Tartu Ülikool
Loodus- ja täppisteaduse valdkond
Tehnoloogiainstituut

Gregory Kuusmik

Postkaartide loomine tehisintellektiga

Bakalaureusetöö (12EAP)
Arvutitehnika eriala

Juhendaja:
Ardi Tampuu

Tartu 2025

Resümee/Abstract

Postkaartide loomine tehisintellektiga

Tehisintellekt (AI, *artificial intelligence*) on vaikselt integreerunud inimeste igapäevaelu osaks. AI kiire areng on loonud inimestele uusi innovatiivseid võimalusi nii töö tegemiseks kui ka meelelahutuse loomiseks. Suured keelemudelid ja tehisnärvivõrgud loovad juba vaevatult pilte, muusikat, videoid ja teksti ning võimaldavad järjest mitmekülgsemaid võimalusi inimeste kasutusse.

Käesoleva bakalaureuse lõputöö eesmärgiks on välja töötada kasutajaliidesega veebirakendus soovitud stiilis postkaartide loomiseks mõne pilte töötleva tehisintellekti lahendusega. Töö koosneb Rahvusraamatukogult saadud postkaartide metaandmete korrastamisest ning organiseerimisest, valitud tehisintellekti lahenduse peenhäälestamisest ning veebirakenduse loomisest.

CERCS: P176 Tehisintellekt;

Märksõnad: tehisintellekt, veebirakendus

Creating postcards using artificial intelligence

AI or artificial intelligence has slowly integrated itself to be a part of people’s everyday lives. AI’s fast development has created new innovative possibilities for people to use in eveyday work as well as for entertainment purposes. Large Language Models and Artificial Neural Networks are already effortlessly creating images, music, videos, and text and offer increasingly diverse possibilities for human use.

The aim of this bachelor's thesis is to develop a web application with a user interface for creating postcards with predefined style with the help of some artificial intelligence solution. The thesis consists of organizing and arranging the metadata of the postcards received from the National Library (RaRa), fine-tuning a chosen artificial intelligence solution, and the creation and development of a web application.

CERCS: P176 Artificial intelligence;

Keywords: artificial intelligence, web application

Sisukord

Jooniste loetelu

Lühendid, konstandid, mõisted

AI/TI (*Artificial Intelligence/Tehisintellekt)* - tehnoloogia, mis jäljendab inimlikku käitumist [1].

RaRa Rahvusraamatukogu

CPU (*Central Processing Unit*) - Arvuti kõige tähtsam riistvara komponent, mis töötleb andmeid jadamisi. Võimaldab operatsioonisüsteemi ja tarkvara kasutamise. [2]

GPU (*Graphics Processing Unit/Graphics card*) - Arvuti riistvara komponent, mis on disainitud töötlema andmeid paralleelselt. Kasutusalad on näiteks videod, 3D graafika ning AI mudelite treenimine [3].

UI (*User Interface*) ehk kasutajaliides.

OAI-PMH (*Open Archives Initiative Protocol for Metadata Harvesting*) - protokoll metaandmete korrektseks tõlgendamiseks [4].

API (*Application Programming Interface*) - tarkvara erinevate osade omavaheliseks suhtlemiseks vajalik vahendaja.

VRAM (*Video Random Access Memory*) - videokaartides kasutatav mäluelement, mis on mõeldud põhiliselt piltide ajutiseks säilitamiseks [5].

RAM (*Random Access Memory*) - mäluelement andmete ajutiseks säilitamiseks.

CNN (*Convolutional Neural Network/Konvolutsiooniline närvivõrk*) - masinõppe mudelite kategooria, mida kasutatakse enamasti pildi klassifikatsiooni ja objektituvastuse otstarbeks.

VGG-19 (*Visual Geometry Group 19*) - 19-kihiline konvolutsioonilise närvivõrgu mudel, mis on loodud Oxfordi visuaalse geomeetria töögrupi poolt [6].

L-BFGS (*Limited-memory Broyden-Fletcher-Goldfarb-Shanno*) - optimeerimis algoritm masinõppe mudelite treenimiseks [7].

1 Sissejuhatus

Tehisintellekt (TI; ingl *artificial intelligence*, AI) on aastakümneid kasutusel olnud mõiste, mis enamasti tähistab tehnoloogiat, mis aitab automatiseerida inimeste igapäevaelu. Viimaste aastate jooksul on AI, peamiselt generatiivsete mudelite, areng aga võtnud uue suuna, muutudes infoallikaks ning meedia sisu loojaks. Kuna (sotsiaal)meedia on järjest suurem osa inimeste elust, siis leidub ka AI-le selles valdkonnas järjest rohkem kasutust [1]. Teksti genereerimine, heli ja muusika genereerimine, sotsiaalmeedia haldamine, videote ning piltide tootmine on mõned näited sellest, milleks arenenumad AI mudelid on võimelised. AI tööriistadest ühed silmapaistvaimad on videoid ning pilte genereerivad programmid, mis suudavad juba inimsilmale päris piltidest eristamatuid reaalelulikke materjale luua. Sellised programmid on ka juba laialdaselt publikule kättesaadavad ning pildi või video loomiseks pole vaja muud kui kasutaja tekstikujul sisendit.

1.1 Töö tutvustus

Käesolev bakalaureusetöö on teostatud koostöös Rahvusraamatukogu (edaspidi RaRa) digilaboriga. RaRa digilabor kogub andmestikke mitmest erinevast allikast ning teeb lihtsamaks Eesti kultuurile juurdepääsetavuse digitaalse arhiivi kujul. Lähtudes digilabori eesmärkidest pakkus RaRa välja bakalaureusetöö teema luua AI lahendus väljamõeldud postkaartide loomiseks, kus arvestatakse nii kasutaja sisendit, kui ka arhiivi kogus olevate postkaartide stiili ja sümboolikat.

Eesti kultuur on maailmas väga ainulaadne ning väärib säilitamist. Tänapäeval on andmete digitaliseerimine avanud uue ning lihtsama ukse tutvuda kultuuriga. Kultuuriga saab tutvuda kirjandusele lisaks ka piltide kaudu. Pildid edastavad konkreetse visuaalse jäljendi mingist sündmusest, olukorrast, emotsioonist, tulemusest või lihtsalt hetkest. Oleme aga jõudnud sinnamaale, kus lihtsalt pilt ei ole enam külastajate jaoks piisavalt huvitav, huvitavaks teeb interaktiivsus. Selle bakalaureusetöö üks eesmärke ongi lisada piltide mõistmisele ja loomisele interaktiivsust.

Postkaardid on väga lihtne ja hea viis edastada emotsioone pildi vormis. Postkaartide kasutamine on aga interneti tõttu välja suremas. Viies kokku postkaardid ning tehisintellekti, saame huvitava koosluse, millega inimesed saavad endale uusi meelepäraseid postkaarte luua ning ajastute erinevusi näha.

1.2 Töö eesmärk ja ülevaade

Töö eesmärk on arendada tehisintellektiga lahendus, mis loob uusi ning huvitavaid postkaarte lähtudes kasutaja sisendist ning andmebaasis eksisteerivatest postkaartidest. Lahendus aga eeldab eelnevat andmetöötlust, planeerimist ning palju testimist.

Töö jaguneb mitmeks etapiks.

1. Postkaartide korrastamata XML formaadis metaandmeid tuleb OAI-PMH protokolli abil kätte saada. Seejärel tuleb andmed korrastada, organiseerida ja normaliseerida selliselt, et neid oleks lõpuks võimalik AI-ga kasutada.
2. Välja töötada kindlate parameetritega ning asjakohase sisu loomiseks kasutajaliidesega veebirakendus kasutaja sisendite jaoks.
3. Peenhäälestada stiiliülekande (ingl *style transfer*) tehisintellekti lahendus organiseeritud postkaartide põhjal. Lõplik lahendus suudab kanda üle postkaartide ajastu- ja stiilipõhised omadused kasutaja valitud pildile.

Töö eesmärk ei ole taotleda kaubanduslikku kasumit vaid eesmärgiks on luua hariduslik eksponaat ning avardada tehisintellekti kasutusala. Samuti luuakse lahendus skaleeruvaks, et andmebaasi kasvades või teiste tehisintellekti mudelite korral oleks võimalik lahendust ümber kohandada.

Kasutasin käesoleva töö koostamise käigus ChatGPT-d, Claude 3.7 ja Gemini 2.5 pro-d järgmistel eesmärkidel:

- Ideede kogumiseks
- Õigekirja kontrollimiseks
- Kokkuvõtete tegemiseks, et teemast parem ülevaade saada
- Õige terminoloogia leidmiseks, et seda korrektselt kasutada
- Töö koodibaasi loomisel
- Koodibaasi komponentide valikuks
- Koodi dokumenteerimiseks
- Koodi paremaks tegevate soovituste saamiseks
- Koodis olevate vigade parandamiseks

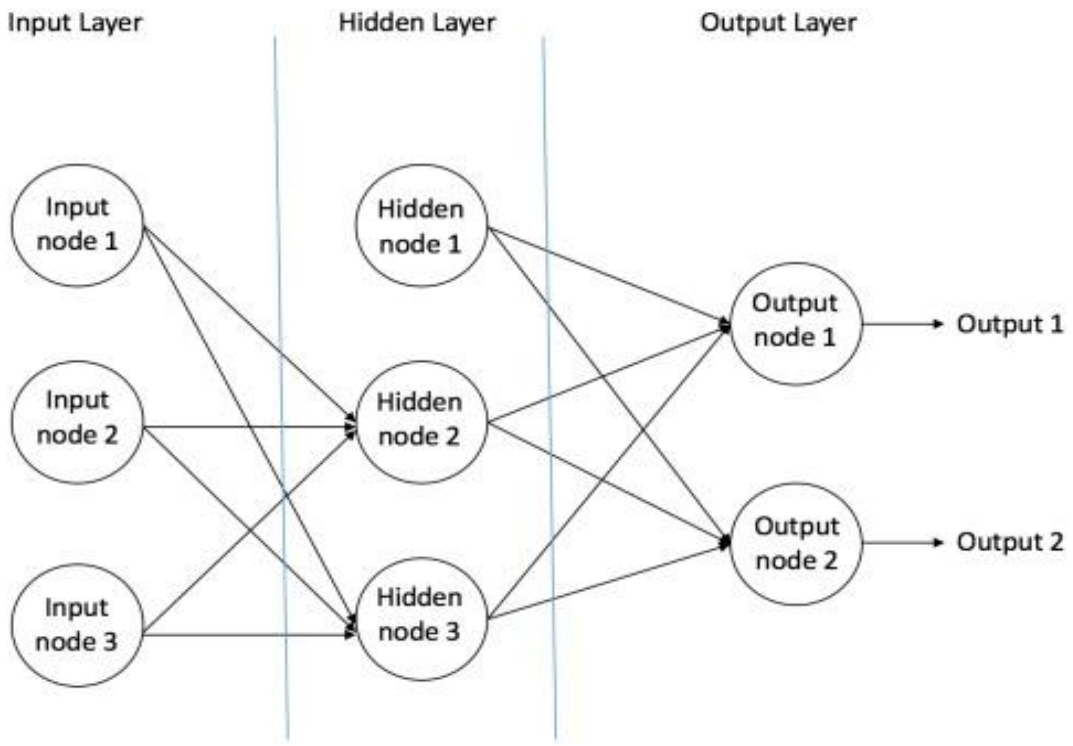
2 Ülevaade probleemist

Töö tehniliselt keerukaimaks eesmärgiks on tehisintellekti mudeli arendamine. AI kontekstis tähendab “mudel” mingit arvutuste võrgustikku, mis läheneb etteantud andmete töötlemisele teatud viisil ning teeb seda autonoomselt. Erinevad AI mudelid on disainitud töötlema erinevaid sisendeid ning kasutavad selleks ka vastavalt erinevat loogikat [8]. Kuna mudelite tööpõhimõtted erinevad teineteisest, võimaldab see meil kasutada neid erinevatel eesmärkidel.

Mudelite põhjal on loodud ning luuakse jätkuvalt juurde erinevaid spetsiifilisi AI mudeleid, mis treenitakse lahendama probleeme, töötlema erinevat tüüpi andmeid ning lõpuks looma/genereerima andmete põhjal vastava väljundi. Lihtsa video- või pilditöötluse jaoks pole vaja keerulist ressursimahukat algoritmi, vaid piisab ka lihtsamast mudelist.

2.1 Tehisintellekti areng

Digitaalne pilditöötlus sai alguse eelmise sajandi keskel ning arenes paralleelselt riistvara täiustumisega. Sajandi lõpuks olid tekkinud esimesed masinõppe mudelid, mida treeniti suurte pildikogumitega ning mis suutsid leida piltidelt üldiseid tunnusjooni ja detaile. Omaduste põhjal suutis algoritm rakendada piltidele algseid pilditöötluse võtteid [9]. Seda algset masinõppe mudelit ei saa aga nimetada veel AI-ks, tegu oli lihtsalt erinevate algoritmide konglomeraadiga, mis kasutas iga töödeldava ühiku osas ettenähtud matemaatilisi valemeid. See lõi aga aluse uuele arenevale tehisnärvivõrkude tehnoloogiale, mis oskas sisendandmete tunnusjoonte ja mustrite põhjal leida korrelatsioone väljundi ja sisendi vahel. Sellest hakkas aastate jooksul arenema meie mõistes tehisintellekt. Kasutusele võeti mitmekihiline andmetöötlus (vt joonis 2.1), mis jagas andmete töötluse mittelineaarselt mitme kihi vahel saavutades efektiivse ja mitmekülgsema tulemuse.



Joonis 2.1: *Feed forward* neuraalvõrgustiku kihiline andmetöötlus [10]

Selle sajandi kümnendatest alates hakkasid tekkima esimesed süvaõppe mudelid koos konvolutsioonilise närvivõrguga (*Convolutional Neural Network*, edaspidi CNN), mis spetsialiseerusid pildi, audio ja kõne töötlemisele. CNN-i edasiarenguga oleme nüüdseks saavutanud järjest paremaid omadusi ja funktsioone, näiteks "*AI upscaling*", ehk pildi resolutsiooni tõstmine ilma detailide kaota, "*Object removal*", ehk pildil oleva valitud elemendi eemaldamine ning ka "*Style Transfer*", ehk pildi stiili ülekanne teisele pildile [9]. Nüüdseks arendatakse juba AI agente, näiteks ChatGPT ja Gemini, mis suudavad üheaegselt töödelda mitut erinevat sorti sisendeid ning nende põhjal järjest intelligentsemaid ja täpsemaid vastuseid anda.

2.2 Esialgsed ressursid

RaRa võimaldas tööd tegemiseks kasutada enda valitud tarkvaralahendusi ja arendada oma kontseptsioon piltide tehniliseks töötlemiseks. Lähtematerjaliks oli XML formaadis Digari veebikeskkonna andmekogum, mis sisaldas kõikide postkaartide metandmeid ja lingi iga postkaardi leheküljele, milles esines kahjuks ka puudulikke andmeid.

2.3. Tehisintellekti mudelite valimine

2.3.1 Lähteülesanne ja piirangud

RaRa esitatud ülesandes nähti ette veebirakenduse loomine, mis võimaldaks genereerida uusi pilte olemasolevate postkaartide põhjal, kasutades tehisintellekti. Kuna RaRa eesmärk on kasutada seda veebirakendust postkaartidega seotud interaktiivse näituse elemendina, siis ei oleks teemaga sobinud näiteks ebatsensuurne sisend. Konkreetne tehisintellekti lahendusviis aga jäeti töö teostaja otsustada. Üheks oluliseks piiranguks projekti realiseerimisel oli ka RaRa nõue hoida rakenduse kasutuskulu minimaalsena ning tagada selle töötamine keskklassi konfiguratsiooniga seadmetel, mis oluliselt mõjutas tehisintellekti lahenduste valikut.

2.3.2 Alternatiivide kaardistus ja hindamine

Pilditöötluse mudelite valikul tuli arvestada mitmete kriteeriumitega:

- Rakenduse kasutusvaldkond interaktiivse näituseelemendina.
- Seadmete ressursipiirangud.
- Tehisintellekt kasutamine lokaalselt või pilvepõhiselt.
- Piiratud eelarve.

Kriteeriumeid arvesse võttes tuli leida sobiv pilditöötluse lahend. Kättesaadavad tehisintellekti lahendused jaotusid töötlemistaseme järgi kolme kategooriasse:

1. Kõrge töötlemistasemega lahendused.

   "*Text-to-image*" pildigeneraatorid (Stable Diffusion, Midjourney, FLUX.1) ning "*image inpainting*" lahendused (Adobe Photoshop, Photopea, Runway) võimaldavad tekstipõhise sisendi alusel luua või täiendada pilte [11][12]. Need mudelid on treenitud suurtel andmekogumitel ning kasutavad loomuliku keele töötlust (*Natural Language Processing*, NLP), et tõlgendada tekst arusaadavaks pildi genereerimiseks mõeldud AI algoritmile [13].

   Sarnaselt teistele arenenumatele mudelitele treenitakse kõrgetasemelisi "*image-to-image*" pildigenereerimise mudeleid suurte andmekogumitega, et tunda ära erinevaid mustreid ja stiile piltides. Selle taseme mudel võimaldab tulemust transformeerida, lisada või eemaldada objekte ja suunata tulemust tekstipõhiselt [14].

See lahendus osutus antud projekti kontekstis ebasobivaks, kuna rakenduse implementeerimine olnuks liialt töömahukas, riistvaranõuded ületasid ettenähtud piirangud, "*text-to-image*" lähenemine ei sobinud kokku ülesandega kasutada olemasolevaid postkaarte ja interaktiivse näituse kontekstis oleks vajalik olnud range sisendi filtreerimine.

2. Keskmise töötlemistasemega lahendused

   "*Style transfer*" on üks "*image-to-image*" variantidest, mis ühendab doonoripildi stiiliparameetrid sisendpildi sisuga [15]. Kuna antud meetod ei nõua täiesti uue sisu genereerimist, sisu on defineeritud sisendpildil olnuga, on selle ressursivajadus palju madalam. Sellise taseme lahendusi leiab tasuta kasutatavatest pilveteenustest, kus tehisintellekti töötlus toimub teenusepakkuja serveri resurssidega erinevalt kõrge töötlemistaseme lahendustest.

3. Madala töötlemistasemega lahendused.

   Lihtsad algoritmid värvifiltrte ja eelseadistatud efektide lisamiseks nõuavad minimaalseid riistvararessursse. Funktsionaalsuse poolest oleks tegemist aga tavalise automatiseeritud pilditöötlusega, mis ei vastaks tööle seatud eesmärkidele ning tulemus oleks ebarahuldav ning igav.

2.3.3 Optimaalse lahenduse valik

Projekti nõuetest lähtuvalt langetati otsus keskmise töötlemistasemega stiiliülekande lahenduse kasuks. Juhul, kui eelpool toodud optimaalseid ressursivajadusi ka täpselt ei kaetaks, oli siiski võimalik leida kompromiss ressursside ja päringu ooteaja vahel.

Pilveteenuste kasutamine otsustati jätta paralleelseks arendusliiniks, mis pakuks kvalitatiivset võrdlust lokaalsele lahendusele. Arendusperioodi ajal oleks võimalik kasutada tasuta mudeleid, ning hiljem on ka töötavale eksponendile realistlik leida täpselt mõõdetud päringute põhjal maksustatav teenus, mis sobiks kulude kokkuhoiu nõude ja rakenduse potensiaalselt suhteliselt madala kasutusaktiivsusega.

3 Süsteemi projekteerimine ja tehniline baas

3.1 Ülesehitus

Töötava veebirakenduse loomiseks sai töö teostaja kasutada vabalt valitud vahendeid ja abinõusid.

Lisaks tehisintellekti kasutamisele oli tellija nõudeks arhiveeritud postkaartide metaandmete esialgne korrastamine ja normaliseerimine. Andmete edasine kasutamine tehisintellektile valitud stiiliülekande protsessi stiili sisendina eeldas andmete täiendavat grupeerimist sarnaste sisendite alamhulkadesse. Selleks tuli töö esimese etapina luua serveris iseseisva komponendina taustateenus. Taustateenuse funktsionaalsus oleks ühtlasi ka varustada andmetega rakenduse teisi komponente. Algupärased postkaartide andmekogumi andmed polnud kuigi hästi organiseeritud ja sisaldasid ebavajalikke ning ebamääraseid märgendeid, millest osa olid ingliskeelsed ning osad eestikeelsed, ning vajasid eraldi töötlust vajaliku info ekstraheerimiseks (vt pilt 4.1.1). Struktuuri ehitades tuli ka arvestada koodi paindlikkust ning edaspidist skaleeritavust.

3.1.1 OAI-PMH andmete kogumine ja normaliseerimine

Postkaartide andmed asetsevad Digari arhiivis ning on kättesaadavad XML formaadis. Selleks, et andmed kätte saada, tuli taustateenusesse esiteks implementeerida OAI-PMH klient [4]. Kasutati teeke nagu axios, mis on HTTP klient brauseri ja Node.js jaoks [16], ja xml2js, mis teisendab XML formaadis andmeid JavaScripti objektideks [17], et postkaartide metaandmeid oleks võimalik mugavalt töödelda. Andmete esialgse kogumise juures oli olulisemateks:

- Tõlgendada korrektselt OAI-PMH XML elementide nimesid.
- Töödelda andmeid partiidena, et tagada veatu failide kirjutamine ja mälu kasutamine.
- Implementeerida korralik vigade käsitlus, mis võimaldaks probleemide korral partiide töötlus uuesti käivitada.

Selle etapi tulemuseks oli kogum normaliseeritud ja töödeldud algandmeid. Need andmed sisaldasid endiselt ainult algupäraseid väärtusi, aga normaliseerimise tulemusena saime lahti algupärase OAI-PMH ebakorrapärasest pesastatud struktuurist ja mittevajalikest

metaandmetetest. Esimene taustateenuste alamteenus on seega ette nähtud konverteerima toore XML struktuuri standardiseeritud ja lamendatud JSON objektiks, valideerides kriitilise tähtsusega andmevälju ja ühendades samasisulised andmed ühe nimetaja alla. Lisaks veakäsitlusele sai konverteerimise protsess kaasa hoiatused, mis ilmutavad olulise tähtsusega andmeväljade puudumisest või tõlgendamatusest.

3.1.2 Normaliseeritud andmete kohandamine

Normaliseeritud andmete põhjal oli järgmiseks tööks andmete kohandamine eesmärgipäraseks kasutamiseks. Kuna andmestik sisaldas üle 7000 ühiku, kerkis juba selles esmases faasis esile probleem, kuidas neid AI jaoks mõistlikusse vormi saada.

Postkaardi kujutise puhul on kõige tähtsam see kasutajale visuaalselt presenteerida, aga rakendus ei võimaldaks mingil juhul kõikide postkaartide näitamist. Seega oli vaja leida loogiline meetod, kuidas andmed kategoriseerida. Meetod, mis tundus nii kasutaja kui tehisintellekti seisukohalt kõige parem, oli kategoriseerida andmed esmalt postkaardi väljaandmise aja põhjal ning järgnevalt postkaardi sisu põhjal.

Postkaardi väljaandmise ajad on kaardistatud 12 erineva Eesti ajaloolise perioodi põhjal. Kuna algandmed sisaldasid äärmiselt mitmesugust tüüpi dateeringuid, näiteks täpne aastaarv, aastakümnend või “sajandi algus”, ja ka vormistus oli erinev, tuli kasutada regulaaravaldiste mustreid, mille alusel sai element tõenäolise perioodi hinnangu. Kui dateering puudus või oli arusaamatu, tuli proovida eraldada see informatsioon muudest metaandmetest.

Postkaardid võivad olla fotod või joonistatud ja kujutada maastikke või portreepilte. Nende andmete põhjal juhindus stiilipõhine kategoriseerimine. Kategooriaid on kokku 4: vaated, portreed, sündmused ja varia, ning need omakorda jagunevad fotodeks ja joonistatud postkaartideks.

3.1.3 Andmestruktuur AI sisendina

Algandmete ümber struktureerimine loob lõpptulemusena ühe JSON andmefaili. See fail koondab kõik normaliseeritud ja kategoriseeritud kirjed eelnevalt täpsustatud struktuuri põhjal kokku. Iga kirje sisaldab metaandmetele lisaks lingi vastava postkaardi Digari arhiivi, kus leidub lõplik postkaardi pildifail. Seda andmefaili kasutatakse veebirakenduse kasutajaliideses stiilipildi valimises, kuvades nii perioodi kui ka perioodi alla kuuluvad

stiilikategooriad. Iga kategooria alt valitakse visualiseerimiseks juhuslik pilt. Valides endale meelepärase stiilipildi ning käivitades stiiliülekande protsessi, saab tehisintellekt valitud stiilipildi andmed JSON andmefailist kätte.

Oluline on tähele panna juhuslikkuse osakaalu protsessi lähtematerjali kogumises. Kasutajaliideses ei ole võimalik näidata kõiki kategooria pilte ja seetõttu ei saa ka valida ühte kindlat eksemplari, mille alusel kasutaja enda sisupilti stiliseeritaks.

3.2 Süsteemi arhitektuur

Selle projekti elluviimiseks valiti teadlikult mitmel väiketeenusel töötav keskkond, eelistades seda monoliitsele koodile, mis sisaldaks endas kogu funktsionaalsust. Põhjuseks pole mitte ainult see, et selline lähenemine pakub paremat hooldusvõimalust, vaid on ka modernsem ja paindlikum ning tulevikus oluliselt lihtsamalt skaleeritav. Luues algusest peale teineteisest sõltumatud teenuseblokid, on neid lihtne asendada vastavalt riistvara ja tellija isikliku infrastruktuuri nõuetele, võimaldades tulevikus mitmekülgsemat skaleeritavust.

Mitmeteenuseline lähenemine võimaldab iga konkreetse ülesande jaoks kasutada kõige sobivamat tehnoloogiapaketti. Andmete kogumise ja API marsruutimise sisend-/väljundipõhised toimingud sobivad Node.js-i asünkroonse, sündmustepõhise mudeli ning TypeScripti tugeva tüübikontrolli jaoks. Seevastu tehisintellekti implementatsioon on oluliselt lihtsam ellu viia Pythonis, mis võimaldab kasutada ulatuslikult masinõppe ning pilditöötluse teeke nagu PyTorch, Pillowja OpenCV. Monoliitne arhitektuur sunniks tegema kompromisse, kasutades tõenäoliselt ühte keelt, mis ei pruugi olla igas süsteemi osas optimaalne.

3.2.1 Ülevaade komponentidest

Süsteem koosneb neljast peamisest teenusest - andmekäsitluse taustateenus, AI teenus, API Gateway ja kasutajaliides:

- **Andmekäsitluse taustateenuse** roll on vastutada kõikide väliste algandmetega seotud toimingute eest. See loeb metaandmed XML formaadis, teostab andmete normaliseerimise JSON formaati, rakendab ajaperioodi ja kategooria kaardistamise loogikat, ning salvestab lõplikult struktureeritud andmed.

- **AI teenus** sisaldab tehisintellekti loogikat. See võtab vastu kasutaja valitud sisu- ja stiilipildi ning teostab stiiliülekande kasutades paralleelselt kolme stiiliülekande lahendust: lokaalselt töötavat VGG-19 põhist närvstiiliülekannet, Magenta mudelit ning pilvepõhist lahendust [18]. Paralleelse implementatsiooni eesmärgiks on tagada optimaalne stiiliülekande kvaliteet erinevates tingimustes.
- **API värav** tegutseb ühtse sisenemispunktina veebiliidese jaoks. See haldab sissetulevad päringud kasutajaliideselt ja suunab need sobivasse kõrvalteenusesse.
- **Kasutajaliides** on veebirakenduse visuaalne kasutajale nähtav teenus. See võimaldab kasutajal üles laadida sisupildi, sirvida ja valida ajaloolisi ajaperioode ning stiili kategooriaid, reguleerida valikulisi häälestusparameetreid, valida töötlusrežiimi (kohalik/pilv), käivitada stiili ülekande protsessi, kuvab töötluse olekut, käsitleb võimalikke vigu sujuvalt, näitab loodud tulemusi, võimaldab tulemuste allalaadimist ning haldab mitme tulemuse kuvamist.

3.3 Kasutatud tehnoloogiad ja valiku põhjendus

Kasutatud tehnoloogia valik toimus arvestades projekti spetsiifilisi omadusi ja nõudmisi, proovides iga valikuga leida lahendatava probleemi jaoks parima lahenduse.

3.3.1 Andmekäsitlus ja API värav

Nende teenuste jaoks valiti Node.js ökosüsteem koos TypeScripti ja Express.js raamistikuga.

Node.js valiti selle väga tõhusa, mittetõkestava, sündmuspõhise sisend-/väljundmudeli tõttu. See on eriti kasulik teenustele, mis tegelevad peamiselt võrgu päringutega, lugemis-/kirjutamistoimingutega ning suudavad efektiivselt hallata samaaegseid ühendusi ilma CPU-d pikaajaliselt koormamata.

TypeScripti võeti kasutusele kõikides JavaScript-il põhinevates teenustes, et lisada range tüüpsus JavaScripti peale. See parandab oluliselt koodi hooldatavust ja vähendab vigu reaalajas, võimaldades tüübikontrolli kompileerimise ajal.

Express.js valiti veebiraamistikuks Node.js jaoks selle minimalistliku disaini, paindlikkuse ja ulatusliku kesktarkvara (*middleware*) ökosüsteemi tõttu. Selle lihtsus võimaldas kiiret API lõpp-punktide arendamist taustateenuse.

3.3.2 AI teenused

AI ehitamiseks kasutati Pythonit, mis võimaldab kasutada mitmekülgselt teeke nagu PyTorch ja FastAPI.

Python on standardkeel arvutamiseks, masinõppeks ja süvaõppeks. Selle äärmiselt lai teekide kogu – näiteks numbriliseks arvutuseks *NumPy*, pilditöötluseks *Pillow* , andmetöötluseks ning eeskätt masinõppe raamistike jaoks *PyTorch* – tegi sellest loomuliku valiku tehisintellekti implementeerimise jaoks.

PyTorch on juhtiv teek generatiivse AI süsteemide kasutamises. *PyTorch* võimaldab ligipääsu erinevatele mudelitele (nt VGG-19), optimeerijatele (nt L-BFGS ja Adam) ja muudele vajalikele ressurssidele. *PyTorch* on optimeeritud kasutama GPU-d ja CPU-d, mis aitab arvutusmahukate mudelite puhul. Lisaks kasutab *PyTorch* sügavõppe raamistikel *Define-by-Run* metoodikat, mis defineerib arvutuskäike dünaamiliselt ja jooksvalt protsessi sees. Võrdluseks on *Define-and-Run* metoodika, mis defineerib arvutuskäigu eelnevalt ära ning alles siis rakendab arvutusgraafil antud sisendit [19].

FastAPI on kõrge jõudlusega veebiraamistik, mis võimaldab *Python*-is API-de ehitamise. *FastAPI* on ehitatud *Starlette* ja *Pydantic* peale, mis pakub *Node.js*-ga võrreldavat jõudlust. See on oluline, et tulla tõhusalt toime potentsiaalselt ajamahukate AI ülesannetega. Sisseehitatud asünkrooniliste päringute tugi võimaldab serverit blokeerimata I/O-toiminguid teostada.

3.3.3 Kasutajaliides

Kasutajaliidese loomiseks osutusid parimateks teekideks UI ehitamist lihtsustav *Lit* ja kiire ehitustööriist *Vite*, mis pakub lisaomadusi veebirakenduse ehitamisel.

Lit on kergekaaluline teek kiirete ja lihtsalt integreeritavate lahenduste loomiseks. *Lit* võimaldab kapseldatud ja taaskasutatavate UI-komponentide loomist, kasutades brauseri sisseehitatud võimalusi. Kuna *Lit* võimaldab kasutada ka reaktiivset loogikat, oli kasutajaliidese valmistamine sellega palju hõlpsam.

Esimene valik kompileerimise tööriistaks ja arenduse serveriks oli *Vite*, kuna see pakub head kiirust ja omab suurepärast kasutajahinnangut. *Vite* pakub praktiliselt kohest *Hot Module*

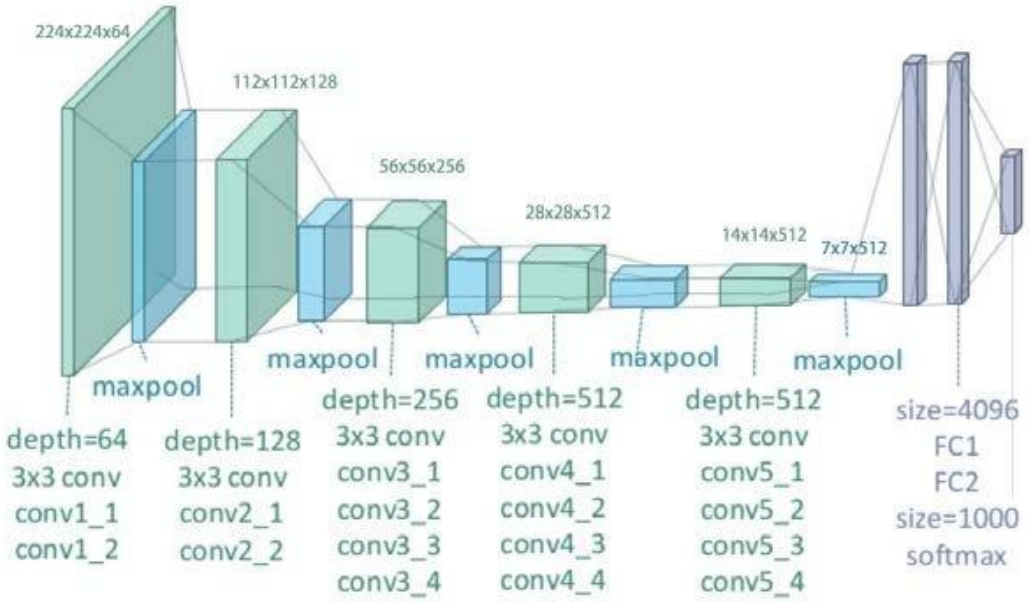
Replacement-i arendamisel ja kokkuvõttes kõrgemat optimeeritust. Selle sisseehitatud proksitugi oli samuti kasulik API *Gateway* päringute kasutamiseks lokaalse arenduse ajal.

3.4 AI baas: *Neural Style Transfer* meetodid

Projekti keskpunktiks on närvivõrkudel baseeruv stiiliülekanne (*Neural Style Transfer*), mis muudab kasutaja pilte ajalooliste postkaartide stiilide järgi. Selle põhifunktsionaalsuse rakendamise strateegia hõlmas paralleelsete mudelite valikut, töötlemise lähenemisviise ja mudelite optimeerimist. Projektis implementeeriti kaks erinevat lokaalset lähenemist: VGG-19 konvolutsioonilise närvivõrgu põhine iteratiivne optimeerimine ning eeltreenitud Magenta mudel kiirema töötlemise võimaldamiseks. Lisaks on alternatiivina implementeeritud ka pilvepõhine lahendus, mille eesmärk on olla põhiliselt võrdluselemendiks. Konkreetne pilveteenus pole aga mõeldud pikemas perspektiivis kasutamiseks, kuna selle olemasolu pole garanteeritud.

3.4.1 VGG-19 konvolutsiooniline närvivõrk

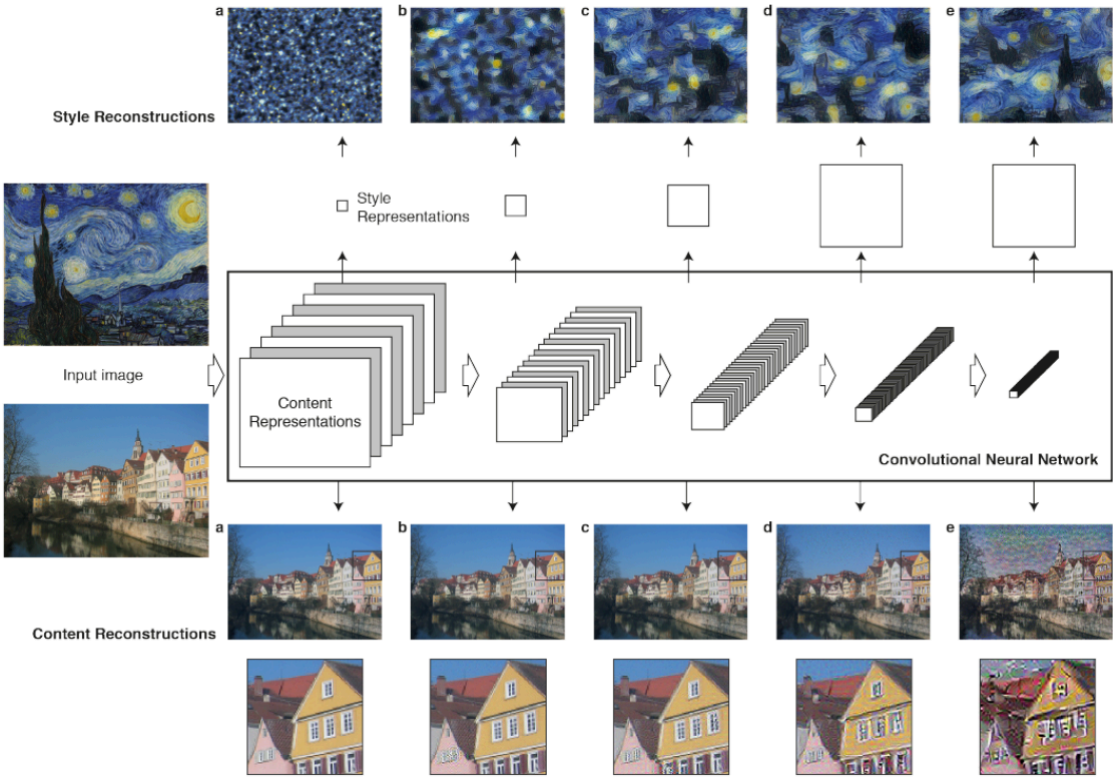
Üheks valitud alusmeetodiks on konvolutsioonilise närvivõrgu (*Convolutional Neural Network*, CNN). VGG-19 võrk valiti selle robustsuse ja efektiivsuse tõttu pildi detailide eraldamisel. VGG-19 on edasi arenenud versioon VGG-16 võrgust ja omab 3 lisakihti, mis aitavad leida piltidest veelgi detailsemaid nüansse. VGG-16 jagab pildi vaid 16-ks kihiks, 13 konvolutsiooni kihti ja 3 täissidusat kihti ning VGG-19 jagab pildi 19-ks kihiks, 16 konvolutsiooni kihti ja 3 täissidusat kihti. Iga kihi vahel koondatakse (*max-pooling*) eelmine kiht väiksemaks järgmise kihi jaoks. Pildi elementide tuvastamiseks rakendatakse väikseid (3x3) konvolutsiooni filtreid iga kihi sees [20]. Madalamad kihid püüavad kinni pildi üldisemaid omadusi, näiteks värvus, tekstuur ja muud stiililised detailid ning sügavamad kihid püüavad kinni abstraktsemat sisuinfot.



Pilt 3.4.1: Illustratsioon VGG-19 CNN tööst [21].

Kihtide eraldatus võimaldab algoritmil määrata eraldi kaotusfunktsioonid sisu säilitamiseks ja stiili sobitamiseks. Väljundi genereerimiseks kasutatakse sisupildi ja stiilipildi kombineeritud kaofunktsiooni, mida optimeerimisel minimeeritakse. Kaofunktsiooni põhimõte on võrrelda piltide omadusi ning leida nende vaheline kadu - mida väiksem on kadu seda sarnasemad on pildid omaduste poolest. Matemaatiliselt väljendub piltide vaheline kogu kadu valemiga $L_{kokku}(\overline{p}, \overline{a}, \overline{x}) = \alpha L_{sisu}(\overline{p}, \overline{x}) + \beta L_{stiil}(\overline{a}, \overline{x})$, kus L_{sisu} ja L_{stiil} on sisupildi ja stiilipildi kaod, α ja β on kaalutegurid kummagi kao jaoks ning $\overline{p}$ sisupilt, $\overline{a}$ stiilipilt ja $\overline{x}$ väljundpilt [22].

Piltide vahelise kao arvutamiseks luuakse esmalt valge müraga pilt, mida võrreldakse sisendpildiga ning hakatakse optimeerima. Optimeerimiseks kasutatakse gradient laskumine (*gradient descent*) meetodit, mis iteratiivselt muudab mürase pildi pikslite väärtusi, et minimeerida kaofunktsiooni väärtust [23]. Stiilipildi ja väljundpildi vahelise sarnasuse mõõtmiseks arvutatakse iga kihi gram-maatriks, mis väljendab kihi tunnuste vahelist korrelatsiooni.



Pilt 3.4.2: Illustratsioon *gradient descent* meetodi tööst stiilipildil ja sisupildil [22].

Pilt 3.4.3: Stiiliülekande tulemus [22].

Optimeerimisel on võimalik kasutada erinevaid optimeerijaid, millest implementatsioonis katsetati nii L-BFGS kui Adam optimeerijaid. L-BFGS (*Limited-memory Broyden–Fletcher–Goldfarb–Shanno*) pakub täpsemat tulemust, kuid on arvutuslikult kallim. Adam optimeerija (*Adaptive Moment Estimation*) koondub tasasemalt ja kiiremini, kuid tulemus ei küündi päris L-BFGS tasemele [24].

Stiiliülekande implementatsioonis on neli erinevat parameetrit, mis mõjutavad väljundpildi tulemust ja mille abil tulemust häälestada:

- content_weight (sisupildi kaalutegur) - määrab kui palju algse pildi sisust säilitatakse.
- style_weight (stiilipildi kaalutegur) - määrab kui tugevalt stiilielemendid rakendatakse.
- tv_weight (*total variation weight*, tulemuspildi siluvusparameeter) - määrab lõpptulemuse silumise määra.
- number_of_steps (iteratsioonide arv) - määrab kui mitu korda optimeerimist teostatakse.

3.4.2 Magenta mudel

Paralleelselt VGG-19 põhise lähenemisega implementeeriti ka eeltreenitud Magenta mudel, mis pakub alternatiivset lähenemist stiiliülekandele. Erinevalt VGG-19 põhisest iteratiivsest optimeerimismeetodist on Magenta eeltreenitud mudel, mis võimaldab oluliselt kiiremat stiili ülekandmist. Magenta mudel põhineb samuti konvolutsioonilise võrgu arhitektuuril, kuid on eeltreenitud nii pilditöötluse kui ka heli- ja muusikatöötluse jaoks ja on mitmekülgsem mudel võrreldes VGG-19-ga [18].

Magenta mudeli eelis on selle kiirus, kuna mudel ei vaja iga sisu- ja stiilipildi paari kohta eraldi pikka optimeerimisprotsessi. Samuti on see vähem tundlik sisendpiltide kvaliteedi ning resolutsiooni osas. Puuduseks on aga väiksem kontroll tulemuse üle ning mõnede stiilide puhul vähem detailsem töötlemine.

3.4.3 Pilvepõhine alternatiiv

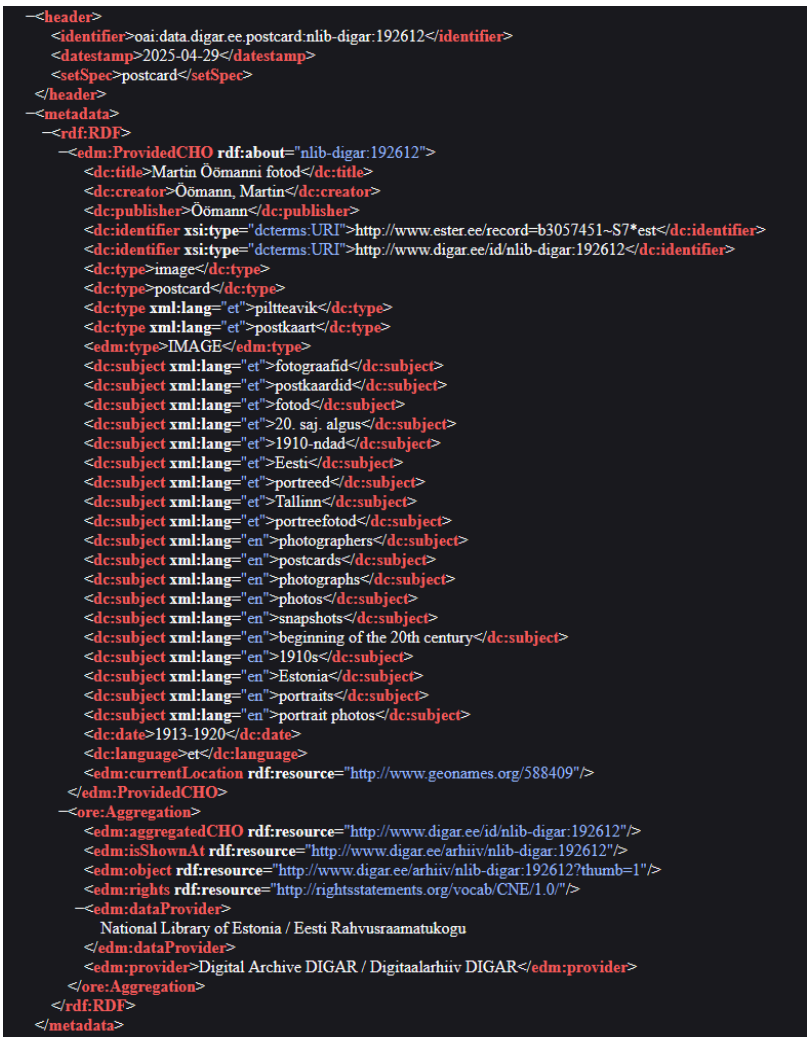
Kuna lokaalse CPU/GPU töötlemise jõudluse piirangud on märgatavad, eriti keskmise võimsusega riistvaral, implementeeriti projektis ka paralleelne pilve töötlemise võimalus. See suunab päringud konkreetsele avalikule *Hugging Face Space*-ile, mis on optimeeritud

samasuguse stiiliülekande sooritamiseks. Pilvepõhine lähenemine pakub märkimisväärset eelist töötlemiskiiruses, võimaldades kiiret tulemuste saamist ka piiratud riistvaraga seadmetel.

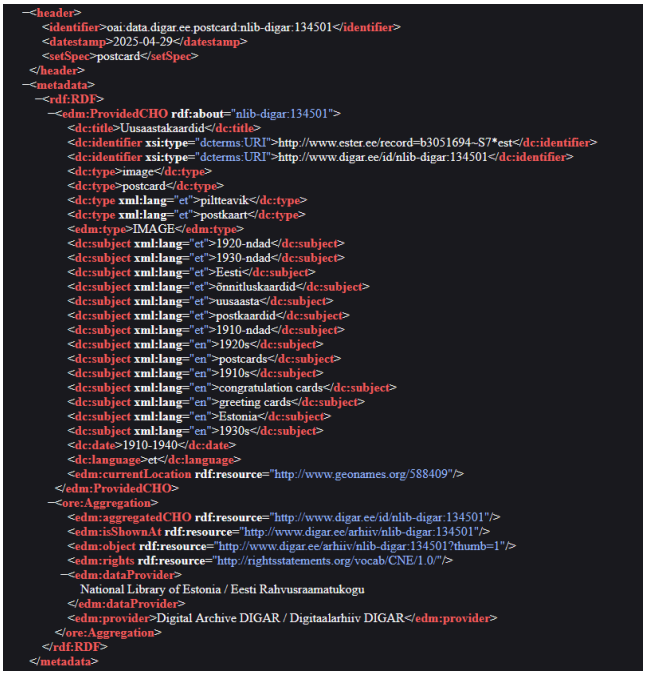
4 Tulemused

4.1 Metaandmete korrastamise tulemused

Digari arhiivi postkaartide metaandmed olid algselt ebamäärased ning korrastamata kujul tehisintellekti rakenduste jaoks ebasobivad. Metaandmed esinesid keerulises XML-struktuuris, mis vajas märkimisväärset töötlust.



Pilt 4.1.1: Esimene näidis Digari metaandmetest XML formaadis.



Pilt 4.1.2: Teine näidis Digari metaandmetest XML formaadis.

Algsetel metaandmetel olemasoleva informatsiooni põhjal oli küllalt raske teha ühest järeldust postkaardi ajastu, sisu ja tüübi kohta. Peatükis 3.1.1 kirjeldatud normaliseerimisprotsessi tulemusena saadud struktureeritud metaandmed kajastavad postkaarte palju selgemalt ja nende üldine andmestruktuur on lineaarne.

Pilt 4.1.3: Normaliseeritud 4.1 pildil olevad metaandmed JSON formaadis.

```
  "title": "Uusaastakaardid",
  "esterUrl": "http://www.ester.ee/record=b3051694~S7*est",
  "imageUrls": {
    "full": "http://www.digar.ee/arhiiv/nlib-digar:134501",
    "resource": "http://www.digar.ee/id/nlib-digar:134501"
  },
  "location": "http://www.geonames.org/588409",
  "subjectsEt": [
    "1920-ndad",
    "1930-ndad",
    "Eesti",
    "õnnitluskaardid",
    "uusaasta",
    "postkaardid",
    "1910-ndad"
  ],
  "subjectsEn": [
    "1920s",
    "postcards",
    "1910s",
    "congratulation cards",
    "greeting cards",
    "Estonia",
    "1930s"
  ],
  "rawDate": "1910-1940",
  "language": "et",
  "dataProvider": "National Library of Estonia / Eesti Rahvusraamatukogu",
  "provider": "Digital Archive DIGAR / Digitaalarhiiv DIGAR",
  "rights": "http://rightsstatements.org/vocab/CNE/1.0/"
},
{
  "identifier": {
    "#": "oai:data.digar.ee.postcard:nlib-digar:107577",
    "@_ns": {
      "uri": "http://www.openarchives.org/OAI/2.0/",
      "local": "identifier"
    }
  },
```


Pilt 4.1.4: Normaliseeritud 4.2 pildil olevad metaandmed JSON formaadis.

Normaliseeritud metaandmeid oli vaja veel täiendavalt kaardistada tehisintellekti ja kasutajaliidese jaoks optimaalsesse vormi. Näiteks postkaartide aastaarvud, perioodid, kirjeldused ja teemad struktureeriti kasutajaliidese jaoks kergesti kättesaadavale kujule, võimaldades nende loogilist kuvamist ja töötlemist.


```
"metadata": {
  "lastUpdated": "2025-05-02T10:53:06.840Z",
  "totalRecords": 7261,
  "periodsPresent": [
    "1946_1953",
    "1906_1917",
    "1940_1945",
    "1930_1939",
    "1965_1986",
    "1923_1929",
    "1894_1905",
    "1918_1922",
    "1954_1964",
    "2003_present",
    "1992_2002",
    "1987_1991"
  ]
},
```


Pilt 4.1.5: Tehisintellekti ja kasutajaliidese jaoks aastaarvude kättesaadavam struktuur JSON formaadis.

```
"periods": {
  "1894_1905": {
    "metadata": {
      "id": "1894_1905",
      "name": {
        "et": "Esimesed postkaardid",
        "en": "Early Postcards"
      },
      "yearRange": {
        "start": 1894,
        "end": 1905
      },
      "description": {
        "et": "Esimesed postkaardid Eestis, tsensuurieelne periood",
        "en": "First postcards in Estonia, pre-censorship period"
      }
    },
    "categories": {
      "drawn_scenery": {
        "records": [
          "http://www.digar.ee/arhiiv/nlib-digar:57027",
          "http://www.digar.ee/arhiiv/nlib-digar:56350",
          "http://www.digar.ee/arhiiv/nlib-digar:57430",
          "http://www.digar.ee/arhiiv/nlib-digar:57431",
          "http://www.digar.ee/arhiiv/nlib-digar:56408",
          "http://www.digar.ee/arhiiv/nlib-digar:56429",
          "http://www.digar.ee/arhiiv/nlib-digar:56422",
          "http://www.digar.ee/arhiiv/nlib-digar:57185",
          "http://www.digar.ee/arhiiv/nlib-digar:334918",
          "http://www.digar.ee/arhiiv/nlib-digar:56513",
          "http://www.digar.ee/arhiiv/nlib-digar:55880",
          "http://www.digar.ee/arhiiv/nlib-digar:56969",
          "http://www.digar.ee/arhiiv/nlib-digar:54777",
          "http://www.digar.ee/arhiiv/nlib-digar:58522",
          "http://www.digar.ee/arhiiv/nlib-digar:58821",
          "http://www.digar.ee/arhiiv/nlib-digar:65449",
          "http://www.digar.ee/arhiiv/nlib-digar:69444",
          "http://www.digar.ee/arhiiv/nlib-digar:71082"
        ],
        "count": 18
      },
      "photo_scenery": {
        "records": [
          "http://www.digar.ee/arhiiv/nlib-digar:68472",
          "http://www.digar.ee/arhiiv/nlib-digar:70480",
          "http://www.digar.ee/arhiiv/nlib-digar:71011"
        ],
        "count": 3
      },
      "drawn_festive": {
        "records": [],
        "count": 0
      },
      "photo_festive": {
        "records": [],
        "count": 0
      },
      "drawn_portrait": {
        "records": [],
        "count": 0
      },
```


Pilt 4.1.6: Andmed sorteeritud perioodide järgi lihtsamaks kasutamiseks.

```
    "photo_portrait": {
      "records": [],
      "count": 0
    },
    "drawn_misc": {
      "records": [
        "http://www.digar.ee/arhiiv/nlib-digar:326989",
        "http://www.digar.ee/arhiiv/nlib-digar:69486",
        "http://www.digar.ee/arhiiv/nlib-digar:69485",
        "http://www.digar.ee/arhiiv/nlib-digar:69484",
        "http://www.digar.ee/arhiiv/nlib-digar:69483",
        "http://www.digar.ee/arhiiv/nlib-digar:69482",
        "http://www.digar.ee/arhiiv/nlib-digar:69481",
        "http://www.digar.ee/arhiiv/nlib-digar:69480",
        "http://www.digar.ee/arhiiv/nlib-digar:69474",
        "http://www.digar.ee/arhiiv/nlib-digar:69473",
        "http://www.digar.ee/arhiiv/nlib-digar:69478",
        "http://www.digar.ee/arhiiv/nlib-digar:69477",
        "http://www.digar.ee/arhiiv/nlib-digar:69472",
        "http://www.digar.ee/arhiiv/nlib-digar:69471",
        "http://www.digar.ee/arhiiv/nlib-digar:69470",
        "http://www.digar.ee/arhiiv/nlib-digar:69469"
      ],
      "count": 16
    },
    "photo_misc": {
      "records": [
        "http://www.digar.ee/arhiiv/nlib-digar:69523"
      ],
      "count": 1
    }
  },
  "statistics": {
    "totalCount": 38,
    "categoryDistribution": {
      "drawn_scenery": 18,
      "photo_scenery": 3,
      "drawn_festive": 0,
      "photo_festive": 0,
      "drawn_portrait": 0,
      "photo_portrait": 0,
      "drawn_misc": 16,
      "photo_misc": 1
    }
  }
```


Pilt 4.1.7: Pildi 4.1.6 jätk.

4.2 Sisendpiltide eeltöötlus

Enne kui sisendpilte sai töödelda stiiliülekande algoritmidega, tuli need viia standardiseeritud kujule. Pilditöötluse algoritm hõlmas:

1. Piltide sisselugemine ja RGB formaati teisendamine
2. Üle 1024px suuruste piltide mõõtmete vähendamine, säilitades originaalpildi kuvasuhte.
3. Pildi üleviimist *PyTorch* Tensor formaati CNN-i mudelile kasutamiseks:
    - Pikslite väärtuste normaliseerimine vahemikku [0, 1]
    - Ploki suuruse määramine (*batch size*)
    - Töötlusseadme (CPU või GPU) määramine

Standardiseerimisprotsess oli ühtne mõlema implementeeritud mudeli puhul (VGG-19 ja Magenta), tagades sisendandmete järjepidevuse kogu süsteemis.

4.3 VGG-19 põhise mudeli arendamise eksperimendid

Arendusprotsessi alustades ilmnesid mitmesugused tehnilised väljakutsed, mis mõjutasid oluliselt stiiliülekande tulemuste kvaliteeti. Töö käigus tuli katsetada ja täiustada mitmeid komponente optimaalse tulemuse saavutamiseks.

4.3.1 Stiilipiltide suuruse problemaatika

Märkimisväärne probleem ilmnes stiilipiltide kvaliteediga. Osadel Digari arhiivi postkaartidel puudusid programmiliselt kättesaadavad täissuuruses eksemplarid, mistõttu tuli testimisel kasutada Digari veebilehel asuvaid postkaartide pisipilte. Need olid aga liiga väikeste mõõtmetega, et pildi stiili detaile täielikult kätte saada.

Probleemi lahendamiseks implementeeriti täiendav võimalus kohalike stiilipiltide kasutamiseks. See laiendus jagab stiilipildi valimise protsessi kaheks, Digari arhiivist pärit postkaartide pisipildid ja kohalik piltide kogumik, mis võimaldab katsetada kõrgema resolutsiooniga stiilipilte.

4.3.2 Gram-maatriksi normaliseerimise täiustamine

Esialgse implementatsiooni testimisel ilmnesid ebastabiilsed kaoväärtused, mis mõnikord viisid *NaN* (*Not a Number*) vigadeni optimeerimisprotsessis.

Algsed visuaalsed tulemused olid kehvad, kaoväärtused olid potentsiaalselt ebastabiilsed ning optimeerimisprotsess ei konvergeerunud ootuspäraselt. Selle põhjal tekkis hüpotees, et Gram-maatriksite arvutamine ja skaleerimine ning nende kasutamine stiili kaofunktsioonis ei olnud optimaalsed stabiilsuse saavutamiseks. Hüpoteesist lähtuvalt implementeeriti täiustatud Gram-maatriksi normaliseerimine, mis arvestab iga tunnuskaardi (*feature map*) dimensioonide kogumahtu. Kui esialgne Gram-maatriks oli normaliseerimata või ebaoptimaalselt normaliseeritud, siis uuendatud versioon normaliseeris Gram-maatriksi tunnuskaardi elementide koguarvuga (värvikanalid × kõrgus × laius). Koos Gram-maatriksi normaliseerimisega kohandati ka stiili kaofunktsiooni arvutust, kasutades keskmist ruutviga (MSE - *Mean Squared Error*) normaliseeritud Gram-maatriksite vahel, keskmistades üle

kõigi stiilikihtide. Tulemusena muutusid kao arvutused stabiilsemaks ja stiili kaalutegurite mõju paremini ennustatavaks, mis olid olulised sammud *NaN* vigade vältimiseks.

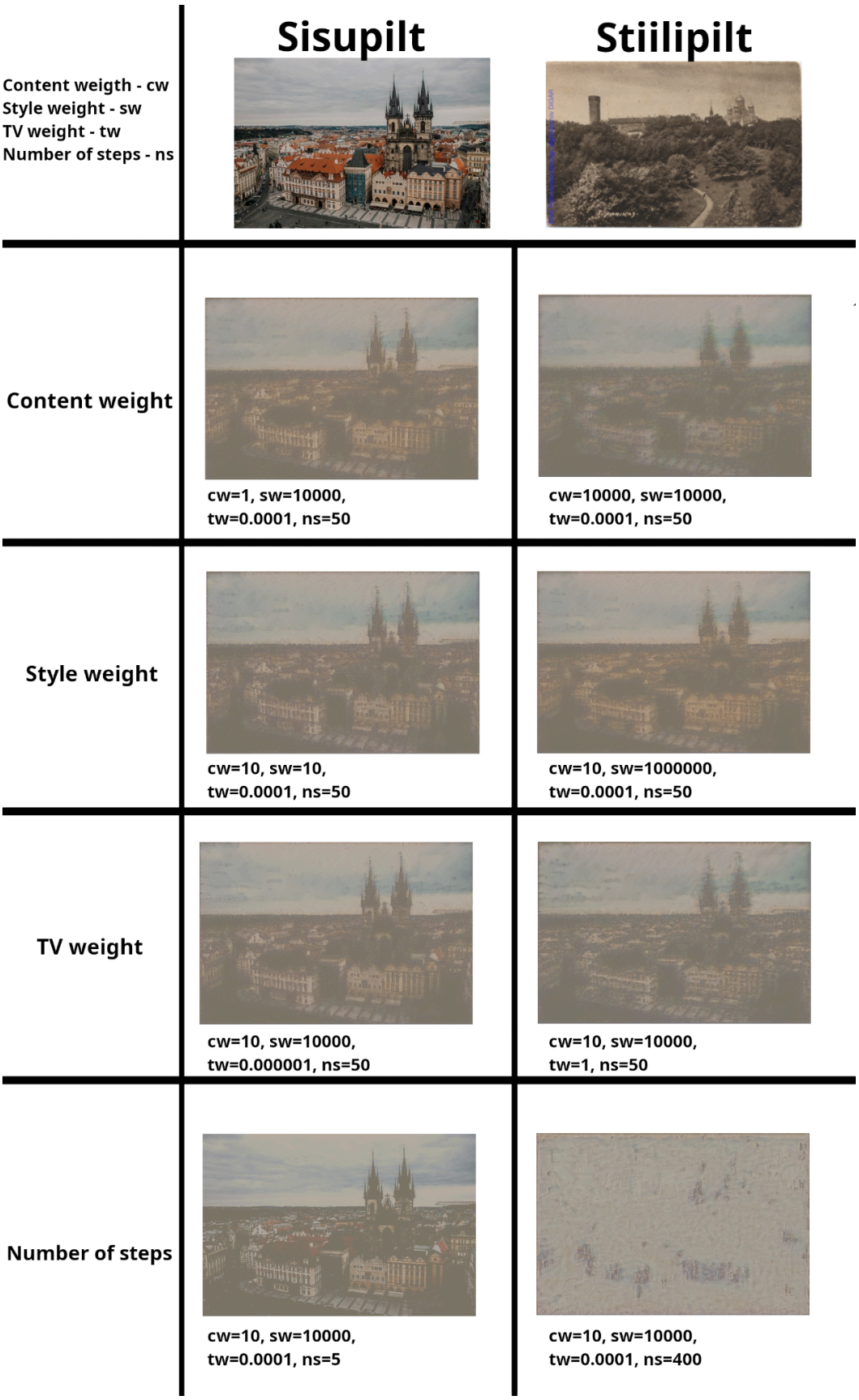
4.3.3 Optimeerijate eksperimendid ja parameetrite häälestamine

Mudeli arendamise käigus katsetati kahte erinevat optimeerijat: L-BFGS ja Adam.

Esialgu implementeeriti stiiliülekande algoritm kasutades L-BFGS optimeerijat, mis on teise järgu optimeerimismeetod ja potentsiaalselt võimeline saavutama paremat konvergentsi mõnes rakenduses [25]. See juhtus peamiselt seetõttu, et algoritm pani liiga suurt rõhku originaalpildi sisu säilitamisele, näidates, et kaalumissüsteem ei olnud korralikult tasakaalus. Parameetrite täiustamiseks katsetati erinevaid meetodeid:

- Stiili kaalu *(style_weight)* märkimisväärne suurendamine (nt 10^5-lt 10^7-le)
- Sisu kaalu *(content_weight*) vähendamine (nt 1.0-lt 0.1-le)
- TV kaalu (*tv_weight*) kohandamine.
- L-BFGS õppimiskiiruse *(learning_rate*) vähendamine (vaikeväärtuselt 1.0 kuni 0.1 või 0.01)

Vaatamata katsetustele muuta parameetrite suuruseid samasse suurusjärku ei õnnestunud L-BFGS optimeerijaga saavutada stabiilseid kuid visuaalselt mitte väga rahuldavaid tulemusi.

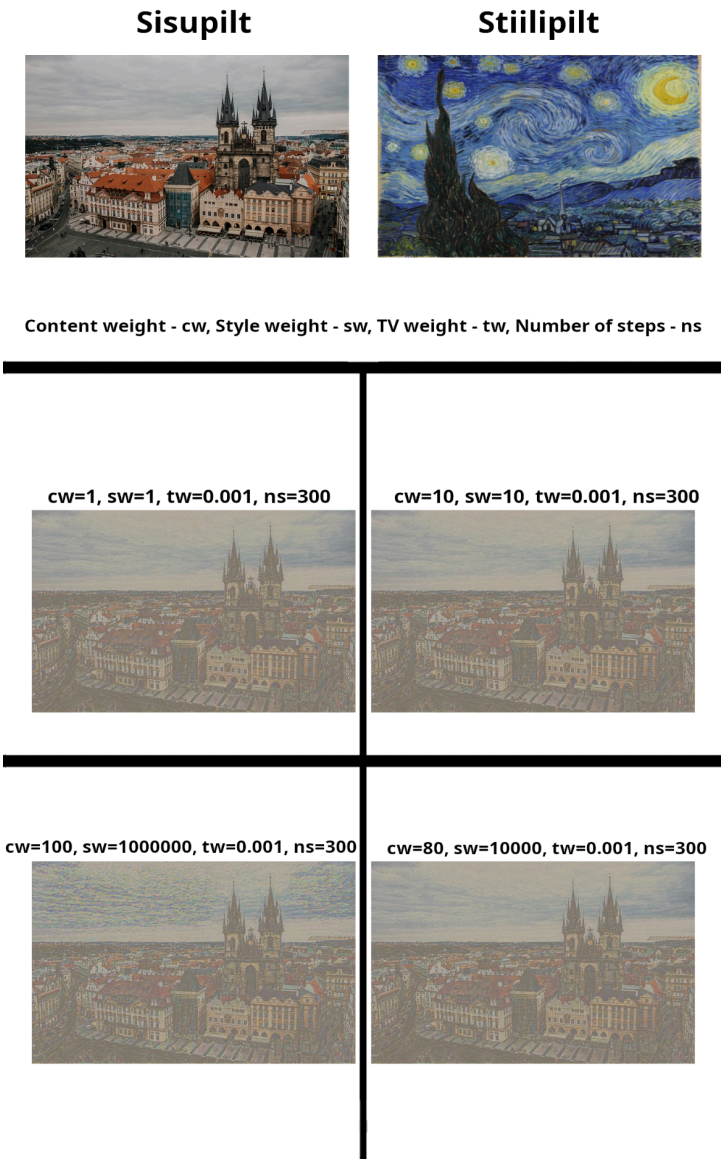


Pilt 4.3.1: Illustratsioon kuidas eri parameetrid muudavad väljundpilti (VGG-19, L-BFGS optimeering)

4.4.2 Optimisaatori vahetamise tulemused

Pärast L-BFGS-iga seotud raskusi otsustati katsetada Adam optimeerijat, mis on esimese järgu optimeerimismeetod ja tihti robustsem keeruliste mittekonvekssete optimeerimisprobleemide korral [24].

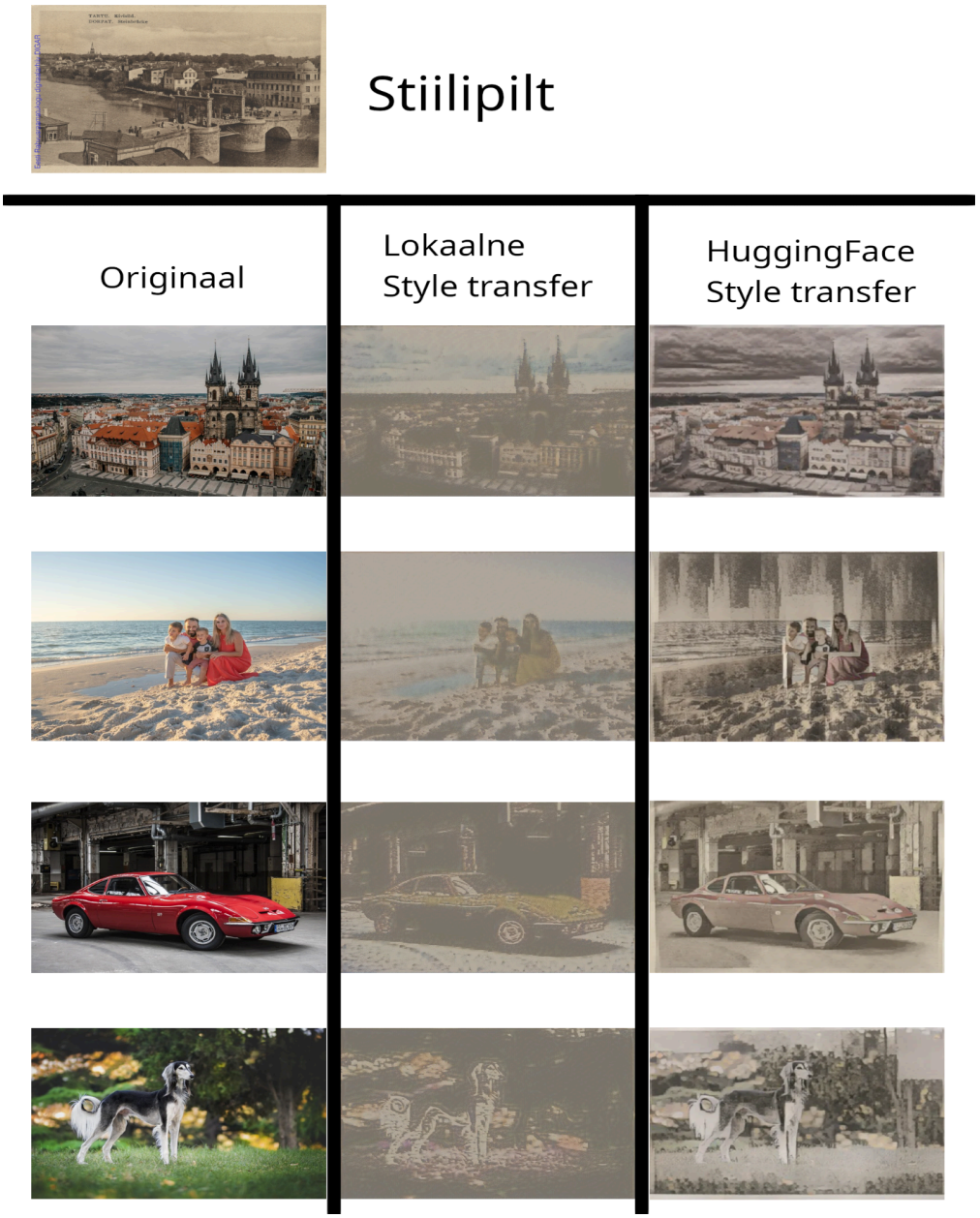
Adam optimeerija kasutusele võtmine andis märgatavalt stabiilsemaid tulemusi. Probleemiks osutus aga vajadus kasutada liiga suurt arvu iteratsioone. Et stiiliülekannet oleks märgata tuli iteratsioonide arvu suurendada, mis tähendas, et piltide genereerimine võttis vastavalt rohkem aega. Vahetulemused dokumenteeriti selgema stiilipildi kasutamisega, et hinnata stiiliülekande algoritmi töötavust.



Pilt 4.3.2: Illustratsioon kuidas parameetrid muudavad tulemust Adam optimiseeringuga.

4.4 Pilvepõhine mudel paralleellahendusena

Testimise osana implementeeriti veebirakenduses alternatiivne pilvepõhine *Hugging Face* stiiliülekande mudel, mis pakkus huvitavat võrdluspunkti lokaalse mudeli kõrval. See võimaldas kasutajal valida kohaliku töötluse ja pilveteenuse vahel, pakkudes paindlikkust erinevate ressursipiirangute korral.

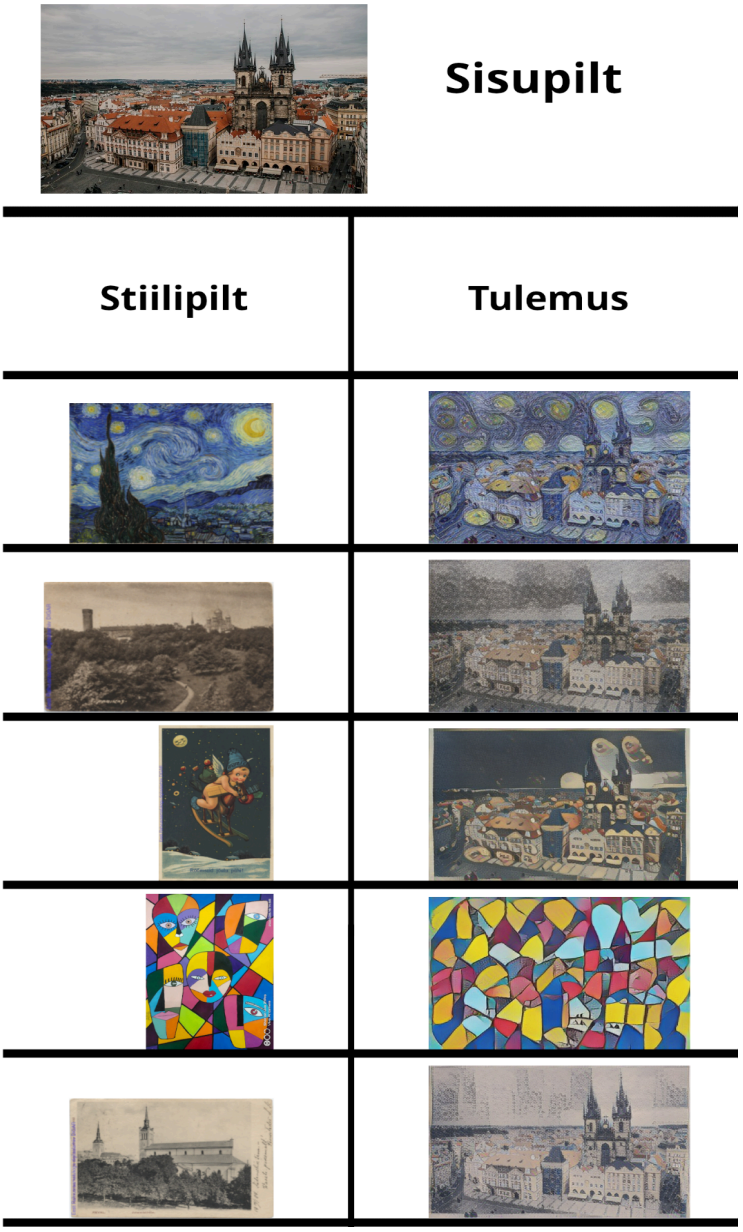


Pilt 4.4.1: Illustratsioon lokaalse stiiliülekande implementatsiooni (L-BFGS) ja veebipõhise *Hugging Face* stiiliülekande vahel.

4.5 Magenta stiili ülekande implementeerimine

Erinevalt VGG-19 mudelil põhinevast iteratiivsest lähenemisest on Magenta eeltreenitud mudel, mis on kohe pärast sisselugemist valmis stiiliülekannet teostama [18].

Magenta mudeli eelisteks olid oluliselt kiirem töötlemisaeg, kuna ei vaja iga pildipaari jaoks eraldi optimeerimisprotsessi, väiksem tundlikkus sisendpiltide kvaliteedi ja resolutsiooni suhtes ning stabiilsed tulemused erinevate sisendite korral. Samas Magenta puudusteks olid väiksem kontroll tulemuse üle, kuna eeltreenitud mudel ei võimalda nii detailset parameetrite häälestamist ja teatud stiilide puhul võib tulemus olla vähem detailne.



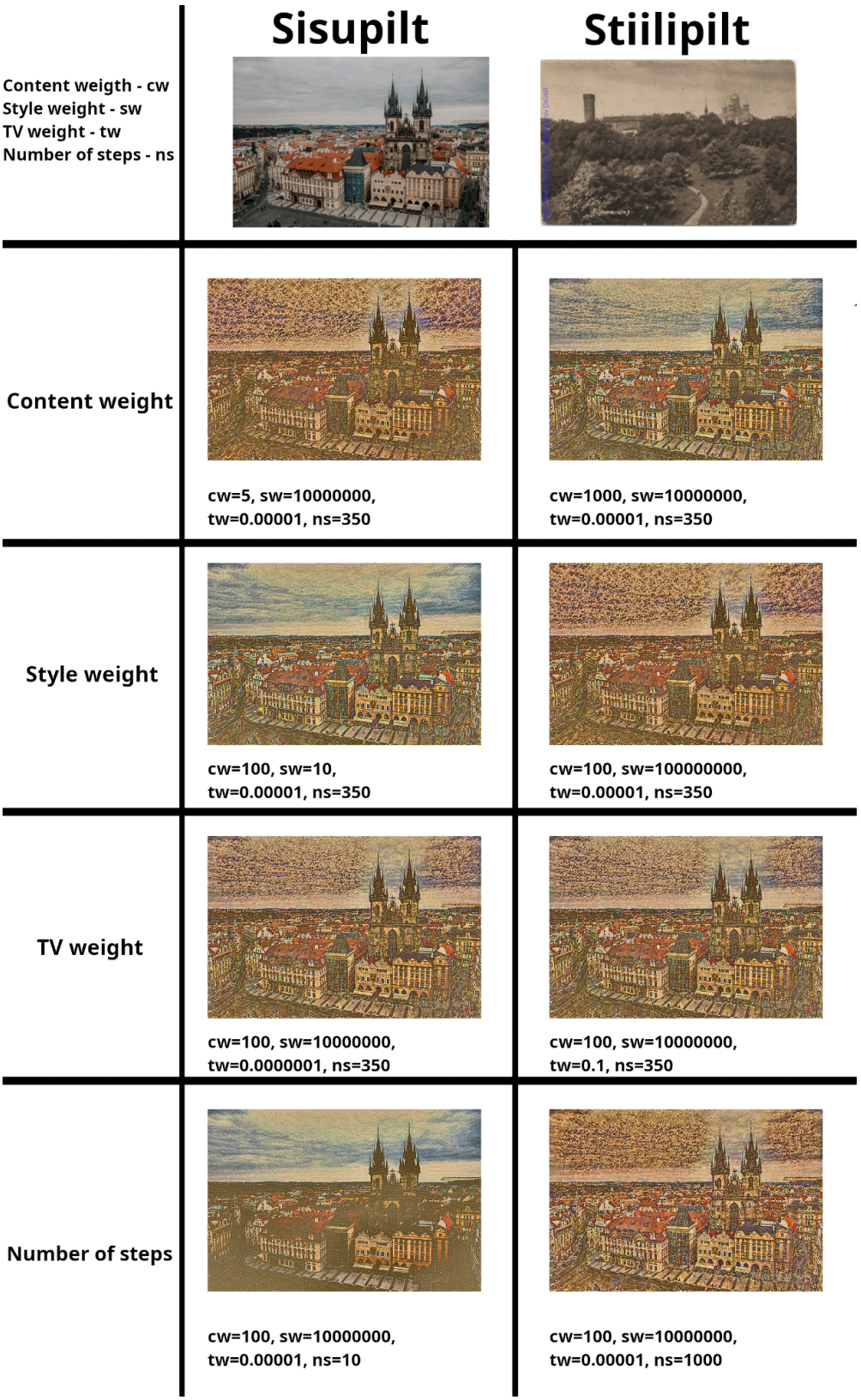
Pilt 4.5.1: Illustratsioon Magenta loodud väljunditest.

4.6 Järeltöötluse implementatsioon

Järjepideva testimise ja implementatsiooni parandamise tulemusena töötati välja lõplik lahendus, mis pildi normaliseerimisele ja tehisintellekti töötlusele lisaks sisaldab ka väljundpildi järeltöötlust. Järeltöötlus sisaldab kolme peamist komponenti:

1. **Pildi teravuse (*sharpness*) reguleerimine** - võimaldab välja tuua pildi detaile, mis stiiliülekande käigus võivad udusemaks muutuda.
2. **Pildi saturatsiooni (*saturation*) kohandamine** - aitab taastada või võimendada pildi värvide intensiivsust.
3. **Pildi kontrasti (*contrast*) täiustamine** - parandab pildi üldist selgust, tuues heleduse ja värvi erinevused paremini esile.

Järeltöötluse parameetrid on kasutajaliideses reguleeritavad, võimaldades lõpptulemust täpsemalt kohandada vastavalt kasutaja eelistustele ja konkreetse pildikombinatsiooni vajadustele.



Pilt 4.6.1: Lõpptulemuse illustratsioon kuidas erinevad parameetrid ja järeltöötlus muudavad väljundpilti (VGG-19, Adam optimaator).

5 Tulemuste analüüs

5.1 Parameetrite mõju analüüs stiiliülekande kvaliteedile

Eksperimentide käigus saadud tulemused kinnitasid osaliselt algseid hüpoteese parameetrite mõju kohta, kuid ilmnesid ka mitmed olulised kitsaskohad, mis vajavad põhjalikumat analüüsi. Esialgne hüpotees põhines järgmistel eeldustel:

- Sisu kaalu (*content weight*) suurendamine peaks säilitama rohkem sisupildi omadusi.
- Stiili kaalu (*style weight*) suurendamine peaks võimendama stiilipildi tekstuuri ja värvuse ülekandumist.
- Tulemuspildi siluvusparameetri (*tv weight*) suurendamine peaks tegema üleminekud sujuvamaks, kuid vähendama detailsust.

Praktilised katsetused (pildil 4.3.1 illustreeritud) näitasid, et parameetrite mõju oli komplekssem kui algselt eeldatud. Eriti märkimisväärne oli tähelepanek, et parameetrite üksikute väärtuste ekstreemsed muutused ei avaldanud tulemustele nii suurt mõju kui mitme parameetri samaaegne kohandamine. See viitab olulisele järeldusele: stiiliülekande kvaliteet sõltub olulisel määral parameetrite omavahelisest tasakaalust, mitte niivõrd üksikute väärtuste absoluutsuurustest.

5.2 Optimeerijate võrdlev analüüs

Stiiliülekande algoritmide optimeerimisel katsetatud kahte erinevat lähenemist – L-BFGS ja Adam optimeerijat – iseloomustasid märkimisväärselt erinevad tulemused.

L-BFGS, kui teise järgu optimeerimismeetod, peaks teoreetiliselt pakkuma kiiremat konvergentsi ja võimalust saavutada parem lõpptulemus võrreldes esimese järgu meetoditega [24]. Analüüsi põhjal võib järeldada, et L-BFGS optimeerija ebastabiilsus oli tingitud kulufunktsiooni omadustest, kaalutegurite tasakaalustamatusest ja Gram-matriksite hälvetest, mis tekkisid normaliseerimise tulemusena.

Adam optimeerija, kuigi teoreetiliselt väiksema konvergentsikiirusega, osutus praktikas stabiilsemaks vahendiks stiiliülekande optimeerimiseks, mida illustreerib pilt 4.3.2. Selle adaptiivne õppimiskiirus praktiliselt välistas *NaN* vigade esinemist. [26].

Optimeerijate analüüs viitab olulisele järeldusele tehisintellekti rakenduste arendamisel: teoreetiliselt võimekama meetodi praktiline kasutamine võib olla piiratud konkreetsete andmete ja probleemi omaduste tõttu. L-BFGS sobib potentsiaalselt paremini keskkondadesse, kus optimeeritavad funktsioonid on sujuvamad ja piisavalt stabiilsed. Adam aga suudab tulla toime ka ebastabiilsete funktsioonide ja mürasemate andmetega ning pakkus meie rakendusjuhul usaldusväärsemat lahendust.

5.3 Erinevate stiiliülekande lähenemiste võrdlus

Projekti käigus rakendatud kolm erinevat stiiliülekande lähenemist – VGG-19 põhine iteratiivne optimeerimismeetod, eeltreenitud Magenta mudel ja pilvepõhine *Hugging Face* implementatsioon – võimaldasid analüüsida nende tugevaid ja nõrku külgi.

VGG-19 põhise lähenemise analüüsil ilmnes, et olenemata püüdlustest optimeerida kulufunktsiooni ja normaliseerimistehnikaid, jäid tulemused oodatust tagasihoidlikumaks. Peamisteks põhjusteks olid**:**

1. Digari arhiivist pärit pisipiltide madal resolutsioon, mis ei võimaldanud algoritmil piisavalt täpselt tuvastada stiilielemente, mida Gram-maatriksite kaudu väljendada.
2. Iteratiivne lähenemine optimeerimisel nõudis hoolikat parameetrite häälestamist.
3. Ajalooliste postkaartide suur erinevus stiilides raskendas üldise optimeerimislähenemise kohandamist.

Pilvepõhise Hugging Face implementatsiooni tulemused (pildil 4.4.1 võrreldud) pakkusid huvitavat kontrasti lokaalse lähenemisega. Analüüs näitas, et pilvepõhised lahendused võivad pakkuda konkurentsivõimelist kvaliteeti ilma lokaalse arvutusvõimsuse piiranguteta, kuid nende tulemused võivad sisaldada defektseid stiilipildilt ülekantud mustreid.

Magenta mudeli kasutamine (tulemused näha pildil 4.5.1) pakkus paremaid tulemusi võrreldes VGG-19 põhise lähenemisega, vaatamata teoreetilisele eeldusele, et spetsialiseeritud iteratiivne lähenemine võiks pakkuda paremat kontrolli ja kvaliteeti. Magenta mudel on eeltreenitud laialdasel pildiandmestikul, mis võimaldab robustsemat stiilielementide tuvastamist ja ülekandmist. Mudel ei sõltu iteratiivsest optimeerimisest konkreetse pildipaari vahel, vaid rakendab üldistatud teadmisi stiiliülekande kohta, mistõttu on Magenta robustsem madala kvaliteediga või ebaselgete sisendite suhtes [18].

Seega, stiiliülekande kvaliteet sõltub märkimisväärselt konkreetsest rakenduskontekstist ja kättesaadavatest ressurssidest. Eeltreenitud mudelite kasutamine võib pakkuda stabiilsemat kvaliteeti kitsendatud ressursside tingimustes. Iteratiivne lähenemine võib potentsiaalselt pakkuda suuremat kohandatavust ideaaltingimustes.

5.4 Jõudluse ja ressursikasutuse analüüs

Alguses L-BFGS optimeerijaga piisas paarikümnest iteratsioonist ning tulemus saadi sekundite vältel. Hiljem, peale normaliseerimise loogika muutmist, kasvas iga optimeerimise sammu ajakulu mitmekordselt.

Magenta mudeli kasutamine pakkus märkimisväärselt paremat jõudlust ning algse implementatsiooniga võrreldavat kiirust. Tänu eeltreenitusele puudusid Magental pikad iteratiivsed optimeeringud, mis võimaldas kiire stiiliülekande.

Pilvepõhise implementatsiooni kiirust mõjutab võrgulatentsust ja kaugserveri koormus. Kui nendega pole probleeme, siis on pilvepõhise implementatsiooni kiirus võrreldav Magenta omaga.

5.5 Järeltöötluse mõju tulemuste kvaliteedile

Järeltöötluse implementeerimine (näha pildil 4.7.1) osutus oluliseks teguriks stiiliülekande tulemuste visuaalse kvaliteedi parandamisel. Analüüs näitas, et järeltöötluse mõju oli märkimisväärne VGG-19 põhise implementatsiooni puhul, kus stiiliülekanne ei saavutanud alati optimaalset värvi ja kontrasti tasakaalu.

Implementeeritud järeltöötluse komponentide (pilt 4.7.2) analüüs näitas erinevat mõju tulemuste kvaliteedile.

Järeltöötluse parameetrite kohandatavus kasutajaliideses andis kasutajatele täiendava kontrolli tulemuste üle, võimaldades kompenseerida stiiliülekande algoritmi piiranguid.

5.6 Edasiarenduse võimalused ja tulevikusuunad

Eesmärgiks oli luua veebirakendus, mida oleks võimalik tulevikus skaleerida ning edasi arendada. Potentsiaalseid edasiarenduse võimalusi, mida võiks tulevikus rakendada:

1. **Metaandmete korrastamise täiustamine** tehisintellekti abil, mis võimaldaks automaatselt tuvastada ja kategoriseerida postkaartide visuaalseid omadusi ilma käsitsi kodeerimisvajaduseta.
2. **Täissuuruses piltide integreerimine** metaandmestikku nõuaks täiendavat koostööd RaRa-ga, kuid pakuks märkimisväärselt paremat sisendmaterjali stiiliülekandeks.
3. **Stiiliülekande mudeli täiustamine** oleks võimalik mitmel viisil. Postkaartide piltide põhjal spetsiifilise mudeli treenimine, tekstiviibete (*text prompts*) integreerimine, Generatiivse adversariaalse võrgustiku (GAN) rakendamine ning alternatiivsete stiiliülekande meetodite integreerimine, näiteks Vision Transformer (ViT).
4. **Kasutajaliidese ja funktsionaalsuste edasiarendamine**, sealhulgas näiteks mitme pildi samaaegne töötlemine ja täiustatud järeltöötluse võimalused.

Need edasiarenduse võimalused pakuksid mitmekülgseid lahendusi praeguse süsteemi piirangute ületamiseks ja rakenduse funktsionaalsuse laiendamiseks.

6 Kokkuvõte

Käesoleva bakalaureusetöö raames valmis koostöös RaRa-ga - kes võimaldas postkaartide andmete kasutamise ning andis kasulikku sisulist infot - veebirakendus, mis utiliseerib *style transfer* tehisintellekti, et luua kasutaja piltide ja Digari arhiivi postkaartide vahel uusi ja huvitavaid pildi kooslusi. Töö kolm faasi - postkaartide sorteerimine ja normaliseerimine, veebirakenduse kasutajaliidese loomine ning tehisintellekti kasutamine uute piltide loomiseks nõudsid põhjalikku uurimistööd ja palju katsetamist, et veebirakendus efektiivselt ja eelduspäraselt tööle saada.

Eesmärgipäraselt valmis hariduslik eksponaat Rahvusraamatukogu õpitubadesse ning vabalt kasutatav veebirakendus Digari veebilehele. Nõuete kohaselt on veebirakendus loodud skaleeritavana, seega on seda võimalik tulevikus edasi arendada ning sellele lisavõimalusi implementeerida.

Viited

[1] Karthikeyan, J., Su Hie, T., Yu Jin, N. (2021). Learning Outcomes of Classroom Research, India: L Ordine Nuovo Publication, Kasutatud 20.05.2025

[2] Karan, O. (2023), Understanding the Central Processing Unit (CPU): The Heart of the Computer, *American Journal of Computer Science and Engineering Survey*, 30. august. Kasutatud 20.05.2025, https://www.primescholars.com/articles/understanding-the-central-processing-unit-cpu-the-heart-of-the-computer-124180.html

[3] Caulfiel, B. (2009). What's the Difference Between a CPU and a GPU? *Nvidia*, 16. detsember. Kasutatud 12.04.2025. https://blogs.nvidia.com/blog/whats-the-difference-between-a-cpu-and-a-gpu/

[4] Khan, N. (2013), Emerging Trends in OAI-PMH Application, *Design, Development and management of Recources for Digital Library Services* (lk. 147-148), november. Kasutatud 20.05.2025, https://www.igi-global.com/chapter/content/72454

[5] Gillis, A. (2021), What is VRAM?, *TechTarget*, juuli. Kasutatud 12.05.2025, https://www.techtarget.com/searchstorage/definition/video-RAM

[6] Simonyan, K., Zissrman, A. (2025), Very Deep Convolutional Networks for Large-Scale Image Recognition, arxiv, 10. aprill. Kasutatud 20.05.2025, https://arxiv.org/abs/1409.1556

[7] Moritz, P., Nisihihara, R., Jordan, M. (2016), A Linear-Convergent Stochastic L-BFGS Algorithm, *Proceedings of Machine Learning Research*, Kasutatud 20.05.2025, https://proceedings.mlr.press/v51/moritz16.html

[8] What Is an AI Model? *IMB*, Kasutatud 17.05.2025, https://www.ibm.com/think/topics/ai-model

[9] Upadhye, S. (2024), AI in Image Editing and Enhancement: Revolutionizing Photography, *International Journal of Modern Engineering & Management Research*, juuni. Kasutatud 20.05.2025, https://www.ijmemr.org/Publication/V12I2/IJMEMR-V12I2-001.pdf

[10] [ujjwalkarn] (2016), A Quick Introduction to Neural Networks, *Ujjwal Karn's blog*, 9. august. Kasutatud 17.05.2025, https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/

[11] Guinness H. (2025). The 8 best AI image generators 2025. *Zapier* 3. aprill. Kasutatud 3.05.2025, https://zapier.com/blog/best-ai-image-generator/

[12] Ashar K. (2025). Best Generative Fill Alternatives for 2025. *smartli* 20. veebruar. Kasutatud 3.05.2025, https://www.smartli.ai/blog/best-generative-fill-alternatives-for-2025

[13] Bie, F., Yang, Y., Zhou, Z., Ghanem, A., Zhang, M., Yao, Z., Wu, X., Holmes, C., Golnari, P., Clifton, D., He, Y., Tao, D., Song, S. (2023) RenAIssance: A Survy into AI Text-to-Image Generation in the Era of Large Model, *arxiv*, 2. september. Kasutatud 20.05.2025, https://arxiv.org/abs/2309.00810

[14] Hoyez, H., Schhockaert, C., Rambach, J., Mirbach, B., Stricker, D. (2022), Unsupervised Image-to-Image Translation: A Review, *sensors*, 6. november. Kasutatud 20.05.2025, https://www.mdpi.com/1424-8220/22/21/8540

[15] Gatys, L., Ecker, A., Bethge, M. (2016), Image Style Transfer Using Convolutional Neural Networks, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, juuni. Kasutatud 20.05.2025, https://openaccess.thecvf.com/content_cvpr_2016/html/Gatys_Image_Style_Transfer_CVPR_2016_paper.html

[16] What is Axios? (2024). *GeeksforGeeks* 30. august. Kasutatud 5.05.2025, https://www.geeksforgeeks.org/what-is-axios/

[17] Serrant D. (2024). xml2js npm (How it works for developers). *ironsoftware* 29. september. Kasutatud 5.05.2025, https://ironpdf.com/nodejs/blog/node-help/xml2js-npm/

[18] Xu, W. (2024). A comparative analysis of VGG19 and Magenta models for neural style transfer, *ResearchGate* märts. Kasutatud 11.05.2025, https://www.researchgate.net/publication/379180556_A_comparative_analysis_of_VGG19_and_Magenta_models_for_Neural_Style_Transfer

[19] Tokui, S., Okuta, R., Akiba, T., Niitani, Y., Ogawa, T., Saito, S., Suzuki, S., Uenishi, K., Vogel, B., Vincent, H. (2019), Chainer: A Deep Learning Framework for Accelerating the Research Cycle, *arxiv*, 1. august. Kasutatud 20.05.2025, https://arxiv.org/pdf/1908.00213

[20] [Farheenshaukat] (2024). Difference between VGG16 and VGG19, *kaggle* juuli. Kasutatud 12.05.2025, https://www.kaggle.com/discussions/getting-started/518971

[21] Zheng, Y., Yang, C., Merkulov, A. (2018), Breast cancer screening using convolutional neural network and follow-up digital mammography, *ReasearchGate*, Kasutatud 17.05.2025, https://www.researchgate.net/publication/325137356_Breast_cancer_screening_using_convolutional_neural_network_and_follow-up_digital_mammography#pf8

[22] Gatys, L., Ecker, A., Bethge, M. (2021), A Neural Algorith of Artistic Style, *Cornell University*, 26. august. Kasutatud 10.05.2025, https://arxiv.org/abs/1508.06576

[23] Shen, F., Yan, S., Zeng, G. (2018), Neural Style Transfer via Meta Networks, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, juuni. Kasutatud 20.05.2025, https://openaccess.thecvf.com/content_cvpr_2018/html/Shen_Neural_Style_Transfer_CVPR_2018_paper.html'

[24] Kashyap, R. (2023), A survey of deep learning optimizers - first and second order methods, arxiv, 27. september. Kasutatud 20.05.2025, https://arxiv.org/abs/2211.15596

[25] Skajaa, A. (2010). Limited Memory BFGS for Nonsmooth Optimization, Magistritöö, New York University, Mathematical Science.

[26] Kingma, D., Ba, J. (2017), Adam: A Method for Stochastic Optimization, 30. jaanuar. Kasutatud 15.05.2025 https://arxiv.org/pdf/1412.6980

Lisad

Veebirakenduse lähtekoodi repositoorium:
https://github.com/NimbusMcManne/RaRa_AI_Postcard_Generation.git

Litsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Gregory Kuusmik**

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

   "**Postkaartide loomine tehisintellektiga**"

   mille juhendaja on **Ardi Tampuu**

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace'i kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ¨arieesm¨argil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Gregory Kuusmik
19.05.2025