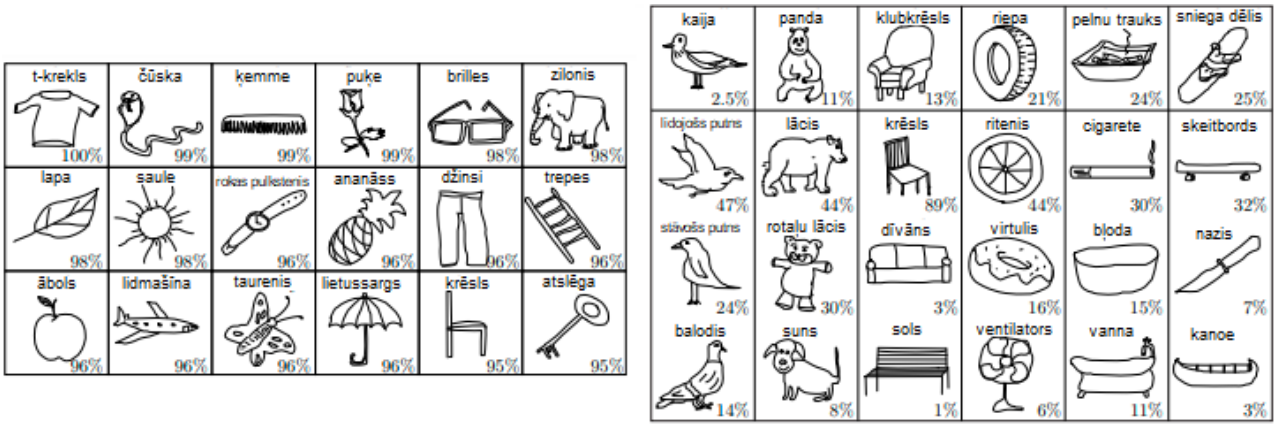
2012. gadā Eitz et al. izveidoja datu kopa *TU Berlin*, kas satur ap 20 000 skiču, iekļaujot 250 kategorijas. *TU Berlin* datu kopas skiču autori bija neprofesionāli zīmētāji, kas zīmējumus atspoguļoja ar vienkāršiem, bet atpazīstamiem objekta elementiem. Skicēšanas laikā autoram priekšā bija fotogrāfija, kas viņam bija jāatspoguļo skices formā. Skices radīšanai tika atveltītas 30 minūtes, tomēr skiču autoriem zīmēšana vidēji prasīja tikai 86 sekundes. Eitz et al. arī novēroja faktu, ka lielākā daļa dalībnieku vienmēr sāka zīmējumus ar divas reizes garākām līnijām, aiz kā sekoja īsākās līnijas, kas piedēvēja zīmējumam vajadzīgās detaļas [4].

Sangklois et al. savā pētījumā pēc līdzīgiem *TU Berlin* skiču ievākšanas principiem arī izveidoja skiču datu kopu *Sketchy*, iekļaujot trīs galvenos *TU Berlin* skices ievākšanas pamatkritērijus: vispusīgums (objektam ir jābūt bieži sastopamam dzīvē), atpazīstamība (skicei nav nepieciešams konteksts, lai atpazītu objektu), specifiskums (piemēram, "dzīvnieks" skicei kā kategorija neder). *Sketchy* vēl pievienoja vienu kritēriju – skicējamība (dažas fotogrāfijas mēdz būt pārāk sarežģītas un detalizētas, lai atveidotu no tās skici, tāpēc tika piemeklētas vienkāršākas fotogrāfijas). Galvenā atšķirība starp abu datu kopu ievākšanas metodēm bija tāda, ka *Sketchy* gadījumā fotogrāfijas paraugs tika uz īsu brīdi parādīts un tad paslēpts, lai veicinātu autoram tieksmi zīmēt skici vadoties pēc atmiņas. Šādi kopsummā tika ievāktas no 12500 bildēm 75471 skices, kas ietver 125 kategorijas [15].

1.4. Skiču atpazīšana

Skiču atpazīšana ir plaši pielietota vairumā problēmu, ja skicēs tiek ietverti labi nodefinētas struktūras elementi, piemēram, elementi elektriskās ķēdēs, matemātiskās shēmas, molekulārās diagrammās un muzikālās notācijas. Plašs pielietojums ir arī sastopams skārienekrāna žestos, kas

parasti sastāv no viena vilkuma līnijām. Salīdzinot ar neprofesionālām mākslas skicēm, skices ir sarežģītāk atpazīt abstrakto īpašību dēļ, piemēram, jaunākie neironu tīklu modeļi kā *GoogleNet, AlexNet* vai *ResNet,* kas koncentrējas uz reālu bilžu atpazīšanu, nav arī piemēroti zīmētu attēlu atpazīšanai, jo skices ģeometriskā līmenī ir ļoti atšķirīgas no reālām fotogrāfijām. Cilvēku veidotās skices parasti attēlo daļēju informāciju no reālā objekta, jo vairumā indivīdiem zīmēšanas prasmes ir ierobežotas, kas būtiski atspoguļojas projekcijas kvalitātē. Salīdzinot ar reālām bildēm, skicēm iztrūkst informācija par krāsām un tekstūru attēlā, kas ierobežo atpazīstamību starp līdzvērtīgām kategorijām, piemēram, dzīvnieku sugām. Skices var arī ļoti atšķirties vienas kategorijas ietvaros lielo variāciju zīmēšanas stilā dēļ, kā arī tādēļ, ka tiek izceltas raksturīgākās objekta īpatnības, bet ignorētas mazsvarīgākās, piemēram, tekstūra. Bet par spīti skiču primitīvajai dabai, skices tiek uzskatītas par, iespējams, augstāko atpazīstamības domēnu, ko cilvēks vēl var atpazīt. Atsaucoties uz alu zīmējumiem, cilvēkiem neatkarīgi no laika un kultūras universāli vizuālo konceptu ir vieglāk saprast caur divdesmit tūkstošus gadus veciem alu zīmējumiem, nekā caur senvalodā vai mūsdienu svešvalodā izteiktiem vārdiem [12]. Mathias Eitzs šo fenomenu pierāda, aicinot dalībniekiem identificēt konkrēto zīmēto skici kāda no definētām 250 kategorijām, kuras tika sadalītas 3 hierarhijas līmeņos. Katram dalībniekam bija jāidentificē 100 dažādas skices. Pētījumā parādījās, ka cilvēki vidējā atpazīst skices ar 73.1% precizitāti. Kamēr cilvēki viegli spēja atpazīt vienkāršas skices kā t-kreklu vai čūsku, tādas skices kā kaiju vai solu atpazina tikai 2.5% vai 1% gadījumos. Šī parādība tiek skaidrota, ka līdzvērtīgas kategorijas kā "kaija" vai "balodis" ir grūtāk atpazīt, sevišķi ja skiču zemā kvalitāte ir būtisks faktors, kas ietekmē kategorizēšanu. Otrs iespējamais faktors tiek uzskaitīts, ka cilvēkiem bija lielāka tendence izvēlēties pirmo tuvāko izvēles kategoriju nekā apskatīt citas 250 dotās alternatīvas [6].



1.6.att. **Reprezentētās skices ar dalībnieku kategorijas atpazīstamības precizitāti**

Figure 1.6 **Representative sketches with human category recognition rate**

(Pārveidots no Human Object Sketches [6])

1.5. Skiču atpazīšanas neironu tīkli

Ir sastopami vairāki materiāli, kas ietver skiču atpazīšanas modeļus. Pirmatnējie modeļi tika bāzēti uz figūru atpazīšanu skicē, tomēr mūsdienās iezīmju atlasīšanai no skicēm vairāk tiek izmantotas dziļās mašīnmācīšanas metodes, piemēram, konvolucionālie neironu tīkli [9].

1.5.1. *WaveNet* neironu tīkls

Monta Gupta un Pulkits Mehndiratta skiču atpazīšanai izmantoja *QuickDraw* datu kopu, lai apmācītu mašīnmācīšanas modeļus. Tā kā *QuickDraw* datu kopa ir atlasīta no vairāk nekā 15 miljoniem dalībnieku, iekļaujot 50 miljonu skices, tad skicēs tika veikta x un y koordinašu atmešana jeb skiču vienkāršošana. Autori saglabāja svarīgākos punktus koordinātu plaknē, kas raksturoja skici vislabāk, pārējie punkti tika atmesti. Šis process tika veikts, aprēķinot minimālo un maksimālo x un y punktu katrā skices vilkumā. Ja vilkumā līnijas orientācijas leņķis būtiski neatšķīrās no iepriekšējām līnijām, tad šo līniju starppunkti tika atmesti. Gala rezultātā visa skiču koordinātu sistēma tika pārveidota vektoros, kas tālāk tika sagatavoti kā ieejas dati mašīnmācīšanas modeļiem. Darbā tika izmantoti četri mašīnmācīšanas modeļi kā *KNN* (*Nearest Neighbor*), *RFC* (*Random Forest Classifier*), *SVC* (*Support Vector Classifier*) un *MLP* (*Multi-Layer Perceptron*) modeļi un trīs dziļie neironu tīkli kā konvolucionālais neironu tīkls (*CNN*), *Long-Short Term Memory* (*LSTM*) un *WaveNet* modelis.

KNN modelis sastāvēja no 5 kaimiņiem, *RFC* saturēja 1200 ansambļa kokus, *SVC* izmantoja *"RBF"* kerneli un *MLP* saturēja 2 slēptos slāņus ar *ReLU* aktivizācijas funkcijām un *"Adam"* optimizatoru. *KNN* no iepriekš minētajiem modeļiem sasniedza vislabāko precizitāti jeb 74% skices atpazīstamību, kur *RFC, SVC* un *MLP* sasniedza 71%, 72% un 69% respektīvi. Dziļie neironu tīkli savukārt guva labākus rezultātus nekā mašīnmācīšanas modeļi. *CNN* un *LSTM* atpazina skices ar 78% un 79% precizitāti. Tomēr *WaveNet* modelis pārspēja ar 82% precizitāti dziļos neironu tīklus. Šī parādība ir izskaidrojama ar to, ka *WaveNet* modelis izmanto sakrautas paplašinātas konvolūcijas slāņus, kas izlaiž *stride* parametrus un *pooling* slāņus, padarot milzīgu skiču datu kopas analīzi efektīvāku par *CNN* un *LSTM* [9].

1.5.2. *ResNet* arhitektūras pielietojums

Veins Lū un Elizabete Trana savā pētījumā apgalvoja, ka *CNN* ir grūtāk apmācīt, kad neirona tīkla dziļums palielinās. Tā kā *RasNet* arhitektūra uzstādīja labākus rezultātus nekā virs tūkstots slāņu *CNN* modelis *ImageNet* un *COCO* datu kopās, tad autori izmantoja savos modeļos *RasNet*

atlikuma kartēšanas (*residual mapping*) pamatprincipu, atlikuma vienību (*residual units)* pielietojumam. Rezultātā tika izveidotas četras *CNN* arhitektūras: parastā, platā, visplatākā un saplūdušā arhitektūra.

Parastā arhitektūra satur 7x7 ieejas konvolūcijas slāni, aiz kura seko četras sērijas ar trīs 3x3 atlikuma vienībām. Katrai sērijai pirmajā atlikuma vienībā kartēšanas izmērs tiek samazināts uz pusi, palielinot *stride* parametru un dublējot sērijas filtru skaitu. Tīkla beigās tiek izmantots vidējais globālais *pooling*, aiz kā seko pilnībā savienots slānis, lai izdotu *softmax cross-entropy loss* vērtību. *Dropout* tiek pielietots tīkla sākumā, katras atlikumu vienību sērijas beigās un pirms pilnībā savienotā slāņa.

Plašā arhitektūra aizvieto 12 atlikuma vienības ar 8 (4 sērijas pa 2), saturot divreiz lielāku filtru skaitu, kopumā veidojot 17 slāņus tīklā. *Dropout* paliek nemainīgās pozīcijās – sākumā, katras sērijas beigās un pirms pilnībā savienotā slāņa.

Visplatākajā arhitektūra aizvieto 8 atlikuma vienības ar 3 (3 sērijas pa 1), saturot *dropout* sērijas beigās. Pēc atlikuma vienību sērijām seko sašaurinājuma slānis un pēc tam 2048 filtru slānis. *Dropout* paliek vēl joprojām sākumā un pirms pilnībā savienotā slāņa.

Visbeidzot saplūdušā arhitektūra ir līdzīga parastajai arhitektūrai, vienīgā atšķirība, ka tiek divkārši palielināts filtru skaits pēdējai atlikumu vienību sērijai uz 1024.

Visām šīm arhitektūrām tika pielietota *batch* normalizācija. Katra arhitektūra tika trenēta 15 reizes ar treniņa ātrumu 0,001, tad 5 reizes ar treniņa ātrumu 0,0001.

Gala rezultātā parastā un saplūdušā arhitektūra abi atpazina skices ar 0,653 precizitāti, kamēr zemākais rādītājs bija plātākajai arhitektūrai ar 0,588 precizitāti. Salīdzinot ar šos rādītājus ar citu pētījumu modeļiem, *DeepSketch* un *DeepSketch2* sasniedz 0,754 un 0,777 precizitāti. Autoru modeļi uzrādīja, ka tomēr dziļāki neironu tīkli ir precīzāki nekā īsie, platie neironu tīkli. Kā arī izmantojot *TU Berlin* datu kopu, skices, kas reprezentēja kategoriju "kaija", tika ievietotas trīs reizes stāvošā putna kategorijā, divas reizes lidojošā putna kategorijā un vienu reizi pīles, kanoe un šļirces kategorijā, šādi akcentējot *TU Berlin* datu skices trūkumus (skat. 1.7. att.) [14].

1.7.att. **No kreisās uz labo pusi skiču kategorijas: kaija, balodis, stāvošs putns**

Figure 1.7 **From left to right: a seagull, a pigeon, and a standing bird**

1.5.3. Fišera vektoru pielietojums

Šneidera un Tuitelarsa pētījumā skiču klasifikatoram tika izmantoti Fišera vektori un telpiskās piramīdas, lai būtiski uzlabotu rezultātus no Mathias Eitza piedāvātā risinājuma, proti, no 56% uz 68,9%. Tāpat arī tika palielināti *SIFT (Scale-invariant feature transform)* lauciņu izmēri, kas uzlaboja rezultātus, jo, atlasot mazos reģionus, tika ievākts tikai viens vilkums reģionā, kas ir vājš informācijas avots analīzei. *TU Berlin* skiču datu kopa tika arī būtiski vienkāršota, atmetot visas skices, kur cilvēks ir kļūdījies atlasīšanā. Šī jaunā, atlasītā datu kopa tika pārbaudīta uz 10 dalībniekiem, uzlabojot skiču atpazīstamības precizitāti no 75,8% uz 93%. Procesā laikā dalībnieki deva padomu, ka, ietverot pilnībā visas skiču kategorijas, atvieglotu atpazīšanas darbu būtiski. Arī mašīnmācīšanas modeļa rezultāti uzlabojās no 68,9% uz 85%. Šo konkrēto faktoru autori izskaidroja, ka iepriekšējā datu kopa sastāv no līdzīgām kategorijām, primitīvām skicēm, tomēr neizslēdz arī cilvēka kļūdas faktoru.

Šneiders un Tuitelarsa arī aktualizēja citu jautājumu skicēšanas problēmai no Mathias Eitza oriģinālā jautājuma "kā cilvēki skicē objektus?" uz "kā cilvēki saprot skices?", pievēršot lielāku uzmanību skiču vilktajām līnijām. Autori secināja, ka katram vilkumam skicē ir daudz lielāka nozīme, salīdzinot ar reālo bilžu piemēriem, jo skicēs dotā informācija ir ierobežota. Šis secinājums pamudināja autoriem izņemt no skicēm atsevišķus vilkumus un izpētīt, kā tas mainīs skices atpazīstamības koeficientu. Un tiešām vilkumi, kas galvenokārt atbildēja par objekta raksturīgo formu, bija daudz nozīmīgāki nekā vilkumi, kas piedeva skicei izteiktāku tekstūru vai raksturojumu. Bieži vien šie papildus vilkumi pat padarīja atpazīstamību modelim sarežģītāku [16].

1.5.4. *Sketch2Tag* sistēma

Ženbangs Sun skiču atpazīšanai piedāvā *Sketch2Tag* sistēmu reālā laikā. Salīdzinot ar iepriekš minētajiem piedāvājumiem, *Sketch2Tag* ļauj ne tikai, uzzīmējot datorā ar roku, atpazīt skici, bet arī atrast līdzīgas bildes, kas atbilst skices formai. Turklāt, lai piedāvātu lielāku kategoriju klāstu skicēm, skices zīmēšanas brīdī, tiek piedāvātas gatavas skiču formas, no kurām lietotājs var izvēlēties, lai izvairītos primitīvām skiču zīmēšanas prasmēm, tādejādi arī iegūstot labākus rezultātus skiču atpazīšanā [20]. Jāņem vērā, ka Mathias Eitzs līdzīgu pētījumu ir veicis, kur dalībnieka uzdevums bija izvērtēt skices ar piedāvāto reālo bilžu kolekciju [5].

1.5.5. *DeepSketch2* neironu tīkls

Seddati et al pētījuma ietvaros *DeepSketch2* neironu tīkls bija uzlabojums no iepriekšējā tīkla *DeepSketch*. Tīkla tika izmantoti dziļie konvolūcijas neironu tīkli (*ConvNets*), lai atpazītu skices no *TU Berlin* skiču datu kopas ar 77,69% precizitāti. Galvenā īpatnība *DeepSketch2* bija tādā, ka skices var tikt atpazītas jau no 20%, 40%, 60%, 80% un 100% skices pabeigtības, kas var būt noderīgi arī reālajā laikā. Autors akcentēja skiču parādību, kur pat ja divu dažādu skiču gala rezultāti var būt līdzīgi, skicēšanas secība un stils var skicēm atšķirties. Tādā ziņā jaunais *DeepSketch2* modelis ir par 23% efektīvāks nekā vecais *DeepSketch* modelis ar 5.29% precizitāti 20% skices pabeigtībā. Savukārt pilnas skices atpazīšanas precizitāti *DeepSketch2* modelis no *DeepSketch* palielināja līdz 77,69% no 75,42% [17].

1.5.6. *Sketch-a-Net* neironu tīkls

Yu et al piedāvā *TU Berlin* skiču datu kopas atpazīšanas risinājumu, kas pārspēj cilvēku atpazīšanas sniegtos rezultātus 73,1%, ar 77,95% precizitāti. Salīdzinot ar iepriekš minētajiem *TU Berlin* datu kopas pētījumiem, autori apgalvo, ka līdz publikācijas brīdim, tomēr netika ievērojami izpētīta skiču zīmēšanas pamatprincipi un atpazīšanas procesi. Tāpat netika izveidots specifisks skiču neironu tīkls, kas specializējas tā laika lielākajai datu kopai *TU Berlin.* Salīdzinot esošo reālo bilžu neironu tīklus, trūkums ir tāds, ka šie tīkli pieprasa daudz lielāku treniņa datu apjomu, lai izvairītos no pārmērīgas pielāgošanās (*overfitting*) treniņu kopai.

Tamdēļ Yu et al izveidoja *Sketch-a-Net* dziļo neironu tīklu ar šādu arhitektūru: pieci konvolūcijas slāņi ar *ReLU* aktivācijas funkciju, kur aiz pirmā, otrā un piektā slāņa seko *Maxpool* funkcija. Pēc tam seko trīs pilnībā savienoti konvolūcijas slāņi, kur aiz sestā un septītā slāņa seko *ReLU* aktivācijas funkcija un *Dropout* legalizācija. Visbeidzot pēdējais slānis sastāv no 250 vienībām ar *softmax* funkciju, lai atbilstu *TU Berlin* datu kopas kategoriju skaitam. Šāda arhitektūra tika izvēlēta sekojošu principu dēļ:

a.) Kopīgi dala īpašības ar *Sketch-a-Net* un *Photo-CNN tīkliem*.
   1.) Filtru skaits – pēdējām tā laika bilžu *CNN* filtru skaits palielinās ar tīkla dziļumu. Pirmais slānis sākas ar 64 un dubultojas pēc katra *Maxpool* līdz sasniedz 512.
   2.) Soļi (*stride*) – lai atlasītu iespējams vairāk informācijas, katra konvolūcijas slāņa solis ir 1, izņemot pirmo konvolūcijas slāni, kas ir 3.
   3.) Papildinājums (*padding*) – Nulles papildinājums (*Zero-padding*) ir izmantoti trešajā, ceturtajā un piektajā konvolūcijas slānī ar nolūku, lai izejas izmērs būt vesels skaitlis.

b.) Unikālās īpašības *Sketch-a-Net* tīklam.

1.) Lielāks pirmā slāņa filtrs – tā kā pirmais konvolūcijas slānis ir pats "jūtīgākais" slānis atšķirībā no pārējiem slāņiem, tad filtra izmērs tiek ņemts uz 15 x 15 no ierastā 11 x 11 izmēra. Šādā veidā tiek ielasīta pilnvērtīgāka informācija, jo skices tekstūras trūkums var būt būtisks faktors atpazīstamībai.
2.) Izņemta vietējās reakcijas normalizācija (*Local Response Normalisation* jeb *LRN*) – *LRN* tiek plaši izmantots reālo bilžu atpazīšanas tīklos, lai normalizētu spilgtumu. Tā kā skicēs spilgtums nav faktors, tas mācīšanos var padarīt ātrāku, nezaudējot veiktspēju.
3.) Lielāks *pooling* izmērs – daudzi reālo bilžu CNN izmanto 2x2 *Maxpooling* ar 2 soļiem. Skiču gadījumā tiek izmantots 3x3 *Maxpooling*, tādejādi radot pārklāšanos *pooling* reģionos. Šādā darbība uzlabo tīklu par 1%.

Yu et al papildus vēl apstrādāja *TU Berlin* skiču datu kopa. Tā kā dažādi cilvēki var uzzīmēt vienu un to pašu skici dažādi, viens no paņēmieniem ir veikt skices lokālās un globālās deformācijas. Lokālā deformācijā vilkumi ar mazāku garumu un izliekumu tiek vairāk deformēti nekā garāki un līkumainākie vilkumi. Globālā deformācijā tiek deformēta visa skice, izmainot skices kontūras formu.

Vēl autors piebilst, ka tā laika esošajos pētījumos netika izmantota informācija par vilkumu zīmēšanas secību. Tā kā pirmie vilkumi, kas ir garāki, ietver svarīgāko informāciju par skici, tad šim nolūkam pēdējie īsākie vilkumi tika izņemti ārā no skicēm par 10-60%. Šī procedūra padarīja ģenerētās skices ar dažādu abstrakcijas līmeni, kas ir tuvāks cilvēka zīmēšanas stila faktors.

Autors arī lai uzlabotu tīkla veiktspēju, izmantoja trīs saplūšanas metodes: raksturiezīmju līmeņa saplūšanu (*CNN* apmācību konkatenācija), vērtējuma līmeņa saplūšanu (arhitektūras ansambļa vidējās *softmax* vērtības) un Džointsa Bejesiana saplūšanu (pārveidots sejas verifikācijas modelis).

Visbeidzot autors papildus šīm uzlabošanas metodēm ierobežotā *TU Berlin* datu kopas skiču skaita dēļ, *Sketch-a-Net* tīklu ietrenēja ar *ImageNet-1K* datu kopu. Šīs bildes tika pirmīt apstrādātas ar fotogrāfijām, kur figurētam vienam objektam tiek izvadīta tikai kontūra. Šāda prakse modeļa inicializēšanu padarīja labāku nekā netīša svaru un nobīdes inicializācija.

Yu et al pētījumā parādījās, ka *Sketch-a-Net* tīkls uzrādīja labākus rezultātus, atpazīstot 1.7. attēlā attēloto kaiju ar 26,92% nekā cilvēks ar 2,5% precizitāti. Salīdzinot dažādus *Sketch-a-Net* modeļus no sākotnējā *Sketch-a-Net* modeļa ar 72,20% precizitāti, modelim pievienojot ģenerētas

skices ar lielāku abstraktumu, precizitāte uzlabojās līdz 74,57%, savukārt modelim pievienojot skices tikai ar deformāciju, precizitāte uzlabojās līdz 75,52%, bet, ja modelim pievienoja visas iepriekš minētās skices, precizitāte uzlabojās līdz 77,06%, un vēl pievienojot iepriekš minētajam modelim visas saplūšanas metodes, precizitāte gala modelim uzlabojās līdz 77,95%. Svarīgi pieminēt, ka no visām modifikācijām, tieši globālā deformācija atstāja lielāko efektu precizitātes uzlabošanai – no 72,20% uz 75,40%, kamēr lokālā deformācija deva līdz 73,86% precizitāti. Savukārt modeļa ietrenēšana ar fotogrāfiju datu kopas kontūrām palielināja precizitāti modelim par 1,0% [22].

1.6. Animācijas tēla klasificēšana

Sakarā ar to, ka animācijas tēlu datu kopa ir maz izplatītas, tad šādā nolūkā Zhou et al izveidoja *ToonNet* datu kopu, kas ietver dažādus animācijas tēla stilus. Ievākšanas gaitā, izmantojot filtrēšanas rīkus kvalitātes uzlabošanai, tika savāktas 4400 bildes, kas ir ļoti mazs skaitlis. Tāpēc šajā nolūkā tika izmantotas divas animācijas tēlu ģenerēšanas metodes. Pirmā metode ģenerē no 3D modeļiem 2D zīmējumus, izmantojot konkrētu leņķi telpā, ēnojuma tehniku, lai ģenerētu no 200 3D modeļiem 2000 bildes. Otra metode ļāva esošās animācijas pārvērst 3D modelī izmantojot *MagicToon* rīku, lai izpildītu jau minēto pirmo metodi, kas rada vēl 2D zīmējumus, bet jau ar citādāku leņķi un ēnojumu, iegūstot vēl 1500 jaunas bildes.

Lai šie animācijas tēli tiktu atpazīti, autori izveidoja *ToonNet* neironu tīklu, kas izmantoja 3 galvenās metodes:

a) Ieejdatu vienotā stilizēšana (*IUS*) – sakarā, ka zīmētu attēlu zīmēšanas stili savstarpēji krasi atšķiras, attēli tika iepriekš vienkāršoti vienotā stilā vieglākai klasificēšanai. Šajā nolūkā tika izmantota aizkrāsošanas algoritms, kas izvēlējās reģionā spilgtāko pikseli, un aizkrāsoja visu laukumu vienā krāsā.
b) Ar atribūtiem ievietots tīkls (*FIN*) – tā kā tradicionāli neironu tīklos nav informācija par *RGB* krāsu histogrammu, kas var būt noderīga animācijas tēlos, šī informācija tika ievietota tīklā priekš labākas veiktspējas.
c) Papildus tīkli tīklā (*NPN*) – izmantojot *ToonNet* datu kopu, no paralēli ietrenētiem tradicionāliem neironu tīkliem tika atlasīti izejas dati, kas tika apvienoti vienā *CNN*, garantējot labāku tīkla stabilitāti un atpazīstamības precizitāti nekā ar vienu neironu tīklu.

Pētījumā *NPN-3* tīkls treniņa laikā, salīdzinājumā ar *GoogleNet* tīklu, guva ātrāku minimālā zaudējumā plato fāzi (2 epohos) nekā *GoogleNet* tīkls (10 epohos) [23].

1.7. Attēlu klasificēšana, izmantojot siluetu

Kamēr nav veikti pētījumi zīmētu attēlu klasificēšanā, izmantojot siluetu, reālo objektu siluetu klasificēšanā tika veikti vairāki pētījumi. Dedeoglul pētījumā galvenais princips reāliem objektiem bija klasificēt un atšķirt dažādus objektus filmēšanas laikā. Pirms klasificēšanas tika manuāli sagatavotas iepriekš definētas kategorijas: "cilvēks", "cilvēku grupa", "transportlīdzeklis" un "troksnis". Pēc tam no video fragmentiem tika ievāktas bildes, no kurām tika izņemtas ēnas un dažādi trokšņi, lai no objektiem iegūtu tīru siluetu, izmantojot dažādus malu detektēšanas metodes kā *SIFT, SURF* vai *ORB*. Pēc tam šim siluetam tika detektēts masas centra punkts, no kā tika noteikts attālums līdz katram punktam silueta perimetrā. Šie rezultāti pēc tam atspoguļoja silueta distances signāla histogrammā. Šādā veidā tika atlasītas vairākas bildes, kas tālāk tika reprezentētas histogrammās, un glabājās datubāzē ar kategorijai atbilstošo nosaukumu. Video filmēšanas laikā, jaunievāktās histogrammas tika salīdzinātas vecajām. Un, ja šo attālumu atšķirības bija minimālas histogrammās, tad tikko uzņemtā kadra objekts ieguva iepriekšējās histogrammas kategorijas nosaukumu. Tomēr autori uzskaita trūkumu, kā šī metode ir no objekta skata leņķa atkarīga [3]. Citos pētījumos tika salīdzināta silueta stūru un to leņķu informācija, lai atpazītu objektus savā starpā [21].

2. MATERIĀLI UN METODES

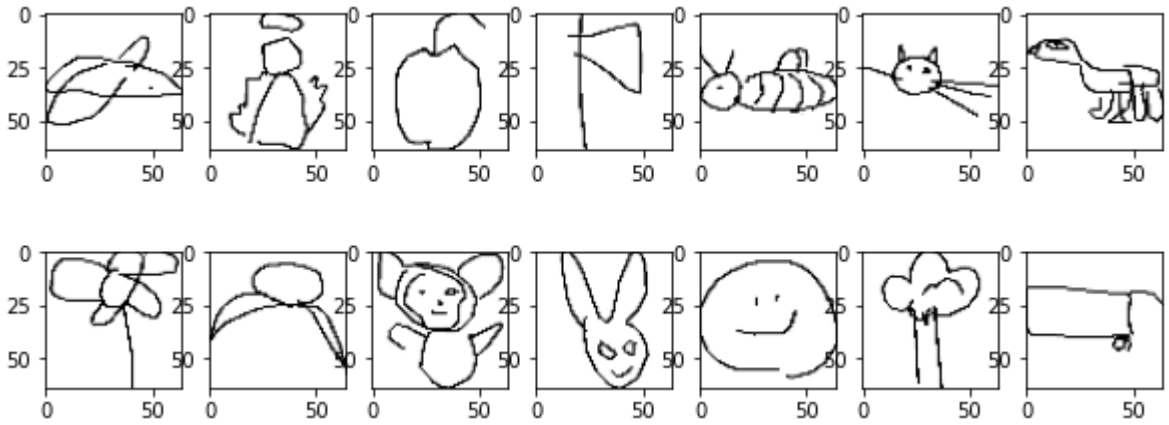
2.1. Datu kopa

QuickDraw datu kopa tiks izmantota kā galvenā datu kopa neironu tīklam. Sakarā ar to, ka kopā tiek ietvertas 50 miljonu skices, kas ir sadalītas 345 kategorijās, treniņu nolūkiem šis ir pietiekami liels apjoms [7, 11]. Salīdzinot ar *TU Berlin* datu kopu, kas ietver 20 000 skices 250 kategorijās, *QuickDraw* gūst pārmērīgu pārsvaru datu apjoma ziņā.

Katrai skicei *QuickDraw* datu kopā ir parametri: unikālais identifikators (*key_id*); kategorijas nosaukums (*word*); pazīme, vai *QuickDraw* spēles laikā modelis atpazina, zīmēto skici (*recognized*); laiks (*timestamp*); spēlētāja valsts kods (*countrycode*); masīvs ar skices informāciju (*drawing*). Masīvs turpretī satur katra vilkuma informāciju skicē ar parametriem x un y koordinātēm plaknē un t, kas ir konkrētā vilkuma zīmēšanas sākuma laiks. Darba nolūkā tiek izmantots kategorijas nosaukums, atpazīšanas pazīme un atbilstošais skices masīvs, no kā tiek atlasītas x un y koordinātas. Skices, kas tika atpazītas spēles laikā, dos lielāku garantiju, ka izveidotās skices ir pietiekami kvalitatīvas, lai apmācītu neironu tīklu, savukārt t vērtība tiek atmesta ar nolūku, jo pikseļattēlos dotā informācija ir gala produkts, kas neietver zīmēšanas secību.

No 345 piedāvātajām kategorijām tika atlasītas 14 kategorijas apmācības un atpazīšanas nolūkos: “lidmašīna” (*airplane*), “eņģelis” (*angel*), “ābols” (*apple*), “cirvis” (*axe*), “bite” (*bee*), “kaķis” (*cat*), “suns” (*dog*), “puķe” (*flower*), “varde” (*frog*), “pērtiķis” (*monkey*), “zaķis” (*rabbit*), “priecīga sejiņa” (*smiley face*), “koks” (*tree*), “kravas auto” (*truck*). Konkrēti šīs kategorijas tika atlasītas, jo pikseļgrafikā ļoti bieži figurē tieši šie minētie elementi, tajā pašā brīdī kategorijas tiek sadalītas līdzvērtīgās proporcijās, ietverot dzīvniekus, priekšmetus, augus un augļus un abstraktākus objektus (skat. 2.1. att.).

Tomēr tā kā ir veikti vairāki pētījumi, kur tika apmācīti neironu tīkli ar *TU Berlin* datu kopu [14, 16, 17, 22], tad darbā apmācītie neironu modeļi tiks salīdzināti arī ar *TU Berlin* datu kopu. Datu kopā ietvertās *TU Berlin* skices atrodas *.png* formātā ar oriģinālā attēla izmēru 1111x1111 pikseļi, kam būs jāizmanto citi, bet līdzīgi pārveidošanas mehānismi nekā *QuickDraw* datu kopai. *TU Berlin* gadījumā tiks izmantotas visas 250 kategorijas, sakarā ar mazo datu apjomu.



2.1.att. **QuickDraw skiču paraugi no katras apmācību kategorijas: lidmašīna, eņģelis, ābols, cirvis, bite, kaķis, suns, puķe, varde, pērtiķis, zaķis, priecīga sejiņa, koks, kravas auto**

Figure 2.1 **QuickDraw sketch training examples from every category: airplane, angel, apple, axe, bee, cat, dog, flower, frog, monkey, rabbit, smiley face, tree, truck**

2.2. Pētījuma gaita

Pētījuma gaita sastāv no 4 galvenām daļām:

1) depikselizācijas algoritma implementācija – depikselizācijas algoritms ir nepieciešams, lai apietu pikseļattēla rakstura galvenos trūkumus, proti, mazā attēla izmērs un ierobežotās informācijas daudzums. Kā galvenās vadlīnijas tiks izmantotas no Kopf et al. publikācijas [13]. Tā kā šāda algoritma implementācija jau tika veikta *Python* valodā un publicēta *github* vidē (https://github.com/vvanirudh/Pixel-Art [pieejams: 29.05.2021]), tad šim nolūkam tiks pārbaudīta šī konkrētā implementācija. Kā treniņdati depikselizācijas algoritmam tiks izmantoti paša Kopfa piedāvātie pikseļattēli vietnē, kur ir parādīta depikselizācijas algoritma salīdzinājums ar *Vector Magic* (https://johanneskopf.de/publications/pixelart/supplementary/comparison_vectormagic.html [pieejams: 29.05.2021]).
2) *Sketch-a-Net* arhitektūras implementācija un analīze – *Sketch-a-Net* arhitektūra tika izvēlēta, jo salīdzinot ar citiem *TU Berlin* datu kopas atpazīšanas rīkiem, *Sketch-a-Net* uzrādīja labākos rezultātus līdz 77.95% atpazīšanas precizitātei [22]. Šis neironu tīkls tiks trenēts gan uz *QuickDraw* datu kopas kā galveno treniņavotu, gan uz *TU Berlin* datu kopas, rezultātu salīdzināšanai ar citiem literatūras avotiem. Pēc tam tiks ģenerēti pašzīmētas skices, ko neironu tīkls nekad nav redzējis, lai pārbaudītu abu modeļu atpazīšanas darbību vispārīgos gadījumos.

3) patvaļīga neironu tīkla implementācija un analīze – tiks meklēta optimālā konvolūcijas tīkla arhitektūra, kas ir spējīga labi atpazīt skices. Salīdzinot vairākus modeļus savā starpā, tiks izvēlēts optimālākais modelis, kas pēc tam tiks pēc līdzīga principa trenēts ar *QuickDraw* un *TU Berlin* datu kopu, kā arī tiks pārbaudīts paszīmētas skices atpazīšanas mehānisms.
4) pikseļgrafikas attēla atpazīšana, izmantojot ģenerētos neironu tīklus – no dažādām interneta vietnēm tiks atlasīti nejauši, bet *QuickDraw* un *TU Berlin* kategorijas ietveroši pikseļattēli, un tie tiks pārbaudīti uz ģenerēta *Sketch-a-Net* un patvaļīga neirona tīkla modeļa. Šie pikseļattēli pirms atpazīšanas tiks apstrādāti, lai pietuvinātu pēc iespējas tuvāk skices zīmēšanas stilam, izmantojot depikselizācijas algoritmu vai citus bilžu pārveidošanas rīkus.

Neironu tīkli tiks darbināti, izmantojot brīvi piejamu *Google Colab* programmēšanas vidi *Python* kodā, kas ir tieši paredzēts, lai trenētu un analizētu mašīnmācīšanas modeļus. *Google Colab* GPU virtuālās mašīnas tehniskie parametri no 2019. gada ir GPU: Nvidia K80 / T4 ar 12 / 16 GB GPU atmiņu un ar 0.82GHz / 1.59GHz GPU atmiņas jaudu. *Python* valodā tiek izmantotas būtiskākās bibliotēkas: datu reprezentācijas rīks *Matplotlib* bibliotēka, attēlu datu masīvu apstrādes rīks *NumPy* bibliotēka, bilžu pirmapstrādes rīks *OpenCV2, skimage* bibliotēkas un mašīnmācīšanas rīks *TensorFlow* bibliotēka ar *Keras* neironu tīklu prototipēšanas bibliotēku.

Pētījuma gaitā netiek izmantota attēlu klasificēšana, izmantojot siluetu [3, 21], taču var būt noderīgs rīks, lai apstrādātu pikseļattēlus labākai atpazīstamībai. Kā arī netiks izmantoti pikseļattēlu ģenerēšanas rīki [8, 10], lai trenētu mašīmācīšanas modeļus šādu iemeslu dēļ:

1. Gerstnera et al piedāvājums, lai arī piedāvā veidot no reālām bildēm pikseļattēlus, ģenerētie pikseļattēli zaudē pikseļattēlam raksturīgākās īpašības, kas ietver ar nolūku ievietotus pikseļus [8].
2. Han et al piedāvājums izveidot no animācijas tēliem pikseļattēlus ir laba alternatīva, lai apmācītu mašīnmācīšanas modeli ar optimālas kvalitātes ģenerētiem pikseļattēliem [10]. Tomēr jāņem vērā faktors, ka arī animācijas tēli nav sevišķi plaši publiski pieejami lielā apjomā un brīvā formātā. Zhou et al, veidojot ToonNet datu kopu, šo faktu apliecināja, tādejādi datu pavairošanai izmantoja konvertēšanas rīkus no 3D uz 2D formātu [23].

3. Apjomīgā *QuickDraw* datu kopa ir ievērojami liela, lai apmācītu efektīvi neironu tīklu. Ja *TU Berlin* no darba izstrādes brīža būtu lielākā skiču kopa, tad papildus pikseļģenerēšanas rīki varētu būtiski uzlabot neironu tīklu apmācīšanas rezultātus.

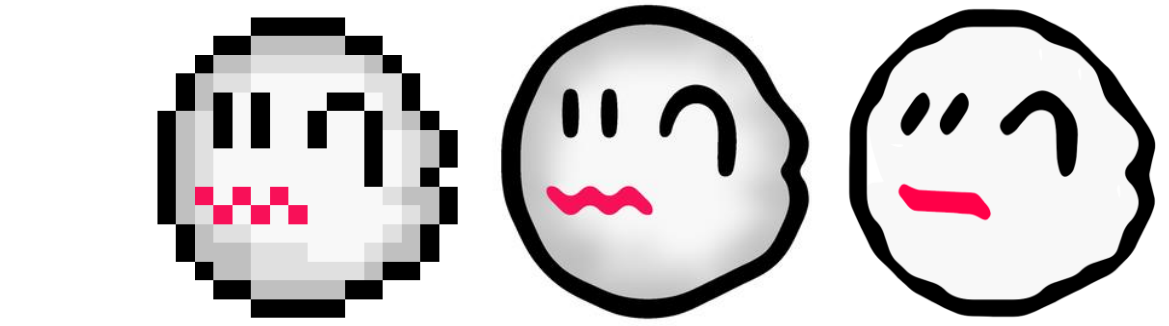
2.3. Datu apstrāde

Attēlu pirmsapstrādē pikseļattēli tiks samazināti līdz 64x64 izmēram, lai mašīnmācīšanas modeļi spētu efektīvāk mācīties un analizēt datus, tāpēc arī treniņnolūkos *QuickDraw* un *TU Berlin* skices tiks samazinātas līdz 64x64 izmēram. Lai nepazaudētu skicei raksturīgo informāciju, skices vilkumi tiks iebiezināti pirms bildes samazināšanas. Var būt gadījumi, kad pašu skici aptver liels tukšais laukums, šādos nolūkos šis laukums tiks nogriezts līdz skices objekta minimālākajam un maksimālākajam punktam. Tomēr, lai saglabātu bildei vajadzīgo 1:1 attiecību pirms attēla samazināšanas uz 64x64 izmēru, tiks pievienots papildus tukšais laukums.

3. REZULTĀTI UN DISKUSIJA

3.1. Pikseļgrafikas tēla depikselizācija, izmantojot gatavu algoritmu

Izmantojot jau gatavu depikselizēšanas algoritmu, sagaidāmais rezultāts neatbilda praksē iegūtajam rezultātam.



a) oriģinālā bilde b) sagaidāmais rezultāts c) iegūtā bilde

a) oriģinālā bilde b) sagaidāmais rezultāts c) iegūtā bilde

3.1.att. **Pikseļgrafikas elementa depiksilizācija, izmantojot gatavu algoritmu**

Figure 3.1 **Pixel art depixelizition using implemented algorithm**

3.1. attēlā spoka variantā elements, lai arī saglabā nosacīti oriģinālās bildes raksturīpašības, iegūtais rezultāts tomēr vietās krasi atšķiras no oriģināla, proti, mute zaudē lauztās līnijas raksturojumu, acu izliekums atšķiras un krāsu pāreja gar malām neeksistē. Delfīna variantā šie trūkumi izpaužas daudz radikālā līmenī, kur krāsojums vienā brīdī pilnībā zaudē kontroli, līdz ar to iegūtā bilde pat tuvu neatbilst sagaidāmajam rezultātam. Līdz ar to secinājums ir patstāvīgi īstenot depikselizācijas algoritmu, vadoties pēc Kopfa un Lisšinska publikācijas [13].

3.2. Depikselizācijas algoritma implementācija

Depikselizācijas algoritms, kā jau iepriekš minēts darbā 1.4 attēlā, sastāv no sešiem galveniem posmiem:

a) pikseļgrafikas attēla ievade - ievadot 18x18 līdzvērtīgu pikseļgrafikas spoka attēlu, tiek izveidots attēla izmēram atbilstošs 18x18 grafs, kur katra virsotne reprezentē attiecīgo krāsu pikseli. Pēc tam katra virsotne tiek savienota ar katru kaimiņu.

b) sākotnējā krāsu reģionu grafa izveide - tālāk seko šķautņu izmešana no grafa, kur, ja savstarpējām virsotnēm ir būtiskas krāsu atšķirības, respektīvi, ja krāsu atšķirība starp virsotnēm *YUV* krāsu kodēšanas sistēmā ir vairāk nekā $\frac{48}{255}$, $\frac{7}{255}$ un $\frac{6}{255}$, tad šīs šķautnes tiek izmestas, tādējādi norobežojot grafā krāsu reģionus.

c) savstarpējo diagonāļu izšķiršana - pirms tiek izšķirtas savstarpējās diagonāles, ejot cauri grafam ar 2x2 apakšgrafu, tiek izmestas abas diagonāles pilnībā savienotā apakšgrafā. Lai izšķirtu pārējos gadījums, tiek ņemti vērā trīs faktori, kad izšķirošā diagonāle apakšgrafā:

1) veido garāku savienojumu – katras diagonāles savienojuma garums jeb svars tiek mērīts līdz brīdim, kamēr diagonāles virsotnes kaimiņu skaits ir divi.
2) ir atribūta nevis fona sastāvdaļa – no grafa tiek atlasīts 8x8 apakšgrafs, kur šīs abas savstarpējas diagonāles atrodas apakšgrafa centrā. Pēc tam katrai diagonālei tiek saskaitīts kopējais mezglu skaits konkrētās diagonāles savienojumā. Ja mezglu skaits vienā diagonāles savienojumā ir lielāks par otru, tad ir pamats domāt, ka šī diagonāle ir visdrīzāk fona sastāvdaļa attiecībā uz otru diagonāli, kas visticamāk ir atribūts attēlā. Kopfa and Lisšinska publikācijā svars tika noteikts pēc savstarpējo mezglu skaita starpības. Konkrētajā implementācijā diagonāles savienojuma mezglu skaits tiek uzskatīts kā negatīvs faktors, kur svars katrai diagonālei ir negatīvs skaitlis no savienojuma mezglu skaita.
3) neveido "saliņu" – ja vienai no diagonālēs mezgliem ir tikai viens kaimiņš, kas ir pati diagonāle, tad, lai neveidotu "saliņas", šai diagonālai tiek piešķirts svars ar vērtību pieci.

Kad visiem trīs faktoriem ir aprēķināts svars, tad izšķirošā diagonāle ir diagonāle ar lielāko kopējo svaru, otra diagonāle tiek izmesta. Ja ir gadījumi, kad abām diagonālēm ir vienāds kopējais svars, tad šīs diagonāles tiek uzskatītas par līdzvērtīgām, un abas diagonāles tiek izmestas no grafa.

d) šūnu formas pārveidošana – šūnas formas izveide notiks jau jaunā grafā ar izmēru (w+1)(h+1) no iepriekšējā grafa. Šīs šūnas formas tiks kartētas pēc Voronoja diagrammas principa, proti, katra šūna ietvers tādu reģionu, kas atrodas vistuvāk centram jeb Voronojas punktam. Depikselizācijas algoritmā Voronojas punkts būs virsotne krāsu reģionu grafā un Voronojas segments jeb punktu kopa būs puse no šķautnes, kas pieder attiecīgajam pikselim. Zinot šos parametrus, tiek aprēķināti attālumi attiecībā pret citu šūnu Voronoja punktiem un segmentiem, un tiek izveidots jauns grafs ar atbilstošām Voronoja šūnām.

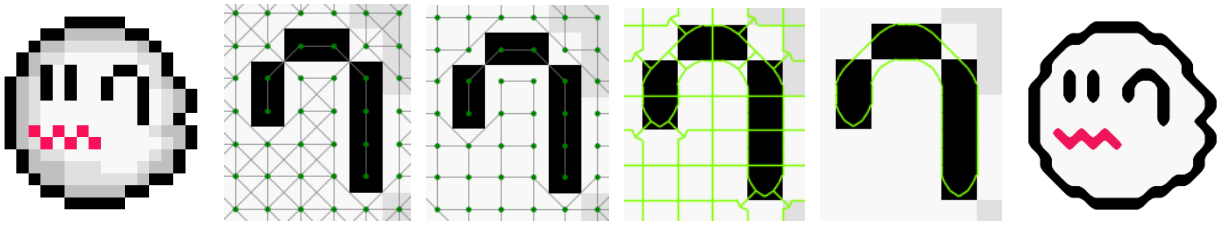
e) kantaino līkņu gludināšana kontūrās – sekojot līdzi krāsa reģionu grafa informācija, no tikko izveidotām šūnām tiek atlasītas šūnas ar līdzīgu krāsojumu, tādejādi veidojot kontūras. Tā kā Kopfa and Lisšinska publikācijā Voronojas diagrammas algoritma apraksts netika ietverts, šūnas

tika veidotas ar *voronoi 0.2.0 Pyhton* pakas palīdzību, kas atsevišķus šūnas apvalka segmentus saskaldīja ļoti mazus. Ja šie segmenti nebija kontūras sastāvdaļa, tie bija jāatsijā.

Arī publikācijā lai izšķirtu kontūras krāsu, kad kontūra ietvēra vairākas krāsas, no šīm krāsām tika izveidota krāsu pāreja. Tā kā skiču datu kopas treniņu dati sastāv no izteikti melna/balta krāsojuma, tad krāsu gradients var pasliktināt atpazīšanas rezultātu. Tāpēc kā kontūras krāsa tika izvēlēta visbiežāk sastopamā krāsa.

B-splaina līkņu implementācija un optimizācija tika izlaista dēļ Voronoja šūnu modificētas implementācijas.

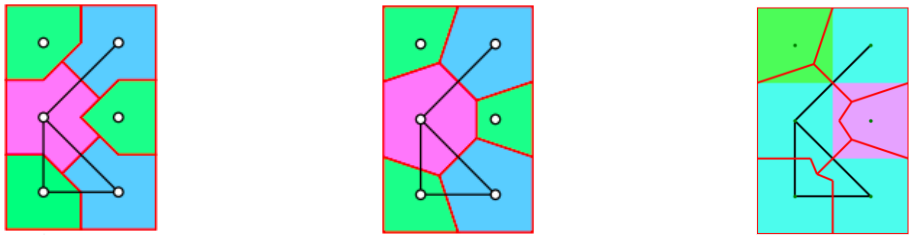
f) gala rezultāta izvade – atbilstošais kontūru grafs tika izvadīts ārā, izmantojot *Matplotlib* paku. Pirms izvades krāsojums tika nomainīts atpakaļ uz *RGB* krāsu kodēšanas sistēmu.



3.2.att. **Depikselizācijas algoritma implementācijas darba gaita**

Figure 3.2 **Overview of depixelization algorithm implementation**

Izstrādājot algoritmu, Kopfa and Lisšinska publikācijā tika konstatētas dažas neprecizitātes: Publikācijā *Figure 3* attēla aprakstā ievadītās bildes izmērs ir 18x18 pikseļi nevis 16x16. Kā arī Voronoja šūnu izveidei uzskatāmais piemērs tikai daļēji ņem vērā pikseļu krāsu reģionu grafu. 3.3. attēlā šo parādību var redzēt, paraugoties, ka Voronoja šūnas formas viena pret otru ir vienādas horizontālā šķērsgriezumā, kamēr krāsu reģionu grafs nav vienāds.



a) precīza Voronoja diagramma b) vienkāršota Voronoja diagramma c) izmantojot *voronoi* paku

3.3.att. **Uzskatāmais piemērs Voronoja šūnu izveidei**

Figure 3.3 **Example for Voronoi cell computation**

Salīdzinot abas algoritma implementācijas, izveidotā depikselizācijas implementācija atbilda tuvāk sagaidāmajam rezultātam nekā gatavā, taču jāņem vērā, ka tika izlaistas krāsu pārejas un B-splainu līknes (skat. 3.4.attēlu).

a) sagaidāmais rezultāts b) izveidotā algoritma implementācija c) gatavā algoritma implementācija

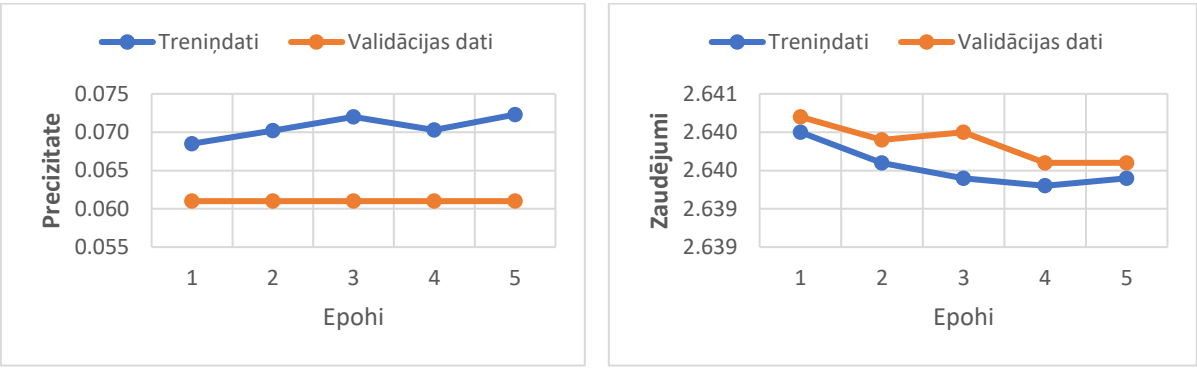
a) sagaidāmais rezultāts b) izveidotā algoritma implementācija c) gatavā algoritma implementācija

3.4.att. **Depikselizācijas algoritma implementāciju salīdzinājums**

Figure 3.4 **Depixelizition algorithm implementation comparison**

3.3. Sketch-a-Net neironu tīkls uz QuickDraw datu kopas

Darbā tika izveidots neironu tīkla modelis atbilstoši *Sketch-a-Net* arhitektūrai (skat. 1. pielikumu). Skices no *QuickDraw* datu kopas tika zīmētas izmērā 255x255 pikseļi, kas tika samazinātas līdz 225x225 izmēram, lai modelis spētu pieņemt datus. Apmācot modeli ar 80% treniņu datiem un 20% validācijas datiem, tika iegūti ļoti zemi rādītāji (skat. 3.5. attēlā).



3.5.att.. **Treniņdatu un validācijas datu precizitāte un zaudējumi, izmantojot Sketch-a-Net arhitektūru ar QuickDraw datu kopu**

Figure 3.5 **Training and validation accuracy and loss using Sketch-a-Net architecture with QuickDraw data set**

Šai parādībai var būt šādi iemesli:

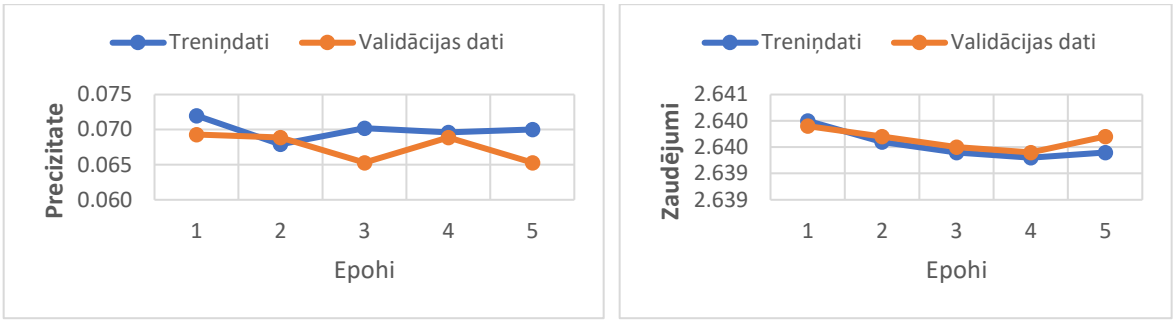
1) *Sketch-a-Net* modelis netika iepriekš sagatavots ar priekšapmācības datiem. Yu et al. pētījumā modelis pirms tam tika apmācīts ar *ImageNet-1K* bildēm, no kurām tika izvilktas malas. Taču pētījumā šāda prakse uzlaboja atpazīšanas rezultātus tikai no 76,06% līdz 77,06% pret nejauši izvēlētām modeļa vērtībām.
2) 225x225 bildes izmēru *Sketch-a-Net* modelim var būt par grūtu uztver un apmācīt, tāpēc šinī nolūkā tiek piedāvāta līdzīga alternatīva oriģinālajam *Sketch-a-Net* modelim (skat. 2. pielikumu). Modificētam *Sketch-a-Net* modelim vienīgā atšķirība ir filtru izmēri, kur sākotnējais filtra izmērs ir samazināts līdz 64x64, un attiecīgi līdzīgās *Sketch-a-Net* arhitektūras proporcijās tiek pārveidoti pārējo filtru izmēri līdz tiek sasniegts 1x1 filtrs. Soļi, papildinājumi un filtru skaiti visos konvolūcijas slāņos paliek joprojām nemainīgi.

Līdz ar *Sketch-a-Net* neirona tīkla modifikāciju, *QuickDraw* skices tiek samazinātas arī līdz 64x64 izmēram. Lai nepazaudētu informāciju par skiču līnijām, līnijas zīmēšanas brīdī tiek pabiezinātas līdz 5 pikseļiem. Tomēr apmācot pārveidoto *Sketch-a-Net* modeli, rādītāji praktiski gandrīz nemainījās no iepriekšējā oriģinālā modeļa, apsverot iespēju, ka vaina iespējams ir priekšapmācības trūkumā (skat. 3.6. att. a variantu).

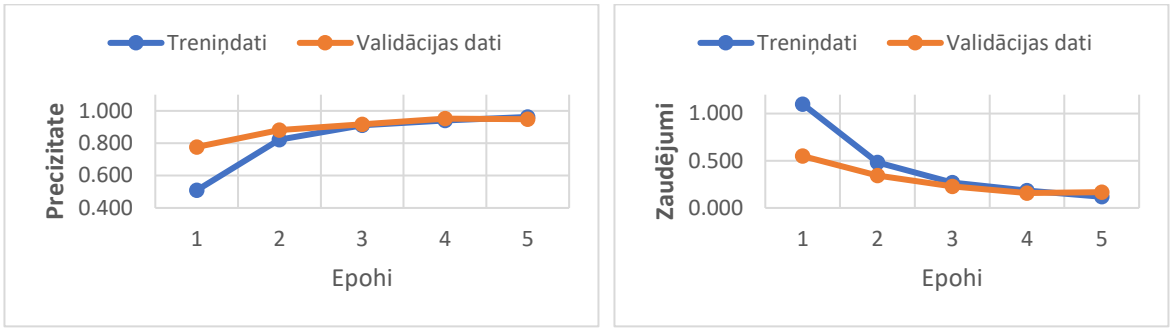
Bet tika novērota interesanta parādība, ka samazinot skiču kategoriju skaitu no 14 uz 5 (“lidmašīna”, “eņģelis”, “ābols”, “cirvis”, “bite”), modelis sāka uzrādīt daudz augstākus rezultātus (skat. 3.6. att. b variantu), kur kopējais skiču skaits samazinājās no 14 x 2500 uz 5 x 2500 bildēm. Validācijas datu zaudējums samazinājās 5 epohos līdz 0,1661 un precizitāte pieauga līdz 0,9488. Arī palielinot bilžu skaitu kategorijā no 2500 uz 10000 bildēm vai nomainot citas piecas kategorijas, nemainīja modificētā modeļa rezultātu būtību. Tomēr kategoriju skaita palielināšana gan lika rādītājiem kristies. Arī, ja šīs pašas 5 kategorijas ar 225 x 255 izmēru tika padotas atkal uz oriģinālā *Sketch-a-Net* modeļa apmācībai, uzlabojoši rezultāti nebija manāmi.

Tiek apsvērts variants, ka *Sketch-a-Net* konvolūcijas astotajā slānī (skat. 3. pielikumu) filtru skaita parametrs var būtiski ietekmēt modeļa rādītājus, jo šis ir vienīgais dinamiskais parametrs, kas ir atkarīgs no kategoriju skaita.

a)



b)

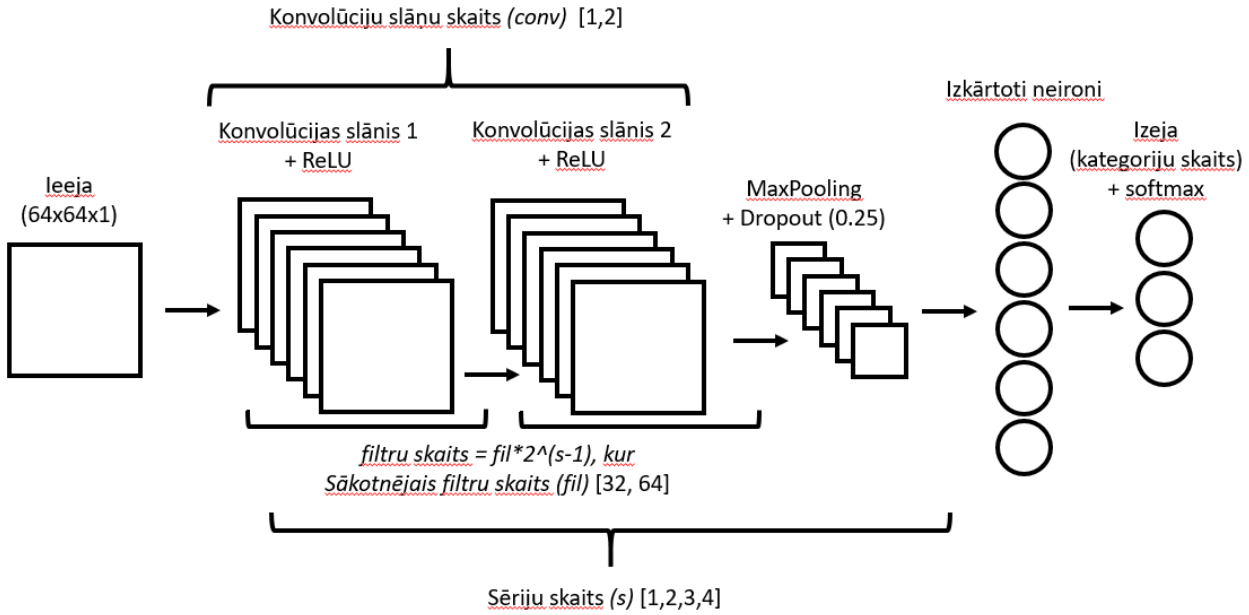


3.6.att. **Treniņdatu un validācijas datu precizitāte un zaudējumi, izmantojot modificētu Sketch-a-Net modeli ar QuickDraw datu kopu, ietverot a) 14 kategorijas, b) 5 kategorijas**

Figure 3.6 **Training and validation accuracy and loss using modified Sketch-a-Net model with QuickDraw data set including a) 14 categories, b) 5 categories**

3.4. Ģenerēts neironu tīkls uz QuickDraw datu kopas

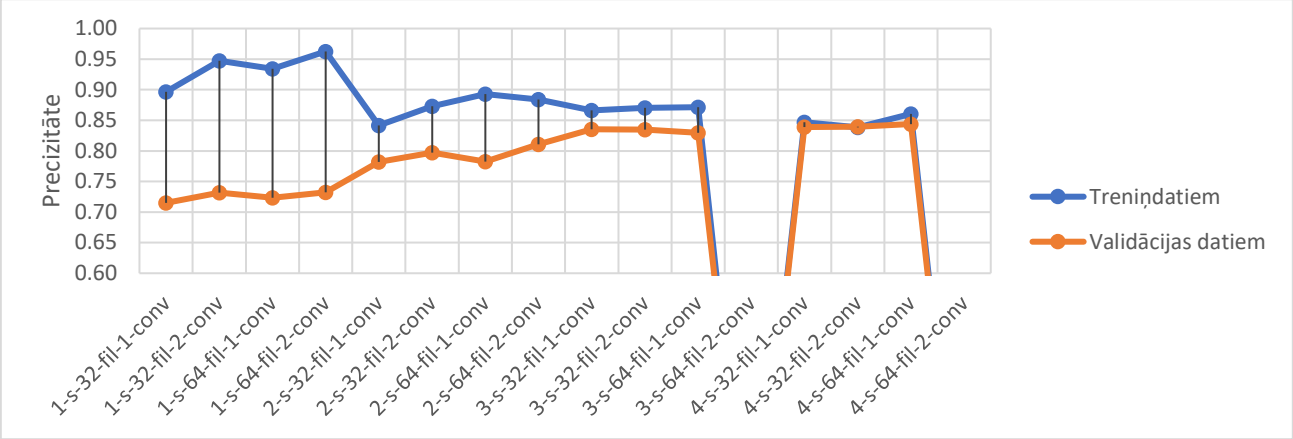
Lai salīdzinātu, kāda arhitektūra skiču atpazīšanai strādā vislabāk, tika izveidoti vairākas *CNN* arhitektūras ar hiperparametriem: sēriju (*s*), sākotnējais filtru (*fil*) un konvolūcijas slāņu skaits (*conv*). *CNN* arhitektūru ģenerēšanas plāns ir parādīts 3.7. attēlā. Ieejas slāņa masīva forma ir 64x64x1 dimensijā, pēc tam seko viens vai divi konvolūcijas slāņi ar filtra izmēru 3x3 un sākotnējo filtru skaitu 32 vai 64. Katrs konvolūcijas slānis satur *ReLU* aktivācijas funkciju. Aiz konvolūcijas slāņiem uzreiz seko *MaxPooling* funkcija ar *Dropout*. *Dropout* ir nepieciešams, ja gadījumā abi secīgi konvolūcijas slāņi piesātinās ar treniņdatu informāciju. Tad iepriekš minētais process atkārtojas līdz četrām reizēm jeb sērijām, kur katrā nākamajā sērijā palielinās filtru skaits par divām reizēm vairāk nekā iepriekšējā sērijā.



3.7.att. **Konvolūcijas neirona tīkla ģeneratora uzbūve**

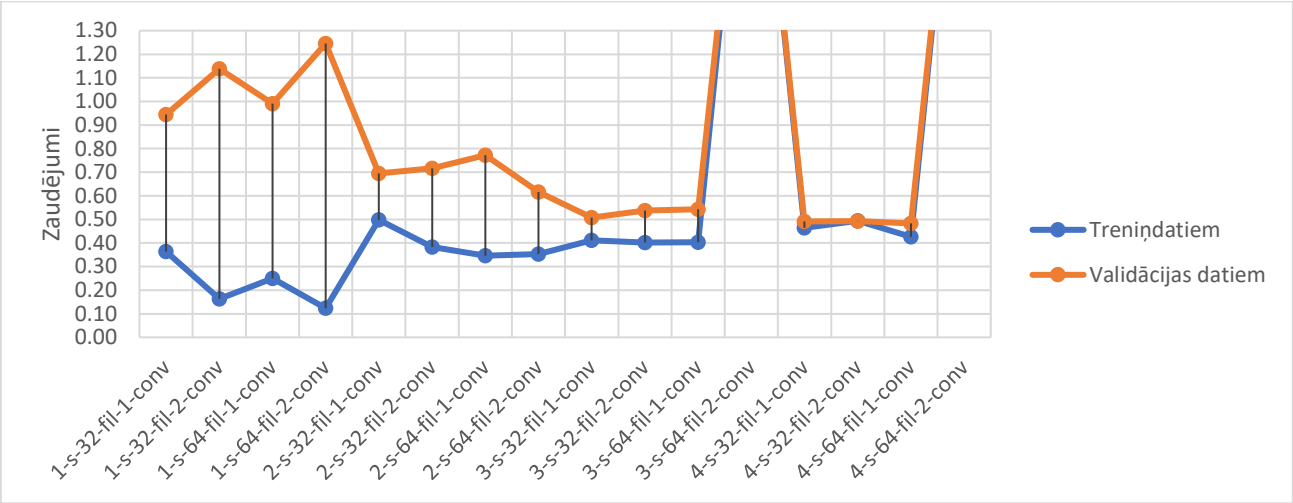
Figure 3.7 **CNN generation plan**

Kopumā tika ģenerēti 16 dažādi modeļi. Katrs modelis tika trenēts un pārbaudīts ar 14 kategorijām pa 1000 *QuickDraw* skicēm, kopā izmantojot apmācībai 14000 skices, kur 80% tika izmantoti treniņu nolūkiem un 20% validācijai. Katrs modelis tika apmācīts ar 5 epohiem, jo izmantojot 10 epohus, modeļi tika pārsvarā pārmācīti ar treniņdatiem. Iegūtie rezultāti ir redzami 3.8. un 3.9. attēlā. Uzreiz pamanāms interesants novērojums - modeļos *3-s-64-fil-2-conv* un *4-s-64-fil-2-conv*, kur sākotnējais filtru skaits ir 64 vienības un satur katrā sērijā divus konvolūcijas slāņus. *3-s-64-fil-2-conv* modelim precizitāte piektajā epohā bija 0,0729 treniņdatiem un 0,0639 validācijas datiem, un zaudējumi 2,6392 treniņdatiem un 2,6396 validācijas datiem. *4-s-64-fil-2-conv* modelim precizitāte bija 0,0697 un 0,0639, un zaudējumi 2,6392 un 2,6395 respektīvi. Interesanti tas, ka *2-s-64-fil-2-conv* modelim, kas ir arī sākotnējā daļa *3-s-64-fil-2-conv* un *4-s-64-fil-2-conv* modelim, šādi rādītāju kritumi nav ievērojami. Liekot apsvert, ka problēmsituācija varēja rasties brīdi, kad tika izmantoti divi sekojoši konvolūcijas slāņi ar 256 filtru skaitu ($64 * 2^{(3-1)}$).



3.8.att. **QuickDraw treniņdatu un validācijas datu precizitāte, izmantojot modeļus ar dinamiskiem hiperparametriem**

Figure 3.8 **QuickDraw training and validation accuracy using models with dynamic hyperparameters**



3.9.att.. **QuickDraw treniņdatu un validācijas datu zaudējumi, izmantojot modeļus ar dinamiskiem hiperparametriem**

Figure 3.9 **QuickDraw training and validation loss using models with dynamic hyperparameters**

Bet tā kā citos gadījumos divi šādi konvolūcijas slāņu rezultāti būtiski neatšķīrās no viena konvolūcijas slāņa rezultātiem, tad tālāka divu konvolūcijas slāņu modeļu izmantošana tiek izslēgta. Tālāk ir redzama parādība, ka izmantojot četras sērijas ar *conv-maxpool-dropout* salikumu, tiek iegūts minimālais zaudējums validācijas datiem 0,4916. Vēl mazliet mazāks zaudējums ir *4-s-64-fil-1-conv* ar 0,4830 rādītāju, taču izskatās, ka tajā pašā laikā modelī treniņdatu zaudējums sāk vairāk samazināties, radot iespaidu, ka modelim var sākties pārmācība ar

treniņdatiem. Seklākos modeļus ar 1-3 sērijām nav pamats izmantot, jo šie modeļi nav spējīgi vispārināt skiču atpazīšanu. To var redzēt arī 3.8. attēlā, kā validācijas datu precizitāte sāk pieaugt dziļākos modeļos, kamēr treniņu datu precizitāte sāk pietuvināties validācijas datiem ar katru nākamo sēriju.

Vēl katram nolūkam tika apmācīts *5-s-32-fil-1-conv* modelis, kas sastāvēja no piecām sērijām. Šis modelis līdzīgi iepriekšējiem tika apmācīts ar 5 epohiem. Apmācību rezultāti ir redzami 3.1. tabulā. Kā redzams dziļāks neironu tīkls atdarina līdzīgus rezultātus, kā *3-s-64-fil-2-conv* un *4-s-64-fil-2-conv* modelis, liekot domāt, ka filtru skaitam ir noteicošā nozīme, jo piecu sēriju modelim maksimālais filtru skaits ir 512 ($32 * 2^{(5-1)}$).

3.1.tabula

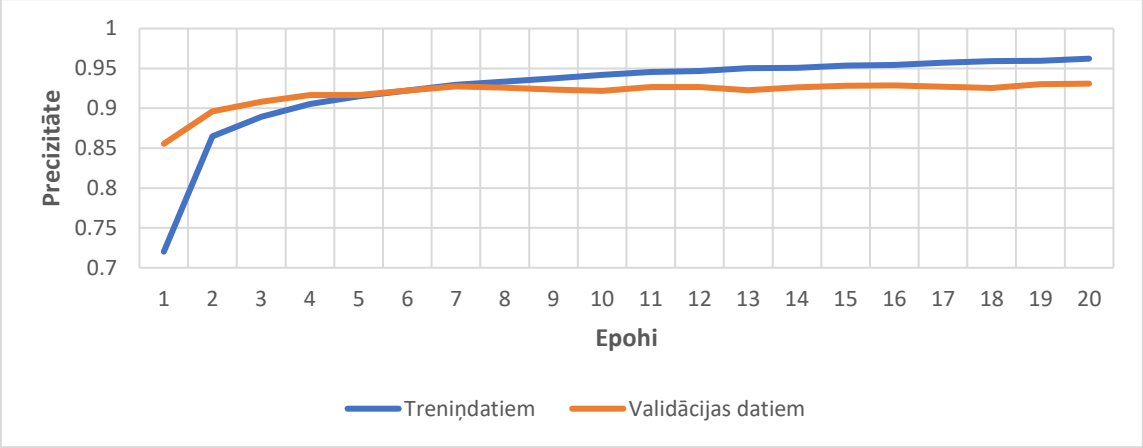
Piecu sēriju CNN modeļa rādītāji

Table 3.1

Results of five series CNN model

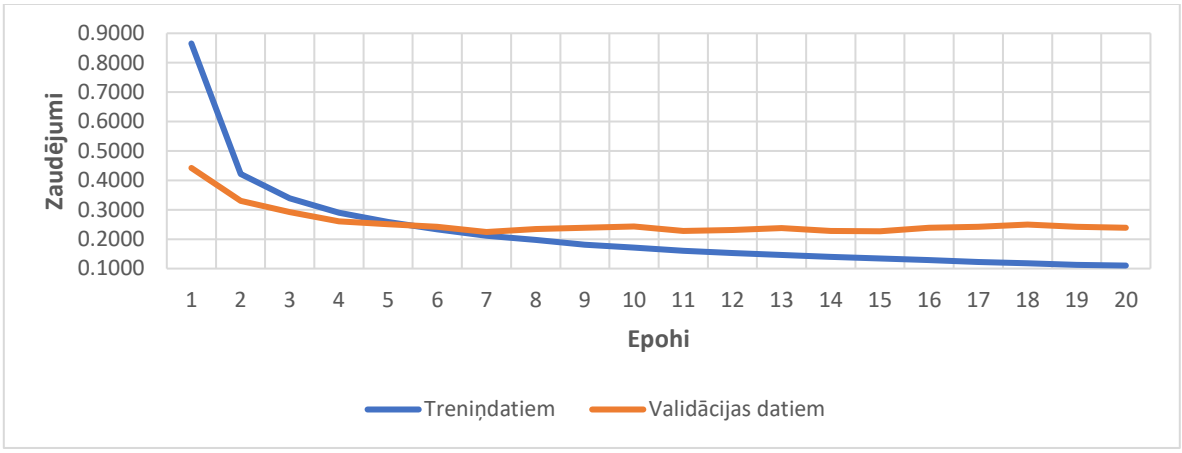
CNN modelis	Precizitāte		Zaudējumi	
	Treniņdatiem	Validācijas datiem	Treniņdatiem	Validācijas datiem
5-s-32-fil-1-conv	0.0733	0.0639	2.6392	2.6396

Ņemot vērā visus ģenerēto modeļu rādītājus skiču atpazīšanai tika izmantots *4-s-32-fil-1-conv* modelis, atstājot *3-s-32-fil-1-conv* modeli kā rezerves plānu. Apmācībām *QuickDraw* datu apjoms attiecīgi tika palielināts no 1000 uz 5000 skicēm katrai kategorijai, līdz ar ko tika palielināts epohu skaits no 5 līdz 20. Pārējie parametri palika nemainīgi. Kā ir redzams 3.10. un 3.11. attēlā validācijas dati pārbaudē ieguva līdz 96,22% lielu precizitāti, kamēr zaudējums bija 23,90%. Var arī ievērot, ka 6. epohā treniņu līkne gan zaudējumos, gan precizitātē krusto validācijas datu līkni, secinot, ka ar sešiem epohiem ir pilnībā pietiekami, lai pilnvērtīgi apmācītu modeli, iegūstot 92,22% un 24,20% precizitāti un zaudējumu validācijas datiem.



3.10.att. **QuickDraw treniņdatu un validācijas datu precizitāte, izmantojot 4-s-32-fil-1-conv modeli**

Figure 3.10 **QuickDraw training and validation accuracy using 4-s-32-fil-1-conv model**

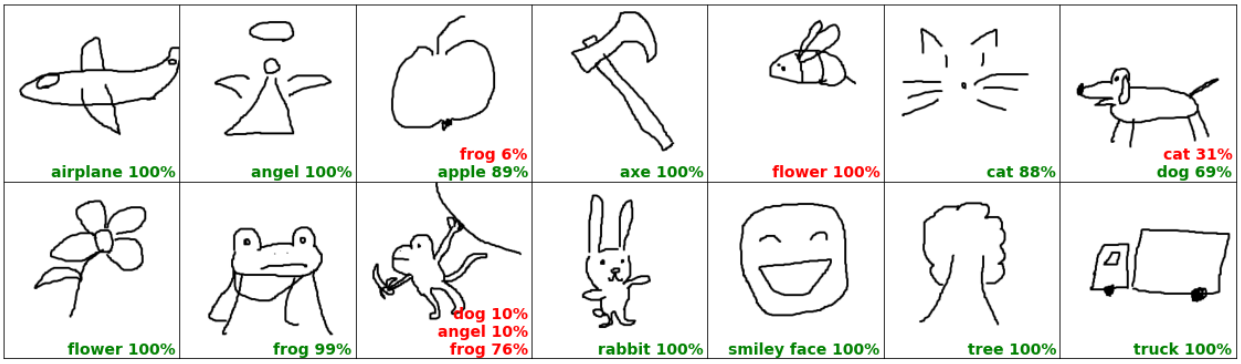


3.11.att. **QuickDraw treniņdatu un validācijas datu zaudējums, izmantojot 4-s-32-fil-1-conv modeli**

Figure 3.11 **QuickDraw training and validation loss using 4-s-32-fil-1-conv model**

Skiču pārbaudei uz doto *3-s-32-fil-1-conv* modeli tika uzzīmētas 14 paszīmētas skices. Skiču atpazīšanas prognozes rezultāti ir vērojami 3.12. attēlā. Kā var redzēt no 14 kategorijām 12 attēli tika atpazīti pareizi ar gandrīz 100% precizitāti. Tomēr bites un pērtiķa skices tika atminētas nepareizi, kur pērtiķis tika piemērots vardes kategorijai, iespējams, kopīgojot vardes sejas vaibstus, un kur bite tika piemērota ar 100% precizitāti puķes kategorijai, iespējams, kopīgojot puķes ziedlapiņas īpašības ar bites spārniem. Šī parādība ir arī novērojama arī suņa kategorijā, kur suns tika atminēts pareizi ar 69% precizitāti, tomēr 31% dala ar kaķa kategoriju, apsverot, ka līdzīgs dzīvnieka ķermenis var būt arī kaķim no trenētām skicēm. Tomēr, veicot atsevišķas korekcijas bites un pērtiķa paszīmētajai skicei, ir iespējams iegūt precīzāku minējumu. Arī ir svarīgi pieminēt, ka

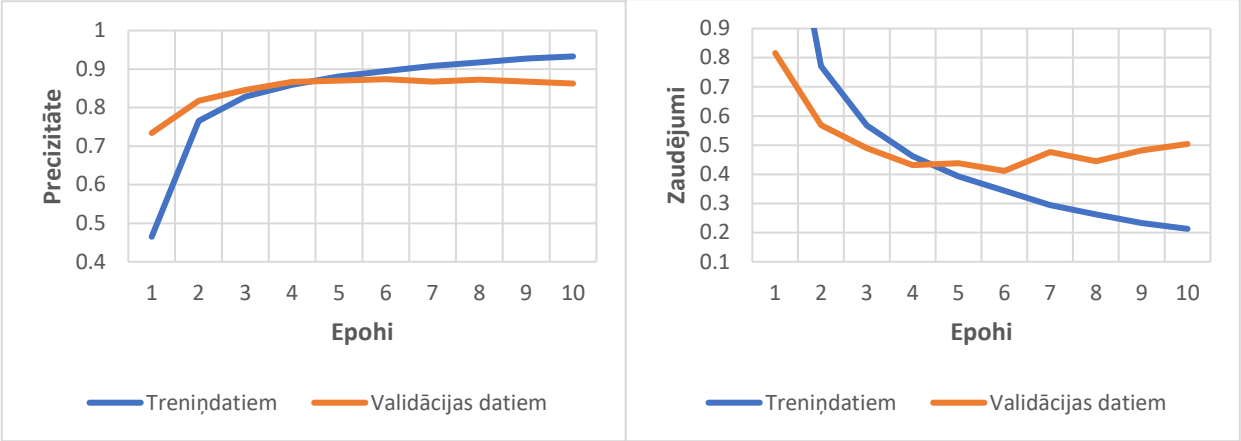
izvēloties lielāku kategoriju skaitu vai līdzīgākas skiču kategorijas, modelis varētu ne tik viennozīmīgi atpazīt prognozējamo skici.



3.12.att. **Paszīmētu skiču atpazīšana, izmantojot 4-s-32-fil-1-conv modeli 14 QuickDraw kategorijās: lidmašīna, eņģelis, ābols, cirvis, bite, kaķis, suns, puķe, varde, pērtiķis, zaķis, priecīga sejiņa, koks, kravas auto**

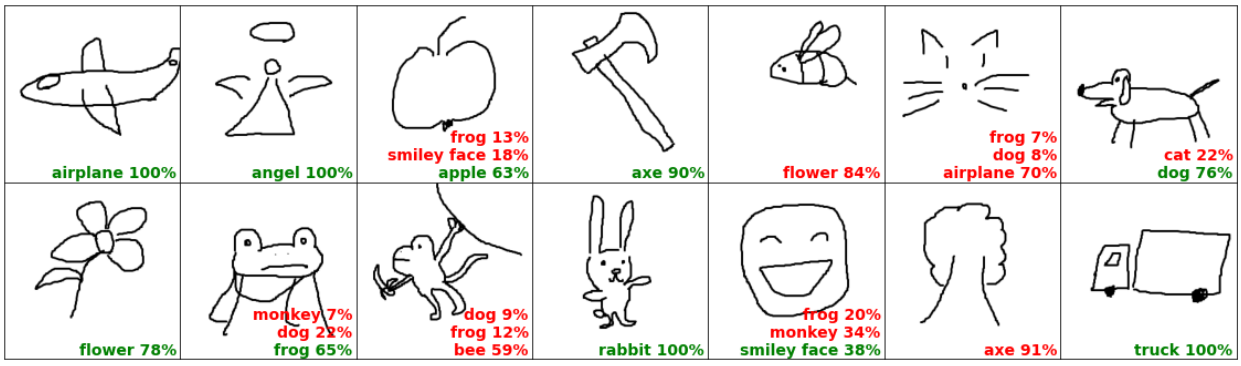
Figure 3.12 **Self-drawn sketch recognition using 4-s-32-fil-1-conv model in 14 QuickDraw categories: airplane, angel, apple, axe, bee, cat, dog, flower, frog, monkey, rabbit, smiley face, tree, truck**

Ņemot vērā *4-s-32-fil-1-conv* modeļa efektīvos rezultātus un filtru skaitu kā ietekmējošo faktoru, modificētais *Sketch-a-Net* modelis (skat. 2. pielikumā) ir ticis vēlreiz pārveidots, tagad samazinot katrā slāni filtru skaitu uz pusi, tādejādi uzstādot maksimālo filtru skaitu no 512 uz 256 vienībām. Interesanti, ka tieši šāda darbība tiešām uzlaboja modeļa rezultātus, kas ir vērojams 3.13. attēlā uz 14 kategorijām pa 5000 skicēm. Līdzīgi, kā tas bija ar *4-s-32-fil-1-conv* modeli, 5 epohi sniedz visoptimālākos rezultātus ar 0,8804 un 0,8697 prezicitāti un 0,3932 un 0,4383 zaudējumiem treniņdatos un validācijas datos respektīvi, tāpēc modelis tika no jauna mācīts ar 5 epohiem. Tomēr par spīti veiksmīgai apmācībai, izmēģinot atpazīt paszīmētās skices, rezultāti bija vājāki nekā *4-s-32-fil-1-conv* modelim, kur no 14 kategorijām tika atpazītas 10 kategorijas. Lai arī atkārtojās līdzīgas pazīmes kā *Sketch-a-Net* modelī, piemēram, bite tiek nodēvēta par puķi, tomēr radās vairākas jaunas neprecizitātes, koka, sejas, vardes, kaķa un ābola kategorijās.



3.13.att. **QuickDraw treniņdatu un validācijas datu precizitāte un zaudējumi, izmantojot modificētu Sketch-a-Net modeli un uz pusi samazinot filtru skaitu**

Figure 3.13 **QuickDraw training and validation accuracy and loss using modified Sketch-a-Net model with half of size filters**



3.14.att. **Paszīmētu skiču atpazīšana, izmantojot modificētu Sketch-a-Net modeli 14 kategorijās**

Figure 3.14 **Self-drawn sketch recognition using Sketch-a-Net model in 14 categories**

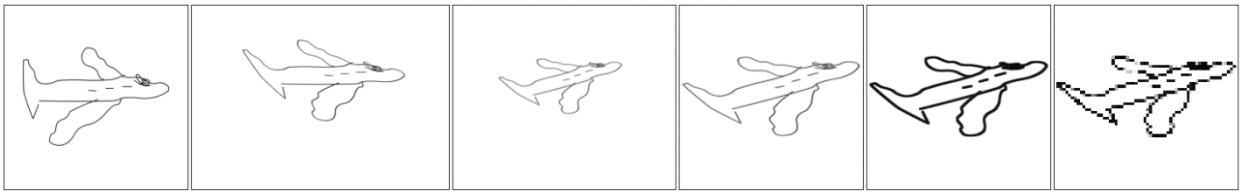
Salīdzinot *4-s-32-fil-1-conv* modeli ar Gupta et al pētījumā izmanto *CNN* modeli, tad *4-s-32-fil-1-conv* guva augstāku validācijas precizitāti ar līdz 96,22% nekā pētījumā minētais ar 78% precizitāti [9]. Tomēr šādu apgalvojumu ir aplami izteikt, kad pētījumā minētais *CNN* modelis tika trenēts uz vienkāršotas *QuickDraw* datu kopas, ietverot visas 345 kategorijas 14 kategoriju vietā.

3.5. *Sketch-a-Net* un pašģenerēts neironu tīkls uz *TU Berlin* datu kopas

Tā kā *TU Berlin* datu kopas skices ir bilžu formātā nevis koordināšu formātā, tad bilžu sagatavošanas process *TU Berlin* datu kopai bijis citādāks. Katrai skicei tika izgriezts baltais laukums un pati skice tika samazināta 1:1 attēla attiecībā. Šī procedūra tika veikta ar nodomu, lai

neironu tīkls spētu darboties ar pareizas formas datiem. Tomēr tāpēc, ka skice oriģināli bija lielākā izmērā, tad samazinot bildi var zaudēt informāciju. Šim nolūkam tika pabiezinātas līnijas informācijas saglabāšanai.

Galvenais trūkums šai datu kopai ir nepietiekams skiču daudzums kategorijā, proti, 80 skices, kas ir ļoti mazs treniņdatu apjoms. Līdzīgā veidā kā to darīja Yu et al., arī šeit dati tika palielināti, globāli deformējot bildi ar izliekumu maiņu un rotācijām. Šādi tika palielināts datu apjoms no 80 uz 240 skicēm kategorijā, kas arī nav liels apjoms, bet ir mazs potenciāls uzlabojums apmācībai. Datu palielināšanas soļus var redzēt 3.15. attēlā.



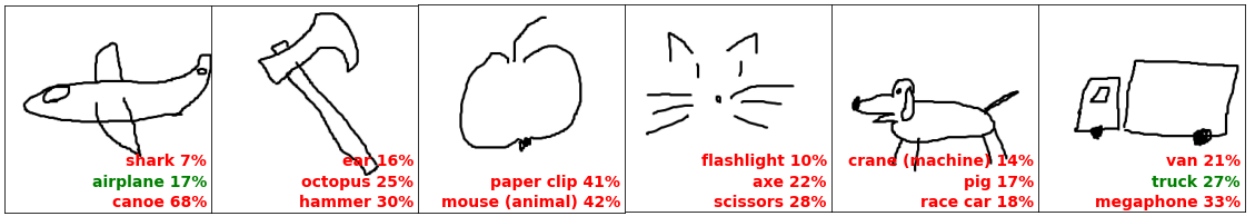
3.15.att. **TU Berlin datu palielināšanas process, sākot no kreisās puses: skices ievade; skices izliekuma maiņa; skices rotācija; balto laukumu izgriešana, saglabājot 1:1 attēla attiecību; malu pabiezināšana; attēla samazināšana uz 64x64 pikseļiem;**

Figure 3.15 **TU Berlin data augmentation starting from left: input; affine transformation; sketch rotation; crop white space out and adjust sketch to 1:1 image ratio; thicken lines; resize to 64x64px;**

Rezultātā tika apmācīts modificēts *Sketch-a-Net* un *4-s-32-fil-1-conv* modelis ar 20 epohiem ar 90% un 10% procentu treniņdatu un validācijas datu attiecību. Kategoriju skaits ar *TU Berlin* datu kopu tika izvērsts līdz 250 kategorijām, sakarā ar mazo skiču apjomu. Kamēr *4-s-32-fil-1-conv* modelis 20. epohā uzradīja daudz maz sološus rezultātus ar 0,8706 un 0,5063 prezicitāti treniņdatos un validācijas datos respektīvi, zaudējumi treniņdatos uzrādījās ar 0,4007 koeficientu, bet validācijas datos ar 3,0422 koeficientu. Arī šis zaudējums validācijas datiem līdz 20 epoham bija tikai ar 2,4779 minimālo robežo, kas apliecina *TU Berlin* datu kopas mazā apjoma trūkumu, lai atpazītu neredzētas skices. Turklāt modificētais *Sketch-a-Net* modelis nebija pat spējīgs mācīties no *TU Berlin* datu kopas, uzrādot 0,0236 un 0,0225 prezicitāti un 5,5236 un 5,5239 zaudējumu treniņdatos un validācijas datos respektīvi. Gala rezultāts, atpazīstot paszīmētās skices ar *4-s-32-fil-1-conv* modeli, ir redzams 3.16. attēlā.

Salīdzinot ar Mathias Eitza modeļa rādītājiem, neizmantojot datu palielināšanas metodes *TU Berlin* datu kopai, precizitāte Eitzam bija 56%, kur *4-s-32-fil-1-conv* ir 50,63%. Lai arī šie rezultāti ļoti neatšķiras viens no otra, tomēr tiek ierosināts turpmāk izmantot *QuickDraw* nevis *TU Berlin* datu kopu, lai apmācītu skiču atpazīšanas neironu tīklus. Šāda opcija arī izslēdz īpašu vajadzību

izmantot datu palielināšanas metodes, jo katrā kategorijā skiču skaits ir nevis 240, izmantojot datu palielināšanas metodes, bet 5000, kur vēl ir opcija palielināt *QuickDraw* kategorijā esošo skiču skaitu līdz pat 120 000 vienībām, bet, ja tiek izvēlētas visas skices, ne tikai tās, kuras *QuickDraw* spēles modelis atpazina, tad līdz 150 000 vienībām.



3.16.att. **Pašzīmētu skiču atpazīšana, izmantojot 4-s-32-fil-1-conv modeli 6 kategorijās: lidmašīna, cirvis, ābols, kaķis, suns, kravas auto**

Figure 3.16 **Self-drawn sketch recognition using 4-s-32-fil-1-conv model in 14 categories: airplane, axe, apple, cat, dog, truck**

3.6. Pikeļgrafikas attēla atpazīšana, izmantojot *Sketch-a-Net* un pašģenerētu modeli

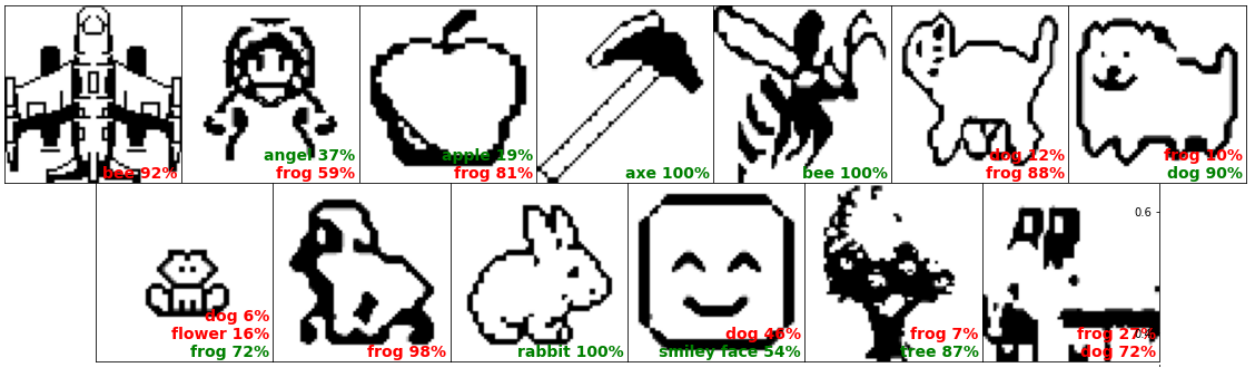
Lai atpazītu pikseļgrafikas attēlus ar skicēm trenētu neironu tīkla modeli, pikseļgrafikas attēls vispirms tiek pārvērsts skicei līdzīgā formā. Šim nolūkam tika izmantota depikselizācijas algoritms, lai pikseļotos attēla fragmentus padarītu gludākas. Katrai *QuickDraw* treniņkopas kategorijai tika piemeklēti pikseļgrafikas attēli, kas tikuši depikselizēti ar 3.2. apakšnodaļā pieminētā algoritma implementāciju (skat. 3.17.att.).

3.17.att. **Pikseļattēlu depikselizācija: augšējais slānis oriģinālattēls, apakšējais depikselizētais slānis**

Figure 3.17 **Pixelart depixelization: upper layer original image, bottom layer depixelated image**

Pēc tam katrs depikselizēts attēls tiek pārvērsts melnbalts ar kontrastējošās krāsas sliekšņa algoritmu, proti, ja konkrētās krāsas pikseļa spilgtums sasniedza, konkrētu sliekšņa vērtību, tad attiecīgais pikselis pārtapa vai nu melns, vai nu balts. Šis parametrs tika mainīts katram attēlam

manuāli, pieturoties pie galvenajām vērtībām 50, 100 un 150, atkarībā kopējā attēla elementa spilgtuma. Pēc tam visi šie attēli tika doti *4-s-32-fil-1-conv* modelim atpazīšanai, kur rezultāti ir novērojami 3.18. attēlā.

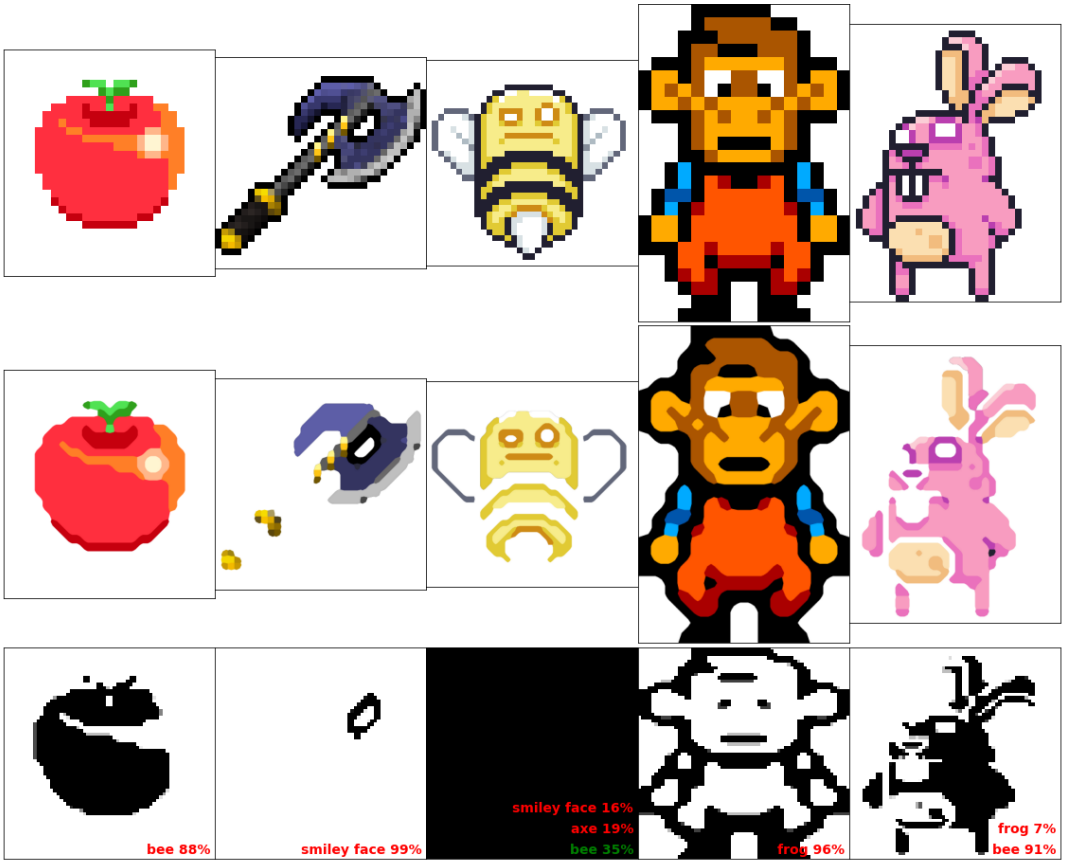


3.18.att. **Pikseļattēlu atpazīšana, izmantojot 4-s-32-fil-1-conv modeli 13 kategorijām: lidmašīna, eņģelis, ābols, cirvis, bite, kaķis, suns, varde, pērtiķis, zaķis, priecīga sejiņa, koks, kravas auto**

Figure 3.18 **Pixelart classification with 4-s-32-fil-1-conv model in 13 categories: airplane, angel, apple, axe, bee, cat, dog, frog, monkey, rabbit, smiley face, tree, truck**

Atkarībā no depikselizācijas un sliekšņā rezultāta, tika atminēti 7 no 13 pikseļattēliem, kur lielāko pārsvaru guva pikseļattēls "cirvis", "bite" un "zaķis". Sliktākie rādītāji bija pikseļattēliem "lidmašīna", "kaķis", "pērtiķis" un "kravas mašīna". Viena no kļūdu pazīmēm varētu būt tāda, ka skices parasti tiek zīmētas vienkāršā formā, it sevišķi ņemot vērā faktoru, ka *QuickDraw* skices spēlē tika zīmētas līdz 20 sekundēm, tādejādi pikseļattēli ar izteiktākām detaļām kā pikseļattēlam "lidmašīna" ir retāka parādība. Otrs iemels ir silueta un kontūra trūkums pikseļattēlā. Ņemot vērā, ka krāsas sliekšņa algoritmā tika izņemtas, tad, ja attiecīgajam attēlam, piemēram, "kravas mašīna" kontūras saplūst ar paša elementa krāsu, tad pastāv iespēja, ka kontūras tiek pilnībā izņemtas, un modelis vairs nespēj atrast skicei līdzīgās pazīmes. Šis fakts ir vairāk vērojams 3.19. attēla ābola gadījumā. Kamēr ābols ir ticis depikselizēts pilnībā bez jebkādiem trūkumiem, krāsu sliekšņa algoritms līdz ar visu krāsu, atbrīvojās arī no kontūras, rezultātā modelis vairs nespēja atpazīt pikseļattēlu. Cirvja, bites un zaķa gadījumā implementētais depikselizācijas algoritms izdeva kropļojumus, kā rezultātā tika zaudētas tumšās līnijas. Šis fakts varētu būt izskaidrojams ar to, ka depikselizācijas algoritmā krāsu grafa reģioni, ir tikuši nepareizi savienoti, kas var būt implementācijas vaina.

Darbā tika izmēģinātas dažādas bilžu pirmsapstrādes metodes, kā malu uztveršanas algoritms, taču problēma radās pārmērīgi daudz līniju uztveršanā, kas kopējo attēlu padarīja neskaidru. Tika mēģināts atpazīt pikseļattēlu oriģinālā formātā, taču dēļ pārmērīgi maziem izmēriem, palielinot attēlu 64x64 pikseļiem, bilde izplūda un zaudēju konkrētās bildes raksturojumu. Viens no ierosinājumiem ir izcelt bildē kontūras, sevišķi, ja malas krāsas saplūst ar pašu elementu. Otrs ierosinājums ir optimizēt depikselizācijas algoritma implementāciju, iekļaujot B-splainu līknes un krāsu pāreju. B-splainu līknes padarītu depikselizētu attēlu mazāk kantainu, līdz ar to rezultāts būs tuvāk skices gludajam zīmēšanas stilam. Krāsu pāreja arī atvieglotu krāsu sliekšņa algoritma darbu. Kā arī būtu ieteicams samazināt detaļas attēlā, bet saglabāt raksturīgākās pikseļattēla īpašības, lai izveidotu tuvāk skicei līdzīgu rezultātu [16].

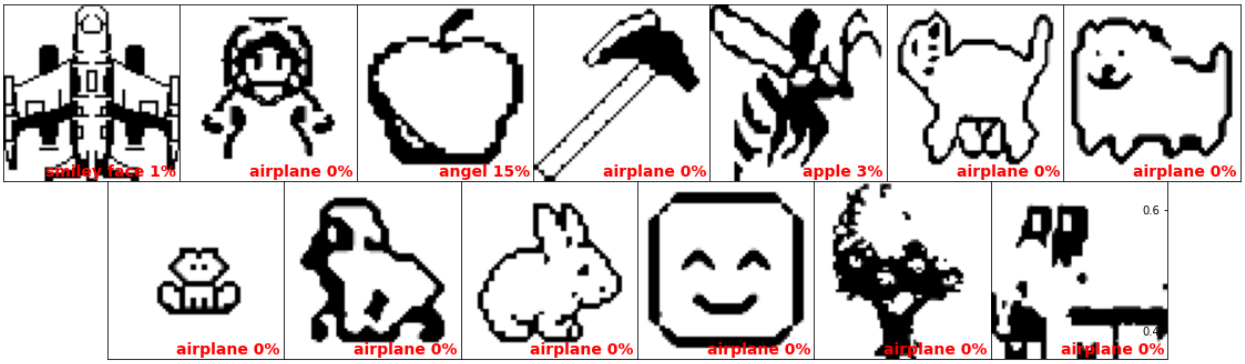


3.19.att. **Sliktākie piemēri (ābols, cirvis, bite, pērtiķis, zaķis) pikseļattēlu klasifikācijā, izmantojot 4-s-32-fil-1-conv modeli**

Figure 3.19 **Worst examples (apple, axe, bee, monkey, rabbit) for pixelart classification using 4-s-32-fil-1-conv model**

Savukārt *Sketch-a-Net* modeļa rezultāti skiču atpazīšanā bija nesekmīgi visos gadījumos (skat 3.20. att.). Ne tikai dotās prognozes bija nepareizas, bet pats minējums kā tāds netika izdarīts.

Gala rezultātā pierādījās, ka klasisks konvolūcijas neirona tīkls ir spējīgs atpazīt pikseļattēlus, pirms tam esot apmācītam ar lielu daudzumu skiču datu. Šis fakts dod arī jaunas iespējas mēģināt atpazīt zīmējumus, kas ir abstraktāki par pikseļattēliem, ja ir iespējams datus daudz maz pārveidot skicei līdzīgā formā.



3.20.att. **Pikseļattēlu atpazīšana, izmantojot Sketch-a-Net modeli 13 kategorijām**

Figure 3.20 **Pixelart classification with Sketch-a-Net model in 13 categories**

SECINĀJUMI

1. Dziļāki konvolucionālie neironu tīkli ir spējīgāki atpazīt skices, tomēr liels dziļums var apgrūtināt neironu tīklu apmācību.
2. Turpmākos neironu tīklu pētījumos, kur tiek izmantota skiču datu kopa, ierosinājums ir izmantot *QuickDraw* datu kopu *TU Berlin* datu kopas vietā dēļ lielā apjoma.
3. *QuickDraw* skiču datu kopa var tikt pielietota arī abstraktu zīmējumu un animācijas tēlu atpazīšanā, kur līnija ir pamatkomponente zīmējumā.
4. Lai no pikseļattēla apstrādes laikā iegūtu pēc iespējas skicei līdzvērtīgāku attēlu, būtu nepieciešams veikt depikselizācijas algoritmā arī krāsu pāreju un B-splaina līkņu implementāciju. Kā arī ir svarīgi vairāk akcentēt kontūras pikseļattēlos, kur kontūra saplūst ar elementa krāsu, un samazināt pikseļattēlam mazsvarīgās detaļas.
5. Pikseļattēla kantainums īpaši netraucēja, lai atpazītu pikseļattēlu. Šādā veidā ierosinot atrast vienkāršākas attēla pirmapstrādes alternatīvas, kas koncentrējas uz pikseļgrafikas tēla kontūru izvešanu.
6. Darba hipotēze apstiprinājās, jo darbā piedāvātais *4-s-32-fil-1-conv* neironu tīkls bija spējīgs atpazīt pikseļgrafikas attēlu, pirms tam apstrādājot pikseļattēlu ar depikselizācijas un kontrastējošās krāsu sliekšņa algoritmu. Tomēr netiek izslēgta pikseļgrafikas ģenerēšanas rīku efektivitāte uz neironu tīkliem.

LITERATŪRAS APSKATS

1. Azzi M. (2019). Pixel Logic - A Guide to Pixel Art: 242
2. Dawe J., Humphries M. (2019). Make Your Own Pixel Art: Create Graphics for Games, Animations, and More! No Starch Press: 200
3. Dedeoğlu1 Y., Töreyin B. U., Güdükbay U., Çetin A. E. (2006). Silhouette-Based Method for Object Classification and Human Action Recognition in Video. ECCV 2006: Computer Vision in Human-Computer Interaction: 64-77
4. Eitz M., Hays J., Alexa M. (2012). How do humans sketch objects? ACM Transactions on Graphics: 31, 4, 10
5. Eitz M., Hildebrand K., Boubekeur T., Alexa M. (2011). Sketch-Based Image Retrieval: Benchmark and Bag-of-Features Descriptors. IEEE Transactions on Visualization and Computer Graphics 17(11):1624 – 1636
6. Eitz M. (2012). Human Object Sketches: Datasets, Descriptors, Computational Recognition and 3d Shape Retrieval. Technische Universität Berlin, Fakultät IV - Elektrotechnik und Informatik: 110
7. Fernandez-Fernandez R., Victores J. G., Estevez D., Balaguer C. (2019). Quick, Stat!: A Statistical Analysis of the Quick, Draw! Dataset. Eurosim 2019: 12
8. Gerstner T., DeCarlo D., Alexa M., Finkelstein A., Gingold Y., Nealen A.(2012). Pixelated Image Abstraction. NPAR 2012, Proceedings of the 10th International Symposium on Non-photorealistic Animation and Rendering: 8
9. Gupta M., Mehndiratta P. (2019). Analysis and Recognition of Hand-Drawn Images with Effective Data Handling. Big Data Analytics 2019: 389-407
10. Han C., Wen Q., He S., Zhu Q., Tan Y., Han G., Wong T (2018). Deep unsupervised pixelization. ACM Transactions on Graphics (TOG): 37, 6, 1—11
11. Ha D., Eck D. (2017). A Neural Representation of Sketch Drawings. ICLR 2018: 6
12. Jiang Y. (2017). Sketch image recognition using deep features. University of Adelaide, School of Computer Science: 55
13. Kopf, J., Lischinski D. (2011). Depixelizing Pixel Art. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2011): 30, 4, 99:1--99:8.
14. Lu, W., Tran, E. (2017). Free-hand Sketch Recognition Classification. Stanford University: 7

15. Sangkloy P., Burnell N, Ham C., Hays J. (2016). The Sketchy Database: Learning to Retrieve Badly Drawn Bunnies. ACM Transactions on Graphics (proceedings of SIGGRAPH): 12
16. Schneider R. G., Tuytelaars T. (2014). Sketch classification and classification-driven analysis using Fisher vectors. ACM Transactions on Graphics: 174, 9
17. Seddati O., Dupont S., Mahmoudi S. (2016). DeepSketch 2: Deep Convolutional Neural Networks for Partial Sketch Recognition. Conference: Proc. of 14th International Workshop on Content-based Multimedia Indexing (CBMI 2016): 6
18. Serpa Y. R., Rodrigues M. A. F. (2019). Towards Machine-Learning Assisted Asset Generation for Games: A Study on Pixel Art Sprite Sheets. 2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames): 533-542
19. Silber D. (2016). Pixel Art for Game Developers (1st Edition): 256
20. Sun Z., Wang C., Zhang Li.., Zhang Le. (2012). Sketch2Tag: Automatic Hand-Drawn Sketch Recognition. ACM Conference on Multimedia: 2
21. Yadav P., Nagmode M. S. (2017). Silhouette Object Recognition Using Edge-Based Method. Innovations in Electronics and Communication Engineering: 251-259
22. Yu Q, Yang Y., Liu F., Song Y., Xiang T., Hospedales T. M. (2017). Sketch-a-Net: A Deep Neural Network that Beats Humans. International Journal of Computer Vision volume 122, 411–425
23. Zhou Y., Jin Y., Luo A., Chan S., Xiao X., Yang X. (2018). ToonNet: A cartoon image dataset and a DNN-based semantic classification system. VRCAI '18: Proceedings of the 16th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry: 30, 1-8

PIELIKUMI

1. pielikums

Sketch-a-Net arhitektūra

Appendix 1

The architecture of Sketch-a-Net

Indekss	Slānis	Tips	Filtra izmērs	Filtru skaits	Solis	Rāmis	Izejas izmērs
0		Input	-	-	-	-	225 x 225
1	L1	Conv	15 x 15	64	3	0	71 x 71
2		ReLU	-	-	-	-	71 x 71
3		Maxpool	3 x 3	-	2	0	35 x 35
4	L2	Conv	5 x 5	128	1	0	31 x 31
5		ReLU	-	-	-	-	31 x 31
6		Maxpool	3 x 3	-	2	0	15 x 15
7	L3	Conv	3 x 3	256	1	1	15 x 15
8		ReLU	-	-	-	-	15 x 15
9	L4	Conv	3 x 3	256	1	1	15 x 15
10		ReLU	-	-	-	-	15 x 15
11	L5	Conv	3 x 3	256	1	1	15 x 15
12		ReLU	-	-	-	-	15 x 15
13		Maxpool	3 x 3		2	0	7 x 7
14	L6	Conv(=FC)	7 x 7	512	1	0	1 x 1
15		ReLU	-	-	-	-	1 x 1
16		Dropout (0,50)	-	-	-	-	1 x 1
17	L7	Conv(=FC)	1 x 1	512	1	0	1 x 1
18		ReLU	-	-	-	-	1 x 1
19		Dropout (0,50)	-	-	-	-	1 x 1
20	L8	Conv(=FC)	1 x 1	14	1	0	1 x 1

2. pielikums

Modificēta Sketch-a-Net arhitektūra

Appendix 2

Modified architecture of Sketch-a-Net

Indekss	Slānis	Tips	Filtra izmērs	Filtru skaits	Solis	Rāmis	Izejas izmērs
0		Input	-	-	-	-	64 x 64
1	L1	Conv	5 x 5	64	3	0	20 x 20
2		ReLU	-	-	-	-	20 x 20
3		Maxpool	2 x 2	-	2	0	10 x 10
4	L2	Conv	3 x 3	128	1	0	8 x 8
5		ReLU	-	-	-	-	8 x 8
6		Maxpool	2 x 2	-	2	0	4 x 4
7	L3	Conv	3 x 3	256	1	1	4 x 4
8		ReLU	-	-	-	-	4 x 4
9	L4	Conv	3 x 3	256	1	1	4 x 4
10		ReLU	-	-	-	-	4 x 4
11	L5	Conv	3 x 3	256	1	1	4 x 4
12		ReLU	-	-	-	-	4 x 4
13		Maxpool	2 x 2		2	0	2 x 2
14	L6	Conv(=FC)	2 x 2	512	1	0	1 x 1
15		ReLU	-	-	-	-	1 x 1
16		Dropout (0,50)	-	-	-	-	1 x 1
17	L7	Conv(=FC)	1 x 1	512	1	0	1 x 1
18		ReLU	-	-	-	-	1 x 1
19		Dropout (0,50)	-	-	-	-	1 x 1
20	L8	Conv(=FC)	1 x 1	14	1	0	1 x 1

3. pielikums

Modificēta Sketch-a-Net modeļa koda paraugs

Appendix 3

Modified Sketch-a-Net model code snippet

```
### Layer 0:
model.add(InputLayer(input_shape=X.shape[1:]))
#--------------------------------------------------------------------------
### Layer 1:
model.add(Conv2D(kernel_size=(5, 5), filters=64, strides=(3, 3),
padding="valid"))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2), padding="valid"))
# #-------------------------------------------------------------------------
# ### Layer 2:
model.add(Conv2D(kernel_size=(3, 3), filters=128, strides=(1, 1),
padding="valid"))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2), padding="valid"))
# #-------------------------------------------------------------------------
### Layer 3:
model.add(Conv2D(kernel_size=(3, 3), filters=256, strides=(1, 1),
padding="same"))
model.add(Activation('relu'))

### Layer 4:
model.add(Conv2D(kernel_size=(3, 3), filters=256, strides=(1, 1),
padding="same"))
model.add(Activation('relu'))

### Layer 5:
model.add(Conv2D(kernel_size=(3, 3), filters=256, strides=(1, 1),
padding="same"))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2), padding="valid"))
# #-------------------------------------------------------------------------
# Layer 6:
model.add(Conv2D(kernel_size=(2, 2), filters=512, strides=(1, 1),
padding="valid"))
model.add(Activation('relu'))
model.add(Dropout(rate=0.5))
# #-------------------------------------------------------------------------
# Layer 7
model.add(Conv2D(kernel_size=(1, 1), filters=512, strides=(1, 1),
padding="valid"))
model.add(Activation('relu'))
model.add(Dropout(rate=0.5))
# #-------------------------------------------------------------------------
# Layer 8
model.add(Conv2D(kernel_size=(1, 1), filters=len(CATEGORIES), strides=(1, 1),
padding="same"))
model.add(Flatten())
model.add(Dense(units=len(CATEGORIES)))
model.add(Activation('softmax'))
```

Bakalaura darbs „Zīmētu attēlu klasificēšana ar neironu tīkliem” izstrādāts LU Datorikas fakultātē.

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

Autors: Edgars Beļevičs 31.05.2021.

Rekomendēju darbu aizstāvēšanai

Vadītājs: docents Dr.sc.comp. Kārlis Freivalds 31.05.2021.

Recenzents: pasniedzējs Bc.philol. Maksims Ivanovs

Darbs iesniegts Datorikas fakultātē 31.05.2021.

Dekāna pilnvarotā persona: vecākā metodiķe Ārija Sproģe

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē

__.06.2021. prot. Nr. ___.

Komisijas sekretārs: Jevgēnijs Vihrovs