MINDTITAN

Juhend koodi lugemiseks

Lähtekood on failis source-code.zip.

Juhend koodi jooksutamiseks

Aknatuvastusmudel võtab sisendiks .tif faili ja leiab sealt aknad. Aknatuvastusmudeli väljundiks on .gpkg fail. Lahenduse kasutamiseks on loodud dockeri konteineri.

Nõuded käivitamiseks

Käivitamiseks on vajalik docker. https://www.docker.com/. Testitud versiooniga 24.0.6. Testitud operatsioonisüsteemidel Ubuntu 22.04 ning Windows 11.

Kasutamine

Kasutamiseks on vaja kahte parameetrit:

1. sisendkaust, kus asub sisendfail - {sisendkaust}

2. sisendfaili nimi - {sisendfail}

Jooksutamiseks: *docker run -v {sisendkaust}:/mnt -e INPUT_FILE={sisendfail} -it madiskarli/window-detection-proovitoo*

Väljundiks on sisendkaustas fail {sisendkausta}/{sisendfail}_windows.gpkg. Näiteks `/home/kasutajanimi/projektinimi/sisendid/pildinimi_windows.gpkg`

Ubuntu näide (terminalis): *docker run -v /home/kasutajanimi/projektinimi/sisendid:/mnt -e INPUT_FILE=pildinimi.tif -it madiskarli/window-detection-proovitoo*

Windows 11 näide (powershellis): *docker run -v c:\Users\Kasutaja\projektinimi\sisendid:/mnt -e INPUT_FILE=pildinimi.tif -it madiskarli/window-detection-proovitoo*

Graafikakaardiga kasutamise näide (Ubuntu): docker run --gpus all -v /home/mkkoppel/maaamet/test_input:/mnt -e INPUT_FILE=img_12.tif -it madiskarli/window-detection-proovitoo

Märkused

Vaikimisi jookseb kood CPU (protsessori) peal. Kuid see on kõvasti kiirem GPU (graafikakaardi) peal. Meie testides töötab ideaalselt kui panna dockeri käsule juurde `--gpus all` ja seejärel on võimalik jooksutada lahendust kasutades GPUd. Selle jaoks on vaja teha mõned muudatused dockerile, aga neid ei ole selles juhendis kirjeldatud. See aga on edasijõudnutele mõeldud kasutusjuht.

Üks 10652 x 14204 hankepilt võtab RTX 2080 Ti graafikakaardil aega alla minuti. Windows 11 sülearvutil, millel pole graafikakaarti, võtab sama pilt 40 minutit.

Mudeli tehnilise toimimise kirjeldus

Mudeli arhitektuuri põhjendus

Proovitöö hindamiseks viidatud täpsuse hindamise link https://pro.arcgis.com/en/pro-app/3.0/tool-reference/image-analyst/compute-accuracy-for-object-detection.htm ning sealt edasi https://pro.arcgis.com/en/pro-app/3.0/tool-reference/image-analyst/how-compute-accuracy-for-object-detection-works.htm viitavad, et oleks olnud variant ka teha objektituvastust antud proovitöös. Otsustasime segmentatsiooni ülesande kasuks, kuna see ühtib suuremal määral plaanitud tegeliku töö sisuga. Teadvustame, et tõenäoliselt meie enda arvutatud mõõdikud ei ühti 1:1 mõõdikule, mida proovitöö hindaja kasutab. Huvitava võrdluse võimaluse pärast alustasime ka objektituvastuse põhise lähenemisega proovitöös, kuid see jäi ajasurve tõttu praegu pooleli.

Mudeli treenimise metoodika selgitus ja põhjendus

Valitud mudel võttis sisendiks 512×512 pilte. Kuna algsed pildid olid oluliselt suuremad, lõikasime need eelnevalt Pythonis kirjutatud skripti abil 512×512 alampiltideks. Lõikamiseks kasutasime mitte-kattuvat liikuvat akent (non-overlapping sliding window), kuna see meetod on kiire, lihtsasti hallatav ning vähendab dubleeritud andmete teket.

Kuigi ühe alternatiivina oleks olnud võimalik kasutada kattuvat liugakent (overlapping sliding window), tähendanuks see oluliselt suuremat andmehulka ning ohtu, et sama pildi piirkond võib sattuda treening-, valideerimis- ja testandmestikku korraga. Meie andmestikus oli aga iga algne pilt iseseisev ning seetõttu võis neid ohutult erinevateks andmestikeks jagada.

Lõime kaks treeningandmestikku:

- Esimene andmestik sisaldas kõikidest piltidest lõigatud alampilte ja võimaldas hinnata mudeli üldist täpsust kogu andmestiku ulatuses.
- Teine andmestik põhines lihtsustatud leave-one-out cross-validation meetodil, kus üks pilt jäeti kõrvale ja ülejäänuid kasutati treeninguks. See võimaldas hinnata, kui hästi mudel töötab pildil, mida ta pole varem näinud.

Kuigi esialgu oli plaanis kasutada täismahus ristvalideerimist (cross-validation), jäi see teostamata ajapuuduse tõttu – mudeli treenimine kahe GPU peal võttis ligikaudu 8 tundi.

Treeningu käigus viisime läbi mitmeid katseid erinevate mudeliarhitektuuride ja andmetäiendusmeetoditega (data augmentation). Lõppvalikuks osutus ResNet50-UNet, mis andis parimaid tulemusi. UNet on laialdaselt kasutatav segmentatsioonimudel, eriti satelliit- ja aerofotode kontekstis (nt: https://github.com/satellite-image-deep-learning/techniques).

Kuna andmestik oli väike, kasutasime rohkem andmetäiendusvõtteid. Kasutatud augmentatsioonid:

- VerticalFlip
- HorizontalFlip
- RandomRotate90
- RandomBrightnessContrast
- HueSaturationValue
- GaussianBlur
- GaussNoise

Täiendavalt oleks olnud võimalik rakendada ka keerukamaid transformatsioone.

Segmentatsiooni ülesandes esines märkimisväärne klasside tasakaalutus: enamus pikseleid kuulusid taustale (mitte-aknad). Selle tasakaalustamiseks arvutasime klasside kaalud, kasutades normaliseeritud pöördsageduse (Normalized Inverse Frequency) valemit:

klassi_kaal = (kõikide näidete arv) / (klasside arv * klassi_näidete_arv).

Lõplikud kaalud olid:

- Aken: 61.44
- Mitte-aken: 0.5

Hinnang edukusele

Mudeli tulemused testandmestiku peal on paljulubavad. Järgnevad tulemused pärinevad treeningust, kus testpildiks oli pilt 12.

Üldised hindamismõõdikud	
mIoU	84.7957
Overall Accuracy	99.8392
Fscore	91.0505
Precision	92.8188
Recall	89.4237

Klassipõhised hindamismõõdikud					
klass	Klassi kaal (*support*)	IoU	F-score	Precision	Recall
aken	61.44	69.7529	82.1817	85.7372	78.9093
mitte-aken	0.5	99.8386	99.9192	99.9005	99.9380

Tulemused treeningutel, kus hindamine toimus kõikide sisendpiltide peal, olid võrreldavad ülaltoodud tulemustega. Kuna tegemist on tulemuste hindamisega ainult ühel pildil, siis võib eeldada, et sarnased tulemused on ka hinnangus kasutatavale testpildil.

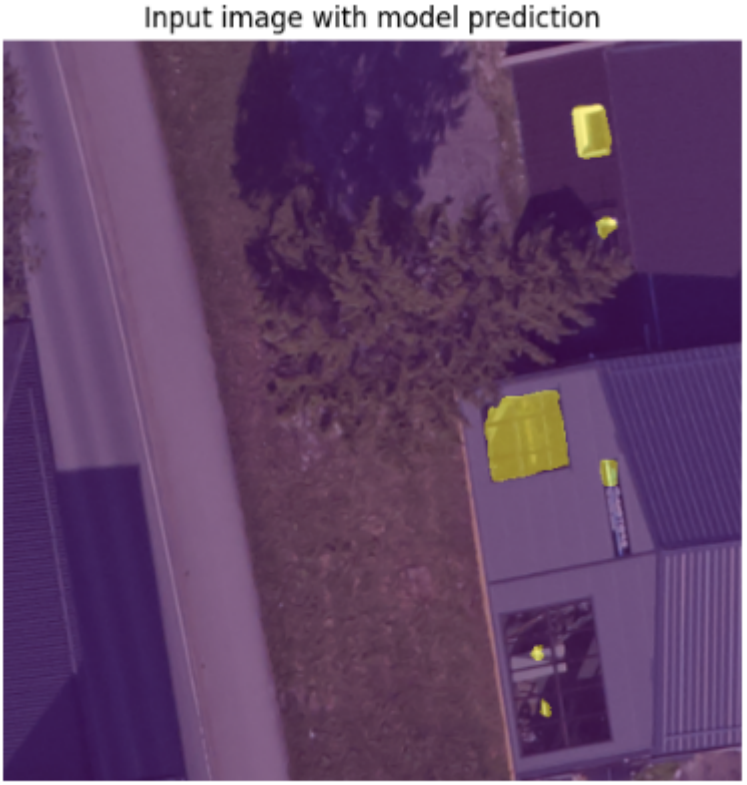
Juhime siinkohal tähelepanu sellele, et antud mõõdikud on segmentatsioonimõõdikud ning on pikslitasemel, mistõttu ei pruugi need otseselt kattuda objektipõhiste hindamismeetoditega.

Kui võtta kasutusele IoU lävend ning võrrelda tuvastatud akna vektoreid märgendatud vektoritega on numbrid teistsugused. Iou=0.05:

- F1 Score: 67.70
- Precision: 67.72
- Recall: 67.67

Mudeli ennustusi üle vaadates täheldasime mitmeid olukordi, kus mudelil esines raskusi.

Näiteks ei leidnud mudel hästi üles akent, millest nägi väga hästi tuppa sisse.

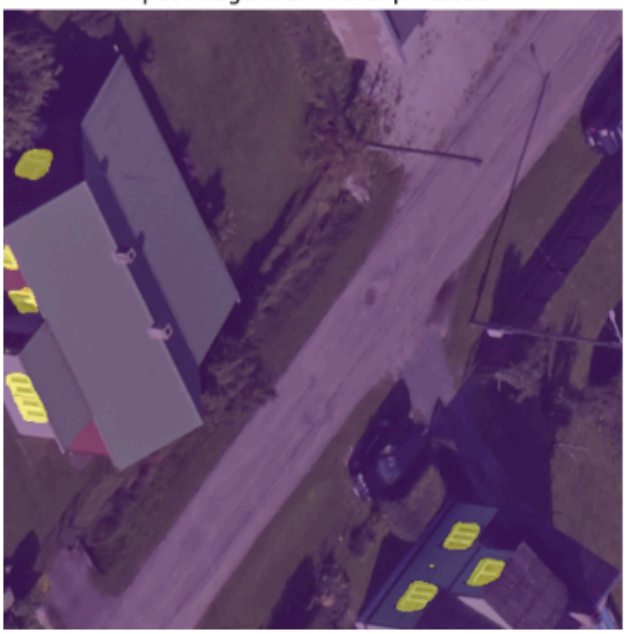


Pildil on kollasega mudeli tuvastatud aknad

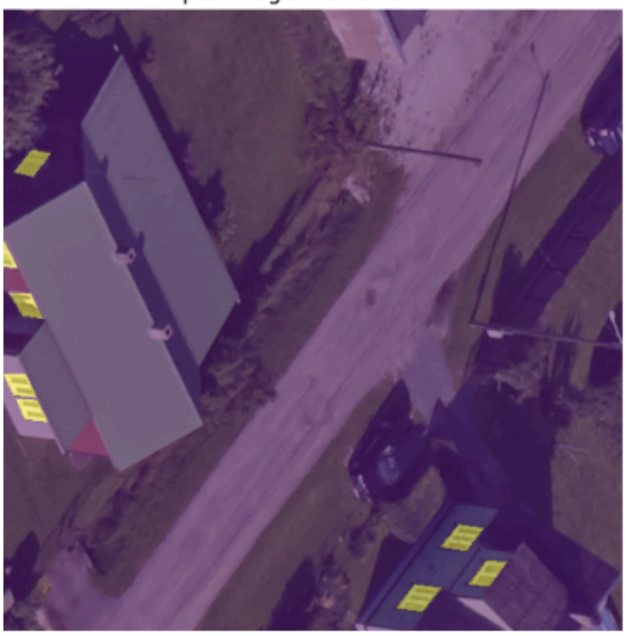
Katusel esinevad tugeva kontrastiga toonimuutused (näiteks rooste) põhjustasid valepositiivseid tuvastusi, mida mudel pidas ekslikult katuseakendeks. Õnneks olid sellised eksimused väga väikeste segmentide kujul ja neid saab lihtsalt eemaldada järeltöötluse käigus.

Lähestikku asuvad aknad: mõnikord sulatas mudel need üheks aknasegmendiks. Näiteks ülemisel pildil on mudeli väljund, alumisel aga märgendatud sisendpilt. Vasakpoolse maja kõige alumised kaks akent on märgendatud eraldi, kuid mudel tuvastas need ühe aknana.

Input image with model prediction



Input image with true label



Üldiselt on mudeli tulemused head ning neid saab järeltöötluse abil lihtsalt veel paremaks teha. Väikeste segmentide eemaldamine (nt valepositiivsed alad, mis on liiga väikesed, et olla aknad) parandab lõpptulemust märgatavalt ja suurendab usaldusväärsust.