Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2025

Voitto Vuorio

Paikkatiedoilla koulutetun GAN-verkoston kuvasta-kuvaan muunnos arkkitehti- ja kaupunkisuunnittelun työkaluna

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Opinnäytetyö (AMK) | Tiivistelmä

Turun ammattikorkeakoulu

Tieto- ja viestintätekniikka

2025 | 55 sivua

Voitto Vuorio

Paikkatiedoilla koulutetun GAN-verkoston kuvasta-kuvaan muunnos arkkitehti- ja kaupunkisuunnittelun työkaluna

Kaupunkisuunnittelu hyödyntää yhä enemmän digitaalisia työkaluja monimutkaisten suunnitteluprosessien tukemiseksi. Opinnäytetyössä kehitettiin prototyyppisovellus arkkitehti- ja kaupunkisuunnittelun avuksi toimeksiantona arkkitehtitoimisto Ark Brut Oy:lle. Tavoitteena oli tutkia generatiivisen tekoälyn soveltuvuutta visuaaliseen rakennussijoitteluun ja tuottaa yksinkertainen työkalu. Työssä käytettiin EU:n avoimen datan direktiivin (2019/1024/EU) tukemaa MML:n aineistoa. Prototyyppisovellukseen koulutettiin Pix2Pix-arkkitehtuurin mukainen generatiivinen kilpaileva verkko (GAN) kuvasta-kuvaan muunnoksen suorittamiseen. Aineisto koostui tonttien rajaviivoista ja niitä vastaavista rakennuksista.

Koulutettu malli tuotti rakennusten jalanjälkiä tonttien rajojen sisälle vaihtelevalla tarkkuudella. Tulokset osoittivat, että malli oppi sijoittamaan yksinkertaisia geometrisia muotoja rajojen läheisyyteen, mutta muotojen yksityiskohtien generoinnissa oli haasteita. Sovellus osoitti potentiaalia alustavassa visualisoinnissa, mutta käytännön hyöty edellyttäisi jatkokehitystä, kuten laajempaa monipuolisempaa tietojoukkoa ja jälkikäsittelymenetelmiä.

Asiasanat:

syväoppiminen, kaupunkisuunnittelu, generatiivinen kilpaileva verkosto, avoin data, tietojoukko, paikkatietoaineisto, ohjelmistokehitys

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Information and Communications Technology

2025 | 55 pages

Voitto Vuorio

Automatic building placement within plot boundaries using GAN networks and Image-to-Image translation

Urban planning increasingly utilizes digital tools to support complex design processes. In this thesis, a prototype application was developed to assist architectural and urban planning as a commission for the architectural firm Ark Brut Oy. The objective was to investigate the suitability of generative artificial intelligence and to produce a simple tool that achieves visualization, where buildings are fit into plot boundaries. The work utilized open geospatial data from the National Land Survey of Finland, the availability of which is supported by the EU’s Open Data Directive (2019/1024/EU).

In the prototype application, a generative adversarial network (GAN), specifically the Pix2Pix architecture, was used to perform an image-to-image translation. The model was trained on a dataset of image pairs consisting of plot boundaries and corresponding buildings from the matching area, collected from the National Land Survey’s APIs. The results showed that the model learned to place simple geometric shapes near plot boundaries, but the precision and detail remained limited. The application demonstrated potential for preliminary visualization, but its practical utility requires further development, such as a broader, more complex dataset and more refined post-processing methods.

Keywords:

generative adversarial network, image-to-image translation, dataset, open data, deep learning, software development, urban planning

Sisältö

Käytetyt lyhenteet **7**

1 Johdanto **9**

2 Data ja tekoäly rakennusalalla **11**
2.1 Rakennusalan digitalisaatio 11
2.2 Suunnitteluprosessin vaiheet ja datan rooli 12
2.3 Avoimen datan merkitys ja saatavuus 12
2.4 Tekoälypohjaiset generatiiviset suunnittelutyökalut 15

3 Sovelluksen suunnittelu ja rakenne **17**
3.1 Alkuperäinen ajatus 17
3.2 Siirtyminen tekoälypohjaiseen ratkaisuun 18
3.3 Työkalun toimintaperiaate 19
3.4 Rakenne ja tietoliikenne 21

4 Käytetyt menetelmät ja teknologiat **23**
4.1 Tekoälymalli 23
4.1.1 Tekoälymallin rakenne 23
4.1.2 Neuroverkot 24
4.2 Työn tekoälymalli 25
4.2.1 Pix2pix ja GAN 26
4.2.2 Tietojoukon rooli 27
4.2.3 Kuvasta kuvaan muunnos 28
4.3 Tietojoukon kokoaminen ja esikäsittely 28
4.3.1 Datan hakeminen rajapinnoista 29
4.3.2 Koordinaatistot ja XML-spesifikaatio 30
4.3.3 Kuvapariaineisto 31
4.3.4 Suodatusmenetelmät 32
4.3.5 Vektoridata 32
4.4 Mallin koulutus 33
4.4.1 Koulutusympäristö ja parametrit 34

4.4.2 Tietojoukon käyttö 35

5 Tulokset ja analyysi **39**
5.1 Tietojoukkoon liittyvät valinnat 39
5.2 Koulutuksen tuloksia selkokartta tietojoukolla 41
5.3 Koulutuksen tulokset vektoripiirretyllä tietojoukolla 42
5.4 Generoitujen kuvien jälkikäsittely 46
5.5 Lopullinen sovellus 47

6 Johtopäätökset **49**
6.1 Saavutukset ja menetelmän arviointi 49
6.2 Työn merkitys ja oppimiskokemus 50
6.3 Johtopäätökset 50
6.4 Jatkokehitysmahdollisuudet 50

Lähteet **52**

Kuvat

Kuva 1. Oulun vireillä olevia asemakaavoja. (Oulun kaupunki, 2024). 14
Kuva 2. Helsingin 3D-kaupunkimalli, rautatieasema. (Helsingin karttapalvelu, 2024). 15
Kuva 3. Forman tarjoama esittely demo. (Autodesk Forma, 2024). 16
Kuva 4. Generatiivisesti tuotettuja tonttivaihtoehtoja. (Autodesk Forma 2024). 16
Kuva 5. Sovelluksen aikaisen vaiheen käyttöliittymä ”mockup”. 17
Kuva 6. Syöte- ja tulosalue käyttöliittymässä. 20
Kuva 7. Esimerkki kuvapari, jossa kiinteistöraja ja selkokartta. (MML, 2024). 21
Kuva 8. Sovelluksen tietoliikennekaavio. 22
Kuva 9. Erään neuroverkkotyypin kerroksia ja rakenne. 25
Kuva 10. GAN verkon arkkitehtuuri. (AI+EDL Blog, UNL College of Architecture, 2019). 26
Kuva 11. Edges2shoes tietojoukon kuvapareja. (Yu & Grauman, 2014). 27
Kuva 12. Kuvasta-kuvaan-muunnoksia Pix2Pix -paperissa. (Isola et al, 2017). 28

Kuva 13. Postman sovelluksella suoritettu GET-kutsu kiinteistöraja-rajapintaan. 29
Kuva 14. Python -koodi indeksilukujen laskemiseen. 30
Kuva 15. Vektori-rasteri tietojoukon kuvapari. (MML, 2024). 31
Kuva 16. Vektori-vektori tietojoukon kuvapari. (MML, 2024). 31
Kuva 17. Näyteotos koulutusvaiheesta paikkatietodatalla. 50. epookki. 36
Kuva 18. Eri esitysmuotojen kandidaatteja kuvapareiksi. 39
Kuva 19. Kuvaparissa liian suuri mittakaava. Ei kerrostaloja. 40
Kuva 20. Osittainen rajaviiva. Kohdekuvassa tekstiä ja vähän rakennuksia. 40
Kuva 21. Rajaviivat-Selkokartta tietojoukon näyte, 7 900. koulutusaskel. 41
Kuva 22. Generoituja artifaktinäytteitä, 10 300. koulutusaskel. 42
Kuva 23. Mallin generoimia "läiskiä". 43
Kuva 24. Vektoripiirretty tietojoukon näyte, 6 600. koulutusaskel. 44
Kuva 25. Rajaviivojen puute generoinnin tuloksissa. 44
Kuva 26. Viimeisemmät generoinnin tulokset. 45
Kuva 27. Sisääntulo - Generoinnin tulos – Sijoitus sisääntulon päälle 46
Kuva 28. Lopullinen suorakulmiomuunnoksella jälkikäsitelty tulos. 46
Kuva 29. Sovelluksen lopullinen käyttöliittymä. 48

Käytetyt lyhenteet

Adam	Stokastisen optimoinnin menetelmä
AEC	Architecture, Engineering and Construction, arkkitehtuuri-, insinööri- ja rakennusala
API	Application Programming Interface, rajapinta
BIM	Building Information Modeling, rakennustietomallinnus
CAD	Computer-Aided Design, tietokoneavusteinen suunnittelu
CRS	Coordinate Reference System, koordinaatistojärjestelmä
CUDA	Compute Unified Device Architecture, NVIDIA:n rinnakkaislaskenta-alusta
ETRS-TM35FIN	Suomen kansallinen tasokoordinaattijärjestelmä
GAN	Generative Adversarial Network, generatiivinen kilpaileva verkosto
Geopandas	Python-kirjasto geospatiaalisen datan käsittelyyn
GeoJSON	Tiedostomuoto geospatiaalisen vektoridatan tallentamiseen
GPU	Graphics Processing Unit, grafiikkaprosessori
HTML	HyperText Markup Language, verkkosivujen merkintäkieli
HTTP	Hypertext Transfer Protocol, verkkoliikenneprotokolla
HVD	High-Value Datasets, EU:n määrittämät arvokkaat tietoaineistot
JSON	JavaScript Object Notation, kevyt tiedostomuoto tiedonvälitykseen

JHS 180	Julkisen hallinnon suositus paikkatiedon sisältöpalveluista
Matplotlib	Python-kirjasto visualisointien luomiseen
MML	Maanmittauslaitos
MSE	Mean Squared Error, keskineliövirhe
Node.js	JavaScript-ajoympäristö palvelinpuolen sovelluksille
OSM	OpenStreetMap, avoin yhteistyöllinen kartoitushanke
PATINE	Paikkatietoasiain neuvottelukunta
PatchGAN	Erityinen diskriminaattoriarkkitehtuuri GAN-verkoissa
PBF	Protocolbuffer Binary Format, tehokas binäärimuoto datan serialisointiin
Pillow	Python-kuvankäsittelykirjasto
Pix2Pix	GAN-verkkoihin perustuva malli kuvasta-kuvaan muunnosten toteuttamiseen
Postman	Työkalu API-rajapintojen testaamiseen
PyTorch	Avoimen lähdekoodin koneoppimiskehys
REST	Representational State Transfer, rajapintatyyppi
Skip-yhteys	Neuroverkkoarkkitehtuurin rakenneosa
U-Net	Eräs neuroverkkotyyppi, konvoluutioverkkoarkkitehtuuri
VRAM	Video Random Access Memory, näytönohjaimen muisti
WMTS	Web Map Tile Service, standardi esitallennettujen karttatiilien jakeluun

1 Johdanto

Tekoäly on mullistanut teknologian kehitystä ja ongelmanratkaisua laajasti eri aloilla, mukaan lukien arkkitehtuuri ja kaupunkisuunnittelu. Kehityksen taustalla ovat teknologiset edistysaskeleet, kuten tehokkaat näytönohjaimet ja erikoistuneet tekoälyprosessorit (Huang et al., 2020) sekä ohjelmakirjastojen ja materiaalien saatavuus. Avoimet tietoaineistot, kuten EU:n avoimen datan direktiivin (2019/1024) mukaiset paikkatietoaineistot, tukevat uusien datalähtöisten innovaatioiden kehittämistä. Datan yhdistäminen tekoälyyn on tunnistettu keskeiseksi politiikkatavoitteeksi (Open Government Partnership, n.d.). Avoimen paikkatiedon saatavuus on hyödyllistä rakennusalalla kaupunkeihin liittyvässä tutkimuksessa, kaupunki-, arkkitehti- ja rakennussuunnittelun ratkaisuissa (European Commission, 2019).

Opinnäytetyön tavoitteena on tutkia generatiivisten adversaaliverkkojen soveltuvuutta ja kehittää tekoälyä hyödyntävä prototyyppisovellus. Työ hyödyntää avointa dataa tietojoukon kokoamiseen, mikä sisältää rakennusten pohjia karttatasoilla sekä kiinteistörajoja. Työn tarkoituksena ei ole kehittää uutta tekoälyarkkitehtuuria, vaan kokeilla Pix2Pix-menetelmän soveltamista suomalaiseen paikkatietokontekstiin. Tavoitteena on kehittää työkalu arkkitehdeille ja kaupunkisuunnittelijoille ja selvittää sen mahdollista hyötyä rakennusten sijoitteluvaihtoehtojen visualisoinnissa.

Prototyyppi käyttää generatiivista adversaaliverkkoa (GAN), joka muuntaa käyttäjän syöttämät tontin rajat pohjapiirroksiksi, joissa näkyvät rajat ja niihin sijoitetut rakennukset. Mallin kouluttamista varten tarvittiin tietojoukko (engl. Dataset), joka kerättiin Maanmittauslaitoksen avoimen datan rajapinnasta. Tietojoukko koostuu aiempien tutkimusten esimerkkejä mukaillen kuvapareista (Isola et al., 2017), jotka tässä työssä ovat tontin rajat sekä vastaavat rakennussuunnitelmat.

Työ jakautuu johdannon jälkeen viiteen sisältö lukuun, ja loppulukuun. 2. luvussa tarkastellaan lyhyesti tekoälyä rakennusalan kontekstissa. Käydään pääpiirteittäin läpi arkkitehtien nykyaikainen suunnitteluprosessi sekä avoimen

datan merkitys. 3. luvussa käsitellään prototyyppisovelluksen suunnitteluvaihe ja toimintaperiaate. 4. luku kattaa työssä käytetyt menetelmät ja teknologiat, tekoälyn teoriaa ja tietojoukon roolin. 5. luvussa esitellään konkreettisimmat tekoälyllä generoidut tulokset, tietojoukkoon liittyviä valintoja, sekä sovelluksen lopullinen ulkomuoto.

Loppuluvussa arvioidaan työn onnistumista sen tavoitteisiin nähden, sovelluksen hyödyllisyyttä nykytilassa ja käydään läpi tekemisen aikana vastaan tulleita ajatuksia sekä jatkokehitysmahdollisuuksia.

Aiheen valinta perustuu aikaisempaan kosketukseen arkkitehtuuri- ja kaupunkisuunnittelun kanssa ja kiinnostuksesta ohjelmointiin ja sovelluskehitykseen. Tekoälyn nopea kehitys muuttaa suunnittelualaa (Rahimian et al., 2021), ja sen vuoksi on tärkeää tutkia, miten avoimet tietoaineistot ja tekoäly voivat tukea käytännön työkaluja.

2 Data ja tekoäly rakennusalalla

Rakennusala, johon kuuluvat arkkitehtuuri, insinööritoiminta ja rakentaminen (AEC-ala), on perinteisesti ollut ala, jossa suunnittelu- ja toteutusprosessit ovat monivaiheisia ja vaativat laajaa erikoisosaamista. Viime vuosikymmeninä digitalisaatio on muokannut syvällisesti alan toimintatapoja, tuoden mukanaan uusia työkaluja ja menetelmiä. (Eastman 2008). Samanaikaisesti tekoälyn ja koneoppimisen kehitys on alkanut avata uusia näkymiä myös AEC-alan prosesseihin ja mahdollistaa tehostamista, innovaatioita ja datan parempaa hyödyntämistä. (Sacks et al., 2018). Tässä luvussa tarkastellaan rakennusalan digitalisaation nykytilaa, suunnitteluprosessin luonnetta, datan merkitystä sekä avoimen paikkatietodatan ja tekoälyn näkyvyyttä alan nykyaikaisissa työkaluissa.

2.1 Rakennusalan digitalisaatio

Suunnittelutoimistot ovat käytännössä siirtyneet lähes kokonaan digitaalisten työkalujen aikakauteen, mikä heijastuu kaikissa suunnitteluprosessin vaiheissa. Tämä muutos alkoi voimistua 1990-luvulla, kun tietokoneavusteinen suunnittelu (CAD) ja myöhemmin rakennuksen tietomallinnus (BIM) alkoivat vakiinnuttaa asemansa alan standardityökaluina (Eastman, 2008)

Tietokoneavusteiset suunnitteluohjelmistot (Computer-Aided Design, CAD) mullistivat aluksi kaksiulotteisen piirtämisen ja dokumentoinnin, korvaten perinteiset piirustuslaudat ja manuaaliset menetelmät. CAD-ohjelmistot mahdollistivat tarkemman, nopeamman ja helpommin muokattavan suunnitteludokumentaation tuottamisen. Alan johtavia CAD-ohjelmistoja ovat olleet esimerkiksi Autodeskin AutoCAD ja Bentley Systemsin MicroStation. (Kousa, 2012).

Rakennuksen tietomallinnus (Building Information Modeling, BIM) vei digitalisaation askeleen pidemmälle. BIM ei ole pelkästään 3D-mallinnusta, vaan se on prosessi, jossa luodaan ja hallitaan rakennuksen digitaalista

esitystä, joka sisältää geometrian lisäksi myös runsaasti ei-graafista tietoa (attribuutteja) rakennusosista, materiaaleista, kustannuksista ja aikatauluista. BIM-mallit mahdollistavat paremman yhteistyön eri suunnittelualojen välillä, törmäystarkastelut, määrälaskennat ja rakennuksen elinkaaren aikaisen tiedonhallinnan. Keskeisiä BIM-ohjelmistoja ovat esimerkiksi Graphisoftin Archicad ja Autodeskin Revit. (Nordic BIM).

Nämä toimialastandardeiksi muodostuneet ohjelmistot ovat vuosikymmenten kehitystyön tulosta ja tarjoavat suunnittelutoimistoille tehokkaat välineet luonnosten muuntamiseen tarkoiksi ja informatiivisiksi rakennussuunnitelmiksi. Suomessa BIM-standardien merkitys on korostunut erityisesti ISO 19650 -standardin myötä, joka on keskeinen standardi rakennusalan tietomallinnukselle (Pietilä, 2021).

2.2 Suunnitteluprosessin vaiheet ja datan rooli

Rakennusalan suunnitteluprosessi on iteratiivinen ja koostuu useista vaiheista, joissa digitaaliset työkalut ja data ovat avainasemassa. Prosessi käynnistyy tarveselvityksellä, jossa kerätään tiedot tontin ominaisuuksista, ympäristöstä, asiakkaan tarpeista ja budjetista (RT 10-11222, 2016). Seuraavassa vaiheessa luonnostellaan rakennusten pohjapiirroksia ja niiden sijoittelua tontille, minkä jälkeen suunnitelmat tarkentuvat laskelmien ja teknisten määreiden kautta. Lopuksi suunnitelma viimeistellään rakennuslupavaihetta varten, ja hyväksynnän jälkeen se etenee toteutukseen.

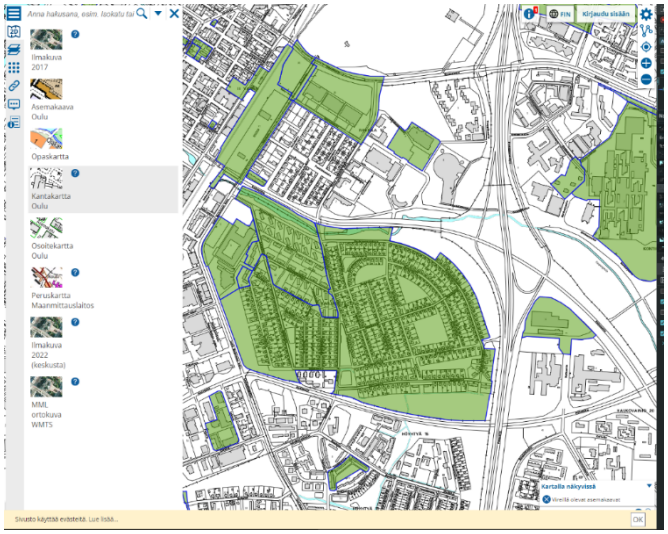
2.3 Avoimen datan merkitys ja saatavuus

Avoin data tarkoittaa Euroopan unionin määritelmän mukaan tietoa, jota kuka tahansa voi vapaasti käyttää, muokata ja jakaa mihin tahansa tarkoitukseen. Tyypillisesti ainoina ehtoina avoimen datan käytölle ja jakamiselle ovat lähdemerkintä ja datan jakaminen samoin ehdoin kuin millä se oli alun perin jaettu ”share-alike" tai ”copyleft”-lisenssin mukaisesti.

Euroopan Unionin avoimen datan direktiivi (2019/1024/EU), velvoittaa julkisen sektorin toimijoita asettamaan tietyt "arvokkaat tietoaineistot" (High-Value Datasets, HVD) saataville maksutta, koneluettavassa muodossa ja selkeiden rajapintojen kautta. Direktiivin tavoitteeksi on mainittu muun muassa talouskasvun edistäminen sekä yhteiskunnallisen osallistumisen helpottaminen. (The World Bank, 2014). Elinkeinoelämän tutkimuslaitos Etla sekä Open Knowledge Finland ry:n tutkimuksen mukaan avointa dataa ja massadataa hyödyntäneet yritykset tekevät muita yrityksiä huomattavasti todennäköisemmin uusia tuoteinnovaatioita. (Avoindata.fi, 2023).

Direktiivi tukee vahvasti myös tämän opinnäytetyön kaltaisia projekteja ja hankkeita, ja on suuri askel digitalisaation innovaatiohyötyjen realisoimiseen. Avoimet paikkatietoaineistot, kuten Maanmittauslaitoksen kartta- ja kiinteistörajadata tarjoavat ylläpidetyn kansallisen maastotieto- sekä kartta-aineisto tietokannan. Näiden rajapintojen kehittämisessä ja yhtenäistämisessä noudatetaan paikkatietoasiain neuvottelukunnan julkaisemia ohjeistuksia (PATINE, 2020). Aineistoja on myös muunlaisia, kuten ilmakuvia ja laserkeilausaineistoja koko Suomen alueelta (MML Verkkosivut, 2024). Suuri osa näistä aineistoista on direktiivin mukaisesti saatavilla avoimina lisensseinä MML:n rajapintapalveluista ja latauspalvelusta.

Avointa dataa näkyy tyypillisesti käytössä esimerkiksi kaupunkien karttasovelluksissa, kuten kuvassa 1, jotka kytkeytyvät edellä mainittuihin aineistoihin.



Kuva 1. Oulun vireillä olevia asemakaavoja. (Oulun kaupunki, 2024).

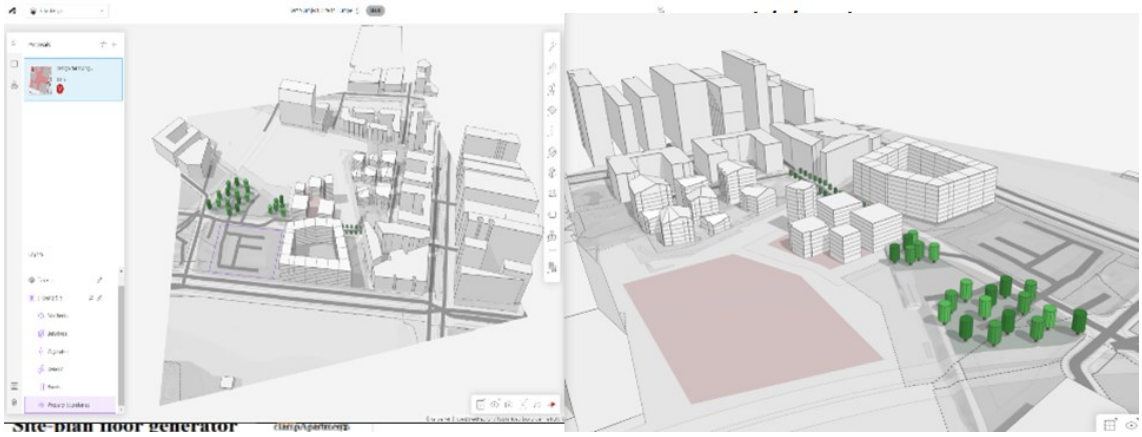
Aineistojen dataa kerätään ja jaetaan julkisista lähteistä, kuten viranomaisten rekistereistä, tutkimuslaitosten mittauksista ja kansalaisjärjestöjen toimesta. Suomessa keskitetty palvelu avoimen datan löytämiseksi ja hyödyntämiseksi on avoindata.fi-portaali.

Saatavilla olevat aineistot eivät aina ole kaksiulotteista paikkatietodataa, vaan myös rakennusten ulkomuotoja voi löytyä joskus yllättävänkin tarkasti mallinnettuna. Kuvassa 2 nähdään Helsingin kaupungin rakennuksien 3D-malli, joka on saatavilla avoimena datana.

Kuva 2. Helsingin 3D-kaupunkimalli, rautatieasema. (Helsingin karttapalvelu, 2024).

2.4 Tekoälypohjaiset generatiiviset suunnittelutyökalut

Tekoälyn, erityisesti generatiivisten mallien, kehitys on avannut uusia mahdollisuuksia suunnittelualalla. Kaupalliset ja tutkimukselliset työkalut pyrkivät hyödyntämään tekoälyä suunnitteluvaihtoehtojen nopeaan generointiin, analysointiin ja optimointiin. Yksi esimerkki tällaisesta kaupallisesta sovelluksesta on Autodesk Forma (aiemmin Spacemaker). Forma hyödyntää tekoälyä luodakseen erilaisia kolmiulotteisia rakennusmassavaihtoehtoja valitulle tontille (kuvat 3 ja 4), perustuen erilaisiin ympäristötiedon analyyseihin kuten päivänvalo, melu ja tuuliolosuhteet. Sovellus siirtää laskentakuorman pilveen, mikä mahdollistaa nopeat tulokset ja tehokkaan visualisoinnin. (Morán, 2023).



Kuva 3. Forman tarjoama esittely demo. (Autodesk Forma, 2024).

Kuva 4. Generatiivisesti tuotettuja tonttivaihtoehtoja. (Autodesk Forma 2024).

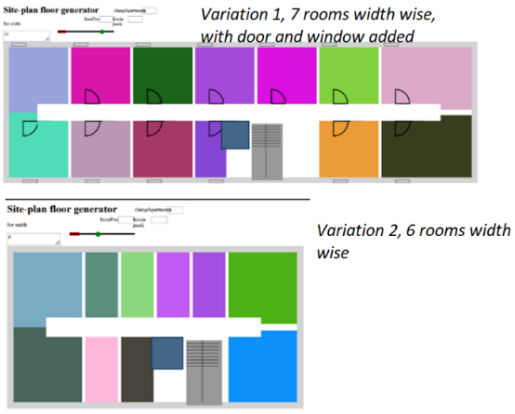
3 Sovelluksen suunnittelu ja rakenne

Työssä toteutetaan arkkitehti- ja kaupunkisuunnittelun tueksi tarkoitettu tekoälypohjainen työkalu. Tässä luvussa käydään läpi suunnittelun lähtökohdat ja muutokset työn edetessä, sekä sovelluksen käyttötarkoitus ja toimintaperiaate. Lisäksi esitellään käyttöliittymän rakenne sekä sovelluksen tekninen arkkitehtuuri ja osien välinen tietoliikenne.

Sovelluksen suunnittelu aloitettiin sen käyttöliittymän hahmottelemisesta.

3.1 Alkuperäinen ajatus

Työn alussa, toimeksiantoon liittyvässä keskustelussa ei vielä ollut varmuutta lähdettäisiinkö koittamaan tekoälyn tuomista sovellukseen. Alkuperäinen ajatus oli rakentaa sovellus, luonnos kuvassa 5, joka pystyisi näyttämään säännönmukaisesti luotuja huonepohjia annettujen mittaiseen suorakaiteen muotoiseen kerrostaloon. Puhuttiin huonejaon, sekä muiden kerrostaloelementtien, kuten kerros- ja rappukäytävän, sekä hissin sijainnin vaihtelua syötettyjen parametrien mukaan. Annettujen parametrien jälkeen sovellus olisi laskenut mahdollisia vaihtoehtoja asuntojen muodoille ja esittäisi nämä visuaalisesti käyttäjälle. Muita ennalta tiedettyjä parametreja olisi sisä- ja ulkoseinien paksuudet. Sovelluksesta olisi saatu kerroksen ja huoneiden tarkat alalaskelmat kullekin parametrien mukaan piirretylle vaihtoehdolle.



Kuva 5. Sovelluksen aikaisen vaiheen käyttöliittymä ”mockup”.

3.2 Siirtyminen tekoälypohjaiseen ratkaisuun

Suunnitteluprosessin edetessä heräsi ajatus tekoälyn käytöstä, mikä vastasi työn tekijänä ja kehittäjänä omaa kiinnostusta. Pohdittiin mahdollisia lähtökohtia. Millaisen tekoälyn tuominen olisi realistista opinnäytetyön laajuuden puitteissa? Keskustelu kääntyi näin GAN-mallin käyttöön, jolla voitaisiin suorittaa esimerkiksi kuvasta-kuvaan muunnos.

Tekoälymallin tiedettiin vaativan dataa, ja GANin käyttö rakennuspohjiin ja huonesijoitteluun edellyttäisi kuitenkin jonkinlaista olemassa olevaa tietojoukkoa suorakulmaisista rakennuksista, ja niiden pohjapiirroksista. Ensihaulla tällaista joukkoa ei löytynyt, ja näin ollen täytyi miettiä muu ratkaisu, jos tekoälyä haluttaisiin käyttää.

Kaupunkisuunnitteluun relevantin avoimen datan tutkimisesta selvisi paikkatietojen hyvä saatavuus Suomen alueella, ja päätettiin, että siihen perustuvan ratkaisun kehitys olisi hyvä tavoite. Myös GAN-mallin esimerkkitoteutukset satelliittikuvien muunnoksesta kartoiksi vaikuttivat lupaavilta. Täten työn tutkimuskohteeksi muodostui siis GAN-mallin hyödyntäminen kuvasta-kuvaan muunnoksessa, jonka pohjana käytettäisiin Maanmittauslaitoksen avointa paikkatietodataa.

Sovelluksen vaatimukset hahmottuivat kuvaamaan järjestelmää, joka pystyisi vastaanottamaan käyttäjän määrittelemät tontin rajaviivat kuvamuodossa ja tuottamaan näiden pohjalta ehdotelman rakennusten sijoittelusta kyseisten rajojen sisällä. Alkuperäisen suunnitelman mukainen rakennusten pohjapiirrosten luonnostelu algoritmillisen piirtotekniikan mukaan jäi syrjään.

Tämän kautta syntyi keskustelua muista arkkitehtien kokeilemista uusista tekoälyratkaisuista, ja esiin nousi Spacemaker-niminen sovellus, jolla oli mahdollista generoida rajatulle alueelle vaihtoehtoja rakennuksista ikään kuin tyhjästä. Millaista tekoälyä Spacemakerin (nyk. Autodesk Forma) kaltainen sovellus mahdollisesti käytti? Paikkatietodataa ainakin.

3.3 Työkalun toimintaperiaate

Todellisuudessa tekoälymallit eivät keksi mitään tyhjästä, kuten Spacemakerin tapauksessa ensisilmäykseltä saattoi vaikuttaa. Mallit ovat koulutettuja, mikä tarkoittaa sitä, että ne käytetään koulutusvaiheen läpi, jonka aikana malli oppii tietojoukoista minkälaisia tuloksia millekin käyttäjän syötteelle halutaan.

Tämän työn malli koulutetaan tunnistamaan syötekuvasta sille opetettuja piirteitä, tässä tapauksessa rajaviivoja, ja generoimaan niiden perusteella uuden kuvan, joka esittää ehdotelman rakennusten sijoittelusta näiden rajojen sisälle. Koulutusvaiheessa malli oppii vastaavuuksia syötekuvien (pelkät rajat) ja kohdekuvien (rajat ja niihin sijoitetut rakennukset) välillä, käyttäen laajaa Maanmittauslaitoksen paikkatietoaineistosta koostettua kuvaparijoukkoa. Generoitu tuloskuva palautetaan käyttäjälle, joka näkee sen sovelluksen käyttöliittymässä.

Vaikka lopputulos on visuaalinen ehdotelma, se perustuu mallin oppimiin säännönmukaisuuksiin todellisista rakennussijoitteluista. Prosessi voi onnistuessaan mahdollistaa nopeaa, interaktiivista, alustavan tason visualisointia ilman manuaalista piirtämistä tai monimutkaisten parametrien syöttämistä.

3.4 Tekoälymallin integroiminen käyttöliittymään

GAN-mallin integroiminen visuaaliseen interaktiiviseen käyttöliittymään edellyttää ainakin seuraavia osia:

- Käyttäjän syötteen vastaanottava komponentti (esim. kuvan lataus tai piirtoalue).
- Tekoälymallin tuotoksen visualisointi, generoitu kuva.

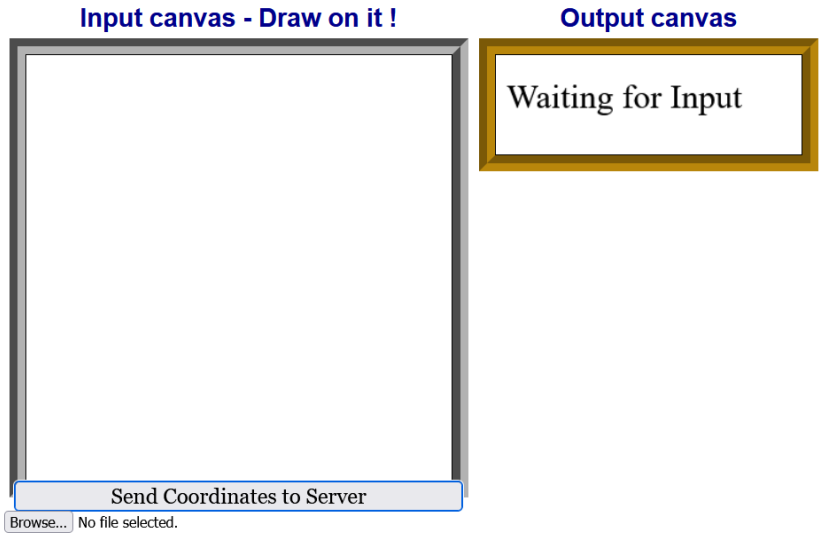
Käytettävyyden kannalta edulliset osat:

- Indikaattori mallin prosessoinnin käynnistymisestä, ja käynnissä olosta
- Virheilmoitus virhetapauksessa tai työn odottamattomassa keskeytyessä

- Välivaiheet, esim. latauspalkki koulutuksen etenemisen havainnollistamiseksi, mikä sopii luonnostason sovelluksiin.

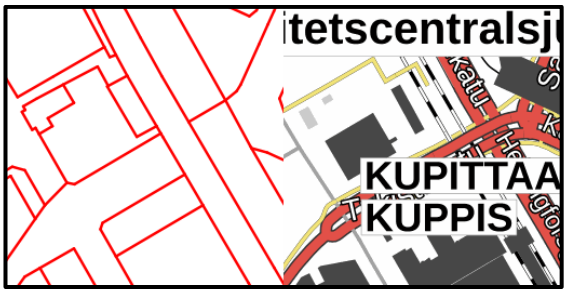
Sovelluksen käyttö onnistuu tavanomaisen verkkosovelluksen tapaan hiirellä ja näppäimistöllä. Kuvatiedosto voidaan tuoda tiedostovalikon kautta syötteeksi. Ladattu kuvatiedosto piirtyy näkyviin nelikulmioiselle canvas-elementille. Syötteeseen voi kokeilla myös piirtää viivoja hiirellä.

Suunniteltu syötealue antaa käyttäjälle kaksi vaihtoehtoa; joko ladata kuva suoraan tiedostosta, tai piirtää rajoja hiirellä. “Send to server” -painikkeesta voi sen lähettää syötteeksi tekoälymallille. Tekoälymalli generoi syötteen mukaan vastauksen, joka siirtyy kolmannelle Ulostulo -canvasille, ja päästään kolmanteen ja viimeiseen sovelluksen osaan. Canvasien luonnos on kuvassa 6.



Kuva 6. Syöte- ja tulosalue käyttöliittymässä.

Sopivanlainen ja mittakaavaltaan oikeanlainen syöte näyttää kuvan 7 kuvaparin vasemman osan kaltaiselta. Oikeanlainen syöte riippuu tekoälymallin tietojoukosta. Työn tietojoukkoa on esitelty tarkemmin luvussa 4.3.

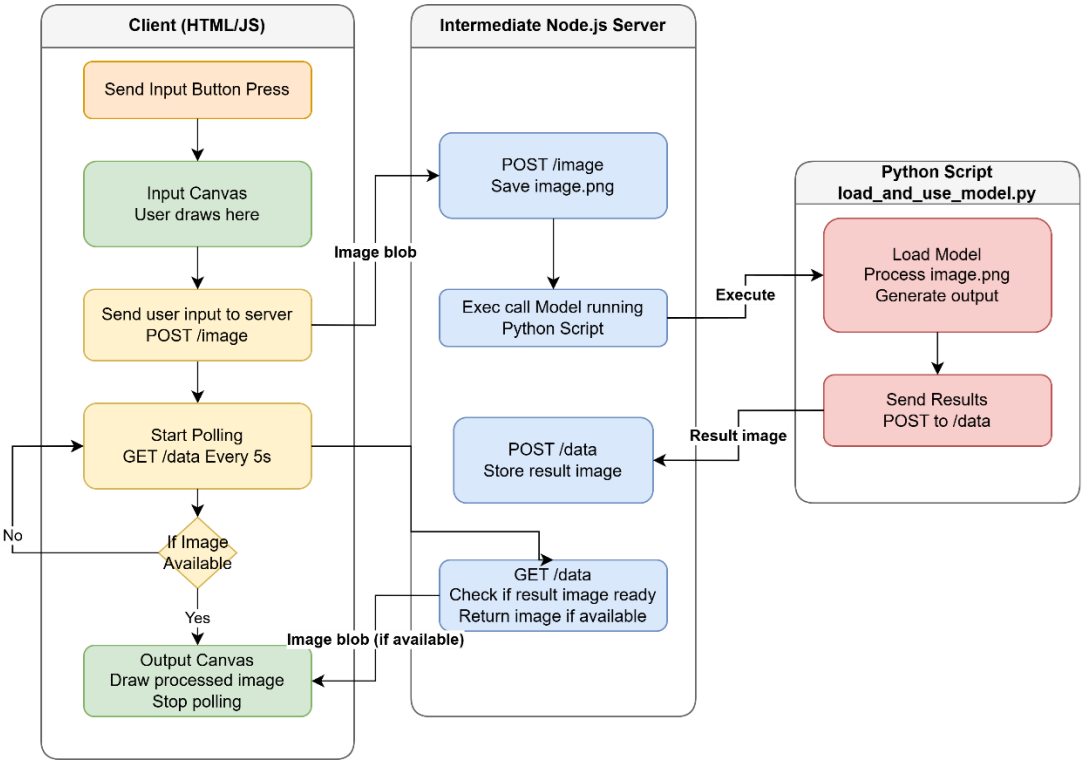


Kuva 7. Esimerkki kuvapari, jossa kiinteistöraja ja selkokartta. (MML, 2024).

3.4 Rakenne ja tietoliikenne

Verkkosovelluksen käyttöliittymän koostuu HTML-elementeistä. Käytettyjä elementtejä ovat otsikot, teksti kappaleet, painikkeet ja canvas-elementit, jotka toimivat graafisen sisällön piirtokanvaaseina.

Kuvassa 8 kuvautuu työkalun rakenne ja sen tiedonkulku. Sovellus on kolmiosainen: Verkkosivu käyttöliittymä, NodeJS palvelin, ja tekoälymallin käytön suorittava python aliohjelma. Tieto liikkuu sovelluksen osien välillä minimaalisen REST API:n avulla http-kutsuin. Python aliohjelmalla käytetään käyttäjän lähettämä syöte tekoälyn lävitse, jonka prosessoinnin jälkeen lähetetään vastauksena generoitu kuva.



Kuva 8. Sovelluksen tietoliikennekaavio.

Tässä vaiheessa käyttäjä näkee generoidun lopputuloksen, ja sovellus jää odottamaan seuraavaa syötettä, jolloin koko prosessi alkaa alusta.

Syötteen käsittelyn aikana käyttöliittymään lisätään käsittelyä indikoiva animoitu gif-kuva, josta käyttäjälle välittyy tieto, että sovellukselta voidaan odottaa vastausta. Jos tapahtuu virhe, näytetään virheilmoitus ja yksinkertainen diagnoosi virheestä käyttäjälle. Virhe voi olla esimerkiksi, jos serveri ei löydä python skriptiä määrätystä koodissa määrätystä sijainnista, tai jos verkkoliikenteen portit ovat jo toisen prosessin käytössä.

4 Käytetyt menetelmät ja teknologiat

Sovelluksen kyky generoida rakennusten pohjapiirroksia annettujen tonttirajojen perusteella perustuu neuroverkkoihin ja syväoppimisen menetelmiin. Tämä luku esittelee työssä hyödynnetyt keskeiset tekoälykonseptit ja -työkalut, alkaen neuroverkkojen perusteista ja edeten generatiivisiin kilpaileviin verkkoihin (GAN) sekä erityisesti työssä käytettyyn Pix2Pix-arkkitehtuuriin. Lisäksi käsitellään mallin koulutusvaiheen sekä käyttöönoton käytännön eri kohtia.

Luvun teoria etenee käsitteiden kautta GAN-tyyppisen mallin rakenteeseen, ja päättyy teknologian käyttöönoton edellytyksiin. Teorian pyrkii esittelemään GAN-mallin teknistä toteutusta, sekä avaamaan neuroverkkoarkkitehtuurin konseptin lukijalle.

4.1 Tekoälymalli

Tekoälyratkaisun sisällyttämiseen pyrkivä sovellus ei tietenkään toimi ilman varsinaista tekoälymallia, joten sellainen pitää tuoda sovellukseen. Käytännössä tekoälymalli[1] koostuu tiedostoon tallennetuista optimoiduista painoarvoista, jotka voidaan lukea ja hyödyntää koneoppimiskirjastojen, kuten TensorFlow’n tai PyTorchin avulla.

4.1.1 Tekoälymallin rakenne

Painoarvot ovat yksinkertaisesti kokoelma desimaalilukuja, jotka toimivat kertoimina neuroverkon matemaattisissa laskutoimituksissa. Lisäksi jokaisessa neuronissa on yleensä bias-arvo, joka lisätään laskutoimituksiin säätämään neuronin aktivaatiota, mikä voi parantaa mallin tarkkuutta.

[1] Tekoäly malli on laaja käsite ja kattaa erilaisia laskennallisia malleja, jotka jäljittelevät ihmisälyä. Työssä käsiteellään pääasiassa vain neuroverkkopohjaista tekoälyä, ja neuroverkkojen suosion vuoksi työn yhteydessä tekoälymallilla viitataan neuroverkkomalliin.

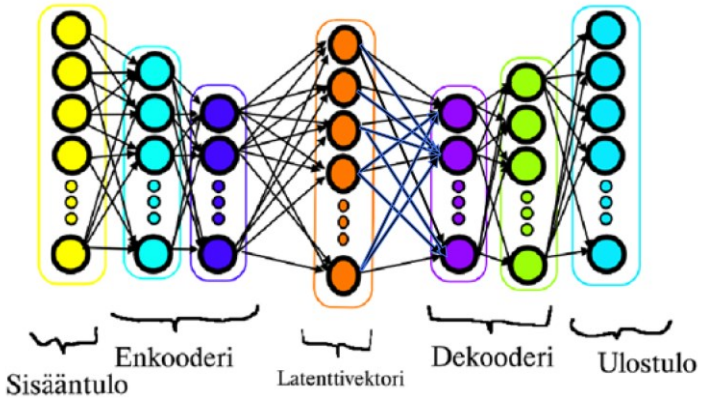
Tekoälymallin tuominen sovellukseen edellyttää valmiin tekoälymallin olemassaoloa. Tekoälymallitiedosto voidaan tässä tapauksessa käsitellä ohjelmalla, joka lataa ja asettaa painoarvot neuroverkkoon. Huomataan kuitenkin, että vaikka painoarvot voidaan käsitellä, on ennalta tiedettävä minkälaiseen neuroverkkoon painoarvot voidaan sovittaa. Tämä ongelma voidaan ratkaista tuntemalla mallin rakenne, eli sen neuroverkkoarkkitehtuuri. Jotta painoarvoja voidaan käyttää, arkkitehtuurin on oltava täsmälleen sama, kuin mallin koulutusvaiheessa käytetty arkkitehtuuri. Neuroverkon kerrosten koot ja niiden väliset yhteydet täytyy siis olla tarkasti määriteltynä ennen mallin käyttöä.

4.1.2 Neuroverkot

Neuroverkko on matemaattinen malli, joka koostuu toisiinsa yhteydessä olevista laskentayksiköistä, eli neuroneista, jotka on järjestetty kerroksiin (Tuominen, 2019).

Neuroverkossa on tyypillisesti kolme pääkerrosta: sisääntulokerros, joka vastaanottaa syötteen (esimerkiksi kuvan pikseliarvot); piilokerrokset, jotka käsittelevät syötettä matemaattisten operaatioiden kautta; ja ulostulokerros, jolle lopputulos, kuten generoidun kuvan pikselit, saadaan tuotettua. Neuronit ovat yhteydessä toisiinsa painoarvojen kautta, jotka määrittävät, miten vahvasti kunkin neuronin arvo vaikuttaa seuraavaan, eli kuinka vahvasti neuronit aktivoivat toisiaan.

Kerrokset on kuvattu moniulotteisina matriiseina, jotka määrittävät niiden ulottuvuudet ja yhteydet. Kerroksellisuutta on kuvattu kuvassa 9.



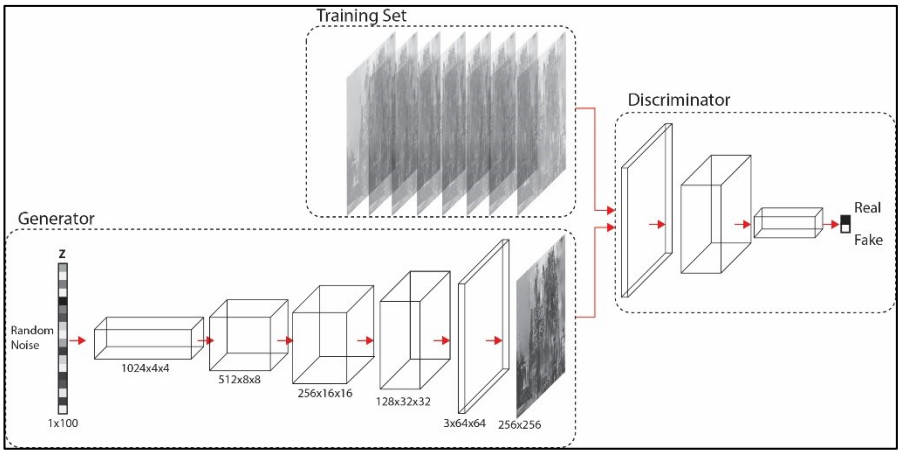
Kuva 9. Erään neuroverkkotyypin kerroksia ja rakenne.

Neuroverkon kyky oppia, esimerkiksi tunnistamaan kuvioita tai generoimaan uutta dataa, ei ole yksittäisen neuronin tai edes kerroksen ominaisuus. Sen sijaan nämä kyvyt ovat mallin emergenttiä käyttäytymistä, joka syntyy kerrosten hierarkkisen ja tarkoin suunnitellun yhteistoiminnan tuloksena.

Neuroverkon oppimisella tarkoitetaan sen koulutusvaiheen prosessia, jonka aikana optimisaatio-algoritmi iteratiivisesti laskee minkä kaltaisilla ns. sysäisyillä painokertoimiin saatiin tuloksia joko parantumaan tai huonontumaan. Negatiivinen tulos johtaa painoarvojen säätämiseen päinvastaiseen suuntaan. Kun malli taas toimii halutunlaisesti, jatketaan painoarvojen säätämistä siihen niihin suuntiin (jokaisella neuronilla on omat painoarvonsa, ja niiden, millä parempaa tulosta saatiin syntymään. Samaa koulutusaskeleiden ketjua jatketaan koko koulutusprosessin ajan.

4.2 Työn tekoälymalli

Tässä työssä sovelluksen käyttämä tekoälymalli on pix2pix-arkkitehtuurin GAN-verkon generaattori. Jotta ymmärretään, mitä tämä tarkoittaa, on syytä ensin avata pix2pix:n tarkoitus ja GAN-verkon toimintaperiaate.



Kuva 10. GAN verkon arkkitehtuuri. (AI+EDL Blog, UNL College of Architecture, 2019).

4.2.1 Pix2pix ja GAN

GAN-mallit, kuten pix2pix, eroavat muista malleista siten, että arkkitehtuuriin kuuluu kaksi neuroverkkoa: generaattori ja diskriminaattori. Tyypillinen GAN-verkon arkkitehtyyri näkyy kuvassa 10. Pix2pix arkkitehtuuri on suunniteltu kuvasta-kuvaan muunnoksen suorittamiseen. (Isola et al, 2017).

Generaattorin tehtävänä on tuottaa tulos syötteen perusteella, kun taas diskriminaattorin on tarkoitus arvioida ja erottaa, onko generaattorin tuottamat kuva aitoja, vaiko synteettisiä, eli generoituja tuloksia. Tämän vastakkaisasettelun periaatteen kautta generaattoria kannustetaan huijaamaan diskriminaattoria, ja tuottamaan mahdollisimman hyvin todellisten mallikuvien kaltaisia kuvia. Kun diskriminaattori oppii samanaikaisesti erottamaan aidot kuvat generoiduista, syntyy verkkojen välille kilpaileva vuorovaikutus. Generaattorin lopputavoite eli kyky tuottaa aidon kuvan näköisiä tuloksia, on siis epäsuora, ja päärään päädytään diskriminaattorin tehokkaan hämäyksen kautta tuottamalla dataa, joka näyttää mahdollisimman aidolta.

Kiinnostava piirre GAN-mallien toiminnassa on tilanne, jossa diskriminaattori on liian tehokas. Tällöin generaattori ei saa riittävää kehittävää palautetta, koska sen hävioarvo pysyy jatkuvasti suurena. Toistuvien epäonnistumisten vuoksi

generaattorin tuottamat tulokset voivat jäädä heikoiksi. Toinen GAN-verkoille ominainen ilmiö, joka poikkeaa monista muista neuroverkkoarkkitehtuureista, on ylikoulutetun mallin suorituskyvyn heikkeneminen. Generaattori saattaa saavuttaa parhaan tasonsa jossain koulutuksen vaiheessa, mutta liiallinen koulutus voi heikentää sen laatua diskriminaattorin kehittyessä liikaa. Tällaisia mallien epätasapainotiloja kutsutaan englanninkielisellä termillä mode collapse. Tämän vuoksi GAN-verkkoa koulutettaessa on tärkeää tallentaa tarkistuspisteitä (checkpoints) eri koulutusvaiheista, jotta voidaan varmistaa parhaan tekoälymallin säilyminen.

4.2.2 Tietojoukon rooli

GAN-verkon periaatteesta nähdään mihin kerättyä tietojoukkoa tarvitaan, ja käytännössä käytetään.

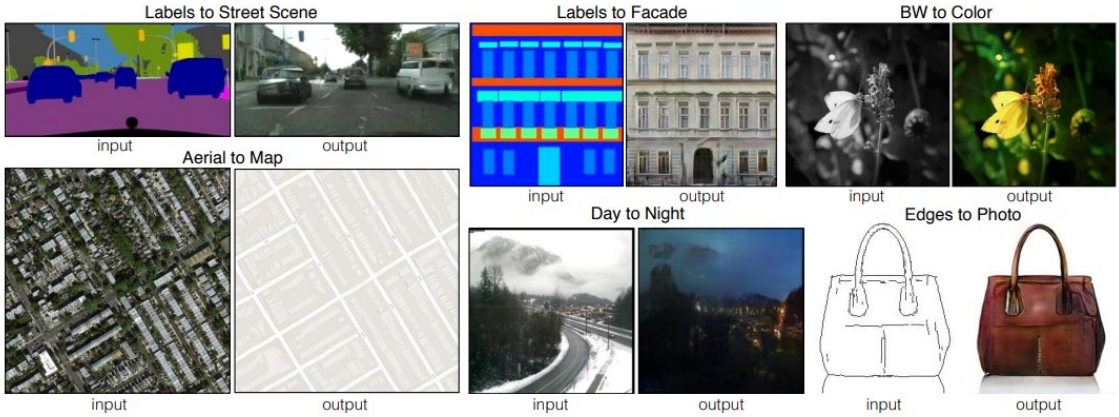
Kuva 11. Edges2shoes tietojoukon kuvapareja. (Yu & Grauman, 2014).

Kuvapari -tietojoukon tapauksessa toinen kuvista edustaa aitoa kuvaa, kuten kuvan 11 oikeanpuoleiset kuvat. Diskriminaattorin täytyy tehdä valinta aidon, ja generaattorin epäaidon tuotoksen välillä. Kuvaparin toinen osa, syötekuva, vasemmalla kuvassa 11, annetaan generaattorille. Se vastaa myös minkälaista syötettä tekoälymalli odottaa käyttäjältä. Tavoitekuva (aito kuva) toimii kannustimena: mitä samankaltaisemman kuvan generaattori onnistuu tuottamaan, sitä suuremmalla todennäköisyydellä diskriminaattori arvaa väärin. Tämä muodostaa GAN-verkon oppimissilmukan, jossa generaattori ja diskriminaattori kehittyvät kilpailevasti. Kun generaattori tuottaa kuvan, jonka

diskriminaattori luokittelee virheellisesti aidoksi, optimointialgoritmi (esim. gradienttimenetelmä, kuten Adam) päivittää generaattorin painojakertoimia vahvistaen niitä piirteitä, jotka johtivat onnistuneeseen harhautukseen.

4.2.3 Kuvasta kuvaan muunnos

Tässä työssä vastaava periaate soveltuu karttakuvien muuntamiseen: syötteenä toimii kuva tonttien ja kiinteistöjen rajoista, ja tulosteena saadaan kuva, johon on generoitu rakennusten pohjia näiden rajojen sisälle oikeanmaailman tapauksia mukaillen. Kuvassa 12 nähdään muita esiteltyjä kuvasta-kuvaan muunnoksen käyttötapauksia.



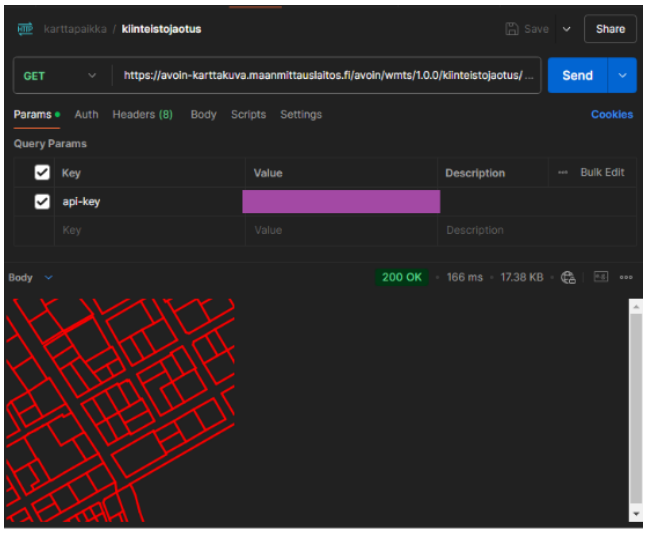
Kuva 12. Kuvasta-kuvaan muunnoksia Pix2Pix -paperissa. (Isola et al, 2017).

4.3 Tietojoukon kokoaminen ja esikäsittely

Tekoälymallin koulutuksen onnistuminen edellyttää laadukasta ja tarkoituksenmukaista tietojoukkoa. Tässä alaluvussa kuvataan prosessi, jolla tämän opinnäytetyön Pix2Pix -mallin koulutukseen tarvittava paikkatietopohjainen kuvapariaineisto koottiin ja esikäsiteltiin. Keskeisenä datalähteenä toimi Maanmittauslaitoksen (MML) avoin data, jonka hyödyntäminen rajapintojen kautta oli olennainen osa prosessia. Lisäksi käsitellään datan eri muotojen ja koordinaatistojärjestelmien merkitystä.

4.3.1 Datan hakeminen rajapinnoista

Maanmittauslaitoksella on useita rajapintoja (API, Application Programming Interface) avoimen paikkatietodatan jakeluun. Näitä rajapintoja käytetään tyypillisesti HTTP-protokollan mukaisilla REST-kutsuilla, joissa tiettyyn URL-osoitteeseen lähetettävät parametroidut kyselyt määrittävät haettavan data-alueen ja -tyypin. Onnistuneen kutsun tuloksena saadaan paikkatietoa pyydetyssä tiedostomuodossa. Kuvassa 13 Postman-sovelluksella suoritettu rajapintakutsu. Tässä työssä käytettyjen MML:n API-palveluiden käyttö edellytti API-avaimen, jonka rekisteröitynyt käyttäjä voi generoida Maanmittauslaitoksen palvelusta.



Kuva 13. Postman sovelluksella suoritettu GET-kutsu kiinteistöraja-rajapintaan.

Datan hakeminen API:n kautta edellyttää useiden parametrien määrittelyä. Näihin kuuluvat tyypillisesti haluttu karttatyyppi tai -taso, käytettävä projektio ja koordinaatistojärjestelmä, tarkkuusaste eli zoom-taso, haettavan karttatiilen X- ja Y-indeksit sekä tiedostomuodon pääte. Esimerkiksi URL-kutsu voi rakentua muotoon:

https://avoin-karttakuva.maanmittauslaitos.fi/vectortiles/taustakartta/wmts/1.0.0/taustakartta/default/v20/WGS84_Pseudo-Mercator/14/4740/9326.png?api-key={API-avain}

On huomioitava, että X- ja Y-parametrit viittaavat tässä karttatiilten ruudukon indekseihin, jotka vaihtelevat valitun tarkkuusasteen mukaa. Luvut noudattavat seuraavassa kappaleessa käsiteltyä logiikkaa JHS 180 -tiilimatriisi spesifikaation mukaisesti.

4.3.2 Koordinaatistot ja XML-spesifikaatio

Tämä osa on relevantti lähinnä sen takia, että tietojoukkoa kasaava henkilö voi haluta dataa joltain tietyltä alueelta, tietyssä mittakaavassa. Tiedon haku ohjelmallisesti API:n kautta vaatii indeksilukujen ja koordinaatiston yhteyden ymmärtämistä. Seuraavan menetelmän avulla on mahdollista hakea karttatiiliä mistä tahansa Suomen alueelta.

MML:n spesifikaatio rajapinnasta haettavan tiedon jaotuksesta löytyy (tarvitsee API avaimen): https://avoin-karttakuva.maanmittauslaitos.fi/avoin/wmts/1.0.0/WMTSCapabilities.xml

XML spesifikaatiosta löydetään logiikka millä indeksiluvut lasketaan, ja niiden yhteys koordinaattijärjestelmään. Kuvassa 14 näkyy koodi, mikä perustuu spesifikaatioon, ja jonka avulla voidaan laskea indeksiluvut annettujen ETRS-TM35FIN x- ja y-koordinaattien perusteella. Suuret luvut koodin laskukaavoissa vastaavat tietoalueen ääripäitä (Suomen alueen rajat) koordinaatein. Kun tiedetään kaupungin sijainti ja äärirajat koordinaattilukuina, voidaan apuohjelmalla hakea sen kaupungin alueelta dataa eri tarkkuustasoilla.

```
def get_tile_idx(x=None,y=None, zoom=14):
    if x and y:
        return int(((x+548576)*(2**zoom)//2097152), -1*((y-8388608)*(2**zoom)//2097152))
    if x:
        return int(((x+548576)*(2**zoom)//2097152))
    if y:
        return int(-1*((y-8388608)*(2**zoom)//2097152))
```

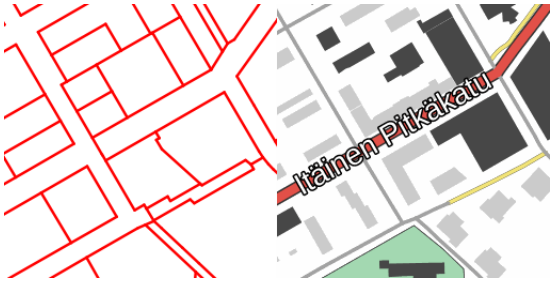
Kuva 14. Python -koodi indeksilukujen laskemiseen.

Hain tätä metodia käyttäen vektoritiiliä API:sta, valikoivasti eri kaupunkien alueilta, ja käsittelin ne Geopandas-kirjaston avulla kohdakkain rasteridatan kuvien kanssa.

4.3.3 Kuvapariaineisto

Tietojoukon kokoamisen tavoitteena oli kerätä kuvien 15 ja 16 näköisiä kuvapareja. Keskityttiin Suomen kaupunkien kerrostaloalueisiin.

Haasteena oli löytää sopiva mittakaava eli Zoom-taso, jotta rakennukset erottuisivat selvästi, mutta yksittäisillä tiilillä näkyi ainakin osittain kokonaisia rakennuksia. JHS 180 dokumentti selventää Zoom-tasojen mittakaavaa, ja tasolla 14 mittakaava on 0,5 m pikseliä kohden. (JUHTA, 2019). Yhden laatan sivut ovat leveydeltä ja korkeudeltaan 256 pikseliä. Kullekin laatalle mahtuu tällä arvolla kootulla datasetissä nyt 128 x 128 metrin kokoinen alue. Näin rakennusten pohjatkin alkavat mittakaavassa piirtymään tarkemmin.



Kuva 15. Vektori-rasteri tietojoukon kuvapari. (MML, 2024).

Kuva 16. Vektori-vektori tietojoukon kuvapari. (MML, 2024).

Prosessi alkoi rasteri- ja vektoridatan hakemisella valituilta alueilta, kuten Turun Kupittaalta. Tärkeä huomio ”vektori” termin käytöstä tässä yhteydessä on, että molemmissa tietojoukossa kuvaparit ovat rasterimuodossa, mutta vektorien tapauksessa toteutettiin rasterointi erikseen vektoritiedon pohjalta. Rasterikuva kuvassa 15 saatiin rajapinnasta sellaisenaan.

4.3.4 Suodatusmenetelmät

Vektoridatan avulla saatiin esitystapa selkeämmäksi poitamalla ylimääräiset tekstit ja merkit kartalta. Kohdekuvissa saatiin näkymään vain halutut rajat ja rakennukset, kuten kuvassa 16, tai pelkästään rakennukset. Lopulliset kuvaparit saatiin yhdistämällä ne ohjelmallisesti käyttäen Pillow-kuvankäsittelykirjastoa.

Tietojoukkoa saatiin suodatettua vektoritasoilta löytyneen rakennukset ja rakenteet -tason avulla. Kunkin tiilen “kerrostalojen kokoisten” multipolygonien määrää käyttäen tietojoukosta voitiin poistaa tiilet, joilla taloja ei ollut, ja näin tietojoukkoon saatiin mahdollisimman paljon materiaalia kerrostaloalueilta.

Python-ohjelmaan piti tehdä muutos tietojoukon kuvankäsittelyn vaiheessa, sillä muisti alkoi loppua kesken ajon aikana. Piilevänä vikana koodista löytyi matplot kirjaston kaavion luontimetodista puuttuvat parametrit “num” ja “clear”. Kullekin arvoilla 1 ja True päästiin eteenpäin. Arvojen asettaminen sai pythonin Garbage collectorin poistaamaan silmukoissa luodut tuhannet kaaviot, ja muisti vapautui ajon aikana oikein. Ratkaisu löytyi Matplotlibin muistinhallintaongelmaa käsittelevästä Stackoverflow-artikkelista (Huang, 2021).

4.3.5 Vektoridata

Rasteridatan vahvuutena oli, että se sopii pix2pix mallin tietojoukoksi käytännössä suoraan. Rastereita on saatavilla eri tasoina, kuten selkokartta, (yksinkertaistettu kartta) ja kiinteistöjaotus (kiinteistö ja tonttirajat).

Vektoritasojen vahvuutena oli kuitenkin ominaisuudet, joiden avulla pysytään valikoiden näyttämään vain halutut elementit, kuten rakennelmien jalanjäljet, ja säätämään niiden visuaalista tyyliä kuvaparin tavoitekuvaa varten.

Vektoridata voidaan jakaa kolmeen päätyyppiin:

- LINESTRING: Viivat, joilla on selkeä alku- ja loppupiste.
- POLYGON: Monikulmiot, joissa on määritelty sisä- ja ulkopuoli.
- MULTIPOLYGON: Useista erillisistä monikulmioista koostuva rakenne.

Kun tieto saadaan muodossa, joka on valmiiksi semanttisesti jaoteltu, on sen pohjalta helpompaa lähteä toteuttamaan analyysiä, tutkimusta, tai kuten tässä työssä, muuntaa se tietojoukoksi.

Vektorit vievät myös vähemmän tilaa, mutta niiden visualisointiin tarvitaan erillinen piirto-ohjelma.

Vektoridata saadaan Maanmittauslaitokselta eri muodoissa. Suosin (Protocolbuffer Binary Format), koska sen sai kätevästi haettua HTTP-pyynnöillä vektoritietolaattoja indeksilukuja käyttäen. Pbf on kehittämä OSM (OpenStreetMap) yhteisön kehittämä formaatti johon paikkatieto saadaan pakattua tiiviisti.

4.4 Mallin koulutus

Seuraava keskeinen vaihe oli itse Pix2Pix-mallin kouluttaminen räätälöidyllä paikkatietoaineistolla. Tässä luvussa kuvataan mallin koulutusympäristö, käytetyt parametrit, sekä iteratiivinen prosessi, jolla malli kouluttui. Luku on tekniseltä tiedoltaan työn raskain osuus, ja sisältää paljon syväoppimisen käsitteitä. Näiden avaaminen täysin olisi tässä työssä mahdotonta, mutta pix2pix arkkitehtuurista ja GANien koulutksesta löytyy runsaasti lisämateriaalia verkosta.

4.4.1 Koulutusympäristö ja parametrit

Tekoälymallin koulutus on laskennallisesti vaativa prosessi. Tässä työssä koulutus suoritettiin paikallisessa ympäristössä kannettavan tietokoneen NVIDIA RTX 3070 Mobile -näytönohjaimella, joka sisälsi 6 GB VRAM-muistia. Näyttöohjaimella voitiin käyttää CUDA-rinnakkaislaskentaa. Ohjelmistopuolella käytettiin Erik Lindernoren python-kielellä kirjoittamaa koodipohjaa PyTorch-GAN (2019), jossa käytetään PyTorch-syväoppimiskehykstä. Työssä käytetty Pix2Pix-mallin toteutus perustui kolmeen yhdessä toimivaan python-skriptiin. GAN-verkon generaattori- ja diskriminaattoriarkkitehtuurit on kuvattu models.py-tiedostossa. Koodista nähdään, että generaattori hyödyntää U-Net-arkkitehtuuria (GeneratorUNet), jossa on kahdeksan enkooderikerrosta ja seitsemän dekooderikerrosta skip-yhteyksillä. Tämä rakenne on tyypillinen kuvasta-kuvaan muunnoksissa, ja auttaa sekä matalan tason piirteiden että korkeamman tason kontekstin välittymisessä syötteestä tulosteeseen. Pix2Pix:n diskriminaattori on PatchGAN-arkkitehtuurin mukainen verkko, mikä arvioi kuvan aitouden paikallisesti pienempien kuva-alueiden perusteella. Toisin kuin perinteiset diskriminaattorit, jotka arvioivat koko kuvan aitoutta yhdellä kertaa, PatchGAN jakaa kuvan pienempiin, päällekkäisiin alueisiin ("patcheihin") ja tekee aitouspäätöksen kullekin alueelle erikseen. Lopputuloksena saadaan matriisi arvioita koko kuvan eri osista.

Molempien mallien painot alustetaan käyttäen weights_init_normal-funktiota, joka alustaa painot normaalijakaumaa käyttäen. Tärkeä yksityiskohta on, että bias-arvot asetetaan tässä toteutuksessa nollaan. Mallin suorituskyky perustuu siis ainoastaan koulutettuihin painoarvoihin.

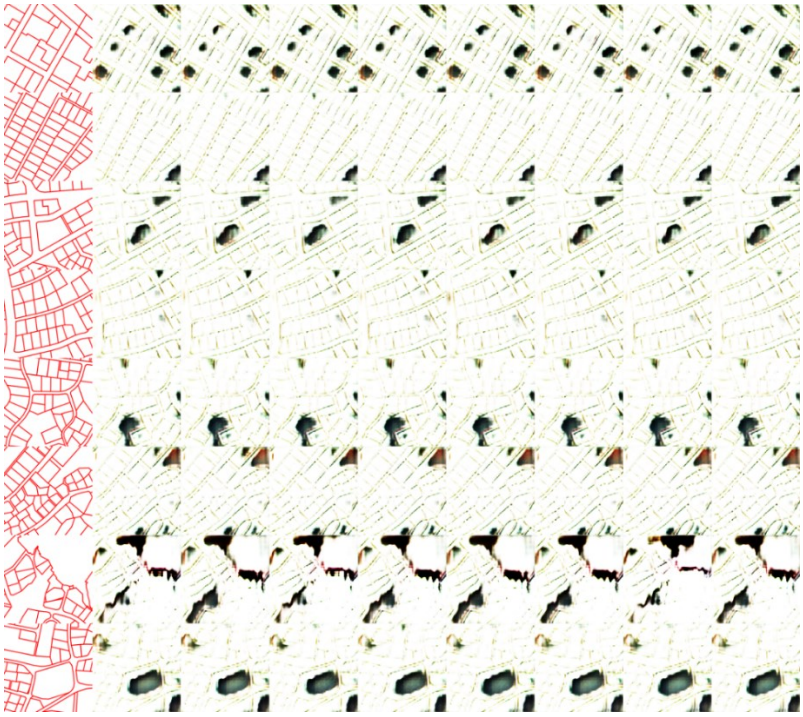
Koulutusprosessia ohjaavat useat parametrit. Epookkien määräksi asetettiin tyypillisesti 200. Eräkoko (batch size) Pix2Pix-koulutuksessa oli usein 1, mikä tarkoittaa, että mallin painoja päivitettiin jokaisen yksittäisen kuvaparin käsittelyn jälkeen. Oppimisnopeudeksi (lr) asetettiin 0.0002 ja optimoijana toimi Adam-optimoija (Kingma & Ba, 2015) parametreilla b1=0.5 ja b2=0.999. Oppimisnopeutta lasketaan (decay_epoch, esim. 100 epookin jälkeen) kohti

nollaa loppujen epookkien aikana. Oppimisnopeuden asteittaisen pienentämisen (learning rate decay) tarkoituksena on auttaa mallia konvergoitumaan optimiin, kun se on ensin oppinut datan pääpiirteet suuremmalla oppimisnopeudella.

Keskeiset häviöfunktiot Pix2Pix-mallissa olivat adversariaalinen häviö (criterion_GAN), joka laskettiin käyttäen Mean Squared Error-funktiota (MSE). Pikselitason L1-häviö (criterion_pixelwise) laskettiin generaattorin tuottaman kuvan (fake_B) ja todellisen kohdekuvan (real_B) välillä. L1-häviöllä on myös painokerroin (lambda_pixel), millä voidaan kontrolloida pikselien vertailun vaikutusta suorituskykyyn. Sen arvo on oletuksena 100. Suuri painoarvo kannustaa generaattoria tuottamaan kuvia, jotka ovat rakenteellisesti hyvin lähellä kohdekuvia. Lopuksi diskriminaattorin kokonaishäviötä (loss_D) skaalataan 0.5 kertoimella, millä voidaan koittaa tasapainottaa generaattorin ja diskriminaattorin oppimisnopeuksia.

4.4.2 Tietojoukon käyttö

Kerätty paikkatietoaineisto jaettiin koulutus- ja validointijoukkoihin. ImageDataset-luokkaa käytetään lataamaan kuvat train-kansiosta, ja erillinen val_dataloader käyttää kuvia val-kansiosta. Validointidataa käytetään koulutuksen aikana säännöllisin väliajoin (sample_interval), jonka mukaan epokkien välillä generoituu näytekuvia. Kuvan 17 kaltaisista näytekuvista voidaan tarkastella mallin generaattorin oppimista.



Kuva 17. Näyteotos koulutusvaiheesta paikkatietodatalla. 50. epookki.

Oletetaan, että saatavilla on sopiva tietojoukko. Kuvia voidaan käsitellä ennen syöttämistä neuroverkkoon, mutta tärkeintä on, että neuroverkko saa juuri sellaisen syötteen, jonka rakenne sallii. Yleisin koulutusmenetelmä on vastavirta-algoritmi (backpropagation), joka koostuu häviöfunktion laskemisesta, gradienttien määrityksestä ja mallin painokertoimien päivittämisestä gradienttipohjaisen optimoinnin avulla. (Kallio, 2021). Tämä iteratiivinen prosessi suoritetaan useiden epookkien ajan, jotta malli oppii tuottamaan tarkkoja ennusteita tai haluttuja tuloksia. Pix2Pix-mallin kaltaisissa GAN-arkkitehtuureissa koulutus sisältää generaattorin ja diskriminaattorin samanaikaisen optimoinnin, mikä edellyttää niiden oppimisnopeuksien ja häviöfunktioiden huolellista tasapainottamista korkealaatuisten tulosten saavuttamiseksi.

Tämä kilpaileva vuorovaikutus, jota ohjaa adversariaalinen häviö ja usein myös L1-häviö (pikselien lineaarinen ero), johtaa generaattorin kykyyn tuottaa laadukkaita kuvia.

4.4.3 Koulutusalgoritmin prosessi

Varsinainen koulutusprosessi aloitetaan ajamalla pix2pix.py-skripti määritellyillä parametreilla. Koulutus etenee epookki kerrallaan, ja jokaisessa epookissa koko koulutusaineisto käydään läpi erissä. Koulutusaineiston koolla on siis huomattava vaikutus koulutuksen kestoon. Jokaisessa erässä algoritmi käy lävitse seuraavat vaiheet:

1. Syötekuvan real_A syöttö generaattorille, joka generoi kuvan fake_B.
2. Generaattorin koulutus:
    - Diskriminaattori arvioi generoidun kuvan fake_B ja syötekuvan real_A parin (pred_fake).
    - Lasketaan adversariaalinen häviö (loss_GAN) pred_fake-arvon perusteella.
    - Lasketaan pikselitason L1-häviö (loss_pixel) generoidun kuvan fake_B ja todellisen kohdekuvan real_B välillä.
    - Generaattorin kokonaishäviö (loss_G) muodostetaan painotettuna summana loss_GAN:sta ja loss_pixel:stä.
    - Päivitetään generaattorin painot (optimizer_G.step()).
3. Diskriminaattorin koulutus:
    - Diskriminaattori arvioi todellisen kuvaparin (real_A, real_B) ja laskee häviön (loss_real).
    - Diskriminaattori arvioi generoidun kuvaparin (real_A, fake_B) ja laskee häviön (loss_fake).
    - Diskriminaattorin kokonaishäviö (loss_D) lasketaan loss_real- ja loss_fake-arvojen summana, skaalattuna kertoimella (esim. 0,5).
    - Päivitetään diskriminaattorin painot (optimizer_D.step()).

Koulutuksen etenemistä seurataan konsolitulosteista, jotka raportoivat eri häviöarvoja (D loss, G loss, pixel loss, adversarial loss) kunkin erän jälkeen. Kuvassa 18 näkyy aiemmin mainitun sample_interval-välein tallennettuja validointidatalla generoituja näytekuvia. Visuaalisista näytekusvista nähdään generaattorin tuotoksia konkreettisesti.

Lisäksi koulutuksen aikana mallin painot (checkpointit) voidaan määrätä tallentumaan säännöllisesti checkpoint_interval-parametrin mukaan epookkien välissä. Voidaan palata tekoälymallin määritelmään, ja tässä nähdään miten tekoälymalli on tiedosto, joka koostuu painoarvoista. Diskriminaattorin sekä generaattorin sen koulutushetken painoarvot (ja vastaavat bias-arvot), tallentuvat säännöllisin eepokki-väliajoin (checkpoint_interval) serialisoituun PyTorch pth-tiedostoon, jonka avulla koulutusta voidaan jatkaa tallennuksen vaiheesta. Tällä tavoin tallennettu generaattorin checkpoint on myös käyttökelpoinen tekoälymalli (luku 4.1.), millä kuvasta-kuvaan muunnos voidaan suorittaa.

Tässä kohtaa työn tekoälyosuus on valmis. Käytännössä malli voidaan nyt ottaa käyttöön.

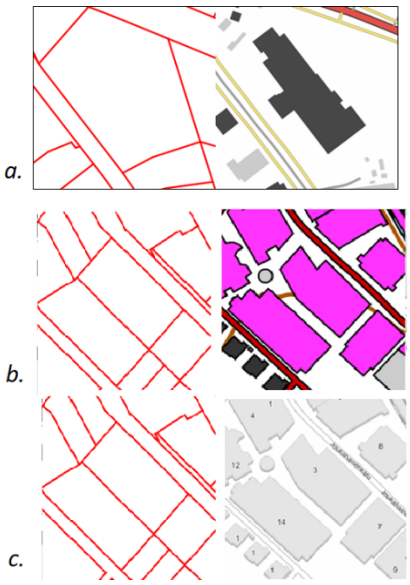
5 Tulokset ja analyysi

Tulokset ja analyysi -luku jakautuu kahteen osaan. Ensimmäisessä käsitellään valintoja tietojoukon suhteen tehtiin, ja millä perustein. Jälkimmäisessä osuudessa katsotaan miten kokeilut etenivät tekoälymallin koulutuksen aikana, ja arvioidaan koulutuksen onnistumista ja generoinnin tuloksia.

Arviointi keskittyy pääasiassa visuaaliseen analyysiin generoiduista kuvista ja niistä tehtyihin havaintoihin mallin oppimisesta ja suorituskyvystä. Lisäksi tarkastellaan kuvien mahdollista jälkikäsittelyä. Jälkikäsittely huomataan auttavan tulosten laatua niiden käyttöä luonnostyökalu sovelluksessa ajatellen.

5.1 Tietojoukkoon liittyvät valinnat

Työn kaltaisessa on paljon valintoja sen suhteen millaisella datalla mallia lähdetään kouluttamaan. Kuvaparin kohdekuvaksi voisi sopia kuvan 18 erilaiset karttatasot.



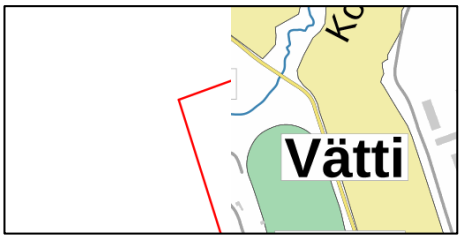
Kuva 18. Eri esitysmuotojen kandidaatteja kuvapareiksi.

Ennen laajemman datankeruun aloitusta oli valittava millaista dataa lähdettäisiin keräämään. Olennaisiksi tekijöiksi ilmenivät jo työn menetelmissä esille nousseet mittakaavan määräävä zoom-taso, sekä oikeanlainen esitysmuoto.

Yksittäisiä tiiliä tarkastellessa löytyi alueesta riippuen epäkohtia. Kuvassa 19 nähdään dataa, jossa rakennukset piirtyvät häilyvän harmaalla. Kyseessä ei myöskään ole kerrostaloalue, ja vaikka tämänlaisen alueen rakennelmia haluttaisiin generoida, on mittakaava liian suuri. Mallin olisi vaikea tuottaa yksityiskohtaista geometriaa tältä ”korkeudelta”.

Kuva 19. Kuvaparissa liian suuri mittakaava. Ei kerrostaloja.

Kuvassa 20 on vaikea tilanne ympäristön suhteen. Rajaviiva leikkautuu pois eikä näytä kokonaista tontin aluetta rajatussa alueessa. Kohdekuvan esitystavassa nähdään että rakennuksia on hyvin vähän kuvan vasemmassa reunassa. Kuvassa on myös värialueita ja tekstiä, jotka voidaan lukea epätoivotuiksi elementeiksi eli ns. artifakteiksi koulutusdatassa.



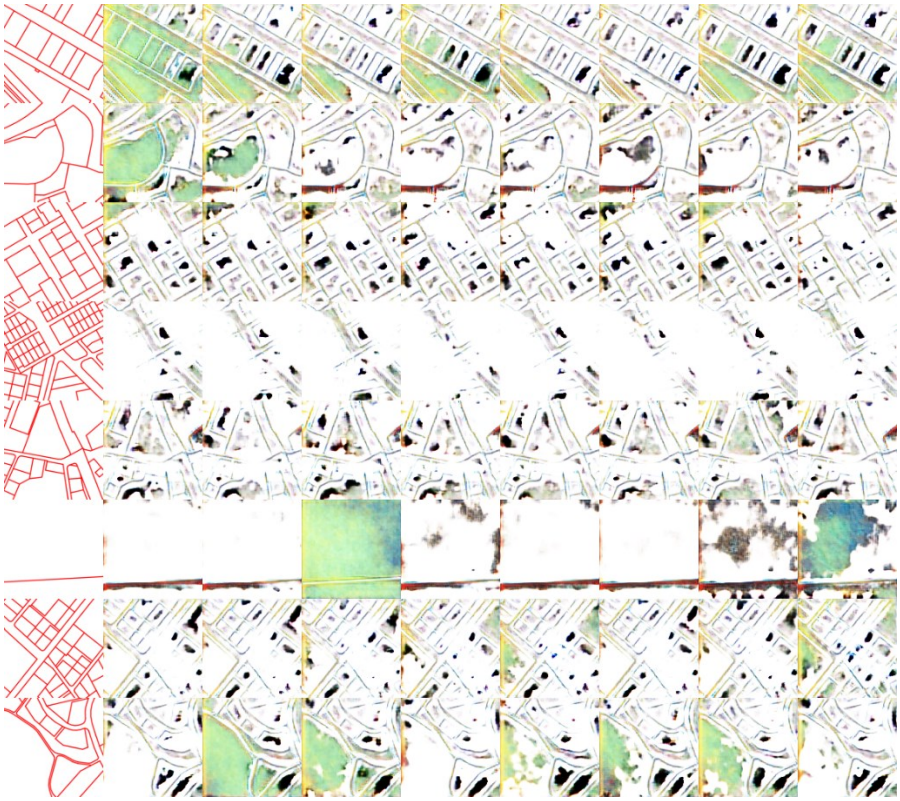
Kuva 20. Osittainen rajaviiva. Kohdekuvassa tekstiä ja vähän rakennuksia.

Päädyttiin joka tapauksessa kokeilemaan koulutusta kuvan 18. a. vaihtoehdon kaltaisella tietojoukolla. Vaihtoehdon kohdetukan esitysasu on sama kuin kuvissa 19 ja 20. Kun esitysmuoto on valittu ja tietojoukko koottuna luvun 4.3.

menetelmien mukaisesti, voidaan mallin koulutusta lähteä suorittamaa luvussa 4.4. kuvatulla menetelmällä.

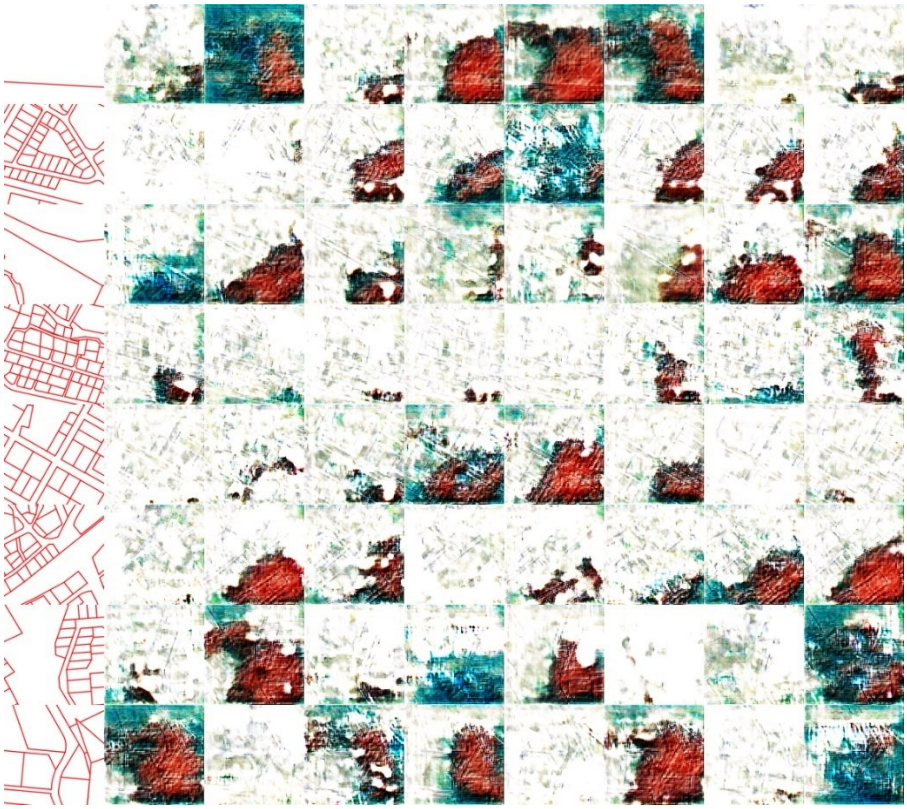
5.2 Koulutuksen tuloksia selkokartta tietojoukolla

Huomataan että selkokarttakuvan käyttö tietojoukkona saa mallin hypoteesin mukaan generoimaan mainittuja artifakteja, ja moni tuloksista sisältää värisekamelskaa. Joissain tapauksissa voidaan kuitenkin n. 30. epookin jälkeen nähdän mustia täpliä piirtyvän rajaviivojen sisään, kuten kuvan 21 generoinneissa. Näytteiden selitteissä käytetään epookkien sijaan koulutusaskeleen määrää, sillä tallentuneissa kuvissa tämä luku oli kätevästi tiedostonimessä.



Kuva 21. Rajaviivat-Selkokartta tietojoukon näyte, 7 900. koulutusaskel.

Kouluttamisen pidemmälle vienti kyseisellä tietojoukolla ja parametreilla johtaa kuitenkin vain kuvan 22 näytteen kaltaisesti lisääntyvään värisotkuun.

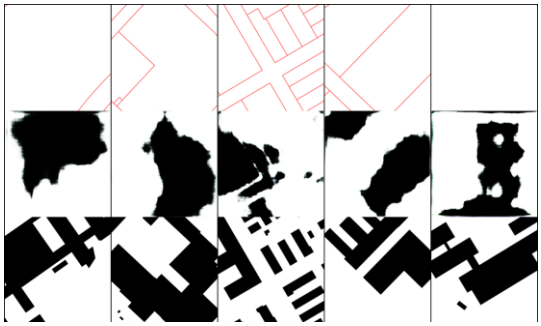


Kuva 22. Generoituja artifaktinäytteitä, 10 300. koulutusaskel.

Huomioitavaa on, että kokeilu selkokarttajoukolla tehtiin ennen oikean mittakaavan, zoom-tason, löytämistä, joten huomataan, että käytetty tietojoukko on liian suuressa mittakaavassa ja rajaviivat liian tiheässä. Tietojoukossa suuri osa tonttirajojen sisällä olevista rakennuksista piirtyy hyvin pieninä tiileen nähden. Tulokset voisivat näyttää erilaisilta säätämällä mittakaava pienemmäksi, mutta artifaktien esiintyminen on tietojoukon karttalehti luonteesta väistämätöntä. Artifaktit hankaloittavat mahdollista jälkikäsittelyä.

5.3 Koulutuksen tulokset vektoripiirretyllä tietojoukolla

Selkokartta tietojoukon jälkeen käytettiin siistitympää vektoreista piirrettyä kuvaparijoukkoa, jossa kuvissa näkyi pelkästään rajaviivat, sekä rakennukset ilman muita merkintöjä. Teiden ja alueiden nimet, numerot ja ympäristön ja aluejaon värit jäivät uudesta joukosta pois.

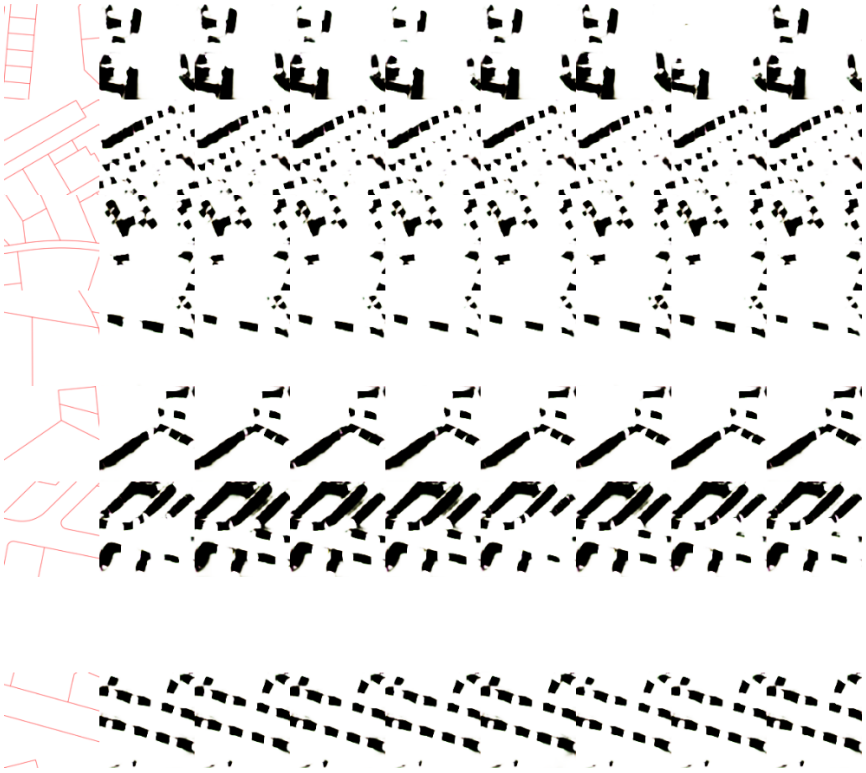


Kuva 23. Mallin generoimia "läiskiä".

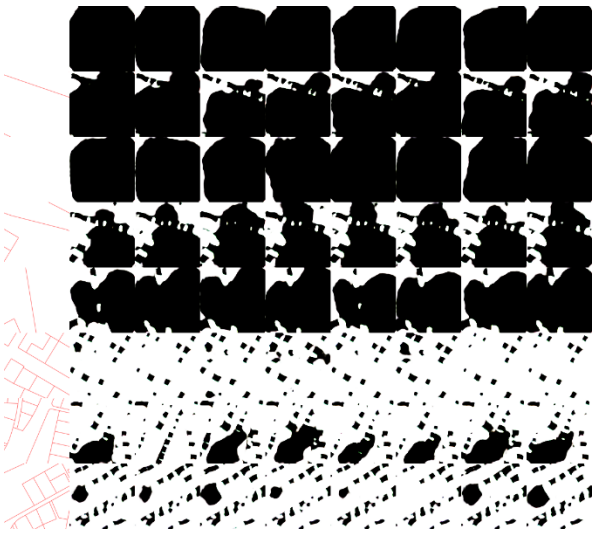
Kuvassa 23 nähdään vektoritietojoukolla tuotettuja mustia läiskiä ja nähdään artifaktien hävinneen. Mallin voidaan huomata generoivan jonkinlaisia muotoja myös rajaviivattomaan ”tyhjään” syötteeseen.

Koulutusta jatkettiin ja vektoritietojoukkoon lisättiin suuremmalta alueelta ja Turun kaupungin lisäksi dataa Hämeenlinnasta, Helsingistä ja Tampereelta.

Kuvassa 24 voidaan havaita suurennetun tietojoukon toimivan paremmin toivottujen tuloksien saamiseen, ja jo 6 600. askeleen kohdalla täplät alkoivat näyttää suhteellisen konkreettisilta rakennuspohjilta. Kuvissa 24 ja 25 näytteen tallentuessa variaatio jostain syystä hävisi, joten riveillä toistuu sama generaatio useampaan otteeseen. Relevantti osa on vasemman laidan syöte, ja sitä seuraavat kunkin rivin tulossarake.



Kuva 24. Vektoripiirretty tietojoukon näyte, 6 600. koulutusaskel.

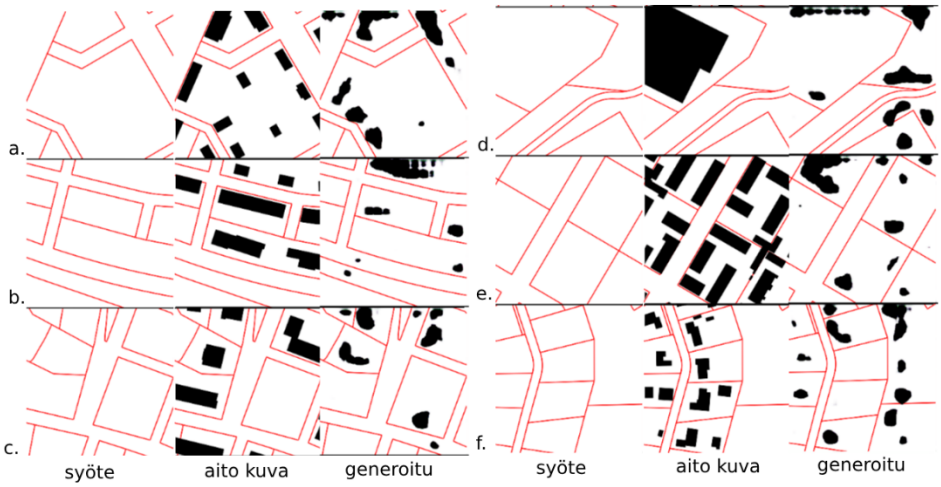


Kuva 25. Rajaviivojen puute generoinnin tuloksissa.

Osassa kuvan 25 syötekuvissa nähdään jälleen hyvin vähän rajaviivoja, joka johtaa suuriin kohdekuvan täyttävien mustien alueiden generointiin. Ongelmaa

yritettiin korjata kappaleen 4.3.3 käsitellyllä tietojoukon suodattamismenetelmällä.

Suodattamisen jälkeen arvioitiin tulosten parantuneen jokseenkin huomattavasti. Kuvassa 26 nähdään generoitujen täplien sijoittuvan tonttien rajojen reunuksille oikean maailman datan omaisesti, joskin ei täysin samankaltaisesti. Generoidut täplät ovat pyöristyneitä, mutta edistystä oli havaittavissa. Jälkikäsittelyvaiheessa kokeillaan vielä, mihin generoinnin tuloksilla voidaan yltää varsinaista sovellukseen siirrettävää lopputulosta ajatellen.

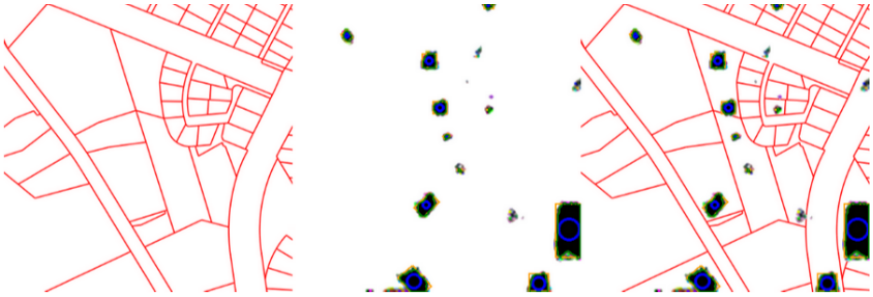


Kuva 26. Viimeisemmät generoinnin tulokset.

Satoja näyteotoksia tarkastellessa huomattiin vaihtelun onnistumisten ja epäonnistumisten näytteiden välillä edelleen suureksi, eikä merkityksellistä kehitystä tapahtunut seuraavien satojen epookkien aikana. Mallin oppimisvaikeuksien voidaan päätellä johtuvan puutteellisesta tietojoukosta, sekä mahdollisesti luvun 4.2.1. mainitusta mode-collapse -ilmiöstä. Tietojoukon parantamisesta kerrotaan loppuluvun jatkokehitysosassa.

5.4 Generoitujen kuvien jälkikäsittely

Generoiduille läiskille suoritettiin algoritmilla kontourien havaitseminen, jonka mukaan saatiin laatikkojen geometria laskennalliseen muotoon. Kuvassa 27 vasemmalla tekoälymallin syöte. Keskelle piirettynä vihreällä kontourit, kontourien sisään mahtuva ympyrä (sinisellä), sekä oranssilla laatikko, jonka sivut ja paikka lasketaan ympyrän halkaisijan ja keskipisteen avulla.



Kuva 27. Sisääntulo - Generoinnin tulos – Sijoitus sisääntulon päälle

Kuvassa 28 nähdään jälkikäsittelyn tuloksena laatikot, jotka ohjelma palauttaa varsinaiseen käyttöliittymän esitykseen.

Kuva 28. Lopullinen suorakulmiomuunnoksella jälkikäsitelty tulos.

Jälkikäsitelty lopputulos (kuva 19) näyttää kiinteistörajojen sisälle syntyneitä mukailtuja mustia suorakulmioita. Tässä kohtaa olin malliin ja käsiteltyihin tuloksiin jo tyytyväinen, ja siirryin kehitysvaiheessa generaattorin integroimiseen prototyyppisovellukseen.

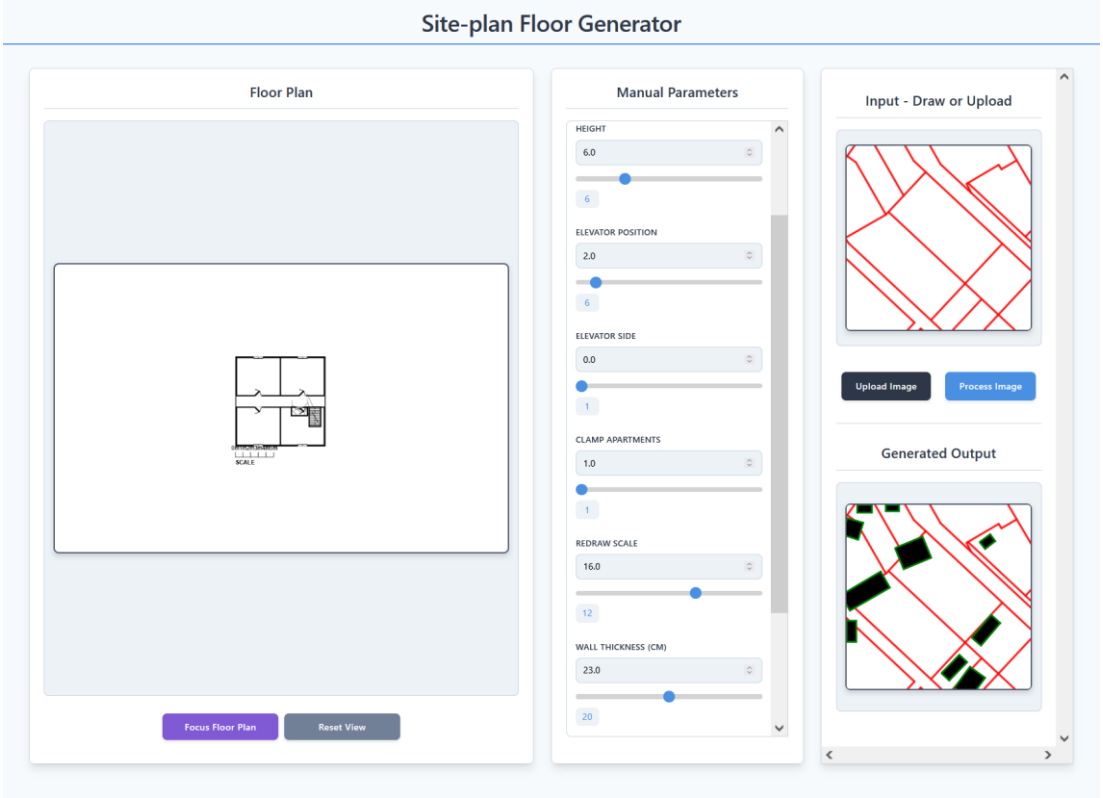
5.5 Lopullinen sovellus

Työn aikana suunniteltiin ja toteutettiin prototyyppisovellus, joka näyttää mahdollisuuden rakennusten pohjapiirrosehdotuksien generoinnille syötettyjen kiinteistörajojen perusteella.

Työn tulos osoittaa, että avoimesta datasta voidaan koota tietojoukko kuvasta-kuvaan muunnoksen tekevän tekoälymallin käyttöön. Tulokset ovat lupaavia, ja generoiduissa kuvissa rakennukset hahmottuvat tonttirajojen sisään, kun syöte vastaa koulutusaineiston tyyliä. Lisäksi generoituvat rakennukset sijoittuvat useimmiten rajojen ja kulmien lähelle, mikä näyttää mallin oppineen ainakin yhden rakennusten paikoituksen luonteenpiirteen.

Vaikka jälkikäsittely paransi tulosten selkeyttä, niiden todellinen hyödyllisyys suunnittelijalle vaatisi laajempaa käyttäjätestausta ja validointia todellisissa suunnitteluprojekteissa.

Lopullisessa sovelluksen käyttöliittymä jakautuu kolmeen sarakkeeseen, jotka nähdään kuvassa 29. Vasemmalla alkuperäisen idean parametrinen pohjapiirrostyökalu. Keskellä sen kontrolloimiseen parametrit. Oikealla tekoälyominaisuus eli syöte ja tulos canvasit painikkeineen.



Kuva 29. Sovelluksen lopullinen käyttöliittymä.

Huomiona siitä, miksi käyttöliittymän vasemmalla puolella näkyy talon pohjapiirustuksen näköinen kuva, vaikka pohjapiirustuksen piirtäminen parametrien mukaan ei ollut työn keskiössä. Halusin liittää sen kuitenkin tähän, sillä se päätettiin jättää käyttöliittymään. Suurin osa työn tunneista käytettiin tekoälymallin kanssa, ja käyttöliittymän viimeistely jäi toissijaiseksi tavoitteeksi. Tosin parametrisen pohjapiirrosesityksen jättöä voidaan perustella, sillä sovelluksen jatkokehitystä hahmotellessa rakentui ajatus tekoälyn sitomisesta toimeksiannon alkuperäiseen suunnitelmaan (työkalu suorakulmaisten kerrostalojen pohjapiirrosten generointiin). Talojen kulmien pisteet saatiin tuotua generoidun kuvan mukana käyttöliittymään, ja sovitettua rakennusten kohdille ”klikkausalueiksi”. Käyttäjä pystyisi täten hiirellä painamaan mitä tahansa generoitua rakennusta, josta sovellus voi laskemalla tuoda sivujen pituudet suoraan parametreiksi pohjapiirrosesitykseen.

6 Johtopäätökset

Tämän opinnäytetyön tavoitteena oli tutkia, miten generatiivisia kilpailevia verkkoja (GAN), erityisesti Pix2Pix-arkkitehtuuria, voisi käyttää rakennusten alustavaan sijoitteluun Maanmittauslaitoksen avoimen paikkatietodatan avulla. Työssä kehitettiin prosessi, joka muuntaa paikkatietoaineiston Pix2Pix-mallille sopiviksi kuvapareiksi, ja koulutettu malli tuotti yksinkertaisia rakennusjalanjälkiehdotuksia. Tässä luvussa arvioidaan saavutettuja tuloksia, pohditaan työn merkitystä opiskelijan näkökulmasta ja reflektoidaan prosessia kriittisesti.

6.1 Saavutukset ja menetelmän arviointi

Työn keskeinen saavutus oli prosessin kehittäminen, joka mahdollisti avoimen paikkatietodatan muuntamisen Pix2Pix-mallille sopivaksi ja mallin kouluttamisen tuottamaan alustavia rakennusehdotuksia. Malli kykeni oppimaan perustavanlaatuisia säännönmukaisuuksia, kuten rakennusten sijoittumisen kiinteistörajojen läheisyyteen. Vaikka tulokset eivät ole valmiita suunnittelukäyttöön, ne osoittavat mallin kyvyn tuottaa visuaalisesti tunnistettavia ja yksinkertaisia geometrisia muotoja. Tämä on opinnäytetyöltä hyvä tulos.

Menetelmän haasteet liittyivät erityisesti generoitujen muotojen tarkkuuteen ja yksityiskohtien puutteeseen. Mallin riippuvuus koulutusdatan laadusta korostui. Vektoripohjaisella, huolellisesti esikäsitellyllä datalla malli saatiin tuottamaan parempia tuloksia kuin kohinaista rasteridataa käytettäessä. Tämä osoittaa vektorimuotoisen datan vahvuuksia sekä datan esikäsittelyn keskeisen roolin. Pix2Pix-arkkitehtuuri, joka keskittyy kuvasta kuvaan -muunnokseen, oppii visuaalisia vastaavuuksia, mutta sen kyky hahmottaa syvällisempiä suunnitteluperiaatteita edellyttäisi monipuolisempaa dataa tai kehittyneempiä malleja.

6.2 Työn merkitys ja oppimiskokemus

Opinnäytetyö oli laajalti oppimiskokemus, jossa pääsin syventymään avoimen dataan ja tietojoukon käsittelyyn, syväoppimismallien koulutuksen ja konkreettiseen sovelluskehitykseen. Erityisen palkitsevaa oli syväoppimisen teoreettisten konseptien muuttuminen toimiviksi prototyypin palasiksi. Iteratiivinen työskentelytapa, jossa ongelmia ratkottiin askel kerrallaan, osoittautui tehokkaaksi lähestymistavaksi työn tekemiseen. Työn jälkeen jäi vahva ymmärrys siitä, miten tekoälysovellusten onnistuminen edellyttää paitsi teknistä osaamista myös ongelman tarkkaa määrittelyä ja jatkuvaa kriittistä arviointia.

6.3 Johtopäätökset

Työ osoittaa, että avoimen paikkatietodatan ja generatiivisten tekoälymallien yhdistämällä on mahdollista luota soveltavia ratkaisuja arkkitehtuuri- ja kaupunkisuunnitteluun. Prototyypin hyödyllisyyttä suunnittelutyössä tulisi arvioida tarkemmin käyttäjätestauksen ja validointien avulla, mutta tämä jäi opinnäytetyön laajuuden ulkopuolelle.

6.4 Jatkokehitysmahdollisuudet

Mallin kehittämiseksi voisi kokeilla häviöfunktioita, jotka korostavat geometrista tarkkuutta tai reunojen selkeyttä. Vaihtoehtoisten GAN-arkkitehtuurien testaaminen voisi tuoda uusia näkökulmia. Näiden yhteydessä koulutusprosessin seurantaan voitaisiin myös keskittyä tarkemmin. Esimerkiksi häviöarvojen graafinen esitys käyrinä konsolin sijaan tai näyteotosten animointi voisi tukea mallin performanssin arviointia.

Nykyinen tietojoukko kattoi vain Turun, Helsingin ja Hämeenlinnan keskusta-alueita, mutta sitä voisi laajentaa muihin kaupunkeihin ja erilaisiin kaupunkirakenteisiin, kuten pientaloalueisiin. Suomesta ja mahdollisesti myös

muista maista voitaisiin luoda monipuolisempi ja edustavampi tietojoukko. Jos datankeruu ja koulutusprosessi pidetään samakaltaisina, mikään ei estä useamman tekoälymallin koulutusta eri alueiden paikkatiedoilla.

Jatkokehityksessä voitaisiin koittaa myös laajentaa koulutusjoukkoa ylimääräisin kaupunkisuunnittelussa käytettävin parametrein, joita on listattu Erlendssonin artikkelissa ”15 parameters in Urban Design” (2020). Dataan voitaisiin koittaa sisällyttää esim. korkeusmalli-, historia- ja kaavoitustietoja, mikä voisi parantaa mallin kykyä tuottaa oikeaa maailmaa mukailevia ehdotuksia. Olisi kiinnostavaa nähdä miten kaupunkisuunnittelussa esiintyviä määreiden tuominen dataan vaikuttaisi tekoälymallin oppimiseen ja tuloksiin. Muita tällaisia parametreja ovat mm. käveltävyys, liikenne, rakennusten spatiaalinen suhde sairaaloihin, kouluihin ja muihin keskeisiin kaupungin julkisiin rakennuksiin (virastot, paloasema, poliisiasema jne.).

Lähteet

Anker K. ARCHICAD AI VISUALIZER – INNOVATIIVINEN AI-TYÖKALU (12/2023). NordicBIM-verkkoartikkeli. Viitattu 18.12.2024. https://www.nordicbim.com/fi/uutiset/archicad-ai-visualizer

Avoindata.fi. 2023. Avoin data liiketoiminnassa. Verkkosivu. Viitattu 12.5.2025. https://www.avoindata.fi/fi/tietoa-avoimesta-datasta/avoin-data-ja-liiketoiminta

Dealroom.co. 2024. Kaavio. Viitattu 18.12.2024.

Eastman, C. M. 2008. BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors. John Wiley & Sons. ISBN 978-0470185285

Creative Tools. Explore the Future of Property Design with Autodesk Forma. N.d. Arkistoitu tuotekuvaus, Internet Archive. 19.4.2024. Viitattu 6.1.2025. https://web.archive.org/web/20240419185240/https://www.creativetools.fi/software/autodesk-forma

Erlendsson, G. T. 2020. 15 Basic parameters in urban design. Thor Architects. Viitattu 11.11.2024. https://www.thorarchitects.com/15-parameters-urban-design/

Euroopan komissio. 2019. Euroopan parlamentin ja neuvoston direktiivi (EU) 2019/1024. Euroopan unionin virallinen lehti. https://eur-lex.europa.eu/legal-content/FI/TXT/?uri=CELEX:32019L1024

Euroopan komissio. 2024. KOMISSION TÄYTÄNTÖÖNPANOASETUS (EU) 2024/2952. Euroopan unionin virallinen lehti. https://eur-lex.europa.eu/legal-content/FI/TXT/?uri=CELEX:32024R2952

Huang J.; Kossaifi J.; Anandkumar, A. & Yue, Y. 2020. HAGO: Hardware-Aware Graph Optimization for Neural Network Inference. CVPRW. https://doi.org/10.48550/arXiv.2104.12766

Huang J. 2021. “How to clear memory completely of all matplotlib plots”, Käyttäjän vastaus, Stackoverflow. Viitattu 26.12.2024 https://stackoverflow.com/a/65910539

Isola, P; Zhu, J-Y; Zhou, T. & Efros, A. A. 2017. Image-to-image translation with conditional adversarial networks. Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). https://doi.org/10.1109/CVPR.2017.632

JUHTA - Julkisen hallinnon tietohallinnon neuvottelukunta. 2019. JHS 180 Paikkatiedon sisältöpalvelut. Viitattu 18.2.2024. https://docs.google.com/document/d/1CT173PSOLf_6EQJuHbWtkx6P0zqxySxX

Kallio, R. 2021. Neuroverkon opettaminen vastavirta-algoritmilla. Opinnäytetyö. Turun Yliopisto. https://urn.fi/URN:NBN:fi-fe2021052030929

Kingma, D. P. & Ba, J. 2015. Adam: A Method for Stochastic Optimization. Proceedings of the 3rd International Conference on Learning Representations (ICLR). https://doi.org/10.48550/arXiv.1412.6980

Kousa, R. 2012. MicroStationin käyttökartoitus Pöyry Finland Oy:lle. Opinnäytetyö, Kymenlaakson ammattikorkeakoulu. https://urn.fi/URN:NBN:fi:amk-2012052810387

Lindernoren E. 2019. PyTorch-GAN. Github-repositorio. https://github.com/eriklindernoren/PyTorch-GAN/

Maanmittauslaitos. Tekninen kuvaus (WMTS). Viitattu 10.1.2024. https://www.maanmittauslaitos.fi/karttakuvapalvelu/tekninen-kuvaus-wmts

Morán J. A., Ricoy P. M. 2023. Autodesk Spacemaker: Diseño Generativo e Inteligencia Artificial para desarrollos inmobiliarios más sostenibles. Webinaari. Arkance Iberia.

Nordic BIM. N.d. BIM eilen, tänään, huomenna. Verkkoartikkeli. Viitattu 25.5.2025. https://www.nordicbim.com/fi/bim-eilen-tanaan-huomenna

Open Data and AI Policy (FI0033), Open Government Partnership. EU avoin data.

Oulun kaupunki. 2024. Karttapalvelu. Viitattu 12.11.2024. https://kartta.ouka.fi/

PATINE, Suositus julkisen hallinnon paikkatietoja koskevista rajapinnoista, Paikkatietoasiain neuvottelukunta. Toukokuu 2020., Viitattu 18.12.2024

Pietilä, V. 2021. ISO-19650 – rakennusalan toimintakulttuurin systeeminen muutos. Nordic BIM Group blogiartikkeli. Viitattu 25.5.2025. https://www.nordicbim.com/fi/bimblogi/iso-19650-rakennusalan-toimintakulttuurin-systeeminen-muutos

RT 10-11222. 2016. Suunnitteluprosessin vaiheet. Rakennustieto Oy. Viitattu 25.4.2025.

The World Bank. 2014. OPEN DATA FOR ECONOMIC GROWTH. Verkkodokumentti. Viitattu 18.12.2024.

Tuominen, H. 2019. Tekoälyn perusteita ja sovelluksia. Jyväskylän yliopisto. ISBN 978-951-39-7796-2

Turku.fi. N.d. Avoimen datan lupa. Viitattu 12.11.2024.

Yu A. & Grauman K. Fine-Grained Visual Comparisons with Local Learning. In CVPR, 2014. https://doi.org/10.1109/CVPR.2014.32