LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

CILVĒKA UZMANĪBAS SIMULĒŠANA TĪMEKĻA MĀJASLAPU DIZAINA PROCESĀ, IZMANTOJOT MAŠĪNMĀCĪŠANOS

BAKALAURA DARBS

Autors: **Arnolds Bogdanovs**
Studenta apliecības Nr.: ab14050
Darba vadītājs: Dr. dat. Renārs Liepiņš

RĪGA 2018

ANOTĀCIJA

Bakalaura darba mērķis ir izveidot sistēmu, kas spētu simulēt cilvēka vizuālo uzmanību tīmekļa lapu kontekstā un iegūto informāciju attēlot lietotājiem ērtā veidā.

Tā kā cilvēku uzmanība kļūst par arvien svarīgāku resursu, ir svarīgi informāciju strukturēt efektīvi un uzmanību piesaistošā veidā. Izmantojot mašīnmācīšanās paņēmienus, darba autors izveidoja tīmekļa lapu attēlu apstrādes aizmugursistēmu, kas paredz kuros tīmekļa lapas apgabalos lietotājs pievērsīs visvairāk uzmanības. Rezultāts tiek attēlots autora veidotajā priekšgalsistēmā - pārlūkprogrammas spraudnī kā pārklājums virs tīmekļa lapas. Lai uzlabotu jau pieejamo mašīnmācīšanās modeli, autors izveidoja jaunu datu kopu, ar kuru to trenēja. Darbs demonstrē visus nepieciešamos soļus mašīnmācīšanās darbināta risinājuma izveidē, izmantojot jau esošu mašīnmācīšanās modeli.

Atslēgvārdi: mašīnmācīšanās, vizuālā uzmanība, tīmekļa dizains, datu kopa

ABSTRACT

HUMAN ATTENTION SIMULATION IN WEBPAGE DESIGN USING MACHINE LEARNING

The aim of this thesis is to create a system that could simulate human visual attention in webpages and display the resulting information in a user-friendly way.

Since human attention is becoming a valuable resource, it is imperative to structure information efficiently and attractively. The author created a webpage image processing back-end system, that predicts the most salient regions of the webpage. The results are displayed by a front-end system in the form of a web browser plugin that creates a visual overlay over the webpage displaying information. In order to improve an existing model, author created a new dataset used for training. This paper demonstrates all the necessary steps to create a machine learning enabled solution using an existing model.

Keywords: machine learning, visual attention, web design, dataset

SATURA RĀDĪTĀJS

APZĪMĒJUMU SARAKSTS

base64 – Attēlu reprezentācijas veids, kas attēlu pārveido par simbolu virkni.

CSS – Cascading Style Sheets (Stila lapas kaskadēšana) – Tīmekļa lapu stila apraksta veids.

DOM – Document Object Model (Dokumenta objekta modelis) – Tīmekļa lapu uzbūves struktūra, kas satur tīmekļa lapas objektus.

GAN – Generative Adversarial Networks (Ģeneratīvi konkurējoši tīkli) – Neuzraudzītas mašīnmācīšanās paveids, kurā divi tīkli savā starpā konkurē uzlabojot viens otra sniegumu.

HTML – HyperText Markup Language (Hiperteksta iezīmēšanas valoda) – Tīmekļa lapās izmantota tehnoloģija, ar kuru define tīmekļa lapas struktūru.

HTTP – HyperText Transfer Protocol (Hiperteksta pārsūtīšanas protokols) – Protokols, ar kuru veic datu apmaiņu globālajā tīmeklī.

MVT – Model View Template (Modelis Skats Veidne) – Tīmekļa lietotņu projektēšanas shēma.

Neironu tīkls – Datu struktūra, kas sastāv no tenzoru masīva un kuru bieži izmanto mašīnmācīšanās nolūkos.

WebGL – Bibliotēka grafikas apstrādei tīmekļa lapās.

IEVADS

Mūsdienās uzmanība ir kļuvusi par vienu no svarīgākajiem resursiem. Mediji un reklāmu aģentūras cīnās par cilvēku uzmanību gan tradicionālajās platformās (Televīzija, avīzes un radio), gan arī Internetā. Daudzi futūristi un tehnoloģiju entuziasti uzskata, ka pašlaik aizsākas jauna ekonomiskās sistēmas ēra – uzmanības ekonomika, kurā vissvarīgākais un retākais resurss ir uzmanība, nevis kapitāls[36]. Līdz ar sociālo mediju izplatīšanos, eksponenciāli pieauga arī satura veidošana, tādā veidā vēl nopietnāk palielinot informācijas pārsātinājumu. Informācijas ir kļuvis tik daudz, ka cilvēkiem vairs fiziski nav laika apskatīt visu saturu.

Līdz ar to uzņēmumiem un satura veidotājiem ir ļoti svarīgi mācēt pārvaldīt lietotāju uzmanību, kas var pat nebūt konkrētajā brīdī pilnībā vērsta uz viņu produktu. Viens veids kā pārvaldīt cilvēku uzmanību ir izmantojot dizainu. Efektīvs dizains spēj dabiski vadīt cilvēka uzmanību tā, lai viņam nenāktos pavadīt pārāk daudz laika mēģinot saprast kā darbojas produkts un panākot sev tīkamo rezultātu. Laba dizaina nozīme produktos pieaug ļoti strauji, kas liek uzņēmumiem un satura veidotājiem pielāgoties jaunākajām dizaina tendencēm ātrāk kā jebkad agrāk. Arī zinātnē uzmanība ir aktuāls temats. Līdz ar kognitīvās zinātnes attīstību, attīstījās arī dažādi praktiskie pielietojumi, piemēram, industrijā plaši pielietotais kognitīvais dizains, kas paredz dažādu produktu izstrādes procesā ņemt vērā kognitīvās zinātnes gūtās atziņas par cilvēku apziņas procesiem.

Darba autors savā darbā vēlas izveidot programmatūras risinājumu, kas varētu palīdzēt tīmekļa lapu izstrādātājiem veidot tīmekļa lapas ar tādu dizainu, kas ņem vērā cilvēka uzmanības tendences un spētu sniegt atgriezenisko saiti gandrīz reālā laikā. Tādā veidā tiek panākta metodoloģiska pieeja dizaina veidošanai, kas taupa dizaineru laiku un varētu vieglāk ievērot kognitīvā dizaina principus. Darba ietvaros ir paredzēts izpētīt kādā veidā ir iespējams noteikt kur ir vērsta cilvēka uzmanība un izveidot tādu sistēmu, kas varētu simulēt cilvēka uzmanību aplūkojot mājaslapu ar labāku precizitāti kā gadījuma rezultātu sniegto un prezentēt to viegli uztveramā veidā un šo sistēmu arī praktiski realizēt. Kaut arī šāda tipa pakalpojums jau eksistē[15], tas ir dārgs un nespēj sniegt atgriezenisko saiti reālā laikā.

Darba pirmajā nodaļā autors apraksta cilvēka uzmanības teorētiskos pamatus un paņēmienus, kurus izmanto cilvēka uzmanības mērīšanā un pētniecībā. Otrajā nodaļā darba autors apraksta risinājuma izpētes procesu. Autors izvēlējās izmantot mašīnmācīšanās paņēmienus, lai simulētu cilvēka vizuālo uzmanību. Tādēļ tika atrasta vizuālās uzmanības simulēšanai paredzēta datu kopa un mašīnmācīšanās modelis, kas ar to tika trenēts. Trešajā nodaļā tiek pārbaudīta izvēlētā mašīnmācīšanās tīkla modeļa atbilstība izmantošanai sistēmā. Ceturtajā nodaļā tiek parādīts projekta pārskats un plans. Tajā var redzēt kādi elementi ir

sistēmas pamatā un kā tie mijiedarbojas. Piektajā nodaļā ir paskaidrota priekšgala sistēmas darbība un projektējumā pieņemtie lēmumi. Sestajā nodaļā ir detalizēti aprakstīta aizmugurgala sistēmas darbība. Septītajā nodaļā ir aprakstīts datu kopas izveides process, kuru autors nolēma izveidot pārbaudot vai ir iespējams uzlabot modeļa precizitāti ar autoram pieejamajiem resursiem. Tajā ir iekļauts apraksts par programmatūru, kas tika izmantota attēlu uzņemšanā, kā arī programmatūra un servisi, kas tika izmantoti uzmanības datu iegūšanai. Visbeidzot pēdējā nodaļā tiek apskatīta modeļa trenēšanas process ar autora izveidotu datu kopu.

1. CILVĒKA UZMANĪBA

Šajā nodaļā ir aprakstīti cilvēka uzmanības teorētiskie pamati. Pirms var runāt par cilvēka uzmanības modelēšanu, ir jāsaprot kādi parametri un paveidi tai ir. Daudzi pētniecībā izmantotie paņēmieni ir nepieciešami simulēšanā, jo ir jābūt saprotamam veidam kā var kvantificēt uzmanības datus un interpretēt to nozīmi, lai varētu veidot efektīvu simulācijas programmatūru.

1.1. Kas ir cilvēka uzmanība, kādi parametri tai ir?

Cilvēka uzmanība ir ļoti plašs jēdzienis. Kaut arī ikdienā lietojot šo vārdu cilvēki aptuveni nojauš par ko ir runa, zinātnē runājot par cilvēka uzmanību ir jāspecificē dažas lietas. Pirmkārt, uzmanība var būt dažāda veida un iedarboties ar dažādiem maņu orgāniem, piemēram audiālais, vizuālais u.c. uzmanības veids. Eksistē arī 2 dažādi uzmanības pievēršanas mehānismi – no augšas lejupejošais un no apakšas augšupejošais. No augšas lejupejošais (top-down) ir tāds uzmanības mehānisms, kas palīdz cilvēkam pievērst uzmanību uz objektiem, kad cilvēkam ir konkrēts mērķis, piemēram, meklējot savu draugu cilvēku pūlī. Šī uzmanības mehānisma ietvaros var runāt par vizuālās meklēšanas paradumiem un iemaņām. No augšas lejupejošais mehānisms tiek iedarbināts apzināti. Savukārt, no apakšas augšupejošais (bottom-up) mehānisms ir tāds mehānisms, kas tiek iedarbināts no zemapziņas un parasti tiek aktivizēts, kad cilvēkam nav konkrēts mērķis vai arī notiek kaut kāda ārkārtas situācija. Piemēram, kad cilvēka virzienā lido kaut kāds objekts, tad šis mehānisms iedarbina cilvēka reakciju izvairīties no objekta vai arī nogrand pērkons un tad gribot vai negribot tam tiek pievērsta uzmanība. Tas arī nozīmē, ka šie abi uzmanības mehānismi eksistē vairākos uzmanības veidos. No apakšas augšupejošais mehānisms parasti tiek iedarbināts neapzināti. Konkrēti šī darba ietvaros autors min cilvēka uzmanību tieši kā no lejas augšupejošu vizuālo uzmanību.[50]

Darba autors izvēlējās fokusēties tieši uz neapzināto uzmanību, jo tā ir labāk izpētīta un neapzinātās uzmanības simulēšana zinātnē ir aktuāls temats. Apzinātās uzmanības simulēšana un izpēte ir daudz sarežģītāks process, jo apzināto uzmanību ietekmē daudz vairāk faktoru. Liela daļa cilvēka neapzināto kognitīvo procesu nāk no instinktiem, kas visiem cilvēkiem ir samērā līdzīgi un ir daudz prognozējamāki par apzināto domāšanu. Savukārt, apzinātie procesi ir ļoti atkarīgi no citām cilvēka domām, emocionālā stāvokļa un citiem faktoriem, kas apgrūtina izpēti un līdz ar to arī simulēšanas procesu. Kaut arī cilvēkos

abas šīs sistēmas ir grūti nošķirt un tās viena otru ietekmē, pastāv eksperimentālas metodes, kas palīdz mazināt vienas sistēmas ietekmi uz otru[50].

1.2. Kā tiek pētīta cilvēku uzmanība?

Cilvēka uzmanības pētniecībā tiek izmantoti daudzi dažādi paņēmieni. Pirmkārt, lai novērotu kur cilvēks pievērš savu uzmanību, tiek izmantotas acu kustības novērošanas ierīces, kas var noteikt ekrāna apgabalu, uz kuru cilvēks skatās. Otrkārt, lai varētu atlasīt derīgu informāciju no trokšņa, izmanto īpašus algoritmus, kas atpazīst acu fiksācijas un sakādes. Fiksācijas ir punkti, uz kuriem acs tiek nofiksēta uz kādu brīdi. Savukārt, sakādes apraksta apgabalus, kuriem acs pārslīd pāri.

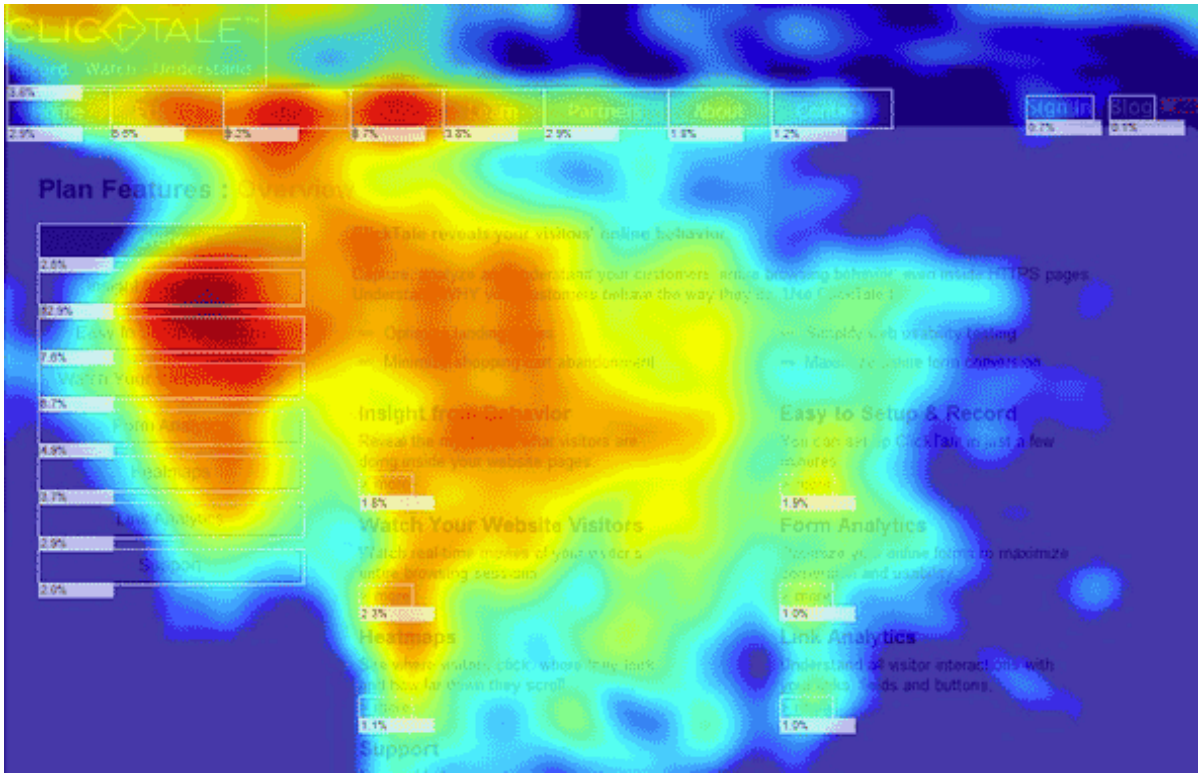
att. 1.1. **Cilvēka acu kustības novērošanas ierīce[1]**

Iegūstot datus no acu kustību novērošanas ierīcēm, ir iespējams noteikt attēla apgabalus, kas piesaista visvairāk uzmanības. Viens no veidiem kā noteikt interesantākos attēla apgabalus ir nosakot cilvēka uzmanības fiksācijas punktus. Aprakstot dažādu objektu spēju piesaistīt vizuālo uzmanību, izmanto izcelšanās (saliency) mērvienību[44]. Izcelšanās parāda to cik ļoti dotais attēla apgabals piesaista cilvēka acu fiksācijas. Attēli eksperimentu dalībniekiem parasti tiek parādīti uz ļoti neilgu laiku (parasti 5 sekundes) un netiek dots nekāds konkrēts uzdevums(tikai brīvi apskatīt attēlu pēc paša iegribas), lai pēc iespējas vairāk tiktu izmantots *bottom-up* uzmanības mehānisms.. Tādā veidā tiek noteikti attēla apgabali, kuriem cilvēki

pievērš skatienu vispirms. Kaut arī cilvēka uzmanībai ir daudz dažādu parametru un acu kustības un fiksācijas var tikai aptuveni noteikt attēlu apgabalus, kur tiek pievērsta uzmanība[49], šī darba ietvaros autors uzskata šos datus par pietiekamiem, lai aptuveni novērtētu cilvēka uzmanības tendences tīmekļa lapu kontekstā.

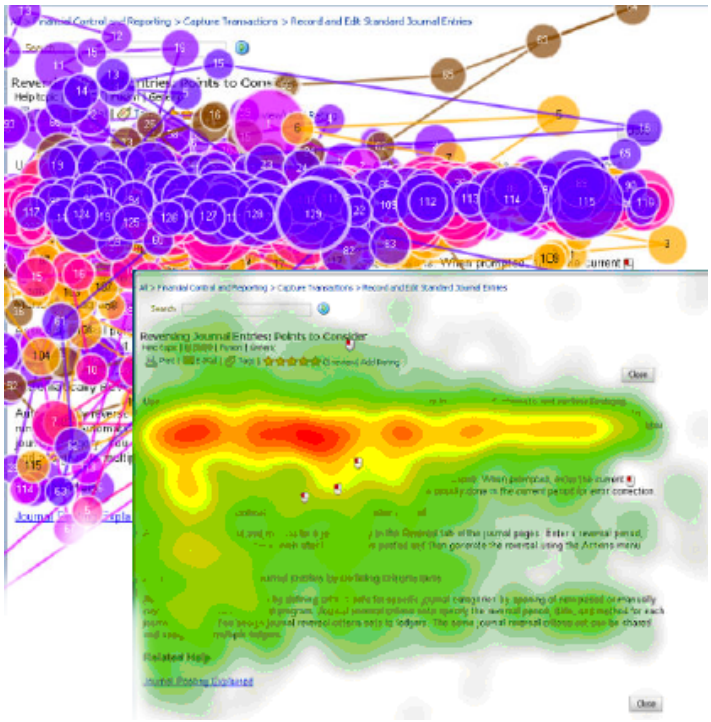
1.3. Uzmanības temperatūras kartes

Dati, kas tiek iegūti no acu kustību novērošanas iekārtām nav ne vizuāli pievilcīgas, ne arī informatīvas. Tā kā iekārtas parasti saglabā savus datus tekstuālā formātā aprakstot cilvēka uzmanību ar skaitļu masīviem, tāpēc efektīvas rezultātu prezentācijas nolūkos ir svarīgi informāciju pārveidot vizuāli uztveramā veidā. Vizuāli attēlojot uzmanību visvairāk piesaistošos attēla apgabalus parasti izmanto temperatūras kartes. Temperatūras kartes tiek attēlotas kā pārklājums jau esošam attēlam veidojot papildus informācijas slāni. Temperatūras karti matemātiski iegūst apvienojot vairāku dalībnieku datus un uz tiem pielietojot Gausa filtru[2]. Var uzskatīt, ka punkti, uz kuriem tika pievērsta uzmanība saņem vienu punktu, bet pateicoties Gausa filtram tam blakus esošie punkti arī saņem punktus, tikai mazākā apmērā. Punkti katrā attēla pikselī tiek summēti no visu dalībnieku rezultātiem un attēla apgabali, kuros ir visvairāk punktu tiek iekrāsoti sarkanā krāsā (vai arī kā citādi tiek parādīts, ka tajos punktos uzmanība ir visintensīvākā).



att. 1.2. **Temperatūras karte[3]**

Ir iespējamas arī dažādas citas vizualizācijas opcijas. Ir iespējams izmantot arī skenēšanas ceļa (scanpath) vizualizāciju. Tā parāda cilvēka acs kustības ceļu cauri attēlam parādot kurās vietās cilvēka acs fokusējas konkrētajos laika posmos. Tāda veida vizualizācijai nav nepieciešams cits datu formāts vai atšķirīgi dati, kā temperatūras kartes izveidei. Taču apskatot salīdzinājumu starp temperatūras kartēm un skenēšanas ceļu, ātri vien var secināt, ka temperatūras karte ir daudz praktiskāks veids kā attēlot uzmanības informāciju.



att. 1.3. **Skenēšanas ceļa (augšā) un temperatūras kartes (apakšā) vizualizācijas[4]**

Temperatūras karti nav obligāti attēlot kā krāsainu 2 dimensiju plankumu pārklātu pāri attēlam. To ir iespējams attēlot arī kā 3 dimensiju izvirzījumu tajos attēlu reģionos, kuros tika pievērsts visvairāk uzmanības. Taču šāda pieeja apgrūtinātu vizualizācijas izveidi, jo būtu nepieciešams 3 dimensiju vizualizācijas satvars vai bibliotēkā, kā arī tas prasītu vairāk atmiņas un skaitļošanas resursu no lietotāja datora, bet netiek iegūti nekādi ieguvumi salīdzinājumā ar 2 dimensiju temperatūras kartēm.

1.4. Secinājumi

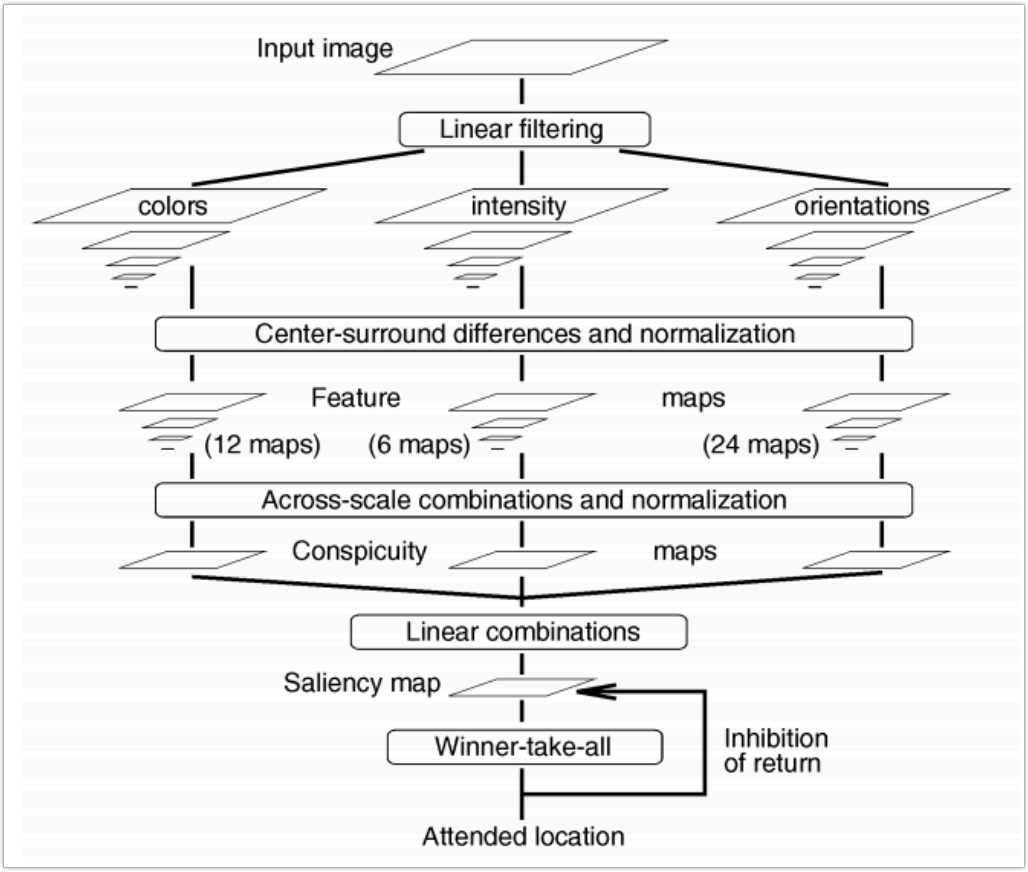
Izpētot vizuālās uzmanības teoriju, autors guva nepieciešamos ieskatus, lai varētu darboties ar uzmanības datiem un tos vizuāli attēlot. Tika noskaidrots, ka vislabāk izpētīta ir *bottom-up* tipa uzmanība, kur viena no galvenajām metrikām ir *saliency*. Ir vairāki veidi kā vizuāli var attēlot *saliency*, taču vissaprotamākais veids autoram šķita temperatūras kartes.

2. RISINĀJUMA IZPĒTE

Risinājuma izpētes nodaļā tiek aprakstīta literatūras izpēte, kura rezultātā darba autors nonāca pie procesa, ka ļauj risināt pētāmo problēmu. Šajā nodaļā tiek aprakstīta teorija cilvēka uzmanības simulēšanai, kā arī praksē pielietotajiem paņēmieniem.

2.1. Kā var simulēt cilvēka uzmanību?

Cilvēka uzmanību ir iespējams simulēt vairākos veidos. Viens veids ir veicot zinātniskus pētījumus saprast kā darbojas cilvēka uzmanība un mēģināt to precīzi simulēt izveidojot specializētus algoritmus, jeb izveidot uz likumiem balstītu sistēmu. Ja programmas un cilvēka uzmanību uzskata par funkcijām, tad izveidot uz likumiem balstītu sistēmu nozīmē ar pētījumu palīdzību noskaidrot cilvēka uzmanības funkciju un tad to realizēt datorprogrammas formā.



att. 2.1. **Itti-Koch modelis[5]**

Attēlā 2.1.1 var redzēt vienu no pirmajiem mēģinājumiem izveidot programmatūras modeli cilvēka uzmanības simulēšanai[5]. Var redzēt, ka risinājums paredz sadalīt uzdevumu sīkākās

daļās un mēģināt atsevišķi risināt katru problēmu – krāsas, intensitāti un objektu orientācijas. Katram no aspektiem tiek izveidotas kartes un tiek veiktas normalizācijas, līdz beigās visa informācija tiek integrēta un tiek noskaidroti visinteresantākie reģioni attēlā. Šīs programmas arhitektūra tika balstīta uz īpašību integrēšanas teoriju, kas paredz vizuālās informācijas sadalīšanu vairākās iezīmēs, piemēram krāsās, intensitātē u. tml. Pēc tam katra no šīm iezīmēm tiek apstrādātas un tiek noteikts kuros attēla reģionos ir visinteresantākās vērtības katrai no iezīmēm. Beigās katras iezīmes interesantākie reģioni tiek sapludināti kopā, tiek noskaidrots attēlā visinteresantākais reģions un tiek pieņemts lēmums pievērst tam uzmanību. Principā šis modelis spēj tikai ļoti aptuveni aprakstīt cilvēka uzmanību, jo tas ir balstīts uz samērā vienkāršu zīdītāju uzmanības modeli un cilvēka uzmanība ir daudz sarežģītāka.

Tādai pieejai ir vairāki trūkumi. Pirmkārt, šāda pieeja ir pilnībā atkarīga no zināšanām, kas ir uzkrātas par cilvēka uzmanību. Tā kā cilvēka uzmanība vēl tiek aktīvi pētīta un ir ļoti daudz mehānismu, kurus nesaprot pat nozares eksperti, tad ir skaidrs, ka tāda pieeja nevar sniegt labu rezultātu. Otrkārt, uz likumiem bāzētām programmām ir nepieciešams veidot ļoti specifisku risinājumu, kas var darboties tikai kādā šaurā apgabalā. Piemēram, vienā gadījumā ir jāizveido programma, kas spēj simulēt cilvēka uzmanību, kad cilvēkam netiek dots mērķis, citā gadījumā kad ir kaut kas jāatrod attēlā.

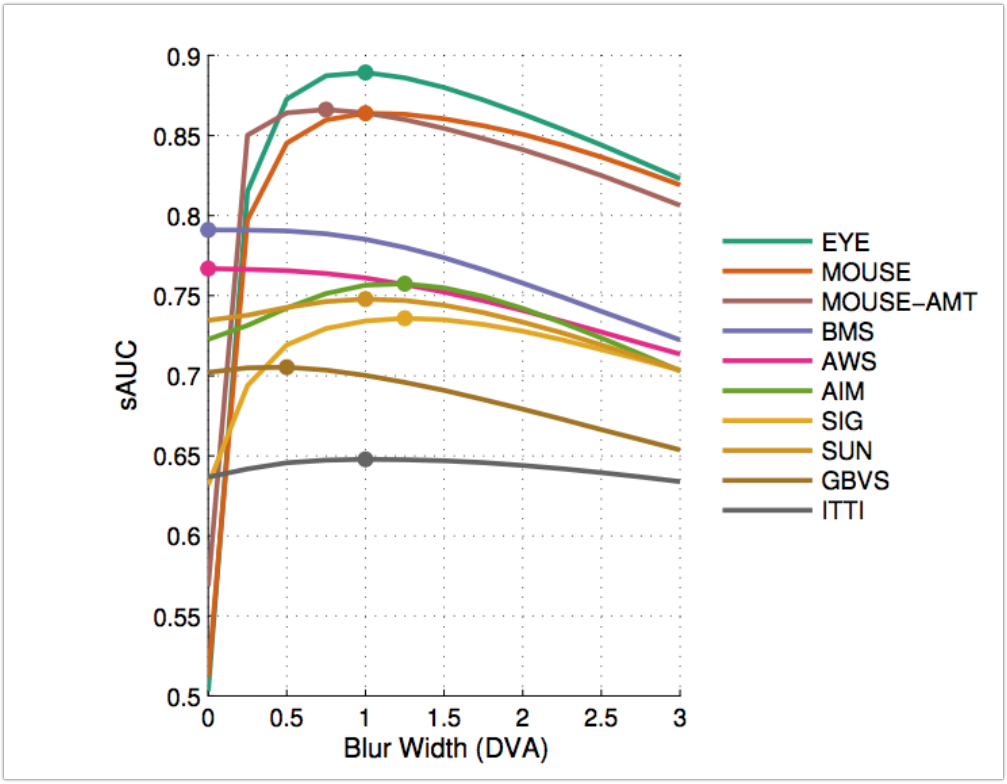
Taču ir iespējams iet arī citu ceļu. Pēdējo gadu laikā ir bijis straujš progress mašīnmācīšanās jomā. Līdz ar dziļās mašīnmācīšanās attīstīšanos, parādījās daudz pētījumu par tās pielietojumiem dažādās sfērās. Savukārt, lielo datu tehnoloģiju attīstība nodrošina datu pieejamību mašīnmācīšanās nolūkiem. Dziļās mašīnmācīšanās paņēmieni balstās pārsvarā uz progresu mašīnmācīšanās nozarē un datu pieejamību, kas nozīmē, ka nav nepieciešams pilnībā saprast pētāmo apgabalu, lai varētu izveidot programmu, kas sniedz labu rezultātu. Vēl viena priekšrocība ir tāda, ka mašīnmācīšanās paņēmieni ļauj pētniekiem atrast jaunas likumsakarības datos daudz ātrāk, līdz ar to radot jaunus ieskatus pētāmajā apgabalā. Pietam, bieži vien, lai pielāgotu jau eksistējošu modeli, ir nepieciešams veikt tikai nelielas izmaiņas kodā un jāievada jauni dati, nevis jāizveido pilnīgi jauns risinājums. Tātad vienīgi ir nepieciešams labs mašīnmācīšanās modelis un datu kopa, lai izveidotu risinājumu. Taču rodas jauna problēma – ir nepieciešama ļoti liela datu kopa.

2.2. SALICON datu kopa

Ņemot vērā to, ka dziļās mašīnmācīšanās modeļu trenēšanai ir nepieciešams milzīgs apjoms datu, dažos gadījumos nepieciešamo datu daudzumu ir ļoti grūti iegūt. Cilvēka uzmanības dati ir tieši tas gadījums. Parasti visvieglāk iegūt liela apjoma datus var digitālajā vidē, piemēram

mājaslapu skatījumu skaitu, dažādu videoklipu popularitāti vai arī pelītes kursora kustības paradumus. Fiziskajā pasaulē, savukārt, rodas daudz sarežģījumu. Konkrēti cilvēka uzmanības datu iegūšanas gadījumā problēma ir tāda, ka datu iegūšanai ir nepieciešams samērā dārgs aprīkojums (acu kustību nosakošas brilles) un ir nepieciešams liels skaits dalībnieku, kas apskatītu attēlus. Tas aizņem daudz laika un naudas. Ir pieejamas vairākas datu kopas ar dabiskiem attēliem un cilvēku uzmanības kartēm, piemēram MIT300[6] un OSIE[7], taču tās ir pārāk mazas, lai varētu efektīvi pielietot dziļo mašīnmācīšanos.

SALICON[2] datu kopā ir 10000 dabisku attēlu(fiziskās pasaules fotogrāfijas), kopā ar cilvēka uzmanības temperatūras kartēm, kas parāda attēlos reģionus, kas piesaista uzmanību visvairāk. Tās izveidē autori izmantoja inovatīvus risinājumus, kas ļauj pārnest datu iegūšanu digitālajā vidē. Publikācijas autoriem ir izdevies pierādīt, ka arī ar pelītes kustībām pa attēlu var aptuveni noteikt kuros attēla reģionos cilvēks pievērsīs visvairāk uzmanības. To panākt darba autoriem izdevās izveidojot programmu, kas izpludina attēlus un rāda skaidro attēla versiju tikai nelielā aplī apkārt pelītes pozīcijai. Tātad lietotājs ir spiests pavirzīt pelīti uz tiem attēla reģioniem, kas viņam šķiet visinteresantākie, lai spētu saskatīt attēlu labāk. Katrs attēls tika rādīts dalībniekiem tikai 5 sekundes, līdz ar to dalībniekiem nebija daudz laika apskatīt visu attēlu. Programma tika publicēta Amazon Mechanical Turk[8] servisā, kas palīdz atrast dalībniekus eksperimentam attālināti par samaksu. Kopā ar attēliem, kuriem vajadzēja noskaidrot uzmanības temperatūras karti, tika attēloti arī attēli ar zināmām temperatūras kartēm, kas tika izmantoti kvalitātes kontrolei. Pētniekiem izdevās pierādīt, ka iegūtās temperatūras kartes ir līdzīgas tām, kuras var iegūt ar laboratorijās izmantotu specializētu acu kustības novērojošu aparatūru.



2.2. att. **Uzmanības simulēšanas modeļu, acs novērošanas un pelītes datu salīdzinājums[2]**

Attēlā 2.2.1 var redzēt, ka ar pelītes izsekošanas palīdzību (MOUSE un MOUSE-AMT (Amazon Mechanical Turk)) iegūtie dati ir ļoti līdzīgi ar acu novērošanas aparatūru iegūtajiem datiem (EYE). Taču pašā apakšā (ITTI) ir uz likumiem balstīts risinājums, kas tika aprakstīts 3.1. apakšnodaļā. Tas nozīmē, ka pieeja SALICON datu kopas veidošanā der liela apjoma datu kopu veidošanai, kurās nav nepieciešama ideāla precizitāte.

Kaut arī pēc gandrīz visiem parametriem SALICON datu kopa ideāli der autora iecerētajiem mērķiem, tā satur dabiskos attēlus, nevis tīmekļa lapu ekrānšāviņus. SALICON datu kopa satur attēlus, kam ir kaut kāda sakritība ar mājaslapu attēliem, taču mājaslapām ir dažādas specifikas, kas nav atrodamas dabiskajos attēlos. Tāpēc vispirms ir jāpārbauda cik kvalitatīvi attēlus spēj pareģot jau ar SALICON datiem uztrenētais modelis, jo var gadīties, ka tas jau rada pietiekami labu rezultātu.

2.3. Kādi neironu tīkli ir vispiemērotākie?

Lai izvēlētos modeli sistēmas izveidei, ir jāizpēta kādus modeļus pētnieki jau izveidoja un uztrenēja izmantojot SALICON datu kopu. Izpētot informāciju par SALICON, izrādījās, ka MIT pētnieki ir apkopojuši informāciju par dažādiem modeļiem[9], kas tika testēti un

salīdzināti savā starpā pēc iegūtajiem rezultātiem simulējot uzmanības temperatūras karti MIT300 datu kopā. Katrs modelis tiek salīdzināts pēc 8 dažādām metrikām:

- Līdzība (Similarity) – Tiek apskatīts abu temperatūras karšu šķēlums, ja tās tiek apskatītas kā histogrammas (Vērtība 1 nozīmē, ka abas kartes ir vienādas)[9]
- Earth Mover's attālums (EMD) – Nosaka cik lielu daļu kartes būs jāmaina, lai abas kartes būtu vienādas (Vērtība 0 nozīmē, ka abas kartes ir identiskas)[9]
- Kullback-Leibler diverģence (KL) – Parāda cik daudz informācijas tiek zaudēts, kad modelis mēģina simulēt cilvēka uzmanību[9]
- Normalized Scanpath Saliency (NSS) – Tiek rēķinātā kā vidējā *saliency* vērtība acu fiksācijas punktos[9]
- Pearson lineārais koeficients (CC) – Lineārais korelācijas koficients starp divām kartēm (0 = nav korelācijas)[9]
- 3 dažādas Area Under ROC Curve implementācijas (sAUC, AUC-Judd un AUC-Borji) – *Saliency* karti izmanto kā bināro klasifikatoru (iedala ieejas datus divās klasēs - atbilstoši vai ne). Metrika nosaka cik liela proporcija ir starp pareizi atpazītiem un nepareizi atpazītiem datiem, mēģinot uzģenerēt jau zināmu karti[33]

Model Name	Published	Code	AUC-Judd [?]	SIM [?]	EMD [?]	AUC-Borji [?]	sAUC [?]	CC [?]	NSS [?]	KL [?]	Date tested [key]	Sample [img]
Baseline: infinite humans [?]			0.92	1	0	0.88	0.81	1	3.29	0		
Deep Gaze 2	Matthias Kümmerer, , Thomas S. A. Wallis, Leon A. Gatys, Matthias Bethge. DeepGaze II: Understanding Low- and High-Level Contributions to Fixation Prediction [ICCV 2017]		0.88 (0.84)	0.46 (0.43)	3.98 (4.52)	0.86 (0.83)	0.72 (0.77)	0.52 (0.45)	1.29 (1.16)	0.96 (1.04)	first tested: 26/11/2015 last tested: 13/09/2016 maps from authors (model without center bias in parentheses)	
SALICON	Xun Huang, Chengyao Shen, Xavier Boix, Qi Zhao		0.87	0.60	2.62	0.85	0.74	0.74	2.12	0.54	first tested: 19/11/2014 last tested: 15/11/2015 maps from authors	
DeepFix	Srinivas S S Kruthiventi, Kumar Ayush, R. Venkatesh Babu DeepFix: A Fully Convolutional Neural Network for predicting Human Eye Fixations [arXiv 2015]		0.87	0.67	2.04	0.80	0.71	0.78	2.26	0.63	first tested: 02/10/2015 last tested: 02/10/2015 maps from authors	
Deep Spatial Contextual Long-term Recurrent Convolutional Network (DSCLRCN)	Nian Liu, Junwei Han. A Deep Spatial Contextual Long-term Recurrent Convolutional Network for Saliency Detection [arXiv 2016]		0.87	0.68	2.17	0.79	0.72	0.80	2.35	0.95	first tested: 16/06/2016 last tested: 27/07/2016 maps from authors	
Saliency Attentive Model (SAM-ResNet)	Marcella Cornia, Lorenzo Baraldi, Giuseppe Serra, Rita Cucchiara. Predicting Human Eye Fixations via an LSTM-based Saliency Attentive Model [arXiv 2016]	python	0.87	0.68	2.15	0.78	0.70	0.78	2.34	1.27	first tested: 10/30/2016 last tested: 03/03/2017 maps from authors	
Saliency Attentive Model (SAM-VGG)	Marcella Cornia, Lorenzo Baraldi, Giuseppe Serra, Rita Cucchiara. Predicting Human Eye Fixations via an LSTM-based Saliency Attentive Model [arXiv 2016]	python	0.87	0.67	2.14	0.78	0.71	0.77	2.30	1.13	first tested: 10/30/2016 last tested: 03/03/2017 maps from authors	
DenseSal	Taiki Oyama, Takao Yamanaka		0.87	0.67	1.99	0.81	0.72	0.79	2.25	0.48	first tested: 14/06/2017 last tested: 14/06/2017 maps from authors	
DPNSal	Taiki Oyama, Takao Yamanaka		0.87	0.69	2.05	0.80	0.74	0.82	2.41	0.91	first tested: 19/04/2018 last tested: 19/04/2018 maps from authors	
SalGAN	Junting Pan, Cristian Canton, Kevin McGuinness, Noel E. Oâ€™Connor, Jordi Torres, Elisa Sayrol and Xavier Giro-i-Nieto. SalGAN: Visual Saliency Prediction with Generative Adversarial Networks [arXiv 2017]	python	0.86	0.63	2.29	0.81	0.72	0.73	2.04	1.07	first tested: 10/30/2016 last tested: 10/30/2016 maps from authors	

2.3. att. **Modeļu rezultātu tabula[9]**

Tabulas ekrānšāviņā netika iekļauti visi modeļi, jo tie ir uzrādījuši vājākus rezultātus salīdzinājumā ar attēlā redzamajiem modeļiem. Attēlā 2.3.1 redzamie modeļi ir uzrādījuši labākus rezultātus paredzot cilvēku uzmanību nekā viena cilvēka uzmanības temperatūras karte. Tabulas pašā augšā ir "bezgalīga cilvēku daudzuma" bāzlīnija. Tā parāda kādas īpašības ir temperatūras kartēm, kad vienā grupā novērotāju skaits tiecas uz bezgalību (ļoti daudz) un tās grupas temperatūras karte tiek izmantota, lai paredzētu citas novērotāju grupas temperatūras kartes. Principā, sasniedzot šīs bāzlīnijas rezultātu, tiek sasniegts labākais iespējamais rezultāts un pēc šīs bāzlīnijas var novērtēt modeļu sniegumu. Apskatot tabulu var redzēt, ka SALICON, Deep Fix un Deep Gaze 2 modeļi dominē šajā tabulā.

Baseline: one human [?]			0.80 min: 0.76 max: 0.83	0.38 min: 0.33 max: 0.46	3.48 min: 2.88 max: 4.18	0.66 min: 0.63 max: 0.71	0.63 min: 0.60 max: 0.67	0.52 min: 0.38 max: 0.65	1.65 min: 1.21 max: 2.10	6.19 min: 4.76 max: 8.37		

2.4. att. **Viena cilvēka uzmanības temperatūras kartes rezultāti[9]**

Taču šiem modeļiem nav pieejams pirmkods. No tabulā redzamajiem modeļiem vienīgi SAM (Saliency Attentive Model) un SalGAN ir atvērtā pirmkoda programmas. Kaut arī pēc 5 no 8 parametriem Saliency Attentive Model pārspēj SalGAN, atšķirības nav lielas un darba autoram SalGAN modelis likās interesantāks, jo tas ietver sevī GAN tīklus, kas uzrādīja lieliskus rezultātus daudzos mašīnmācīšanās pētījumos.

2.4. GAN un to pielietojumi

Ģeneratīvi konkurējošie tīkli (GAN) ir neuzraudzītas mašīnmācīšanās neironu tīklu kopa, kas sastāv no diviem atsevišķiem tīkliem. Vienu tīklu sauc par ģeneratoru, kas ir atbildīgs par jaunu piemēru izveidi. Otrs tīkls ir diskriminators, kas atbild par piemēru šķirošanu starp īstiem piemēriem un ģeneratora izveidotiem piemēriem. Ģeneratora uzdevums ir apmānīt diskriminatoru – tātad izveidot pietiekami realistiskus piemērus, bet diskriminātors mēģina pieķert ģenerētās bildes. Ģenerators ievadā saņem diskriminatora atbildi uz ģeneratora veidotiem piemēriem, savukārt diskriminators saņem ģeneratora veidotus piemērus un īstus piemērus no datu kopas. Lai varētu salīdzināt diskriminatora un ģeneratora sniegumu tiek izmantot zuduma funkcija.[37]

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

2.5. att. **GAN zuduma funkcija[37]**

Šajā formulā var redzēt GAN konkurences pamatprincipa darbību. Pirmā daļa satur logaritmu no varbūtības, ka diskriminators atpazīs patiesos datus no datu kopas *x* kā patiesus. Otrā daļa sastāv no logaritma no inversas varbūtības, ka diskriminators atpazīs ģeneratora bildi kā patiesu. Ģenerators (G) mēģina šo summu samazināt veidojot saturu, kas atgādina īstu saturu, bet diskriminators mēģina šo summu palielināt, tātad kļūstot precīzākam. Ģeneratoram tiek padots arī trokšņa mainīgais *z*, kas ļauj ģeneratoram veidot jaunus attēlus katru reizi, nevis vienmēr vienu un to pašu attēlu.[37]

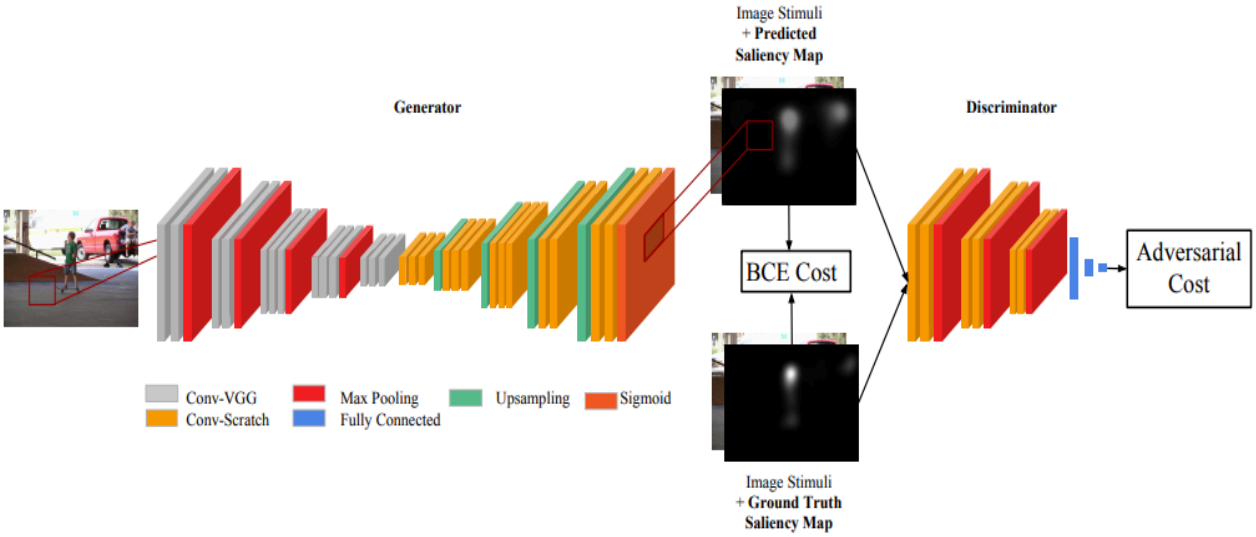
2.6. att. **GAN ģenerēti guļamistabu attēli[45]**

Ģeneratīvos konkurējošus tīklus parasti izmanto satura ģenerēšanas uzdevumos. Tiek sagatavota datu kopa, kas satur objektus, kas ir līdzīgi tiem, kurus tīklam vēlas iemācīt ģenerēt. Attēlā 2.6 ir parādīti GAN veidoti attēli ar guļamistabām, kas nozīmē, ka datu kopā bija līdzīgi attēli ar guļamistabām. GAN dabiski der gandrīz jebkuram uzdevumam, kur ir jāizveido papildus piemēri ar kaut kāda veida datiem, taču ne šo datu precīzas kopijas. Tāpēc GAN var izmantot arī mūzikas[40], video[39], teksta[41] un cita satura ģenerēšanai.

2.5. SalGAN

Arī uzmanības temperatūras kartes var ģenerēt tāpat kā citu saturu. Tā kā temperatūras karti var interpretēt kā attēlu, tad pieeja var būt gandrīz identiska darbam ar attēliem. Ģenerators SalGAN gadījumā ir izveidots pēc konvolūcijas iekodētāja-atkodētāja arhitektūras, kas nozīmē, ka vispirms tīkls samazina ievada dimensijas apvienojot pikseļus nelielos apgabalos, tādā veidā

samazinot attēla izšķirtspēju. Pēc tam šos apgabalus analizējot ir iespējams noteikt cik intensīva uzmanība ir tajos apgabalos. Atkodētājs savukārt palielina iegūtās uzmanības kartes izšķirtspēju atpakaļ līdz oriģinālajam attēla izmēram. Iekodētāja arhitektūra ir VGG-16[42] un SalGAN trenēšanai izmanto jau daļēji uztrenētu VGG-16 modeli, lai paātrinātu trenēšanas procesu. Atkodētāja arhitektūra ir līdzīga, tikai tā slāņi ir apgriezti otrādi un izšķirtspējas samazināšanas slāņu vietā tiek izmantoti izšķirtspēju palielinošie slāņi.[38]



2.7. att. **SalGAN arhitektūra[38]**

Savukārt diskriminators, kura arhitektūra arī ir redzama attēlā 2.7., Tas sastāv no 6 konvolūciajs slāņiem un 3 izšķirtspējas samazināšanas slāņiem. Tad visbeidzot ir 3 pilnīgi savienoti slāņi, kas beigu beigās izvada atbildi (Jā vai nē).[38]

2.6. Secinājumi

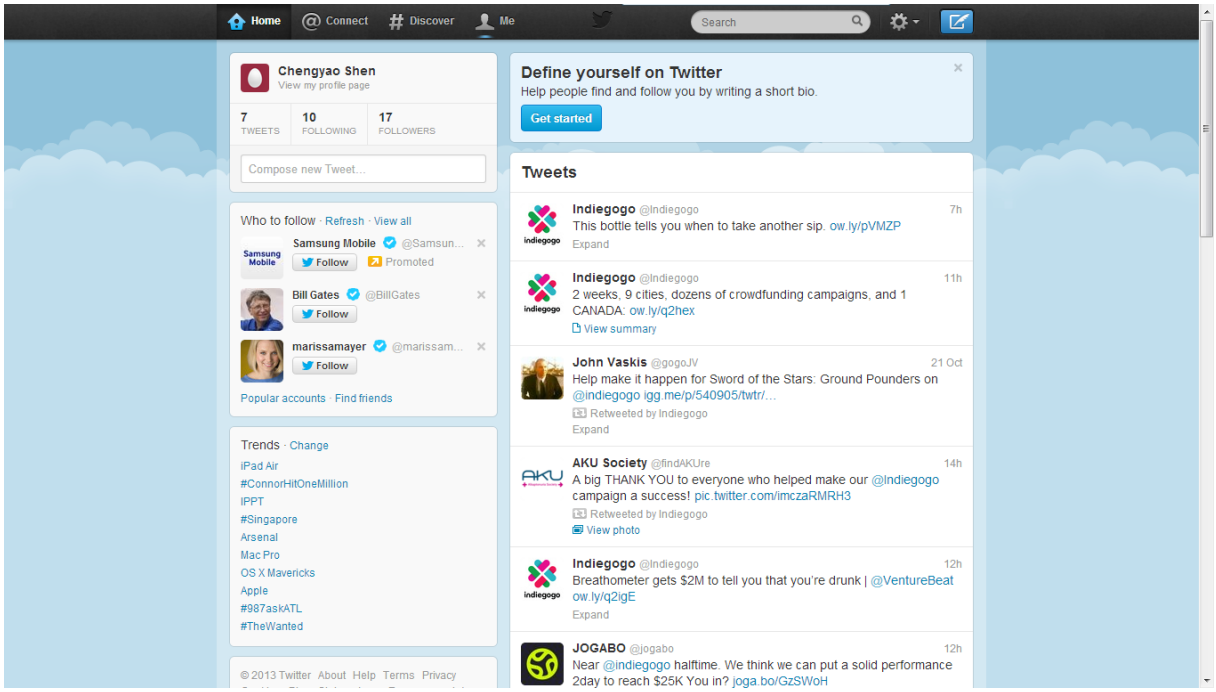
Tika izvēlēta pēc autora domām labākā pieeja uzmanības simulēšanai. Tā kā mašīnmācīšanās strauji attīstās un tās risināmās problēmas ir līdzīgas autora izvēlētajai, tad šī pieeja ir daudzsološa. Mašīnmācīšanās ļauj arī daudz vienkāršāku rezultāta uzlabojumu, kas balstās uz datu kvalitāti un pieejamiem vispārīgiem mašīnmācīšanās algoritmiem, nevis uz specifiskiem šaurai jomai paredzētiem algoritmiem. Datorzinātnē jau tika veikti vairāki pētījumi par uzmanības simulēšanu ar mašīnmācīšanos un ir pieejami modeļi, kas panāk lieliskus rezultātus. Autors savā darbā izvēlējās SalGAN tīklu, kas uzrādā rezultātus ļoti tuvu teorētiski labākajam iespējamam rezultātam un tas pēc autora domām sniegs adekvātu rezultātu.

3. IEPRIEKŠ TRENĒTA MODEĻA PĀRBAUDE

Šajā nodaļā autors pārbauda iepriekš trenēta SalGAN modeļa atbilstību tīmekļa lapu uzmanības temperatūras karšu simulēšanai. Lai varētu pārbaudīt kvalitāti, ir nepieciešams etalons, ar ko varētu salīdzināt risinājuma sniegumu. Darba autors nolēma izveidot pārbaudes datu kopu, kuras zināmās temperatūras kartes tiktu salīdzinātas ar izmantotā modeļa rezultātiem.

3.1. Pārbaudes datu kopas izveide

Lai varētu efektīvi pārbaudīt datu kvalitāti un novērtētu modeļa sniegumu, ir jāizmanto kāda pārbaudes datu kopa. Autoram šajā ziņā bija 2 izvēles – vai nu veidot tādu kopu pašam izmantojot acu kustību novērošanas aparatūru, vai arī atrast un pielāgot tādu datu kopu Internetā. Izrādās, ka šāda datu kopa jau pastāv. *Webpage Saliency*[12] raksts, kas tika publicēts 2014. gadā, satur 149 dažādu tīmekļa lapu attēlus ar 1360x768 pikseļu izšķirtspēju un tiem atbilstošas uzmanības temperatūras kartes. Datu ievākšanas procesā piedalījās 11 cilvēki, kas katrs apskatīja 149 attēlus no trim tīmekļa lapu kategorijām – tekstuālas, attēliem bagātas un jauktas lapas. Dati tika ierakstīti ar 1000 Hz frekvenci, kas rada samērā lielu datu apjomu katram dalībniekam (55 MB).



att. 3.1. **Attēls no *Webpage Saliency* datu kopas[12]**

Datu kopa satur neapstrādātus datus no acu kustību novērošanas aparāta, tīmekļa lapu ekrānšāviņus, no datiem izveidotās uzmanības temperatūras kartes, acu fiksācijas informācija un Matlab kodu, kas ir izmantots datu apstrādei. Kaut arī ir pieejama acu fiksācijas informācija, SalGAN ievaddatos vajag arī acu skatiena vai pelītes lokācijas pozīcijas datus, taču tie ir glabāti neapstrādātā formā un nav uzreiz lietojami. Tādēļ sākumā darba autors uzrakstīja Python skriptu, kas ar regulāro izteiksmju palīdzību spēj izfiltrēt informāciju par acs skatienu koordinātēm konkrētajos laika posmos un izvēlēties katru 10. punktu, lai samazinātu datu kopas frekvenci līdz 100 Hz un samazinot datu kopas izmēru.

```
4293247   687.3   359.7  2872.0     0.5     2.5   34.40   25.80 ...
4293248   687.2   359.6  2872.0     0.4     2.2   34.40   25.80 ...
4293249   687.1   359.6  2871.0     0.2     1.9   34.40   25.80 ...
4293250   687.1   359.6  2871.0     0.2     1.5   34.40   25.80 ...
4293251   687.1   359.6  2872.0     0.3     1.1   34.40   25.80 ...
4293252   687.1   359.7  2874.0     0.3     0.7   34.40   25.80 ...
4293253   687.2   359.7  2876.0     0.4     0.3   34.40   25.80 ...
4293254   687.2   359.7  2877.0     0.6    -0.1   34.40   25.80 ...
4293255   687.1   359.7  2877.0     0.8    -0.4   34.40   25.80 ...
4293256   687.0   359.7  2877.0     1.1    -0.6   34.40   25.80 ...
4293257   687.2   359.7  2877.0     1.4    -0.7   34.40   25.80 ...
4293258   687.4   359.6  2877.0     1.6    -0.8   34.40   25.80 ...
4293259   687.7   359.6  2877.0     1.8    -1.0   34.40   25.80 ...
4293260   687.7   359.5  2877.0     1.9    -1.2   34.40   25.80 ...
4293261   687.6   359.5  2877.0     1.8    -1.3   34.40   25.80 ...
4293262   687.5   359.4  2877.0     1.7    -1.1   34.40   25.80 ...
4293263   687.5   359.3  2877.0     1.6    -1.0   34.40   25.80 ...
4293264   687.6   359.4  2877.0     1.5    -0.9   34.40   25.80 ...
4293265   687.9   359.4  2876.0     1.4    -0.9   34.40   25.80 ...
4293266   688.1   359.4  2877.0     1.2    -1.0   34.40   25.80 ...
4293267   688.2   359.3  2879.0     0.9    -1.1   34.40   25.80 ...
4293268   688.4   359.2  2881.0     0.6    -1.1   34.40   25.80 ...
4293269   688.2   359.1  2883.0     0.3    -1.1   34.40   25.80 ...
4293270   687.9   359.2  2883.0    -0.4    -1.2   34.40   25.80 ...
4293271   687.8   359.4  2883.0    -1.6    -1.5   34.40   25.80 ...
4293272   687.8   359.6  2885.0    -3.0    -2.1   34.40   25.80 ...
4293273   687.9   359.3  2887.0    -5.2    -3.1   34.40   25.80 ...
EFIX R   4292928        4293273 346       687.0   356.5    2795   34.35   25.80
SSACC R  4293274
4293274   688.0   359.0  2888.0    -8.0    -4.5   34.40   25.80 ...
```

att. 6.5.1 **Acu kustību novērotāja dati (1. kolonna – laiks milisekundēs, 2. un 3. kolonnas x un y koordinātas)**

Pēc datu apstrādes tiek iegūta datu struktūra, kas satur sevī masīvu no objektiem, kas satur x, y koordināšu laukus un laika zīmogu. Beigās ir nepieciešams apvienot jauniegūto struktūru ar jau esošo acu fiksāciju informāciju. To autors panāca ar vēl vienu Python programmu, kas izmantojot *mat4py*[51] bibliotēku spēja atvērt acu fiksāciju Matlab bināro failu un apvienojot to vienā vārdnīcā ar neapstrādātiem acu kustības datiem, saglabā jaunos Matlab binārajos failos.

3.2. Rezultāti

Lai iegūtu rezultātus, vispirms ir jāizvēlas metrika, pēc kuras SalGAN modelis tiks salīdzināts ar pārbaudes datu kopas temperatūras kartēm. Apskatot tabulu (att. 2.3), var redzēt biežāk izmantotās metrikas uzmanības modeļu salīdzināšanai. Autors izvēlējās izmantot AUC (nod. 2.3.) metriku, kuru izmanto arī SALICON rakstā[2]. SALICON raksta autori ir izveidojuši atvērtā pirmkoda programmu kopu, kuru izmanto modeļu snieguma pārbaudei trenējoties ar SALICON datu kopu[13]. Šī programmu kopa satur arī Python valodā rakstītu programmu, kas rēķina AUC divām temperatūras kartēm. Ievadot datus programmā (direktoriju ar pārbaudes datiem un direktoriju ar ģenerētajiem datiem), uz ekrāna tiek izdrukāts vidējais AUC rezultāts no visiem attēliem un saraksts ar individuālajiem rezultātiem katram attēlam.

```
(0.6181707334541265, array([0.62603697, 0.65194805, 0.65626459, 0.62622515, 0.51304158,
       0.72006642, 0.63177816, 0.63250739, 0.68619765, 0.61470388,
       0.62289165, 0.65702145, 0.67438897, 0.55536638, 0.62228421,
       0.52214387, 0.63914365, 0.62280382, 0.46987992, 0.65408667,
       0.57080112, 0.57435545, 0.58526684, 0.74899487, 0.60330062,
       0.68906834, 0.53838084, 0.65470604, 0.53681529, 0.64732793,
       0.60174485, 0.49462871, 0.65132702, 0.693912  , 0.68041063,
       0.71880989, 0.69632016, 0.56429008, 0.73261839, 0.54695223,
       0.49429323, 0.58996672, 0.64491352, 0.61088137, 0.63147346,
       0.59549159, 0.69104127, 0.59805306, 0.6180652 , 0.48211194,
       0.54091768, 0.58969773, 0.57230833, 0.59479732, 0.60430398,
       0.46709143, 0.62485871, 0.62762116, 0.57610144, 0.68082746,
       0.64757571, 0.62545852, 0.55220567, 0.72646189, 0.55883828,
       0.64347874, 0.71160502, 0.64093837, 0.70308364, 0.65155175,
       0.55713425, 0.65073066, 0.68789878, 0.58336756, 0.76481795,
       0.65758262, 0.60084595, 0.55008841, 0.66288242, 0.66310567,
       0.64671869, 0.58262492, 0.46994158, 0.65237686, 0.56530795,
       0.49917166, 0.63541289, 0.6750826 , 0.54341079, 0.72608824,
       0.74769851, 0.63082899, 0.72459906, 0.53025213, 0.60440675,
       0.54750275, 0.58211302, 0.54662117, 0.62778304, 0.5553827 ,
       0.57103704, 0.57581328, 0.58072706, 0.58550917, 0.58945106,
       0.6278449 , 0.65776777, 0.57035634, 0.63809065, 0.43758273,
       0.83786604, 0.69304438, 0.58559592, 0.46355949, 0.69653814,
       0.52624703, 0.65125565, 0.58919319, 0.6404005 , 0.59784339,
       0.56341941, 0.69650713, 0.6330052 , 0.52154866, 0.48746241,
       0.67306647, 0.56938249, 0.60608938, 0.65067401, 0.63504565,
       0.66268938, 0.72959381, 0.5699441 , 0.65523163, 0.5997326 ,
       0.61755909, 0.63091201, 0.73646233, 0.75415227, 0.71490194,
       0.60052067, 0.76396943, 0.66092822, 0.50327492, 0.58400232,
       0.4499092 , 0.64491755, 0.67044547, 0.70976331]))
```

att 6.2 **Ar SALICON datu kopas trenēta SalGAN rezultāts**

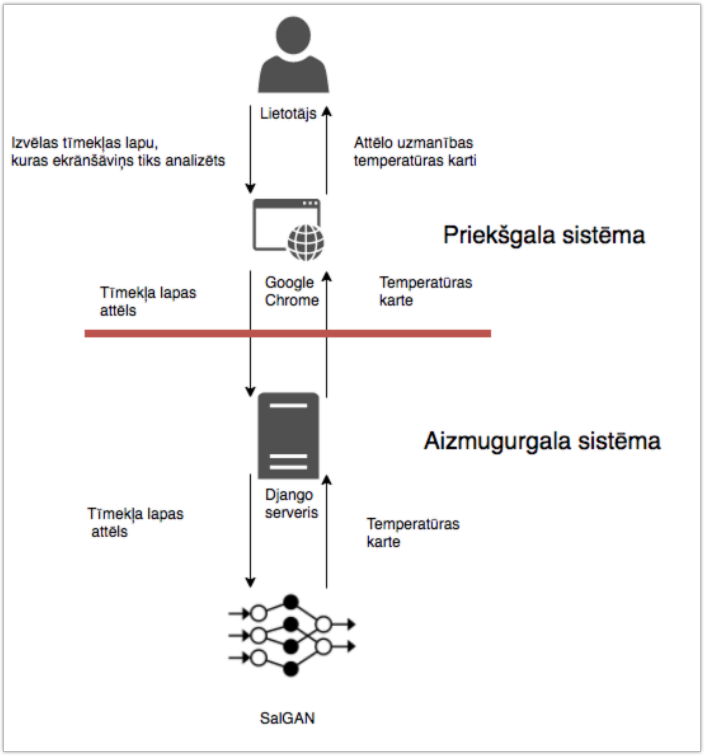
Attēlā 6.2 var redzēt, ka iegūtais rezultāts ir ~0.62 AUC (Pasvītrotais rezultāts). Ņemot vērā teorētisko maksimumu 0.92 AUC, tad var sarēķināt modeļa precizitāti izdalot modeļa AUC vērtību ar teorētiskā maksimuma vērtību, iegūstot ~67% precizitāti. Salīdzinot ar MIT sniegtajiem rezultātiem[9], var secināt, ka šis rezultāts ir labāks par gadījuma rezultātu. Tātad var konstatēt, ka arī ar SALICON datu kopas trenētu SalGAN modeli var veidot *saliency* temperatūras kartes arī tīmekļa lapām.

3.3. Secinājumi

Tā kā ar SALICON datu kopu trenēts SalGAN modelis uzrādīja labākus rezultātus kā gadījuma rezultātus (0.5 AUC), tad var secināt, ka šo modeli var lietot risinājuma izstrādē. Pēc šī rezultāta iegūšanas var sākt veidot pašu sistēmu.

4. DARBA PLĀNS UN RISINĀJUMA ARHITEKTŪRA

Pirms tiek veikts darbs pie produkta izstrādes, ir nepieciešams saprast kā var panākt vēlamos rezultātus. Tā kā ir nepieciešams informāciju attēlot lietotājam pēc iespējas ērtākā veidā, tad viens no aspektiem būtu attēlot informāciju vistuvāk videi, kurā lietotājs veic savu darbu. Tā kā tīmekļa lapu veidotāji izmanto pārlūkprogrammas, lai apskatītu rezultātus, tad autoram vislabākais veids kā pasniegt informāciju šķiet caur paša veidotu pārlūkprogrammas spraudni. Tātad ir paredzēts izveidot pārlūkprogrammas Google Chrome spraudni, kas spēj attēlot pašlaik aplūkotās tīmekļa lapas uzmanības temperatūras karti. Tas nozīmē, ka pirmajā kārtā ir nepieciešams izveidot pašu spraudni. Pats spraudnis ir atbildīgs tikai par tīmekļa lapas ekrānšāviņa uzņemšanu un nosūtīšanu uz serveri, kā arī mijiedarbību ar lietotāju. Serveris ir atbildīgs par ekrānšāviņa saņemšanu, tā nodošanu SalGAN modelim un rezultātā iegūtās uzmanības temperatūras kartes nosūtīšanu spraudnim. Serveris ir realizēts izmantojot Python valodas tīmekļa satvaru Django. Ir pieejami dažādi JavaScript valodas satvari, ar kuru palīdzību ir iespējams iepakot modeļus tādā veidā, lai tos varētu palaist spraudnī, bez nepieciešamības komunicēt ar ārēju modeli. Darba autors nolēma šādu pieeju neizmantot, jo tā padara spraudni pārāk lielu (vairāki simti megabaitu līdz pat gigabaitiem) un var padarīt spraudņa lietošanu nepraktisku (pārāk lielas atmiņas un jaudas prasības lietotāja datoram). Pie reizes tiek atdalīts prezentācijas slānis (vizuālās informācijas attēlošana un mijiedarbība ar lietotāju) no darījumprocesu slāņa (datu apstrāde un sagatavošana).



3.1 att. **Produkta darbības cikls**

Attēlā 3.1 ir redzams produkta darbības cikls, kas apraksta visu informācijas apmaiņas procesu sākot no ekrānšāviņa uzņemšanas, beidzot ar attēla prezentēšanu lietotājam.

5. GOOGLE CHROME SPRAUDNIS

Šajā nodaļā autors apraksta izmantotās tehnoloģijas Google Chrome spraudņa izveidē, kā arī apraksta spraudņa projektējumu. Spraudņa uzdevums ir uzņemt tīmekļa lapas ekrānšāviņu, to nosūtīt serverim un attēlot saņemto rezultātu kā pārklājumu virs tīmekļa lapas. Tiek arī apskatītas alternatīvas, kuras varētu lietot veidojot spraudni un arī iegūtie rezultāti.

5.1. Izmantotās tehnoloģijas

Mūsdienās ir pieejami daudz dažādu pārlūkprogrammu un konkurējošu tīmekļa tehnoloģiju. Vienu uzdevumu var veikt dažādos veidos – daži no tiem ir efektīvāki, citi mazāk efektīvi, tāpēc ir svarīgi izvēlēties pareizos rīkus attiecīgajam uzdevumam. Darba autors izvēlējās veidot tieši Google Chrome pārlūkprogrammas spraudni vairāku iemeslu dēļ. Pirmkārt, Google Chrome ir populārākais pārlūks pasaulē[18]. Otrkārt, spraudņa izveidošanas process ir samērā vienkāršs un maz atšķiras no parastas priekšgala tīmekļa lapas izveides[19].

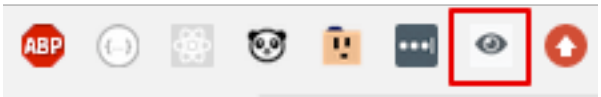
Tā kā Google Chrome spraudnis pamatā ir tīmekļa lapa, tad spraudņa izveidē tiek izmantotas tīmekļa tehnoloģijas. Pēc noklusējuma tiek izmantots HTML, CSS un JavaScript (ECMAScript 5) tehnoloģiju komplekts. Taču pieejami daudzi tīmekļa satvari un kompilatori, kas ļauj atvieglot darbu ar tīmekļa tehnoloģijām, uzlabojot koda lasāmību, uzturamību un automatizējot vairāku uzdevumu izpildi. Viens no labākajiem piemēriem šajā sfērā ir priekšgala sistēmu satvars React[17] JavaScript valodā. Tas ļauj automatizēt vairākus ar interfeisu saistītos uzdevumus, piemēram, informācijas atjaunošanu un programmas stāvokļa pārvaldīšanu. Ir pieejami vairāki citi satvari, kas pilda līdzīgus uzdevumus, piemēram, AngularJS[20] un VueJS[21]. Aplūkojot spraudņa funkcionalitāti ir skaidrs, ka tas nav pietiekami sarežģīts, lai tāda tipa satvars būtu nepieciešams. Tātad autors izvēlējās neizmantot JavaScript satvarus spraudņa izstrādes procesā. Taču ir iespējamas arī vairākas citas modifikācijas noklusējuma tehnoloģiju komplektā. Ir pieejamas vairākas modifikācijas valodai JavaScript, piemēram ECMAScript 6, 7 un 8, kā arī TypeScript[22], CoffeeScript[23] un citas modifikācijas, kas padara to jaudīgāku un vieglāk izmantojamu. Spraudņa izveidē autors nolēma neizmantot valodas JavaScript modifikācijas, jo tam būtu nepieciešams izmantot kompilatorus, kas varētu kompilēt attiecīgo JavaScript paveidu uz JavaScript ECMAScript 5 un tas visu procesu apgrūtina un nesniedz gandrīz nekādas priekšrocības.

Spraudņa funkcionalitātē ir paredzēts manipulēt ar DOM elementiem un JavaScript valodā tās nav implementētas pietiekami ērti pēc autora uzskatiem. Tāpēc spraudnī ir paredzēts izmanto *jQuery*[24] bibliotēku, kas atvieglo darbu ar DOM. Bibliotēkas instalācija ir ļoti

vienkārša, ir nepieciešams tikai saglabāt bibliotēkas kodu lokāli un attiecīgi atsaukties uz to spraudņa konfigurācijas failā. Papildus tam ir nepieciešama bibliotēka ekrānšāviņu uzņemšanai no tīmekļa lapas DOM struktūras. Tam autors izvēlējās bibliotēku *html2canvas*[25], kas spēj pārveidot mājaslapas HTML un CSS informāciju par attēlu base64 formātā.

5.2. Spraudņa projektējums

Spraudnis tika izveidots pēc Google Chrome izstrādātāju vadlīnijām[19]. Tās paredz, ka spraudnim ir jāsastāv no spraudņa *manifesta*, kas ietver sevī informāciju par spraudni un tā komponenetēm, kā arī no pašām spraudņa komponentēm (HTML, CSS, JavaScript un multivides faili). Spraudņa manifests ir JSON formāta fails, kas apraksta informāciju par spraudni – spraudņa nosaukums, versija, atļaujas, kas ir nepieciešamas, lai realizētu spraudņa paredzēto funkcionalitāti, kā arī spraudņa ikonas izskats un funkcijas.



att. 5.1. **Spraudņa ikona rīkjoslā (apvilkta ar sarkanu kvadrātu)**

Papildus iepriekš minētajai informācijai, spraudnis satur arī informāciju par komponentēm. Viena no komponentēm ir fona HTML fails. Fona HTML fails ir parasti paredzēts citu failu ielādei vai arī procesiem, kas notiek visu laiku kamēr spraudnis ir aktīvs. Kaut arī tas ir HTML fails, tā elementi netiek nekur attēloti. Fona fails šajā gadījumā tiek izmantots, lai ielādētu jQuery bibliotēku un kodu, kas nodrošina ekrānšāviņa uzņemšanu un nosūtīšanu uz serveri, kad lietotājs nospiež uz spraudņa ikonas. Alternatīva būtu izveidot notikumu (klikšķis uz spraudņa ikonas) un norādīt kodu, kas tiktu palaists. Taču tas ir sarežģītāk un uz katru klikšķi nāktos veikt bibliotēku ielādi. Attēla uzņemšanas un nosūtīšanas kods tiek injecēts mājaslapās, kas tiek apmeklētas, kad spraudnis ir aktivizēts, lai varētu piekļūt mājaslapas informācijai.

att. 5.2. **estudijas.lu.lv ar uzmanības kartes pārklājumu**

Pēc klikšķa uz spraudņa ikonas, tiek uzņemts attēls base64 formātā. Pēc tam attēls tiek pārveidots par failu, kas sastāv no vairākiem fragmentiem, kas var tikt pārsūtīti uz serveri. Fails tiek sagatavots sūtīšanai vispirms izveidojot faila HTML formu, kurai piesaista sūtāmo failu un pēc tam veidojot HTTP POST pieprasījumu uz serveri, kas nosūta formas datus. Saņemot atbildi no servera attēla formā, tas tiek ievietots lapas DOM struktūrā *body* elementā kopā ar attēlam pievienoto CSS stilu, kas pārklāj to pa virsu visiem pārējiem lapas elementiem un pievieno caurspīdīgumu. Spraudņa JavaScript kods ir ievietots 5. pielikumā.

5.3. Secinājumi

Ir izveidots Google Chrome spraudnis, kas var veikt efektīvu komunikāciju ar serveri un nodrošina lietotājam grafisku saskarni ar rīku, kas ļauj lietotājam uzņemt ekrānšāviņu un nosūtīt to uz serveri apstrādei. Spraudnis kalpo arī kā vizualizācijas attēlotājs, parādot lietotājam uzmanības temperatūras karti kā pārklājumu virs mājaslapas. Tādā veidā spraudnis pilda visus priekšgala sistēmai nepieciešamās funkcijas dotajā sistēmā.

6. SERVERIS

Šajā nodaļā autors apraksta serverī izmantotās tehnoloģijas un servera projektējumu. Servera uzdevums ir spēt saņemt ekrānšāviņu no spraudņa un to saglabāt uz lokālā diska. Pēc tam serverim ir jāizsauc SalGAN un jāsagaida rezultātā iegūtais uzmanības temperatūras kartes attēls, kas pēc tam ir jānosūta spraudnim kā atbilde.

6.1. Izmantotās tehnoloģijas

Serveris tika izveidots Python valodā izmantojot Django tīmekļa satvaru. Python valoda ir ļoti populāra mašīnmācīšanās risinājumu izveidē un vispopulārākā mašīnmācīšanās bibliotēka TensorFlow ir izveidota Python programmēšanas valodā. SalGAN arī ir realizēts Python valodā un izmanto Lasagne[26] mašīnmācīšanās bibliotēku. Tāpēc Python valodas izmantošana ir vislabākais risinājums saderībai. Django tīmekļa satvars automatizē vairākas ar tīmekli saitītās darbības, piemēram, maršrutēšanu un HTTP pieprasījumu pārsūtīšanu. Ir vairāki tīmekļa satvari Python valodā, kas risina līdzīgas problēmas, piemēram, Flask un Pyramid, taču darba autors ir labāk pazīstams tieši ar Django. Lai varētu ģenerēt ekrānšāviņiem unikālus nosaukumus servera pusē, tiek izmantota pakotne UUID, kas var izveidot nejaušu 32 simbolu virkni, ko var izmantot kā attēlu failu nosaukumus. Pakotņu pārvaldīšanai tiek izmantots PIP pakotņu pārvaldnieks kopā ar Virtualenv valodas Python vides pārvaldnieku.

6.2. Servera projektējums

Serveris ir veidots izmantojot Django izstrādes serveri kopā ar noklusējuma iestatījumiem. Serveris sastāv no viena galapunkta "/sendScreen/", uz kuru spraudnis nosūta POST pieprasījumu ar tīmekļa lapas ekrānšāviņu. Galapunkts norāda uz servera vienīgo skatu – *send_screen*. Skats satur kodu, kas saglabā saņemto kodu uz lokālā diska un palaiž SalGAN ar nepieciešamajiem parametriem. Pēc rezultāta saņemšanas no SalGAN, tas tiek iekodēts base64 formātā un nosūtīts kā HTTP atbilde spraudnim. Servera kods ir ievietots 4. pielikumā.

6.3. Secinājumi

Rezultātā tika izveidots serveris, kas darbojas kā starpnieks starp spraudni un SalGAN modeli. Kaut arī serveris nav tehniski sarežģīts risinājums, tas palīdz atdalīt prezentācijas slāni no darījumprocesu slāņa un mazināt spraudņa izmēru. Serveris ir izveidots tā, lai veiktu vienu funkciju – nodot datus SalGAN un pārsūtīt atbildi uz spraudni. Tātad darba galvenais mērķis tika sasniegts – ir izveidots risinājums, kas var paredzēt cilvēka uzmanību tīmekļa lapu kontekstā ar precizitāti lielāku par gadījuma rezultātu (AUC 0.5). Pēc servera pabeigšanas un notestēšanas, spraudnis tika publicēts iekš Google Chrome Store ar saiti: https://chrome.google.com/webstore/detail/fjalpjgpmlabcdhlocglgkmhnocdbifd/.

7. DATU KOPAS IZVEIDES PROCESS

Pēc iecerēto mērķu sasniegšanas, darba autors vēlējās noskaidrot vai ar viņam pieejamajiem resursiem ir iespējams uzlabot sistēmas sniegumu. Lielu datu kopu izveide parasti izmaksā ievērojamas naudas summas. Ja SALICON autori maksāja $0.01 katram darbiniekam par katru apskatīto attēlu, tad 10000 attēli, katrs apskatīti 60 reizes izmaksāja $6000, neņemot vērā Amazon Mechanical Turk platformas komisijas. Neskatoties uz to darba autors iedalīja 150 EUR lielu budžetu šim uzdevumam un vēlējās noskaidrot vai ir iespējams izveidot pietiekami labu datu kopu, lai varētu uzlabot jau esošo rezultātu. Šajā nodaļā autors apraksta datu kopas izveides procesu. Lai izveidotu jaunu datu kopu, sākumā bija nepieciešams izveidot lielu dažādu mājaslapu ekrānšāviņu kopu, kas tika veikts ar autora uzrakstītu NodeJS programmu izmantojot *Puppeteer*[47] pakotni, kas ļauj vadīt Google Chrome pārlūku ar NodeJS programmām bez maksas un ar pilnu pārlūka kontroli. Programmas sākotnējā versijā darba autors mēģināja izmantot *Url2PNG*[46] servisu, lai atomatizētu ekrānšāviņu ievākšanu, taču attēlu kvalitāte nebija apmierinoša. Servisa lietošanā autors iztērēja EUR 20, kas nozīmē, ka datu kopas izstrādei atlika vien EUR 130. Lai ievāktu datus par cilvēku uzmanības kartēm tīmekļa lapu ekrānšāviņos tika izveidota tīmekļa lapa, kas sastāvēja no priekšgala sistēmas, kas attēloja ekrānšāviņus dalībniekiem un ievāca pelītes kustības datus un tos saglabāja serverī[2]. Datu ievākšana norisinājās Amazon Mechanical Turk platformā, kas ļauj nolīgt cilvēkus konkrētiem īstermiņa uzdevumiem.

7.1. Mājaslapu attēlu iegūšanas programmatūra

Datu kopas veidošanas procesu ir nepieciešams sākt ar mājaslapu attēlu iegūšanas programmatūru un mājaslapu saraksta izveidi. Mājaslapu saraksta izveidei autors izvēlējās izmantot *klart.io*[14] mājaslapu, kas satur sarakstu ar mājaslapām, kas pēc *klart.io* radītāja domām ir vizuāli pievilcīgas. Ielādējot visas lapas HTML struktūru Google Chrome pārlūkā un palaižot dažas komandas pārlūka konsolē, tika iegūts saraksts ar saitēm uz mājaslapām, taču ar daudzām liekām rindām. Tādēļ pēc tam ar Python programmas palīzību saraksts tika attīrīts no liekām rindām un jaunais saraksts saglabāts teksta failā.

Pēc mājaslapu saraksta saglabāšanas, tika izveidota programma ekrānšāviņu ieguvei. Programmas ideja ir izmantojot Puppeteer pakotni vadīt neredzamu Google Chrome instanci un katrā mājaslapā ieejot pēc 5 sekunžu pauzes uzņemt ekrānšāviņu, pēc tam pabīdīt lapu uz leju par ekrāna augstuma skaita pikseļiem (720 px), lai iegūtu 1280x720 pikseļu attēlus un pēc vēl 5 sekunžu pauzes uzņemt vēl vienu ekrānšāviņu un atkal atkārtot lapas pabīdīšanas un

ekrānšāviņa uzņemšanas procesu līdz tiek sasniegtas tīmekļa lapas beigas. Pauzes pirms ekrānšāviņa uzņemšanas ir nepieciešamas gadījumā, ja tīmekļa lapas satur animācijas, kas parādās pirmo reizi redzot saturu, lai to atraktīvāk prezentētu. Ar pauzes palīdzību tiek palielināta iespēja, ka animācijas jau beigs spēlēt un tiks uzņemts ekrānšāviņš ar tīmekļa lapas izskatu statiskajā formā. Kopā tika iegūti 1000 attēli.

7.2. Cilvēka uzmanības iegūšanas programmatūra

Viena no svarīgākajām un sarežģītākajām sastāvdaļām datu kopas izveidē ir cilvēka uzmanības iegūšanas programmatūra. Lai varētu izveidot programmu, vispirms ir nepieciešams saprast kā darbojas SALICON rakstā aprakstītais risinājums. Galvenās risinājuma idejas ir pelītes kustību ierakstīšana un dažādu attēla segmentu attēlošanu ar dažādu izšķirtspēju, atkarībā no pelītes atrašanās vietas. Pelītes pozīcijai vistuvāk atrodas attēla segments ar oriģinālo izšķirtspēju, apkārt tam atrodas attēls ar 2 reizes mazāku izšķirtspēju un tam apkārt vēl 2 reizes mazākas izšķirtspējas versija un tā attēls tiek attēlots 6 slāņos. Tādā veidā veidojot 6 slāņu Gausa piramīdu. Lai precīzi definētu kā tiek veikta attēlu pārklāšanās, rakstā ir minētas vairākas formulas.

Pirmkārt, tiek nodefinēts leņķis α, pie kura attēla izšķirtspēja samazinās 2 reizes un tas ir 2.5 grādus liels (atbilst cilvēka acu tīklenes asumam). Pēc tam tiek nodefinēta R funkcija katrā pikselī.

$$R(x,y) = \frac{\alpha}{\alpha + \theta(x,y)}$$

att. 7.1. **Izšķirtspējas kartes funkcija[2]**

Funkcija θ ir pikseļa attālums no pelītes pozīcijas, kas ir pareizināts ar koeficientu $\frac{1}{p}$, kur SALICON autori p noteica ar vērtību 7.5, lai nodrošinātu patīkamu pieredzi dalībniekiem. Pēc tam tiek noteikta pārnešanas funkcija.

$$T_i(f) = \begin{cases} e^{1/2 \times (-2^{i-3} f/\sigma)^2}, & i = 1, \cdots, 5 \\ 0, & i = 6, \end{cases}$$

att. 7.2. **Pārnešanas funkcija[2]**

Pārnešanas funkcija satur samērā daudz parametru. σ ir konstante, kas apzīmē ko autori noteica kā 0.248. Savukārt, i ir Gausa piramīdas slāņa numurs, kuram tiek rēķināta pārejas funkcija. Pārnešanas funkciju izmanto krāsu pārejas funkcijai B(x, y) katrā pikselī.

$$B(x,y) = \frac{0.5 - T_i(x,y)}{T_{i-1}(x,y) - T_i(x,y)},$$

att. 7.3. **Krāsu pārejas funkcija[2]**

Krāsu pārejas funkcija savukārt tiek izmantota transformācijas matricā, kas nosaka kā attēliem vajadzētu tikt sapludinātiem kopā.

$$M_i(x,y) = \begin{cases} B(x,y), & i = i_0 - 1 \\ 1 - B(x,y), & i = i_0 \\ 0, & \text{otherwise} \end{cases}$$

att. 7.4. **Filtra matrica[2]**

Šī matrica beigās tiek sareizināta ar katru Gausa piramīdas elementu (attiecīgajos i slāņos) un visus iegūtos attēlus sareizinot tiek iegūts beigu attēls. Lai noteiktu i0 – jeb kurā Gausa piramīdas slānī atrodas dotais punkts (x, y) – tiek sarēķinātas 6 w_i vērtības, kur $T_i(w_i) = 0.5$ un $w_6 = 0$. Punkts (x, y) pieder i slānim, kad $w_{i-1} > R(x, y) > w_i$. Šis process ir samērā sarežģīts un paļaujas uz filtra elementu implementāciju no pilnīgas nulles – lineārās algebras līmenī. Neskatoties uz to, darba autors mēģināja realizēt šo pieeju.

Lai realizētu SALICON rakstā aprakstīto filtra izveides procesu, vispirms bija jāizvēlas tādu JavaScript bibliotēku, kas atbalstītu lineārā algebras operācijas, jo JavaScript pēc noklusējuma tās neatbalsta. Principā ir 2 galvenie kandidāti – MathJS[27], kas ir veidota kā uzlabojums JavaScript valodā pēc noklusējuma esošai matemātikas bibliotēkai Math. Otra opcija ir lietot mašīnmācīšanās bibliotēkas Tensorflow JavaScript versiju – TensorflowJS[28]. Autors izlēma izmantot TensorflowJS šī darba nolūkos, jo viņam jau bija pieredze darbā ar Tensorflow bibliotēku. Risinājuma izveides procesā autors sastapās ar iepriekš neparedzētām grūtībām – izrādījās, ka TensorflowJS bibliotēka ir pārāk lēna nepieciešamajam risinājumam – viena kadra aprēķināšanai aiziet aptuveni 1 sekunde. Viens attēla izmēra matricas reizinājums TensorflowJS bibliotēkai aizņem aptuveni 100 milisekundes, taču SALICON datu kopas autori veica pelītes informācijas nolasīšanu 100 reizes sekundē. Kaut arī pelītes pozīcijas nolasīšanu un attēla atjaunināšanu var atdalīt, ir skaidrs, ka lēna attēla atjaunošana negatīvi ietekmēs datu

kvalitāti. SALICON autori izmantoja Matlab, lai izveidotu savu datu ievākšanas programmatūru, taču rakstā netika aprakstīts kādā veidā Matlab programma tika publicēta tīmeklī un autoram ir maz pieredzes darbojoties ar Matlab programmatūru. Tāpēc autors izlēma iet nestandarta ceļu.

Mūsdienās ir ļoti populāri izmantot HTML5 Canvas elementa un WebGL[52] piedāvātās iespējas. Līdz ar Adobe Flash novecošanu, vairāk un vairāk tīmekļa pārlūkos bāzētu spēļu izstrādātāji sāka izmantot HTML5 un WebGL, kas mudināja izstrādātājus radīt dažādas uz šīm tehnoloģijām bāzētas datorspēļu veidošanas bibliotēkas, kas atvieglo darbību ar šīm tehnoloģijām. Ir pieejams vesels klāsts ar dažādām bibliotēkām, sākot ar vienkāršām Canvas elementa manipulācijas bibliotēkām kā Caman[29], infografiku un datu vizualizācijas bibliotēku D3.js[30] un beidzot ar 3 dimensiju objektu attēlošanas un manipulācijas bibliotēkām kā Three.js[31]. Taču autora iecerēm vislabāk der bibliotēka PIXI.js[32]. Tā ir specifiski veidota 2 dimensiju datorspēļu veidošanai un nāk komplektā ar vairākiem ļoti labi optimizētiem attēlu filtriem, kur viens no tiem ir tieši Gausa filtrs. Turklāt, bibliotēka ļauj efektīvi ielādēt un pārvaldīt attēlus, iegūt informāciju par pelītes kursoru un attēlot tekstu.

Lai varētu sākt lietot PIXI.js, vispirms ir jāsaprot kā var vienkāršot SALICON autoru lietotās matemātiskās formulas. Apskatot formulas vērīgāk, var nākt klajā ar secinājumu, ka visas w_i vērtības īstenībā ir konstantas, jo tās nav atkarīgas ne no viena attēla vai kursora parametriem – tātad tos ir iespējams aprēķināt iepriekš. Iegūstot vērtības, autors secināja, ka katra nākamā w_i vērtība palielinās 2 reizes. Turklāt apskatot iepriekš apskatītās formulas, var secināt, ka ir iespējams aprēķināt kuri pikseļi piederēs kuram attēla izšķirtspējas līmenim. Tā kā pikseļi vienādā attālumā no pelītes kursora saņem vienādu R(x, y) vērtību (jo vienīgais mainīgais ir attālums no pelītes kursora), tad var secināt, ka katrs attēla slānis tiek attēlots kā aplis un θ ir vienkārši šī apļa rādiuss. Taču paliek vienīgi problēma ar krāsu pārneses funkciju. Tās uzdevums ir padarīt pāreju starp slāņiem nemanāmu un lai nebūtu dīvainu un uzmanību novērsošu artefaktu. Šīs funkcijas rezultātu ir iespējams aptuveni reproducēt izmantojot apļa masku ar malām, kas ir daļēji caurspīdīgas. Tādā veidā 2 blakus esoši slāņi saplūdīs kopā un programmas lietošana kļūs patīkamāka.

Gala risinājumā tika izmantota priekšgala sistēma (kods 7. pielikumā) (HTML lapa ar PIXI.js elementu) un aizmugursistēma (Django serveris, kods 6. pielikumā), kas saglabā datus un atsūta nākamo attēlu. Priekšgala sistēma izmanto PIXI.js un jQuery bibliotēkas. Vispirms, lai inicializētu PIXI.js bibliotēku, tiek izveidota jauna PIXI.js lietotne un tās skats tiek pievienots HTML lapas DOM struktūrā. Tas panāk to, ka tiek izveidots melns 1280x720 (Datu kopas attēlu izmērā) lieluma taisnstūris, kas ir PIXI.js scēna un tiek ielādēta WebGL bibliotēka. Pēc tam scēnai tiek pievienots teksts ar instrukcijām un notikums, kas sagaida peles klikšķi un

ielādē nākamo attēlu pēc klikšķa. Pēc attēla ielādes tam tiek pielietots PIXI.js iebūvēts Gausa filtrs, ar parametriem, kas nosaka, ka filtrs sapludinās kopā tikai 0.001 pikseli (tātad radīs minimālu gludināšanas efektu) un samazinās izšķirtspēju 10 reizes. Tādā veidā tiek izveidots fona attēls, jeb pēdējā Gausa piramīdas slāņa attēls. Kaut arī SALICON raksta autori pēdējam slānim samazināja izšķirtspēju 32 reizes, ir jāpatur prātā, ka SALICON datu kopa satur attēlus ar izšķirtspēju 640x480. Gadījumā ar mājaslapu attēliem starpība ir daudz krasāka un var gadīties, ka daži attēla elementi kļūst neredzami, jo tie ir pārāk mazi. Tas var negatīvi ietekmēt rezultātus. Tāpēc autors nolēma nevis samazināt katru nākamo attēlu divas reizes, bet gan samazināt oriģinālo attēlu par divām reizēm vairāk nekā iepriekšējā slāņa attēlu. Autors arī nolēma palielināt apļu rādiusus 3 reizes salīdzinājumā ar SALICON rakstā lietotajiem apļu rādiusiem, jo tie bija pārāk mazi attiecībā pret attēla izmēru (skaidrākā attēla aplis ir ar 6 pikseļu rādiusu) un mājaslapu attēli satur tieši 3 reizes vairāk pikseļu nekā SALICON attēli. Pēc fona attēla pievienošanas tiek izveidoti 5 papildus attēla kopijas un 5 dažādu rādiusu apļu attēli, kas tiek pa pāriem salikti konteineros (attiecīgajam attēlam attiecīgā rādiusa aplis). Katram attēlam tiek uzlikts attiecīgās izšķirtspējas Gausa filtrs un attiecīgais aplis tiek izmantots kā attēla maska, līdz ar to panākot efektu, ka vienīgais attiecīgās izšķirtspējas attēla reģions, kas ir redzams, atrodas tieši tur kur ir aplis. Panākot to, ka katrs nākamais aplis ar mazāku rādiusu un lielāku attēla izšķirtspēju tiek uzlikts viens otram pa virsu, tiek panākts efekts, ka attēla izšķirtspēja samazinās pieaugot attālumam no apļu centra. Pēc attēlu apstrādes tiek pievienots notikums, kas reaģē pakustinot pelīti un tas pakustina visus piecus riņķus tā, lai to centri būtu tieši pelītes atrašanās vietā. Lai ierakstītu pelītes atrašanās vietu konkrētajos laika brīžos, tiek pievienota funkcija, kas tiek izsaukta ik pēc 10 milisekundēm un ieraksta pelītes koordinātes un laiku. Līdzīga funkcija tiek definēta attēlu maiņai. Tā notīra PIXI.js skatuvi no pašreizējā attēla, saglabā ierakstītos pelītes datus, ielādē nākamo attēlu no servera un izvada paziņojumu par apskatīto attēlu skaitu. Pēc pelītes klikšķa uz skatuves tiek attēlots nākamais attēls. Ik pēc 10 attēliem tiek pieprasīts attēls no pārbaudes datu kopas, lai varētu kontrolēt datu kopas kvalitāti. Attēlā 7.5. ir redzms kā izskatās tīmekļa lapas ekrānšāviņš programmā. Sarkanā riņķa līnija attēlā parāda attēla apgabalu ar vislielāko izšķirtspēju (kursora atrašanās vieta).

att. 7.5. **Tīmekļa lapas ekrānšāviņš datu iegūšanas programmā**

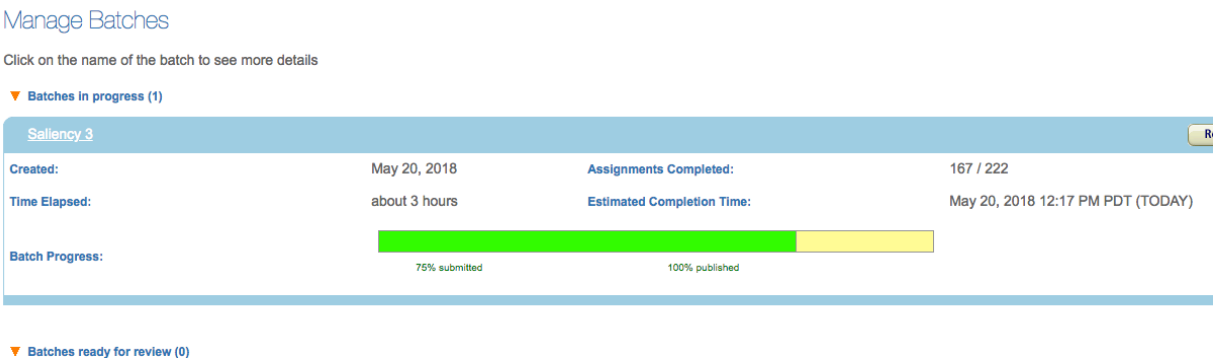
Servera pusē tiek pārvaldīta attēlu izsūtīšana un datu saglabāšana. Katram lietotājam ir iedalīta sava direktorija, kurā tiek glabāti peles kursora dati. Kvalitātes novērtēšanai paredzētie dati tiek glabāti atsevišķā direktorijā. Saņemot pieprasījumu no klienta, serveris sākumā izšķir vai tas ir pieprasījums pēc treniņa attēla vai arī pēc kontroles attēla. Ja pieprasīts tiek kontroles attēls, tad tiek saglabāti kursora dati attiecīgi kvalitātes nodrošināšanas vai arī galvenajā direktorijā, pretējā gadījumā vienkārši tiek nejauši izvēlēts attēls no treniņa direktorijas. Saglabājot peles datus, tiek veikts ieraksts *manifest.txt* failā, kas satur informāciju par to cik reizes kāds attēls vēl ir jāapskata un attiecīgajam failam atņem 1. Pēc tam tiek noteikts vai klients pieprasa attēlu no pārbaudes datu kopas vai arī parastu attēlu un attiecīgi tiek nejauši izvēlēts nākamais attēls no pārbaudes kopas vai arī no parastās kopas.

7.3. Kursora datu ievākšanas process

Datu ievākšanas process norisinājās 2 veidos. Daļa datu tika iegūta pateicoties brīvprātīgajiem un daļa datu tika iegūta Amazon Mechanical Turk platformā. Visus nepieciešamos datus nebija iespējams iegūt caur Amazon Mechanical Turk ar atlikušajiem līdzekļiem (ar EUR 130 varēja iegūt 8840 skatījumu) , tāpēc nācās lūgt palīdzību brīvprātīgajiem (vēl 1160 skatījumu). Darbā ar brīvprātīgajiem tika izmantotas saites ar unikāliem identifikatoriem katram dalībniekam, lai vēlāk būtu viegli tos identificēt un risināt dažādas problēmas, piemēram neskaidrības ar uzdevumu vai tehniska rakstura problēmas.

Katra dalībnieka direktorijai serverī tika dots unikālā identifikātora nosaukums. Eksperimenta laikā serveris tika izvietots izmantojot AWS Elastic Beanstalk[10] mākoņskaitļošanas servisu. Savukārt eksperimenta klienta puse tika publicēta izmantojot *surge.sh* [53] servisu, kas ļauj ātri un ērti publicēt tīmekļa lapas, kas sastāv tikai no priekšgala komponentēm. Katram dalībniekam tika izsūtītas viņiem iedalītās saites un tika pateikts izskatīt tik attēlus cik vien ir iespējams (maksimums visus 1000).

Amazon Mechanical Turk platformā savukārt ir dažas izmaiņas. Platformas pamatideja ir tāda, ka lietotāji, kas vēlas iegūt kaut kāda veida informāciju no cilvēkiem veido īpašus uzdevumus, kurus sauc par cilvēka inteliģences uzdevumiem (Human Intelligence Tasks), kurus var pildīt vai nu jebkurš platformā esošais strādnieks, kas piekrīt atalgojumam apmaiņā pret padarīto darbu vai arī īpaši atlasītie cilvēki, kas pārstāv konkrētu demogrāfiju. Lai izveidotu uzdevumu, ir jānosaka cik daudz cilvēku ir nepieciešams uzdevuma izpildei, cik daudz naudas katrs dalībnieks saņems un kādi papildus nosacījumi ir izpildītāju atlasei. Pēc tam ir jāuzbūvē priekšgala sistēmas risinājums, kurā izpildītāji varēs saņemt uzdevuma instrukcijas un ievadīt savas atbildes. Tā kā atbildes tiek nodotas glabāšanai izmantojot HTML formas elementus, tad uzmanības datu ievākšanas programmatūrai nācās ieviest dažus labojumus. Pirmkārt, bija jāmaina lietotāju identifikācijas metode – HTTP GET parametra vietā tiek izmantots UUID ģenerators un katram lietotājam tiek piekārtota nejauši izveidota UUID virkne. Otrkārt, lai varētu salīdzināt darbinieku saglabātos datus serverī un viņu sniegumu platformā, tad visu attēlu pelītes informācijas glabāšanas masīvā tiek ierakstīta arī lietotāja identifikācijas virkne. Visbeidzot, kad tiek apskatīti 45 attēli (40 parastie un 5 pārbaudes), tad tiek izvadīts paziņojums par to, ka uzdevums ir pabeigts, tīmekļa lapas struktūrā tiek pievienots *input* elements ar pelītes datiem.



att. 7.6. **Uzdevuma progress Amazon Mechanical Turk**

Publicējot uzdevumu, tiek attēlota informācija par uzdevuma izpildi un darbinieku progress. Pateicoties lielam skaitam darbinieku, Amazon Mechanical Turk ir ļoti ātrs veids kā

var iegūt nepieciešamo informāciju. Lai apskatītu 8860 tūkstošus attēlu, kas kopā aptuveni aizņem 17 stundas kopumā, ja pieņem, ka katra attēla ielāde aizņem 2 sekundes, šajā platformā bija nepieciešamas vien 6 stundas. Cenu nosaka pats uzdevuma izveidotājs, taču ir pieņemts, ka 10 centi minūtē ir samērā labs atalgojums. Klāt nāk arī dažādas komisijas, ko piemēro Amazon Mechanical Turk platforma.

Kaut arī daudzi darbinieki ir samērā augstu kvalificēti un pilda šāda veida uzdevumus jau samērā ilgu laiku, dažreiz problēmas izraisa aprīkojums. Daudziem nav pieejams pietiekami labs Interneta savienojums vai arī pats dators ir samērā lēns (WebGL lietotnes ir samērā prasīgas), tad var rasties datu kvalitātes problēmas. Konkrēti šī darba izstrādes laikā autors saskārās ar problēmu, ka daudzu darbinieku pelītes dati jaunu attēlu parādīšanās brīdī kādu laiku netiek ierakstīti pareizi un gan x, gan y koordinātes tiek atzīmētas ar vērtību 0. Šāda problēma netika novērota ne testējot risinājumu, ne dodot uzdevumu izpildīt brīvprātīgajiem. Tāpēc atlika vien datu apstrādes procesā šos datus izfiltrēt.

7.4. Datu formāta pārveidošana

SalGAN ievaddatos sagaida datus konkrētā formātā. Viena daļa datu ir paši attēli, kuri tiek analizēti. Tos nav nepieciešams īpaši apstrādāt, jo SalGAN datu priekšapstrādes posmā attēliem maina izmērus tā, lai tie iederētos neironu tīkla struktūrā. Attēli ir tikai jāsadala attiecīgi pa mapītēm. Otra daļa sastāv no datu struktūras, kas satur informāciju par cilvēku uzmanības tendencēm konkrētajā attēlā. Datu struktūra satur attēla nosaukumu, kura apskatīšanas rezultātā tika iegūti konkrētie dati, attēla izmēru, katra cilvēka acu fiksācijas pozīciju masīvus un neapstrādātu skatiena pozīciju masīvus ar laika zīmogiem. Datnei, kas satur šo struktūru ir jābūt Matlab binārā faila (.mat) formātā. Trešā datu sastāvdaļa ir fiksācijas punktu temperatūras karte, kuru iegūst atzīmējot attēla izmēra (1280x720) matricā ar 1 tos punktus, kur tika reģistrēta fiksācija un ar 0 visus pārējos punktus. Pielietojot šai matricai Gausa filtru, tiek izveidota uzmanības temperatūras karte, kur katrs punkts ir ar vērtību no 0 līdz 1, kas norāda uz uzmanības intensitāti dotajā apgabalā. Arī uzmanības temperatūras kartei ir jābūt Matlab binārā faila formātā.

Criteria		Representative Algorithms				
		I-VT	I-HMM	I-DT	I-MST	I-AOI
Spatial	Velocity-based	X	X			
	Dispersion-based			X	X	
	Area-based					X
Temporal	Duration sensitive			X		X
	Locally adaptive		X	X	X	

att. 6.4.1 **Acu fiksāciju noteikšanas algoritmu salīdzinājums[11]**

Lai iegūtu datu struktūru par uzmanības tendencēm vajadzīgajā formātā, sākumā ir jāizmanto acu fiksāciju algoritmu, kas iegūst acu fiksāciju koordinātas no neapstrādātiem skatienu pozīcijas datiem un laika zīmogiem. Darba autors izvēlējās izmantot I-DT[11] algoritmu (implementācijas kods 8. pielikumā), kas ņem vērā gan fiksācijas ilgumu, gan arī attālumus starp acu skatiena punktiem (dispersiju). I-DT algoritma pamatideja ir tāda, ka datu kopas punkti ciklā tiek pievienoti pēc kārtas līdz tiek sasniegts laika slieksnis (starpība starp mazāko laika zīmogu un lielāko), kas šajā gadījumā ir 150ms, kas nosaka cik ir minimālais fiksācijas ilgums. Ja tajā laikā sakrātie punkti pārsniedz dispersijas slieksni (starpība starp mazākajām un lielākajām koordinātēm), tad tiek secināts ka novērotā parādība ir sakāde, nevis fiksācija un pirmais punkts secībā tiek izdzēsts no saraksta un process atkārtots. Šajā gadījumā dispersijas slieksnis tika noteikts 20px attālumā (tieši mazākā riņķa rādiuss). Ja tomēr dispersijas slieksnis vēl netika sasniegts, tad tiek pievienoti papildus punkti līdz slieksnis tiek sasniegts un to punktu vidū tiek noteikts fiksācijas centrs un fiksācija tiek pievienota atrasto fiksāciju masīvam. Fiksāciju masīvs tiek saglabāts JSON formāta failā. Ar citu Python programmu, autors apvienoja datus par attēlā iegūto uzmanības informāciju vienā struktūrā pēc SALICON publicētā formāta un saglabāja gatavos strukturētos datus Matlab binārā faila formātā ar *Scipy* bibliotēkas starpniecību.

Beigās atlika vienīgi izveidot fiksācijas punktu temperatūras karti. Tam nolūkam autors izmantoja Matlab kodu, kas nāca līdzi raksta *Website saliency* datu kopai un tika izmantots,

lai vizuāli attēlotu acu novērošanas datus. Izrādās, pielietojot tikai minimālus labojumus kodā (datu ieguves procesā, atšķirīga datu formāta dēļ) un palaižot kodu *Octave* vidē un saglabājot iegūtās kartes Matlab binära faila formātā, bija iespējams pārveidot datus nepieciešamajā formātā.

7.5. Datu kvalitātes novērtēšana

Pēc datu kopas izveides, ir noderīgi pārbaudīt tās kvalitātes. Datu kvalitāti var noteikt salīdzinot temperatūras kartes starp acu fiksācijas datiem, kas tika iegūti izmantojot autora izveidotu programmu un datiem, kas tika uzņemti *Webpage saliency* pētījuma rezultātā. Autors izvēlējās 3 mājaslapu attēlus – *gmarket.co.kr* (1. pielikums), *facebook.com* (2. pielikums) un *amazon.com* (3. pielikums). Var redzēt, ka atšķirības ir samērā krasas, taču ir arī līdzības. Vislielākā starpība ir starp temperatūras kartēm tieši *amazon.com* gadījumā. Var redzēt, ka daudzi autora eksperimenta dalībnieki bija vairāk ieinteresēti noskaidrot kāda tīmekļa lapa tiek demonstrēta, apskatot tīmekļa lapas logo, nekā *Webpage saliency* dalībnieki. *Facebook.com* gadījumā rezultāts ir daudz labāks, vispārīgā uzmanības tendence sakrīt, taču ne līdz galam. Savukārt, vislīdzīgākais rezultāts ir saskatāms *gmarket.co.kr* gadījumā. Kaut arī temperatūras kartes plankumu forma īsti nesakrīt (praktiski ļoti grūti panākt), uzmanības aktīvākie reģioni ir tie paši.

7.6. Secinājumi

Kaut arī rezultātā iegūtie dati nav pietiekami kvalitatīvi, lai varētu uztrenēt SalGAN līdz tik pat labam līmenim kā tas tika uztrenēts ar dabiskiem attēliem, tomēr tie spēj parādīt daļējas sakritības ar testa kopas temperatūras kartēm. Šie dati potenciāli var uzlabot modeli, taču nelielos apmēros.

8. SALGAN TRENĒŠANA AR JAUNIEM DATIEM

Šajā nodaļā autors apraksta SalGAN trenēšanas procesu ar jauniem datiem. Tiek aprakstīts trenēšanas process un izmantotais aprīkojums, kā arī iegūtie rezultāti.

8.1. Trenēšanas process un aprīkojums

Ir iespējamas dažādas pieejas modeļa trenēšanai. Parasti, kad tiek veidots jauns modelis, trenēšana notiek no nulles, kas nozīmē, ka modelis tiek inicializēts ar gadījuma svariem un trenēšanas procesa ietvaros svari tiek pielāgoti tā, lai iegūtu pēc iespējas mazāku zuduma funkciju. Taču arvien biežāk nākas sastapties ar pielāgošanas paņēmieniem. Daudzu zinātnisko rakstu neironu tīkli pamatā izmanto sastāvdaļas, kas ir samērā plaši izmantotas citos darbos un kurām jau ir pieejami modeļi ar iepriekš daļēji uztrenētiem slāņiem. Tā kā apakšējie slāņi, kas atbild par sīkām detaļām var būt koplietoti starp līdzīga tipa uzdevumiem un katram uzdevumam specifiski ir jāpielāgo tikai augšējie slāņi, tad pielāgošanas metode ļauj ietaupīt gan trenēšanas laiku, gan arī samazināt nepieciešamo datu daudzumu, lai uztrenētu tīklu. Tā kā autora izveidotā datu kopa ir samērā maza pēc mašīnmācīšanās standartiem un iepriekš uztrenētais modelis bija paredzēts līdzīgai fukncijai, tad samērā dabiski ir izvēlēties pielāgošanas stratēģiju.

Mašīnmācīšanās modeļu trenēšanai ir nepieciešams ļoti liels daudzums skaitļošanas jaudas. Tā kā autoram personīgi nepieder jaudīgs dators un nav cita veida kā varētu bez maksas piekļūt tādam, tad vienīgā izvēle atliek mākoņskaitļošana. Pateicoties dažādām atbalsta programmām autoram izdevās iegūt bezmaksas skaitļošanas kredītus Amazon Web Services platformā. Platformas ietvaros ir īpaša serveru klase, kas ir paredzēta mašīnmācīšanās modeļu trenēšanai – P3[43]. Šīs klases instances ir aprīkotas ar jaudīgām Nvidia Tesla V100 videokartēm, kas ir specifiski paredzētas mašīnmācīšanās modeļu trenēšanai. Autors izmantoja *p3.8xlarge* instances sava modeļa trenēšanai, kas nāk ar 4 Tesla V100 videokartēm, ar 64 GB VRAM, 32 procesora kodoliem un 244 GB RAM, kas maksā $12.24 stundā[43].

Mašīnmācīšanās modeļa trenēšanas process ir samērā vienkāršs. Vispirms ir pareizi jāievieto datu kopa, esošais modelis un trenēšanas programmatūru serverī. Pēc tam ir jāpārliecinās, ka iestatījumu failā ir norādīti pareizie ceļi uz datu kopu. Tad ir jāpalaiž SalGAN iebūvēto datu kopas priekšapstrādes programmu un visbeidzot ir jāpalaiž trenēšanas programma ar pareizajiem parametriem. Lai trenēšana notiktu pēc iespējas efektīvāk, ir

jāuzstāda daži *Theano* ietvara karogi. Tie var tikt uzstādīt izmantojot vides mainīgo *THEANO_FLAGS*[34], kuram piešķir visu karogu nosaukumu un vērtību sarakstu. Pirmais izmantojamais karogs ir *mode* un tam piešķir vērtību *FAST_RUN*, kas nozīmē, ka Theano mēģinās izmantot *C* valodas implementācijas dažādām funkcijām pēc iespējas biežāk. Nākamais karogs ir *device*, kas tiek uzstādīts ar vērtību *cuda*, kas nozīmē, ka trenēšanai tiks izmantota *Nvidia CUDA* tehnoloģija. *FloatX* karogam ir uzstādīta vērtība *float32*, kas norāda Visbeidzot tiek uzstādīts *optimizer_including* karogs ar vērtību *cudnn*, kas parāda kādu optimizācijas tehnoloģiju izmantot trenēšanas procesā[35]. Pēc neironu tīkla ielādes atliek vien gaidīt.

```
23%|2| 23/100 [03:06<10:23,  8.10s/it]Epoch: 23  train_loss-> ('0.6965724780009344', '0.6932639731810644', '0.08261646932134262')
24%|2| 24/100 [03:14<10:14,  8.09s/it]Epoch: 24  train_loss-> ('0.6965637826002561', '0.6932801076999078', '0.07884256025919548')
25%|2| 25/100 [03:25<10:17,  8.24s/it]Epoch: 25  train_loss-> ('0.6952861455770639', '0.6932483590566195', '0.07538144892224899')
26%|2| 26/100 [03:33<10:08,  8.22s/it]Epoch: 26  train_loss-> ('0.6981006356386038', '0.6933479813429025', '0.07213443660965332')
27%|2| 27/100 [03:41<09:58,  8.20s/it]Epoch: 27  train_loss-> ('0.6969290467408987', '0.6932634711265564', '0.06874044354145344')
28%|2| 28/100 [03:50<09:52,  8.22s/it]Epoch: 28  train_loss-> ('0.69635232595297', '0.6932457823019761', '0.06603663887542027')
29%|2| 29/100 [03:58<09:42,  8.21s/it]Epoch: 29  train_loss-> ('0.696685288961117', '0.6932099782503568', '0.06376443660029998')
30%|3| 30/100 [04:05<09:33,  8.20s/it]Epoch: 30  train_loss-> ('0.6958166062831879', '0.6932026056142954', '0.06139747691942066')
31%|3| 31/100 [04:14<09:26,  8.21s/it]Epoch: 31  train_loss-> ('0.6963152954211602', '0.6931943893432617', '0.05958732604407347')
32%|3| 32/100 [04:22<09:17,  8.20s/it]Epoch: 32  train_loss-> ('0.6965467219169323', '0.6931850153666276', '0.05712226377083705')
33%|3| 33/100 [04:30<09:08,  8.18s/it]Epoch: 33  train_loss-> ('0.6969859622992002', '0.6932077889259045', '0.05543026528679407')
34%|3| 34/100 [04:38<09:01,  8.20s/it]Epoch: 34  train_loss-> ('0.6960667807322282', '0.6931912417595203', '0.0536057072190138')
35%|3| 35/100 [04:46<08:52,  8.19s/it]Epoch: 35  train_loss-> ('0.6958377865644602', '0.6931653458338517', '0.05187752183813315')
36%|3| 36/100 [04:54<08:43,  8.18s/it]Epoch: 36  train_loss-> ('0.6956925896497873', '0.693144502548071', '0.05044148575801155')
37%|3| 37/100 [05:03<08:35,  8.19s/it]Epoch: 37  train_loss-> ('0.6957996029120225', '0.6931434732217056', '0.04881068147145785')
38%|3| 38/100 [05:10<08:27,  8.18s/it]Epoch: 38  train_loss-> ('0.6958856720190781', '0.6931527944711539', '0.04763405970655954')
39%|3| 39/100 [05:18<08:18,  8.17s/it]Epoch: 39  train_loss-> ('0.6958085436087388', '0.6931361395579118', '0.04624290406130827')
40%|4| 40/100 [05:29<08:14,  8.24s/it]Epoch: 40  train_loss-> ('0.6963167763673342', '0.6931326779035422', '0.04475659461548695')
41%|4| 41/100 [05:37<08:05,  8.22s/it]Epoch: 41  train_loss-> ('0.6954020009591029', '0.6931219215576465', '0.04359256074978754')
42%|4| 42/100 [05:45<07:56,  8.21s/it]Epoch: 42  train_loss-> ('0.695692261824241', '0.6931259953058683', '0.04258683395500366')
43%|4| 43/100 [05:53<07:48,  8.22s/it]Epoch: 43  train_loss-> ('0.695579680112692', '0.69311546591612', '0.04147285896425064')
44%|4| 44/100 [06:01<07:39,  8.21s/it]Epoch: 44  train_loss-> ('0.6954183922364161', '0.6931172265456274', '0.04036271471816759')
45%|4| 45/100 [06:09<07:31,  8.21s/it]Epoch: 45  train_loss-> ('0.6955192455878625', '0.6931177240151626', '0.0393923527489488')
46%|4| 46/100 [06:17<07:23,  8.21s/it]Epoch: 46  train_loss-> ('0.6952734016455137', '0.6931119492420783', '0.03868056647479534')
47%|4| 47/100 [06:25<07:14,  8.20s/it]Epoch: 47  train_loss-> ('0.6956612055118268', '0.6931153879715846', '0.03779632368913063')
48%|4| 48/100 [06:33<07:06,  8.20s/it]Epoch: 48  train_loss-> ('0.6951004427212936', '0.6931042212706345', '0.0366357436690193')
```

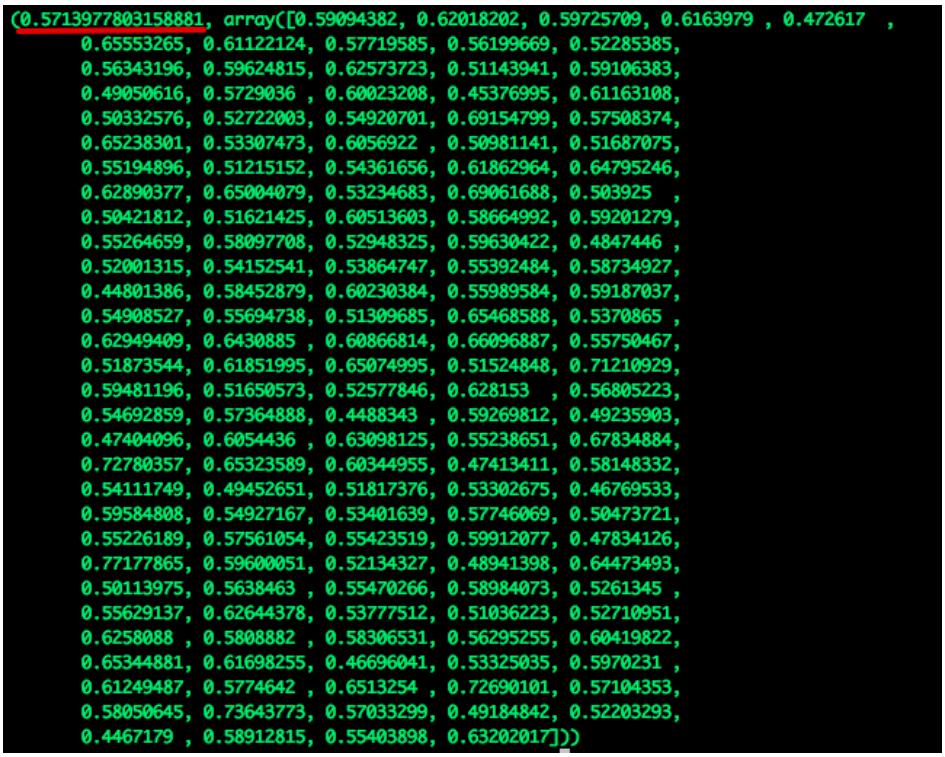
att. 8.1. **SalGAN trenēšanas process**

Šajā attēlā ir redzams trenēšanas process un tā laikā saņemtā informācija par statusu. Tīkla statuss ir izdrukāts iekavās ar nosaukumu *train_loss*. Pirmais parametrs iekavās ir ģeneratora zuduma funkcijas vērtība. Otrais parametrs ir diskriminatora zuduma funkcijas vērtība. Savukārt, trešais parametrs ir tīkla kopējā zuduma funkcija. Var redzēt, ka kopējā zuduma vērtība samazinās, tātad tīkls tiek trenēts bez acīmredzamām kļūdām.

8.2. Trenēšanas rezultāti

Kaut arī modeļu trenēšana parasti ir laikietilpīgs process, izmantojot jau iepriekš trenētu modeli un jaudīgu trenēšanas iekārtu, trenēšanas laiks bija vien 40 minūtes. Tajā laikā modelis

tika atjaunots 300 reizes. Pēc trenēšanas jaunais modelis tika pārnests no trenēšanas servera uz lokālo datoru, kur līdzīgi kā ar SALICON datu kopu trenēts modelis, tas uzģenerēja temperatūras kartes pārbaudes datu kopai.



att. 8.1. **SalGAN rezultāts pēc trenēšanas ar autora datu kopu**

Pēc attēla var secināt, ka rezultāts ir pasliktinājies par aptuveni 0.05 AUC, taču tas joprojām ir labāks par gadījuma rezultātiem. Šāds rezultāts nav pārsteidzošs, ņemot vērā datu kopas kvalitāti un izmērus. Taču veicot visus nepieciešamos soļus, lai varētu izveidot jaunu datu kopu un uztrenētu modeli, nākamais mēģinājums var tikt veikts ātrāk, jo datu ievākšanai un apstrādei nepieciešamā programmatūra tika izveidota.

8.3. Secinājumi

Kaut arī autoram neizdevās uzlabot sava modeļa sniegumu, izdevās noskaidrot, ka ar mazu budžetu (EUR 150) nav iespējams izveidot pietiekami labu datu kopu izmantojot Amazon Mechanical Turk servisu un rezultāta uzlabošanai ir nepieciešami daudz lielāki resursi.

REZULTĀTI

Darba rezultātā tika izveidota sistēma, kas spēj simulēt *saliency* temperatūras kartes ar 67% precizitāti un attēlot datus gandrīz reālā laikā. Sistēma atbilst nodefinētajiem sistēmas mērķiem un spēj veikt savas funkcijas. Darba ietvaros veiksmīgi tika izveidota visa nepieciešamā infrastruktūra mašīnmācīšanās modeļa turpmākai uzlabošanai un tika noskaidrots, ka ir nepieciešams piesaistīt papildus finansējumu turpmākai sistēmas kvalitātes uzlabošanai.

SECINĀJUMI

Tā kā uzmanība ir kļuvusi par ļoti svarīgu tematu mūsdienu sabiedrībā, tās pētniecība kļust arvien svarīgāka. Izstrādājot dažādus produktus arvien svarīgāks kļūst produkta dizains, kas pievērš uzmanību. Līdz ar to rodas interese simulēt cilvēka uzmanību, lai varētu paredzēt tās tendences. Tīmekļa lapas nav izņēmums.

Izpētot vizuālās uzmanības teoriju, autors secināja, ka ir nepieciešams fokusēties tieši uz neapzināto uzmanību, jo tā ir labāk izpētīta. Viena no raksturīgākajām metrikām ir *saliency*, kas nosaka cik pievilcīgs ir konkrēts attēla apgabals un cik liela iespēja, ka uzmanība tiks pievērsta tieši tam. Tika salīdzināti vairāki veidi kā ir iespējams vizuāli attēlot *saliency* un temperatūras kartes autoram šķita par vislabāko attēlošanas veidu.

Pēc teorijas izpētes, darba autors izpētīja *saliency* simulēšanas literatūru un secināja, ka tieši jaunākie mašīnmācīšanās modeļi ir visefektīvākie. Izpētes procesā tika atrasta SALICON datu kopa, kas ir 10000 attēlu datu kopa ar *saliency* informāciju. Izmantojot šo datu kopu tika uztrenēti daudzi mašīnmācīšanās modeļi, kas spēj simulēt *saliency* informāciju attēlos. Autors izvēlējās SalGAN tīklu savam risinājumam. Pēc SalGAN izvēles, tā iepriekš trenēts modelis tika pārbaudīts ar *Webpage Saliency* datu kopu, kas satur acu novērošanas aparatūras datus no 11 dalībniekiem apskatot 149 tīmekļa lapas. Izrādījās, ka SalGAN ar SALICON datu kopu trenēts modelis spēj paredzēt uzmanību ar labāku precizitāti par gadījuma rezultātiem.

Pēc modeļa izvēles, darba autors sāka sistēmas izveidi. Vispirms tika izvēlēta sistēmas arhitektūra, kas sastāv no Django servera un Google Chrome tīmekļa pārlūka spraudņa. Abas komponentes izveidojot un veiksmīgi palaižot darba autors konstatēja, ka darba uzdevums tika izpildīts.

Pēc risinājuma izveides autors mēģināja noskaidrot vai ar viņam pieejamajiem resursiem (EUR 150) ir iespējams uzlabot sistēmas rezultātu. Izpētot SALICON rakstu, darba autors noskaidroja kā var patstāvīgi iegūt uzmanības datus. Vispirms tika iegūti 1000 mājaslapu attēli. Pēc tam tika izveidota speciāla uzmanības datu iegūšanas programma, kas tika nopublicēta Amazon Mechanical Turk platformā un kurā piedalījās arī brīvprātīgie. Iegūstot datus un salīdzinot tos ar pārbaudes datu kopu tika secināts, ka dati nav pietiekami kvalitatīvi, lai veiktu ievērojamu modeļa uzlabojumu. Veicot SalGAN trenēšanu ar autora izveidoto datu kopu netika iegūts uzlabojums un autors secināja, ka ar viņam pieejamajiem resursiem pietiekami kvalitatīvu datu kopu šim uzdevumam nav iespējams izveidot.

Šis darbs ir tikai pirmais veiksmīgais solis risinājuma izveidē. Nākotnē, lai uzlabotu rezultātu ir iespējams izveidot jaunu datu kopu, ar lielāku skaitu piemēru un labāku kvalitāti. Pietam, ir iespējams izveidot jaunu SalGAN versiju ar jaunāku tīklu izmantošanu arhitektūrā.

Turpmāk šo risinājumu var pielāgot arī citiem pielietojumiem, piemēram, plakātu un 3D modeļu uzmanības temperatūras karšu veidošanai. Līdz ar cilvēku uzmanības pētījumu un mašīnmācīšanās tehnoloģiju attīstīšanos varēs attīstīt arī šo risinājumu.

IZMANTOTĀ LITERATŪRA UN AVOTI

1. P. Delmas “Gaussian Filtering”, 2010.
2. Q. Zhao, M. Jing, S. Huang, J. Duan “SALICON: Saliency in Context”, 2015.
3. “3 Free Tools to See a Heat Map of Your Website Visitors” [Tiešsaiste] Pieejams: https://growtraffic.com/blog/2015/03/3-free-tools-see-heat-map-your-website-visitors [Skatīts: 28.05.2018]
4. “Eyetracking scanpath and heatmap” [Tiešsaiste] Pieejams: https://multilingual.com/combined_eyetracking/ [Skatīts: 28.05.2018]
5. L. Itti, C. Koch, E. Niebur “A model of saliency-based visual attention for rapid scene analysis”, 1998.
6. T. Judd, F. Durand, A. Torralba “A benchmark of Computational Models of Saliency to Predict Human Fixations”, 2012.
7. Q. Zhao, M. Jing, S. Wang, J. Xu, M. Kankanhalli “Predicting human gaze beyond pixels”, 2014.
8. “Amazon Mechanical Turk” [Tiešsaiste] Pieejams: https://www.mturk.com/ [Skatīts: 28.05.2018]
9. Z.Bylinskii, T. Judd, A. Borji, L. Itti, F. Durand “MIT Saliency Benchmark”, 2012.
10. “AWS Elastic Beanstalk” [Tiešsaiste] Pieejams: https://aws.amazon.com/elasticbeanstalk/ [Skatīts: 28.05.2018].
11. D. Salvucci, J. Goldberg “Identifying Fixations and Saccades in Eye-Tracking Protocols”, 2000.
12. Q. Zhao, C. Shen “Webpage Saliency”, 2014.
13. “SALICON Saliency Evaluation” [Tiešsaiste] Pieejams: https://github.com/NUS-VIP/salicon-evaluation [Skatīts: 28.05.2018].
14. “Pixels” [Tiešsaiste] Pieejams: https://klart.io/pixels [Skatīts: 28.05.2018]
15. “Feng-GUI” [Tiešsaiste] Pieejams: https://feng-gui.com/ [Skatīts: 28.05.2018]
16. Y. Pinto, A. van der Leij, I. Sligte, V. Lamme, H. Scholte “Bottom-up and top-down attention are independent”, 2013.
17. “React” [Tiešsaiste] Pieejams: https://reactjs.org/ [Skatīts: 28.05.2018]
18. “Statcounter GlobalStats” [Tiešsaiste] Pieejams: http://gs.statcounter.com/ [Skatīts: 28.05.2018]
19. “Google Chrome Developer Getting Started Tutorial” [Tiešsaiste] Pieejams: https://developer.chrome.com/extensions/getstarted [Skatīts: 28.05.2015]
20. “AngularJS” [Tiešsaiste] Pieejams: https://angularjs.org/ [Skatīts: 28.05.2018]

21. “Vue.js” [Tiešsaiste] Pieejams: https://vuejs.org/ [Skatīts: 28.05.2018]
22. “TypeScript” [Tiešsaiste] Pieejams: https://www.typescriptlang.org/ [Skatīts: 28.05.2018]
23. “CoffeeScript” [Tiešsaiste] Pieejams: https://coffeescript.org/ [Skatīts: 28.05.2018]
24. “jQuery” [Tiešsaiste] Pieejams: https://jquery.com/ [Skatīts 28.05.2018]
25. “html2canvas” [Tiešsaiste] Pieejams: https://html2canvas.hertzen.com/ [Skatīts 28.05.2018]
26. “Lasagne GitHub” [Tiešsaiste] Pieejams: https://github.com/Lasagne/Lasagne [Skatīts: 28.05.2018]
27. “Math.js” [Tiešsaiste] Pieejams: http://mathjs.org/ [Skatīts: 28.05.2018]
28. “TensorFlow.js” [Tiešsaiste] Pieejams: https://js.tensorflow.org/ [Skatīts: 28.05.2018]
29. “CamanJS” [Tiešsaiste] Pieejams: http://camanjs.com/ [Skatīts: 28.05.2018]
30. “Data-Driven Documents” [Tiešsaiste] Pieejams: https://d3js.org/ [Skatīts: 28.05.2018]
31. “three.js” [Tiešsaiste] Pieejams: https://threejs.org/ [Skatīts: 28.05.2018]
32. “PixiJS” [Tiešsaiste] Pieejams: http://www.pixijs.com/ [Skatīts: 28.05.2018]
33. A. Borji, H. Tavakoli, D. Sihite, L. Itti “Analysis of Scores, Datasets and Models in Visual Saliency Prediction”, 2013.
34. “Theano Configuration Settings and Compiling Modes” [Tiešsaiste] Pieejams: http://deeplearning.net/software/theano/tutorial/modes.html [Skatīts: 28.05.2018]
35. “Optimizing Recurrent Neural Networks in cuDNN 5” [Tiešsaiste] Pieejams: https://devblogs.nvidia.com/optimizing-recurrent-neural-networks-cudnn-5/ [Skatīts: 28.05.2018]
36. “Decentralised Attention Economies for the Web 3.0” [Tiešsaiste] Pieejams: https://maciejolpinski.gitbooks.io/decentralised-attention-economies-for-the-web-3-0/content/chapter1.html [Skatīts: 28.05.2018]
37. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio “Generative Adversarial Nets”, 2014.
38. J. Pan, E. Sayrol, X. Giro-i-Nieto, J. Torres, C. Ferrer, K. McGuiness, N. O’Connor “SalGAN: Visual Saliency Prediction with Generative Adversarial Networks”, 2017.
39. M. Mathieu, C. Couprie, Y. LeCunn “Deep Multi-Scale Video Prediction Beyond Mean Square Error”, 2016.
40. L. Yang, S. Chou, Y. Yang “MidiNet: A Convolutional Generative Adversarial Network for Symbolic-domain Music Generation, 2017.
41. W. Fedus, I. Goodfellow, A. Dai “MaskGAN: Better Text Generation via Filling in the _____”, 2018.

42. K. Simonyan, A. Zisserman “Very Deep Convolutional Networks for Large-scale Image Recognition”, 2015.
43. “Amazon EC2 P3 Instances” [Tiešsaiste] Pieejams: https://aws.amazon.com/ec2/instance-types/p3/ [Skatīts: 28.05.2018].
44. I. Laicāne, D. Dižpētere, I. Lācis “Acu kustības simbolu skenēšana”, 2013.
45. A. Radford, L. Metz “Unsupervised Representational Learning with Deep Convolutional Generative Adversarial Networks”, 2016.
46. “Url2PNG” [Tiešsaiste] Pieejams: https://www.url2png.com/ [Skatīts: 28.05.2018]
47. “Puppeteer GitHub” [Tiešsaiste] Pieejams: https://github.com/GoogleChrome/puppeteer [Skatīts: 28.05.2018]
48. “University of Kent Eyetracking lab” [Tiešsaiste] Pieejams: https://www.kent.ac.uk/psychology/research/facilities/eyetrack.html [Skatīts: 28.05.2018]
49. M. Carrasco “Visual attention: The past 25 years”, 2011
50. Y. Pinto, A. Leij, I. Sligte, V. Lamme, H. Scholte “Bottom-up and top-down attention are independent”, 2013.
51. “Mat4py package” [Tiešsaiste] Pieejams: https://pypi.org/project/mat4py/ [Skatīts: 28.05.2018]
52. “WebGL wiki” [Tiešsaiste] Pieejams: https://www.khronos.org/webgl/wiki/Main_Page [Skatīts: 28.05.2018]
53. “Surge” [Tiešsaiste] Pieejams: https://surge.sh/ [Skatīts: 28.05.2018]

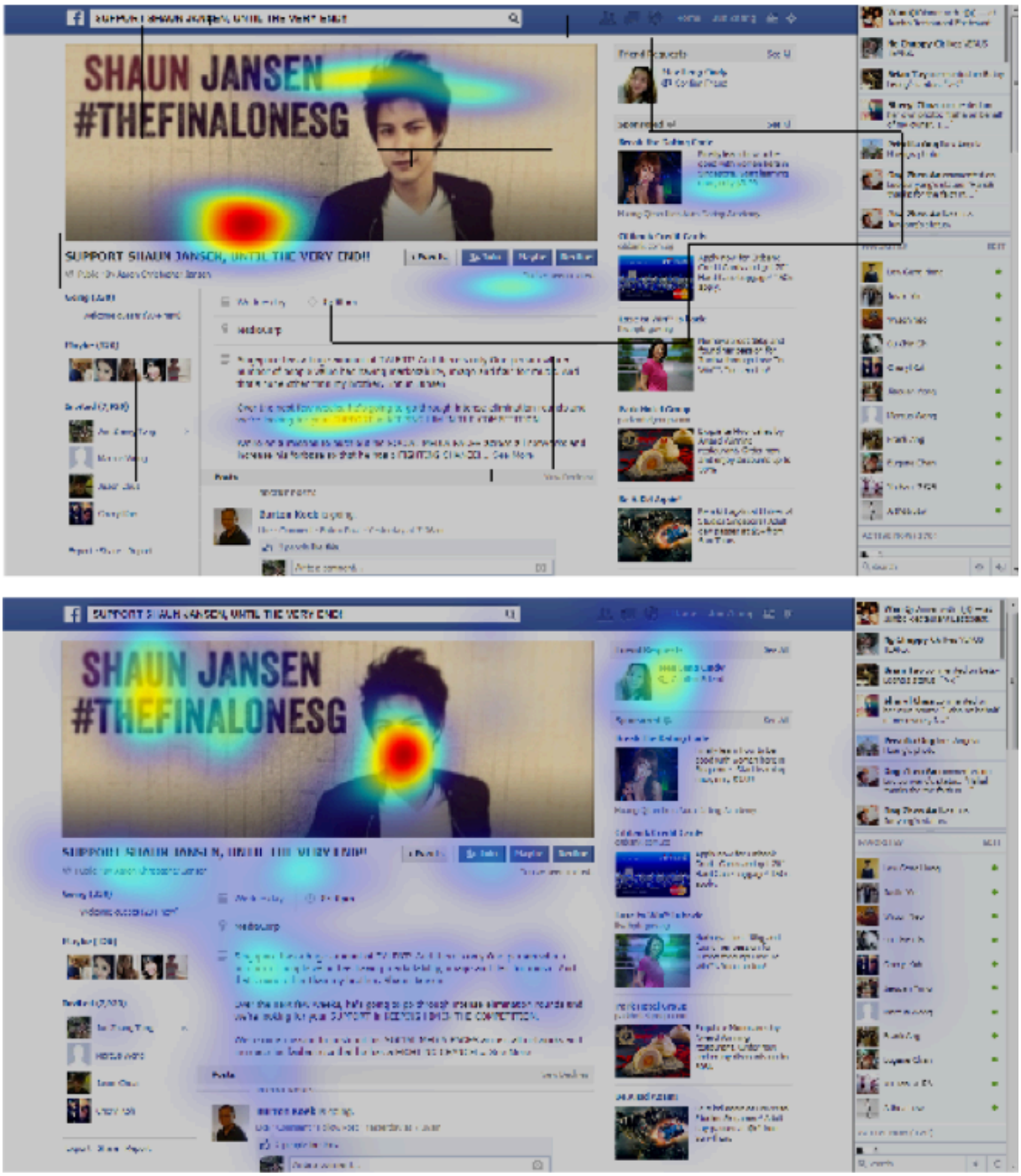
PIELIKUMI

1. PIELIKUMS. GMARKET.CO.KR KARŠU SALĪDZINĀJUMS

Šajā attēlā var redzēt salīdzinājumu starp iegūto temperatūras karti no datu ievākšanas eksperimenta (augšā) un testa kopas temperatūras karti (apakšā) tīmekļa lapai *gmarket.co.kr*.

2. PIELIKUMS FACEBOOK.COM KARŠU SALĪDZINĀJUMS

Šajā attēlā var redzēt salīdzinājumu starp iegūto temperatūras karti no datu ievākšanas eksperimenta (augšā) un testa kopas temperatūras karti (apakšā) tīmekļa lapai *facebook.com*.



3. PIELIKUMS AMAZON.COM KARŠU SALĪDZINĀJUMS

Šajā attēlā var redzēt salīdzinājumu starp iegūto temperatūras karti no datu ievākšanas eksperimenta (augšā) un testa kopas temperatūras karti (apakšā) tīmekļa lapai *amazon.com*.

4. PIELIKUMS SPRAUDŅA DJANGO SERVERA KODS

Šajā pielikumā ir ievietots daļējs spraudņa Django servera kods.

```
from django.http import HttpResponse
from django.views.decorators.csrf import csrf_exempt
from uuid import uuid1
from subprocess import call
import base64
import os


@csrf_exempt
def send_screen(request):
    filename = uuid1().hex + ".jpg"
    with open("saliency/images/" + filename, "w") as f:
        for chunk in request.FILES["file"].chunks():
            f.write(chunk)
        f.close()
    os.environ["THEANO_FLAGS"] =
"mode=FAST_RUN,floatX=float32,lib.cnmem=1,optimizer_including=cudnn"
    call(["python",  "predict.py"])
    call(["rm", "-R", "saliency/images/" + filename])
    with open("saliency/saliency/" + filename, "rb") as f:
        encoded_image = base64.b64encode(f.read())
        f.close()
    return HttpResponse(encoded_image)
```

5. PIELIKUMS SPRAUDŅA JAVASCRIPT KODS

Šajā pielikumā ir ievietots spraudņa priekšgala sistēmas JavaScript kods.

```
function sendScreen() {
    html2canvas(document.body).then(function(canvas) {
      var blob = b64toBlob(canvas.toDataURL().substr(22));
      var data = new FormData();
      data.append("file", blob);
      $.ajax({
          url: 'http://localhost:8001/sendScreen/',
          data: data,
          cache: false,
          contentType: false,
          processData: false,
          method: 'POST',
          success: function(data){
            var image = new Image();
            image.src = "" + data;
            image.style = "position: absolute;\n" +
                "width: 100%;\n" +
                "height: 100%;\n" +
                "top: 0; \n" +
                "left: 0;\n" +
                "right: 0;\n" +
                "bottom: 0;\n" +
                "opacity: 0.85;\n" +
                "z-index: 2;\n";
            document.body.appendChild(image);
          }
      });
    });
  }


function b64toBlob(b64Data, contentType, sliceSize) {
        contentType = contentType || '';
        sliceSize = sliceSize || 512;

        var byteCharacters = atob(b64Data);
        var byteArrays = [];

        for (var offset = 0; offset < byteCharacters.length; offset += sliceSize) {
            var slice = byteCharacters.slice(offset, offset + sliceSize);

            var byteNumbers = new Array(slice.length);
            for (var i = 0; i < slice.length; i++) {
                byteNumbers[i] = slice.charCodeAt(i);
            }

            var byteArray = new Uint8Array(byteNumbers);

            byteArrays.push(byteArray);
        }

      var blob = new Blob(byteArrays, {type: contentType});
      return blob;
}
```

6. PIELIKUMS UZMANĪBAS DATU IEGŪŠANAS PROGRAMMAS SERVERA KODS

Šajā pielikumā ir ievietots daļējs uzmanības datu iegūšanas programmas servera pirmkods.

```
# -*- coding: utf-8 -*-
from __future__ import unicode_literals
from django.http import JsonResponse
import random
import base64
import json
from django.views.decorators.csrf import csrf_exempt
from os import listdir, mkdir, path


@csrf_exempt
def image_request(request):
    if not path.isdir("mouse_data/" + request.POST["id"]):
        mkdir("mouse_data/" + request.POST["id"])
        mkdir("mouse_data/qa/" + request.POST["id"])
    #Decide if person is still in practice mode or not
    if request.POST["practice"] == "false":
        print(request.POST["id"])
        qa_files = [f for f in listdir("./qa_images")]
        if request.POST["img_name"] in qa_files:
            subdirectory = "qa/"
        else:
            subdirectory = ""
            #Update manifest file
            with open("manifest.txt", "r") as f:
                img_data = json.loads(f.read())
                try:
                    img_data[request.POST["img_name"]] -= 1
                    if img_data[request.POST["img_name"]] == 0:
                        img_data.pop(request.POST["img_name"], None)
                except:
                    pass
                f.close()
            with open("manifest.txt", "w") as f:
                f.write(json.dumps(img_data))
                f.close()
        with open("mouse_data/" + subdirectory + request.POST["id"] + "/" +
request.POST["img_name"] + ".json", "w") as f:
            f.write(request.POST["mouse"])
            f.close()
        #Send a QA image or real image
        if request.POST["qa"] == "true":
            qa_files = [f for f in listdir("./qa_images")]
            qa_collected = [f[:-5] for f in listdir("mouse_data/qa/" +
request.POST["id"]) if path.isfile("mouse_data/qa/" + request.POST["id"] + "/" +
f)]
            qa_to_see = [x for x in qa_files if x not in qa_collected]
            selected_file = random.choice(qa_to_see)
            with open("qa_images/" + selected_file, "rb") as f:
                encoded_image = base64.b64encode(f.read())
        else:
            with open("manifest.txt", "r") as f:
                real_images = json.loads(f.read())
                data_collected = [d[:-5] for d in listdir("mouse_data/" +
request.POST["id"]) if path.isfile("mouse_data/" + request.POST["id"] + "/" + d)]
                images_to_see = [a for a in real_images if a not in data_collected]
            if len(images_to_see) == 0:
                return JsonResponse({"Done": True})
            selected_file = random.choice(images_to_see)
```

```
            with open("images/" + selected_file, "rb") as f:
                encoded_image = base64.b64encode(f.read())
    else:
        files = [f for f in listdir("./practice_images")]
        selected_file = random.choice(files)
        with open("practice_images/" + selected_file, "rb") as f:
            encoded_image = base64.b64encode(f.read())
    return JsonResponse({"image": encoded_image, "filename": selected_file})
```

7. PIELIKUMS UZMANĪBAS DATU IEGŪŠANAS PROGRAMMAS PRIEKŠGALA SISTĒMAS KODS

Šajā pielikumā tika pievienots uzmanības datu iegūšanas priekšgala sistēmas pilns kods.

```
<link crossorigin="anonymous"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.1.0.min.js" integrity="sha256-
cCueBR6CsyA4/9szpPfrX3s49M9vUU5BgtiJj06wt/s=" crossorigin="anonymous"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/pixi.js/4.7.3/pixi.min.js"></script>
<script src="pixi-layers.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa"
crossorigin="anonymous"></script>
<script>
    $(function(){
      let images_seen = 0;
      let awaiting_input = false;
      let image_data;
      let image_name;
      let first_image = true;
      let qa_images = 0;
      let practice_images = 5;
      let hostname = "http://eyeserver-master.8panzwsc2m.eu-west-
1.elasticbeanstalk.com/";
      const app = new PIXI.Application({width: 1280, height: 720});
      const RADIUSES = [20, 40, 80, 160, 320];
      const CIRCLES = 5;
      document.body.appendChild(app.view);
      let imageElement = document.createElement("img");
      let url = new URL(window.location.href);
      let id = url.searchParams.get("id");

      app.stage = new PIXI.display.Stage();
      app.stage.interactive = true;

      app.stage.click = function(clk) {
        if(awaiting_input) {
            awaiting_input = false;
            setupImage(image_data);
        }
        else if(first_image) {
          first_image = false;
          app.stage.removeChild(instructions, please_read);
          click_to_continue.x = 350;
          click_to_continue.y = 300;
          image_info.x = 450;
          image_info.y = 250;

          $.post(hostname + "requestImage/", {practice: true, id: id}, function
(data) {
            setupImage(data);
          });
        }
      };

      let please_read = new PIXI.Text("PLEASE READ", {fontFamily: "Arial",
fontSize: 30, fill: 0xffffff});

      let instructions = new PIXI.Text("The purpose of this experiment is to record
the attention " +
```

```
          "of users seeing a website in the first 5 seconds. During this experiment
You will view up to 100 images of different websites." +
          "\n\nData for every image is being saved after each viewing, so don't
worry if you have to restart or you get bored. You can " +
          "opt out at any moment. After 5 seconds of viewing, the picture
disappears and you have to click the primary mouse button " +
          "to load the next image. " +
          "You can rest between images if you wish, so no time pressures. First You
will be presented with 5 practice images, for which your performance" +
          "will not be recorded." +
          "\n\nThe task for each image is to move your mouse cursor to the areas of
the image you wish to explore. " +
          "Try not to read text, as it will slow you down considerably. " +
          "It is recommended to use a mouse and not a touchpad, since mouses are
faster. " +
          "Also, do not jerk your mouse too fast. Quick & smooth is the way to go.
" +
          "The collected data is going to be used to train a neural network to
predict human attention in webpages. " +
          "All collected data is anonymous, unlike Your Facebook profile.",
{fontFamily: "Arial", fontSize: 24, fill: 0xffffff, wordWrap: true, wordWrapWidth:
1000});

      let click_to_continue = new PIXI.Text("CLICK ON THE SCREEN TO CONTINUE",
{fontFamily: "Arial", fontSize: 30, fill: 0xffffff});
      let image_info = new PIXI.Text("", {fontFamily: "Arial", fontSize: 30, fill:
0xffffff});

      instructions.x = 140;
      instructions.y = 140;
      click_to_continue.x = 350;
      click_to_continue.y = 650;
      please_read.x = 530;
      please_read.y = 50;

      app.stage.addChild(please_read, instructions, click_to_continue, image_info);

      function setupImage(data) {
        imageElement.src = "" + data.image;
        image_name = data.filename;
        let texture = PIXI.Texture.from(imageElement);
        let background = new PIXI.Sprite(texture);
        let mouse_x = 0;
        let mouse_y = 0;
        let recordings = [];
        let start_time;
        let picture_interval;
        let snap_inteval;
        let image = [];
        let circle = [];
        let container = [];

        background.filters = [new PIXI.filters.BlurFilter(0.001, 0, 1/10)];
        app.stage.addChild(background);

        //Create image sprites, circles and containers
        for(let i = 0; i < CIRCLES; i++) {
          image.push(new PIXI.Sprite(texture));
          circle.push(new PIXI.Sprite.fromImage('centerblur.png'));
          container.push(new PIXI.Container());
        }

        for(let i = CIRCLES - 1; i >= 0; i--) {
          circle[i].height = RADIUSES[i] * 2;
          circle[i].width = RADIUSES[i] * 2;
          app.stage.addChild(circle[i]);

          if(i > 0) {
            image[i].filters = [new PIXI.filters.BlurFilter(0.001, 0, 1/(2*i))];
```

```
          }

          container[i].addChild(image[i]);
          app.stage.addChild(container[i]);
          container[i].mask = circle[i];
        }

        //Move circles
        app.stage.mousemove = function(mouse) {
          mouse_x = mouse.data.global.x;
          mouse_y = mouse.data.global.y;
          for(let i = 0; i < CIRCLES; i++) {
            circle[i].x = mouse_x - Math.floor(circle[i].width / 2);
            circle[i].y = mouse_y - Math.floor(circle[i].height / 2);
          }
        };

        start_time = Date.now();

        //Record mouse data
        snap_interval = setInterval(function() {
          recordings.push({x: mouse_x, y: mouse_y, timestamp: Date.now() -
start_time});
        }, 10);

        //Manage loading next image
        picture_interval = setInterval(function() {
          clearInterval(picture_interval);
          clearInterval(snap_inteval);

          for(let i = 0; i < CIRCLES; i++) {
              app.stage.removeChild(container[i]);
            }
          app.stage.removeChild(background);

          $.post(hostname + "requestImage/", {mouse: JSON.stringify(recordings),
img_name: image_name,
            practice: practice_images > 0, qa: qa_images === 0, id: id},
function(data) {
            if(practice_images > 0) {
              practice_images--;
              if(practice_images > 0) {
                image_info.text = practice_images + " practice images left to see";
              }
              else {
                image_info.text = "Now get ready for the real deal";
              }
            }
            else {
              images_seen++;
              image_info.text = "You have seen " + images_seen + " image(s)
already!";
              if(qa_images === 0) {
                qa_images = 9;
              }
              else {
                qa_images--;
              }
            }

            awaiting_input = true;
            image_data = data;
          });
        }, 5000);
      }
    });
</script>
```

8. PIELIKUMS I-DT ALGORITMA IMPLEMENTĀCIJA

Šajā pielikumā tika pievienots autora veidotas I-DT algoritma implementācijas kods.

```
from os import listdir, path
import json

dirs = [f for f in listdir("./mouse_data") if f != "qa" and
path.isdir("./mouse_data/qa/" + f)]

DURATION_THRESHOLD = 150 #Average value mentioned in the paper
DISPERSION_THRESHOLD = 20 #Size of the circle

for dir in dirs:
    files = [f for f in listdir("./mouse_data/qa/" + dir) if
path.isfile("./mouse_data/qa/" + dir + "/" + f) and f != ".DS_Store"]
    for file in files:
        with open("./mouse_data/qa/" + dir + "/" + file) as f:
            data = json.loads(f.read())
            if "mouse" in data:
                data = data["mouse"]
            data = [w for w in data if int(w["x"]) > 0 and int(w["y"]) > 0 and
int(w["y"]) <= 720 and int(w["x"]) <= 1280]
            f.close()
        fixations = []
        while not data == []:
            window_x = []
            window_y = []
            duration = 0
            i = 0
            start_duration = data[0]["timestamp"]
            while duration < DURATION_THRESHOLD and i < len(data):
                window_x.append(data[i]["x"])
                window_y.append(data[i]["y"])
                duration = data[i]["timestamp"] - start_duration
                i += 1
            dispersion = (max(window_x) - min(window_x)) + (max(window_y) -
min(window_y))
            #Add more points to the fixation
            if dispersion <= DISPERSION_THRESHOLD and i < len(data):
                while dispersion <= DISPERSION_THRESHOLD and i < len(data):
                    window_x.append(data[i]["x"])
                    window_y.append(data[i]["y"])
                    i += 1
                    dispersion = (max(window_x) - min(window_x)) + (max(window_y) -
min(window_y))
                fixations.append([int(round(sum(window_x) / len(window_x))),
int(round(sum(window_y) / len(window_y)))])
                for j in range(i):
                    data.pop(0)
            else:
                #Drop a point, no fixation
                data.pop(0)
        with open("fixation_qa/" + dir + "_" + file, "w") as f:
            f.write(json.dumps(fixations))
            f.close()
```

Bakalaura darbs „Cilvēka uzmanības simulēšana tīmekļa mājaslapu dizaina procesā, izmantojot mašīnmācīšanos” izstrādāts LU Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: Arnolds Bogdanovs

Rekomendēju/nerekomendēju darbu aizstāvēšanai

Vadītājs: vadošais pētnieks LUMII Dr. dat. Renārs Liepiņš 28.05.2018

Recenzents: prof. Dr. phil. Jurģis Šķilters

Darbs iesniegts Datorikas fakultātē 28.05.2018.

Dekāna pilnvarotā persona: vecākā metodiķe Ārija Sproģe

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē

Komisijas sekretārs(-e):