Norges miljø- og biovitenskapelige universitet

Masteroppgave 2022 30 stp
Fakultet for realfag og teknologi

Semantisk segmentering av objekter på hustak i flybilder

Semantic segmentation of rooftop objects in aerial images

Marius Helsem
Geomatikk

Forord

Med denne masteroppgaven konkluderer jeg min tid som geomatikk-student ved Norges miljø- og biovitenskapelige universitet i Ås. Jeg vil takke min hovedveileder Ivar Maalen-Johansen for god rådgivning og tilgjengelighet gjennom hele prosjektet. Oppgaven er skrevet for Blom Norway As, og jeg vil takke mine tillegsveiledere Floris Jan Groesz og Stian Rostad for all tiden de har satt av til dette prosjektet. Til slutt vil jeg takke mine medstudenter for fem flotte år i Ås.

Ås, mai 2022
Marius Helsem

Sammendrag

I dette prosjektet utforskes muligheten til å detektere objekter gjennom semantisk segmentering av flybilder ved bruk av U-Net med et forhåndstrent ResNet34 som grunnlag.

Det finnes store mengder flybilder av forskjellige typer som er offentlig tilgjengelig og som dekker store deler av Norge. Bildene tatt i bruk i dette prosjektet produseres av Blom Norway AS og er åpent tilgjengelig via karttjenesten til Opplysningen 1881. Dette er en enorm ressurs, og denne oppgaven presenterer en metode for å benytte noe av den informasjonen som er tilgjengelig.

Metoden beskriver nødvendig forarbeid og trening for å lære et nevralt nett å detektere tak-objekter i skråbilder og ortofoto tatt fra fly. Objektene som detekteres i denne oppgaven er begrenset til piper, takstiger og takvinduer, men metoden muliggjør detektering av en hvilken som helst samling klasser. Arbeidet resulterer i to modeller, der den ene er trent med data fra et område i form av skråfoto, mens den andre har blitt trent med data basert på ortofoto av samme området. De to modellene sammenliknes og det konkluderes med at ortofoto er marginalt bedre egnet til bruken i denne oppgaven.

Abstract

This project explores the possibility of detecting objects in aerial images through semantic segmentation using a U-Net with a pre-trained ResNet34 as a backbone.

There are large amounts of aerial photographs of various types that are publicly available and that cover large parts of Norway. The images used in this project are produced by Blom Norway AS and are openly available via the map service from 'Opplysningen 1881'. This is an enormous resource, and this thesis presents a method using some of the information that is available.

The method describes the necessary preparations and training to teach a neural network to detect rooftop objects in oblique images and orthophotos captured from airplanes. The objects detected in this task are limited to chimneys, roof ladders and skylights, but the method enables the detection of any collection of classes. The method results in two models, where one has been trained with data from an area in the form of oblique photos, while the other has been trained with data based on orthophotos of the same area. The two models are compared and it is concluded that orthophotos are marginally better suited for the use in this thesis.

Innhold

Figurer

Tabeller

Kapittel 1

Introduksjon

1.1 Bakgrunn og motivasjon

Fjernmåling kan defineres som innsamling av informasjon om objekter eller miljøer man ikke er i fysisk kontakt med.[6] Fly- og satellittbilder er en form for fjernmåling der det produseres enorme mengder data, og motivasjonen bak denne oppgaven har vært å utnytte noe av potensialet som ligger i denne informasjonen.

I Norge samarbeider kommunene med Kartverket om å utvikle og vedlikeholde vår felles kartdatabase (FKB) [7]. Dette er en standardisert samling kart over Norge med mange forskjellige tema og detaljnivåer. Dagens kartleggingsmetoder er veldig ressurskrevende, og derfor er innholdet i kartene begrenset. For eksempel gjør manuelle metoder at det er uhensiktsmessig å kartlegge bygninger i mer detalj enn kun bygningsomriss.

Automatisk deteksjon av takobjekter i flybilder vil ha stor verdi da dette muliggjør mer detaljert kartlegging av for eksempel bygninger. Der man før kun hadde oversikt over omrisset til en bolig vil man kunne se hva boligen har på taket, som igjen gir mer informasjon om bruken av boligen. Har man en pipe har man et illsted, og har man store takvinduer kan man regne med loftet som bruksareal. Det er mye nyttig informasjon som kan deriveres fra kart, men da må kartene være så detaljerte som mulig!

En mulig fremgangsmåte for å oppnå dette er ved bruk av kunstig intelligens, ettersom nevrale nett kan trenes opp til å detektere objekter i bilder. Disse nettene er i dag dypere og mer komplekse en noen gang, mye grunnet utviklingen av datamaskiner med enorm beregningskraft. Dette åpner for nye muligheter innen objektdetektering i bilder, som skal utforskes i denne oppgaven.

1.2 Forskningspørsmål

Målet med denne oppgaven er å undersøke i hvilken grad kunstig intelligens kan brukes til å detektere tak-objekter i flybilder. Nettet som settes opp har arkitekturen til U-net og bruker ResNet34 som grunnlag. I den forbindelse er det definert to forskningsspørsmål det ønskes å finne svaret på:

- **1:** I hvilken grad kan skrå- og ortofoto brukes som grunnlag for automatisk objekt-detektering ved bruk av et U-net med ResNet34 som grunnlag?
- **2:** Hvilken av de to datatypene skrå- og ortofoto er best egnet til å automatisk detektere objekter på tak i boligområder?

1.3 Oversikt

- **Kapittel 1:** Introduksjon
- **Kapittel 2:** Teori
- **Kapittel 3:** Metode
- **Kapittel 4:** Resultater
- **Kapittel 5:** Diskusjon
- **Kapittel 6:** Konklusjon

Kapittel 2

Teori

2.1 Ortofoto

Ortofoto er bilder tatt fra fly, som har samme geometriske egenskaper som kart.[8] Vanlige flybilder har i utgangspunktet en sentralprojeksjon, der målestokken endrer seg utifra hvor i bildet man ser. Denne effekten, som oppstår fordi punkter på bakken har forskjellig avstand til sensoren, kan unngås gjennom en rektifiseringsprosess basert på bildets orientering og en høydemodell av det aktuelle området.[8]

2.2 Skråfoto

Skråfoto er en type fjernmålings-bilder som er tatt fra fly, der kameraaksen har et avvik fra den vertikale linja mellom flyet og bakken.[9] Flybilder får flere bruksområder når man får sett store områder fra dette perspektivet. For eksempel vil observasjon av fasaden til bebyggelse muliggjøres, og bildene blir mer like den oppfatningen vi mennesker har av et område. [9]

2.3 Kunstige nevrale nettverk

Et kunstig nevralt nett er en beregningsmodell som er bygget opp av enheter kalt nevroner, som er inspirert av hvordan hjernen vår lærer ny informasjon. Nevronene er koblet sammen av vekter og fordelt i forskjellige lag som utgjør strukturen til nettet. Det er hovedsaklig gjennom oppdatering av disse vektene at et nett lærer. Nevronene i ett lag tar vektene fra nevronene i det forrige laget som inngangsverdi, samler disse verdiene, sender den nye verdien gjennom en aktiveringsfunksjon. Da står man igjen med en ny vekt som sendes som utgangsverdi til nevronene i neste lag.[5]

2.3.1 Aktiveringsfunksjoner

Aktiveringsfunksjonens rolle i nevrale nett er å påvirke utgangsverdien til nevronene. Det finnes forskjellige aktiveringsfunksjoner som er egnet for ulike scenarier.

Sigmoid

Sigmoid-funksjonen returnerer et tall mellom 0 og 1. Dette er egnet når man skal finne sannsynligheter, som alle kan legges mellom 0 og 1. I en binær situasjon med to klasser kan man definere en terskel på 0.5, og bestemme klassen til en forekomst utifra hvilken side av terskelen utgangsverdien ender på. [10]

Softmax

Softmax som aktiveringsfunksjon er satt sammen av flere sigmoid-funksjoner. Sigmoid-funksjonen brukes kun i binære klassifiseringsoppgaver, men når flere sigmoidfunksjoner kombineres til en softmax-funksjon kan man utføre klassifisering av flere klasser på en gang. Utgangsverdien er mellom 0 og 1, og beskriver sannsynligheten for at en forekomst tilhører hver klasse.[11]

2.3.2 Tapsfunksjoner (eng. Loss functions)

Tapsfunksjoner måler hvor godt et nettverk klassifiserer. Det finnes flere ulike typer tapsfunksjoner, men de har alle til felles at de måler hvor langt unna perfeksjon nettverket er.[5]

Dice loss

Dice loss er en tapsfunksjon som vektlegger læringsbidraget til klasser uavhengig av størrelse. Dersom en klasse i et datasett består av mindre data enn en annen klasse vil Dice loss fokusere mer på den minste klassen så den ikke blir 'glemt'. Dice loss er også effektivt hvis objekter sin størrelse varierer. Et stort objekt kan bestå av langt flere piksler enn et lite objekt, men Dice loss vektlegger læringsbidraget fra de to objektene likt.[12]

Focal loss

I et datasett kan noen forekomster være lette å lære, og noen mer utfordrende for modellen. Focal loss vektlegger læringsbidraget fra de vanskeligere eksemplene høyere enn læringsbidraget fra de lettere eksemplene. Vektleggingen av bidraget fra de lette eksemplene blir altså skalert ned så modellen får fokusert mer på det den sliter med. Dette er også en egnet tapsfunksjon for datasett med svært ujevn klassefordeling. [13]

2.3.3 Konvolusjonelle nevrale nett

Konvolusjonelle nevrale nett er en type nett som er passende for å håndtere store mengder data. Navnet kommer fra den matematiske operasjonen konvolusjon, som går ut på at to funksjoner kan beskrives av en tredje funksjon [14]. Konvolusjonelle nett er egnet for å lære

av bilder fordi man unngår den enorme mengden vektene man får hvis man bruker et dense neural network.

Konvolusjonelle nevrale nett lærer gjennom en prosess kalt 'feature extraction'. Under treningen av nettet konstrueres ulike filtre for å kjenne igjen ulike former i bildet. For eksempel kan ett filter være egnet til å kjenne igjen horisontale kanter, og et annet være egnet til å kjenne igjen firkanter av en viss størrelse. Forskjeller i skalering og oppløsning gjennom de forskjellige lagene gjør at nettet lager filtre som øker i kompleksitet jo dypere i modellen de ligger.

2.3.4 U-NET

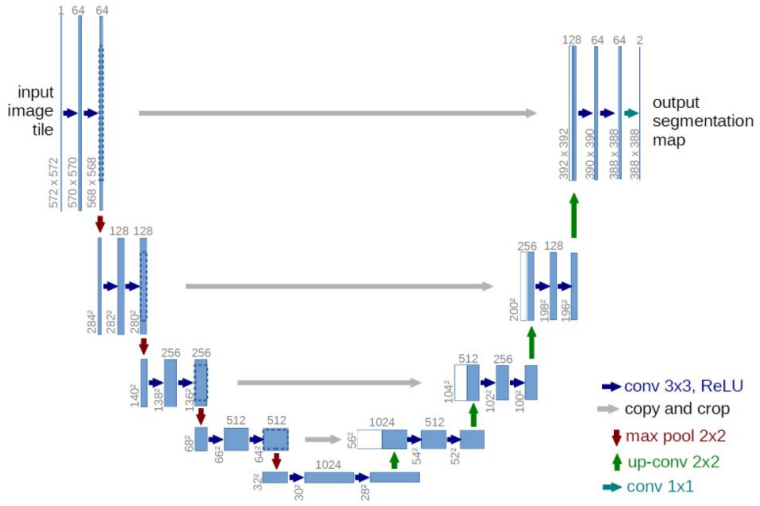
U-net er en nettverksarkitektur foreslått av Ronneberger, Fisher og Bronx i 2015. Arkitekturen bygger på konvolusjonelle nevrale nett, og hensikten er å tilby en løsning som gir nøyaktig segmentering selv om man har få treningsbilder. Motivasjonen bak U-net stammer fra behovet for å segmentere biomedisinske bilder. Segmentering av bilder av cellekjerner er et eksempel på dette.[15] Produsering av treningsdata for slik segmentering er en tidkrevende prosess, da dette krever manuell annotering gjennomført av personell med spesiell kunnskap. Derfor er u-net som metode for segmentering i slike situasjoner svært nyttig, da man oppnår høy presisjon med lite data. [1].

U-net er bygget opp av én del for sammentrekking og én del for utstrekking. Sammentrekkingsdelen vil i likhet med et konvolusjonelt nettverk skalere og endre oppløsningen på inngangsbildene for å lage filtre for forskjellige detaljnivå. Utstrekkingsdelen er der U-net skiller seg ut ved at konvolusjonsprosessen reverseres i like steg som i sammentrekkingsdelen. Bilder med lik oppløsning fra hver del sammenkobles og kombinerer informasjonen fra de to lagene for økt presisjon. Figur 2.1 viser hvordan de forskjellige lagene i U-net er koblet sammen, hvilke operasjoner som påføres hvert lag og hvordan dette påvirker dimensjonene til laget.

2.4 Parametere

Når konteksten er maskin- og dyp læring kan parametere deles inn i to kategorier, trenbare vekter og hyperparametere. [16]

- **Trenbare vekter** er parametere som oppdateres underveis i læringsprosessen. En modell kan bestå av millioner av disse.
- **Hyperparametere** er parametere som defineres før treningen starter, og som påvirker hvordan modellen er satt opp og går frem for å lære.



Figur 2.1: Visualisering av U-net-arkitekturen. Figur hentet fra [1]

2.4.1 Hyperparametere

Læringsrate

Læringsrate (eng: Learning rate) er en koeffisient som påvirker størrelsen på endringen av vektene under læringsprosessen. Med større læringsrate lærer modellen fortere ved at stegene vektene tar mot optimal verdi blir lengre. Høyere læringsrate fører altså til mer effektiv læring, men hvis stegene blir for lange risikerer man å bomme på den optimale verdien. Derfor er det viktig å finne en god læringsrate så vektene når fram så fort som mulig, men ikke går for langt. [5]

Optimaliserer (eng: Optimizer)

Optimaliserer er betegnelsen på en algoritme som påvirker læringsraten underveis i opplæringen av modellen. En fast satt læringsrate kan være for høy og for lav i samme læringsprosess, og derfor er det utviklet metoder for å justere læringsraten så den passer til enhver tid. [5] [10]

- **Adam** er et eksempel på en optimaliserer. Den virker ved å tilpasse læringsraten til hver enkelt parameter slik at de ikke oppdateres for fort eller for sakte. Dette gjøres ved å se på momentet til gradienten til hver vekt.[17]

Oppdeling av datasett

Det må bestemmes hvor stor del av datasettet som settes av til trening og validering. Det er vanlig å sette av 10-20 prosent til validering. Jo mer man setter av til validering, jo mer nøyaktig kan modellens virkelige prestasjon beregnes. Problemet med en høy andel

valideringsdata er at man mister mye data modellen kunne brukt til å lære. Derfor er det vanlig å trene den endelige modellen en siste gang med all tilgjengelig treningsdata før den brukes til problemløsningen den er ment for.

2.5 Gjenværende læring (eng: Residual learning)

Jo dypere nevrale nett er, jo vanskeligere blir de å trene opp. En av hovedutfordringene med dype nett er det vi kaller 'Vanishing gradient decent'. Dette innebærer at vektene i nettet blir så små at de ved neste oppdatering ikke endrer seg nok til å bidra med noe betydelig læring. Gjenværende læring har blitt presentert som en løsning på dette problemet. [2]

Gjenværende læring forenkler nevrale nett ved å bruke 'skip connections'. Vanligvis sender ett lag verdiene sine til neste lag, og kun dette laget. Med 'skip connections' sendes verdiene til både neste lag og et annet lag som er et par steg dypere i nettet. Det mottagende laget kombinerer informasjonen fra laget over seg med laget som brukte skip-connection. Dette motvirker 'vanishing gradient decent' ved at vektoppdatering i sistnevnte lag blir holdt tilbake av informasjonen fra det tidligere laget. [18] Figur 2.2 viser oppbygningen til et typisk residualt nettverk.

2.6 Kunstig syn

Å kjenne igjen hva hva man ser på er noe vi mennesker er ekstremt gode på. Hvordan denne egenskapen kan overføres til maskiner har vist seg å være en stor utfordring, som etterhvert har utviklet seg til å bli et eget fagfelt innen maskinlæring. Kunstig syn har fått mer oppmerksomhet det siste tiåret mye grunnet den økte interessen og utviklingen av autonome kjøretøy. Det er viktig at en selvkjørende bil vet hva den ser, før den kan slippes ut på veien! [19]

Kunstig syn kan implementeres på flere forskjellige måter, her nevnes noen:

- **Bildeklassifisering**

  Et bilde blir presentert og modellen bestemmer hvilken klasse hele bildet tilhører.

- **Objektdetektering**

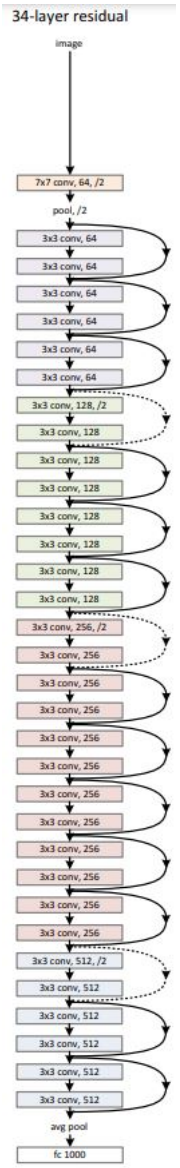
  Et bilde blir presentert og modellen bestemmer hvor relevante objekter er i bildet og hvilken klasse hvert objekt tilhører. Objektets posisjon markeres, vanligvis med en boks.

- **Semantisk segmentering**

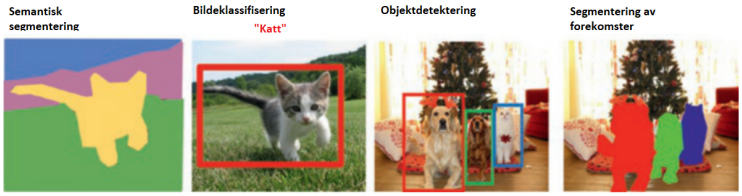
  Et bilde blir presentert og modellen bestemmer hvilken klasse hver piksel i bildet tilhører. Det skilles ikke mellom forskjellige forekomster av samme klasse i bildet.

- **Segmentering av forekomster** (eng. Instanse segmentation)

  Virker på samme måte som semantisk segmentering, men her skilles det også mellom forskjellige forekomster av samme klasse. Så hver piksel tildeles en klasse og forekomst.



Figur 2.2: Oppbygningen til et typisk residualt nettverk. Figur hentet fra [2]



Figur 2.3: Visualisering av ulike typer kunstig syn. Hentet fra [3]

2.7 Semantisk segmentering

Semantisk segmentering er en form for styrt læring. Målet er å definere hvilken klasse hver enkelt piksel i bildet tilhører. For å gjøre dette må man konstruere et datasett med bilder og tilsvarende annoterte masker som treningsdata.

- **Bildene** fremstilles som en 3-dimensjonal matrise med en høyde, bredde og dybde. Høyden og bredden er oppløsningen til bildet, og dybden er antall kanaler bilder består av. Det er mest vanlig med RGB-bilder, som består av 3 kanaler tilsvarende røde, grønne, og blå verdier.
- **Maskene** fremstilles som en 2-dimensjonal matrise med høyde og bredde tilsvarende sitt bilde. Verdiene i matrisen representerer klassen til pikslene. Antall klasser kan variere, fra binære datasett med objekt og bakgrunn, til datasett der alt i bildet er klassifisert. Det er mange tjenester der man kan lage masker til egne datasett ved å definere klasser og tegne dem inn i bildene. Figur 2.4 viser et eksempel på hvordan slike masker ser ut.

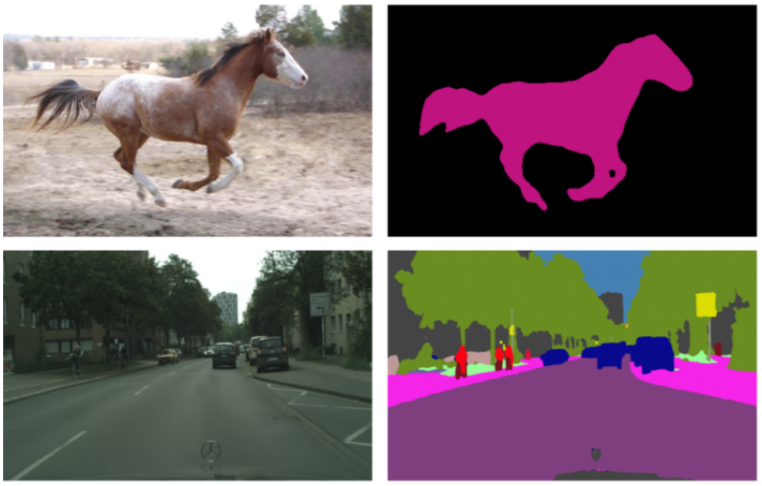
2.8 Prestasjons-beregninger (eng. Training metrics)

For å evaluere hvor godt en modell presterer beregner man ulike verdier som beskriver presisjonen til modellen når den testes på evaluerings-data. Hensikten med dette er å vite hvordan presisjonen påvirkes når parameterene til modellen blir endret.

2.8.1 Forvirringsmatrise (eng. Confusion matrix)

Forvirringsmatrisen er en tabell med rader og kolonner som representerer prediksjoner modellen har gjort mot sanne verdier. Dette er et nyttig verktøy for å skjønne hvordan modellen presterer. [5] Matrisen er kvadratisk med antall rader og kolonner tilsvarende antall klasser i datasettet. Figur 2.5 viser forvirringsmatrisen for en binær klassifisering.

Forvirringsmatrisen for semantisk segmentering vil både vise hvor mange piksler som blir klassifisert korrekt, og hvilken klasse de feilklassifiserte pikslene blir definert til. For eksempel vil rad 1 kolonne 2 vise hvor mange piksler fra klasse 1 som blir feilaktig definert til klasse 2.



Figur 2.4: Eksempel på bilder og binær maske(øverst) og flerklasse-maske(nederst) Figur hentet fra [4]

2.8.2 Nøyaktighet

Nøyaktighet er kanskje den mest intuitive måten å evaluere prestasjonen til en modell. For å beregne nøyaktigheten til en modell deler man antallet korrekte prediksjoner på antallet prediksjoner.[5] Dersom modellen brukes til semantisk segmentering ville man sett på forholdet mellom korrekte klassifiserte piksler mot feil-klassifiserte piksler. Har man datasett med skjev klassefordeling vil man da støte på store problemer, ettersom nøyaktighet i en slik sammenheng vil være veldig misvisende. I et eksempel der 95 prosent av pikslene i et bilde er bakgrunn kan modellen regne seg frem til at hele bildet er bakgrunn og oppnå 95 prosent nøyaktighet selv om ingen andre objekter er funnet. Derfor er det vanlig å se til andre prestasjons-beregninger når man gjennomfører semantisk segmentering.

2.8.3 IoU - Intersection over Union

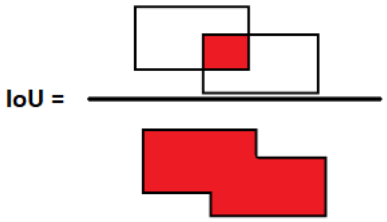
Ved praktisering av objektdetektering og semantisk segmentering er Intersection over Union en mye brukt metode for evaluering av modeller.[20] Metoden går ut på å beregne i hvilken grad predikerte objekter overlapper med sin egen fasit, se figur 2.6. Med denne evalueringsmetoden slipper man det problemet nøyaktighet har hvis klassefordelingen i datasettet er skjev.

2.9 Forhåndstrente nevrale nettverk

Konvolusjonelle nevrale nettverk har vist at de har mulighet til å oppnå klassifisering med høy nøyaktighet. En viktig hake er at modeller basert på CNNs krever store mengder treningsdata for å prestere en akseptabel presisjon. En mulig vei rundt dette er å bruke for-

	P' (Predicted)	N' (Predicted)
P (Actual)	True Positive	False Negative
N (Actual)	False Positive	True Negative

Figur 2.5: Forvirringsmatrise, hentet fra [5]



Figur 2.6: Intersection over Union visualisert

håndstrente nevrale nettverk. Dette er nettverk som har blitt trent over lang tid på veldig store datasett med millioner av bilder og tusenvis av klasser. Et eksempel på et slikt datasett er ImageNet, som består av 1.3 millioner bilder fordelt på 1000 klasser. [21]

Tanken er at nettet skal bruke det det har lært av forhåndstreningen når det trenes på nytt, slik at det har et utgangspunkt å starte fra. Hvis datasettet det forhåndstrenes med er stort nok og dekker mange nok klasser vil mye av det nettet lærer være så generelt at det kan brukes i en hvilken som helst annen sammenheng.

2.10 Styrt læring (eng: Supervised learning)

Styrt læring er en utbredt gren innen maskinlæring som innebærer at en modell lærer ved å kartlegge sammenhengen mellom et sett inngangsverdier og kjente utgangsverdier. Det som læres under denne prosessen kan senere brukes til å finne ukjente verdier.[22] Dette brukes mye i kunstig syn. Hvis man lager et datasett med bilder og definerer klassen til hvert bilde (katt, hund, etc.) kan en modell lære sammenhengen mellom bilde og klasse og bruke dette

til å definere klassen til ukjente bilder.

2.11 Dataøking (eng. Data augmentation)

Dataøking er en vidt brukt metode for å øke treningsgrunnlaget for dype nevrale nett. Metoden går ut på å generere ny, kunstig data med utgangspunkt i den informasjonen man allerede har. [23] Dette gjøres ved å kopiere en forekomst fra datasettet, gjennomføre begrensede endringer på denne forekomsten, og vise den til modellen som treningsdata. Dette er ikke en erstatning for helt nye annoterte data, men det er et nyttig verktøy dersom man ikke har tilgang til eller ressurser til å lage datasett med tilstrekkelig størrelse.

I sammenhenger der den tilgjengelige datatypen består av bilder finnes det en rekke ulike måter å gjennomføre dataøkning på. Noen av de mest grunnleggende er:

- **Flipping** Dette er den enkleste formen for dataøkning, som i sin enkelhet går ut på at man vender bildet om en akse, som oftest horisontalaksen [24]
- **Fargeendring** Fargeendring er en annen enkel form for dataøkning, der man endrer fargekombinasjonene i bilder for å lage realistiske variasjoner fra datasettet man allerede har[24]
- **Rotasjon** Rotering av bilder øker datagrunnlaget, ved at modellen får se det samme objektet orientert i flere retninger.[24]

2.12 Tensorflow

Tensorflow er et open-source programvarebibliotek med verktøy som gjør det mulig for hvem som helst å praktisere maskin- og dyp læring på sin egen maskin. Biblioteket er utviklet av Google, og er skreddersydd for å håndtere data i form av tensorer. Tensorer er data på matriseform som egner seg for å mates til nevrale nett. Tensorflow effektiviserer og korter ned de ofte lange beregningstidene som følger med dyp læring ved å muliggjøre bruk av en eller flere grafikkprosessorer(GPU).

2.13 Keras

Keras er et open-source bibliotek med verktøy for å arbeide med nevrale nett, og bygger på tensorflow. Med Keras-APIen kan man enkelt konstruere egne eller importere ferdiglagde nettverksarkitekturer. Flere av disse ferdiglagde nettene er forhåndstrent på imagenet-datasettet. Med alle disse egenskapene er keras et essensielt verktøy for alle som jobber med kunstig intelligens.

2.14 Segmentation models

Segmentation models APIen gjør det mulig å lage mer avanserte nettverks-arkitekturer med utgangspunkt i forhåndsdefinerte modeller. Dette utgangspunktet kan kalles grunnlag (eng. Backbone), og kan for eksempel være et av de ferdiglagde, ferdigtrente nevrale nettene som tilbys av Keras. Nettverksarkitekturene man kan lage per april 2022 er: UNET, Linknet, FPN(Feature Pyramid Network), og PSPnet(Pyramid Feature Scene Parsing Network).

2.15 Google colab

Google Colab er en tjeneste som tilbyr bruk av eksterne maskiner som er passende for å drive trening av nevrale nett. Brukere kan skrive kode i en Jupyter Notebook og kjøre disse på maskinene til Google, som har betydelig mer beregningskraft enn vanlige personlige maskiner. Tjenesten gjør det mulig for hvem som helst å drive trening og testing av enorme nevrale nett, og er et stort bidrag til fagfeltet kunstig intelligens.

Kapittel 3

Metode

3.1 Datasett

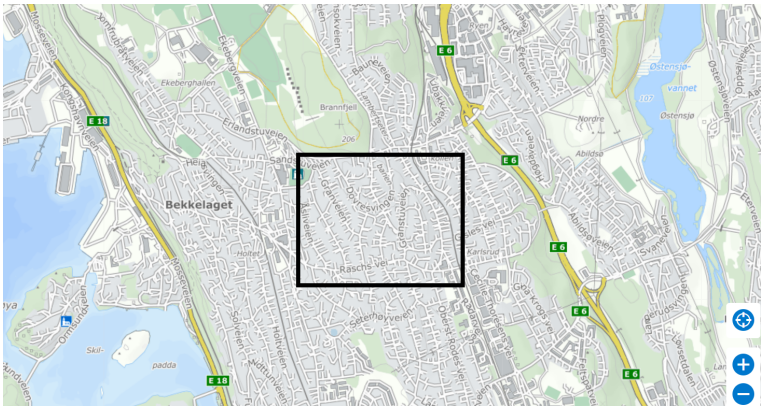
All rådata i form av bilder som er brukt i denne oppgaven er produsert og ferdigstilt av Blom Norway, som er en ledende leverandør innen innsamling, prosessering og modellering av geografisk informasjon.[25] Bildene blir i denne oppgaven brukt til å lage to datasett. Begge datasettene er produsert på samme måte, der den eneste forskjellen er hvilken bildetype man tar utgangspunkt i. Bildetypene er ortofoto og skråfoto. Videre følger en stegvis forklaring av produksjonen av datasettene.

3.1.1 Valg av klasser

Klassene som er valgt for prosjektet er piper, takstiger, og takvinduer. Dette er takobjekter som forekommer ofte, slik at man får et godt læringsgrunnlag for modellen. Det ble også vurdert en siste klasse, solcellepanel, men det ble bestemt at tak med solceller er for sjeldne til å ha som klasse.

3.1.2 Valg av område

Blom produserer flyfoto og skråbilder for store områder i hele Norge. For å få et generelt utvalg i datasettene er det valgt et område som representerer et typisk norsk nabolag. Området består for det meste av eneboliger og rekkehus, og ligger i Ekeberg-Bekkelaget området i Oslo-øst. I figur 3.1 kan man se området markert i et kart. Hvis man har et generelt datagrunnlag har man større sjanse for nøyaktig segmentering av andre områder modellen ikke er trent på. Hadde modellen aldri fått sett hva et vanlig rekkehus ser ut som er det ikke sikkert den ville klart å klassifisere takobjekter om den ble vist et rekkehus.



Figur 3.1: Begge datasettene inneholder bilder fra dette området på Oslo øst.

3.1.3 Bildene

- **Skråfoto:**

  Det er hentet ett skråfoto av området, tatt i retning nord. Oppløsningen på bildet er 14144x10560 piksler før det deles opp i mindre bilder med 512x512 piksler. Dette er for å gjøre bildene håndterbare for annotering og bruk av modell.

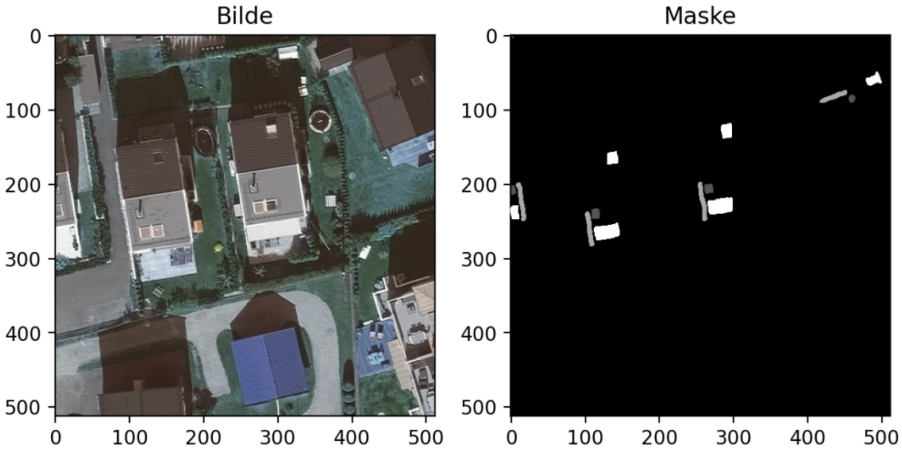
- **Ortofoto:** Det er hentet flere ortofoto som tilsammen dekker det samme området som skråfotoet. Oppløsningen på disse bildene er 10000x7500 piksler. Også disse deles inn i mindre bilder med 512x512 piksler før videre håndtering.

3.1.4 Annotering av bildene

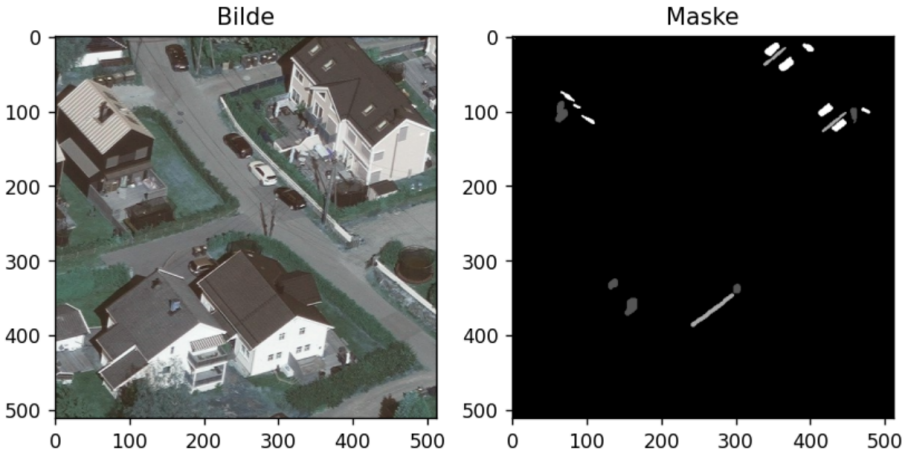
Annotering av bildene er gjort ved bruk av en online platform kalt Apeer, laget av Zeiss. De har en tjeneste der man kan laste opp sine egne bilder, definere egne klasser, og markere hvor i bildene man kan se forekomster av hver klasse. Etter annotering laster man ned en maske der forskjellige pikselverdier representerer hver klasse. En oversikt over klasser og pikselverdier, samt et eksempel på bilde med tilhørende maske vises i hhv. tabell 3.1 og figurer 3.2 og 3.3. Dette er den mest tidkrevende delen av prosessen. Selv etter mange timer med annotering av bilder står man igjen med et relativt lite datasett, når konteksten er kunstig syn.

Klasse	Pikselverdi
Bakgrunn	0
Pipe	1
Takstige	2
Takvindu	3

Tabell 3.1: Klasser og pikselverdier



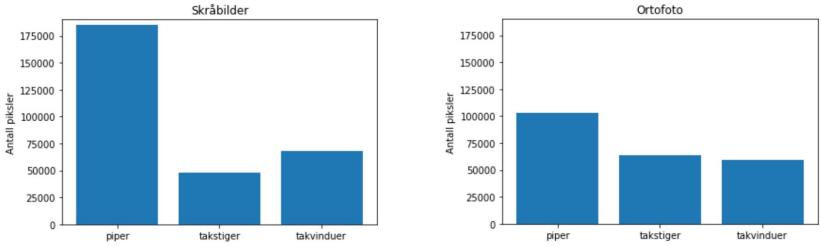
Figur 3.2: Eksempel på ortofoto og tilhørende maske



Figur 3.3: Eksempel på skråbilde og tilhørende maske

3.1.5 Datasett for en balansert sammenlikning.

Hvordan det nevrale nettet presterer når det mates med de to datasettene skal sammenliknes. For at denne sammenlikningen skal bli retterferdig er det bestemt at datasettene skal bestå av et likt antall bilder, og at ressursene som kreves for å produsere datasettene holdes like. Tilgjengelige ressurser er en viktig faktor når man planlegger annotering til prosjekter innen kunstig syn. Hvis man har tilgang til flere datatyper der det viktig å velge den som gir best treningsdata.



(a) Skråbilde-datasett (b) Ortofoto-datasett

Figur 3.4: Fordeling av piksler i hver klasse for hvert datasett.

3.1.6 Gjenstående datasett

Etter stegene over er fullført står vi igjen med to datasett bygget opp på samme måte, der ett er laget med skråfoto og ett med ortofoto. Begge datasettene består av ca. 250 bilder med tilhørende masker.

Fordeling av piksler

Det totale antallet piksler i hvert datasett er tilnærmet helt likt på ca. 65 millioner piksler. Noe det faktisk er forskjell på, er hvordan disse pikslene er fordelt på klasser og bagrunn. Antallet piksler i hver klasse blir påvirket av bildetypen, spesielt pipe-klassen. Figur 3.4 viser hvordan pikslene i hvert datasett fordeler seg på hver klasse, og viser hvordan antallet piksler i pipeklassen er dobbelt så høyt i skråfoto-datasettet, sammenliknet med ortofoto-datasettet. Det er ortofoto-datasettet som har klart flest takstige-piksler, men antallet piksler i takvindu-klassen er veldig likt.

3.1.7 Oppdeling av datasettene

Begge datasettene er delt opp i ett sett for trening og ett sett for validering. For å få et godt grunnlag når nøyaktigheten til modellen skal beregnes, blir 80 prosent av dataene satt av til trening, og de siste 20 prosentene blir satt av til validering. Hvilke par med bilder og masker som havner i hvilken del er tilfeldig, og tabell 3.2 viser fordelingen av bilder og masker i hvert datasett

Denne tilfeldige inndelingen kan påvirke resultatet av valideringen av modellen etter trening. Forskjellige oppdeling av datasettene vil gi forskjellige resultater, så for å motvirke dette gjennomføres treningen 10 ganger slik at resultatene kan snittes ut. Dette vil gi resultater som i større grad representerer den virkelige prestasjonen til modellen.

To modeller blir i tillegg trent opp uten at noe av datasettene settes av til validering. Da blir all tilgjengelig data brukt til trening, som fører til at modellene blir maksimalt effektive når de skal gjennomføre prediksjoner på bilder de aldri har sett før.

Datasett	Trening	Validering
Ortofoto	203	51
Skråfoto	199	50

Tabell 3.2: Fordeling av par med bilder og masker i hvert datasett

3.2 Nettverksarkitektur

Det er gjort en rekke valg angående nettverksarkitektur, av hensyn til den begrensede størrelsen på datasettene. Arkitekturen er basert på U-net strukturen til Ronneberger [1], og det benyttes forhåndstrente vekter for effektivisering av læringsprosessen. Nettet bruker gjenværende læring for å motvirke fenomenet 'vanishing gradient decent' og dermed muliggjøre en større dybde.

3.2.1 ResNet 34

ResNet34 er en predefinert modell importert fra keras sitt modell-bibliotek og brukt som skjellett i et U-net konstruert med Segmentation models-pakken. Modellen er lastet inn med forhåndstrente vekter fra imagenet for å effektivisere læringsprosessen. Navnet er en forkortelse for 'residual net', som betyr at nettet bruker gjenværende læring i form av 'skip-connections' som muliggjør en større dybde og store mengder vekter som skal oppdateres. Antallet trenbare parametere i form av vekter er for ResNet34 24,5 millioner.

3.3 Dataøkning

Det er gjennomført en horisontal flipp-økning som dobler antall bilder og masker ved å 'flippe' hvert bilde med tilhørende maske. På denne måten får man økt mengden data uten å måtte annotere flere bilder. Det er valgt horisontal flipping, ettersom noen annen type flipp eller rotasjon ikke vil gi realistiske orienterte hus i skråbildene. Det er ikke noe poeng for modellen å lære hva objektene på taket av et hus som er rotert opp-ned ser ut, da dette ikke finnes i den virkelige verden.

Det er kun bildene og maskene i trenings-delen av datasettet som blir flippet, for å holde valideringsdelen ukjent for modellen. Hvis splittingen hadde skjedd etter flippingen kunne man risikert å få et par med bilder og masker som treningsdata mens den flippede versjonen av samme bildet kunne blitt valideringsdata. Da ville modellen allerede ha sett et bilde som er identisk med et bilde den skal teste seg på senere. Det ville gitt modellen høyere nøyaktighet enn den egentlig har.

3.4 Hyperparametere

Det er en rekke hyperparametere som bestemmes før læringsprosessen kan settes i gang.

- **Læringsrate**: 0.001

- **Optimaliserer**: Adam
- **Tapsfunksjoner**: Sum av dice- og focal loss
- **Oppdeling**: Trening: 80 prosent - Validering: 20 prosent
- **Aktiveringsfunksjon**: Softmax

3.5 Trening

Det blir trent opp 10 modeller til på hvert datasett, og resultatene av valideringen av disse modellene blir snittet ut. På denne måten sikrer man resultater som viser den virkelige prestasjonen til modellene, upåvirket av datasettenes tilfeldige oppdeling.

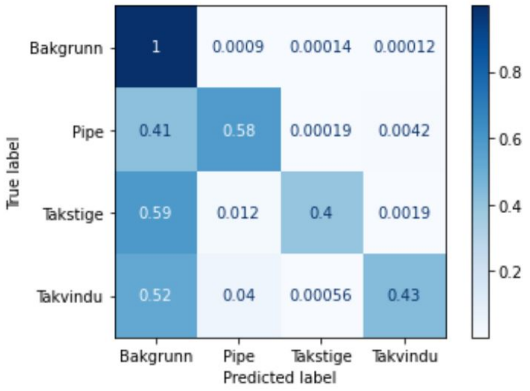
Kapittel 4

Resultat

4.1 Forvirringsmatriser

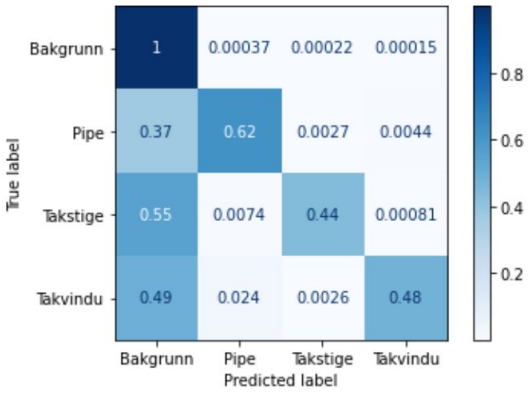
Figurene 4.1 og 4.2 viser snittet av forvirringsmatrisene etter 10 treningsprosesser, normalisert etter de sanne verdiene. Summen av verdiene i hver rad er altså 1. Verdiene som skiller seg ut diagonalt viser andelen piksler i valideringsdatasettet som blir predikert riktig. De andre verdiene viser andelene piksler i valideringsdatasettet som blir klassifisert feil, og hvilken klasse de blir feilaktig tildelt. Vi kan for eksempel se at 58% av pipepikslene i skråfoto-valideringsdatasettet blir predikert riktig, mens 41% blir klassifisert som bakgrunn. De tilsvarende verdiene for ortofoto-valideringsdatasettet er 62 % og 37%, så akkurat her presterer ortofoto-modellen bedre.

4.1.1 Skråfoto



Figur 4.1: Forvirringsmatrise etter trening med skråfoto-datasett

4.1.2 Ortofoto



Figur 4.2: Forvirringsmatrise etter trening med ortofoto-datasett

4.2 IoU for hver klasse

Tabell 4.1 viser gjennomsnittlig IoU etter modellene er trent 10 ganger, samt standardavvik for hver verdi.

Klasser	Datasett			
	Skråfoto		Ortofoto	
	Iou	Std.avvik	IoU	Std.avvik
Piper	0.47	0.07	0.50	0.04
Takstiger	0.34	0.09	0.35	0.06
Takvinduer	0.38	0.11	0.39	0.08

Tabell 4.1: IoU for hver klasse i hvert datasett

4.3 Eksempler

Videre følger et par eksempler på hvordan modellene presterer når de bes detektere takobjekter i ukjente bilder. Å faktisk se på bilder som har blitt segmentert gir et viktig inntrykk av hva modellene får til, og dermed hva de kan brukes til. Under trening før denne siste testen er ingen andel av datasettene holdt tilbake til validering, så modellene får mest mulig data å trene med. Tabell 4.2 viser hvilke farger i bildet som representerer hvilken klasse.

Farge	Klasse
Pipe	Grønn
Takvindu	Rød
Takstige	Blå

Tabell 4.2: Klasser og farger

Figur 4.3: Resultat - Godt eksempel

Figur 4.3 er et godt eksempel på hva skråbilde-modellen kan få til. Alle piper blir markert riktig, og ingen områder i bakgrunnen blir feilklassifisert. Det ene takvinduet i bildet blir markert riktig, og modellen finner pipen og takstigen helt i den øvre kanten av bildet.

Figur 4.4: Resultat - Dårlig eksempel

Figur 4.4 viser et mindre vellykket segmenteringsforsøk fra skråbilde-modellen. De to tydeligste pipene i bildet blir funnet, men ingen av takstigene blir sett. Det blir også feilaktig markert en pipe i øvre høyredel av bildet. Dette viser at modellene sliter dersom objektene ikke stikker tydelig ut i bildet, og at det kan forekomme feilklassifiseringer.

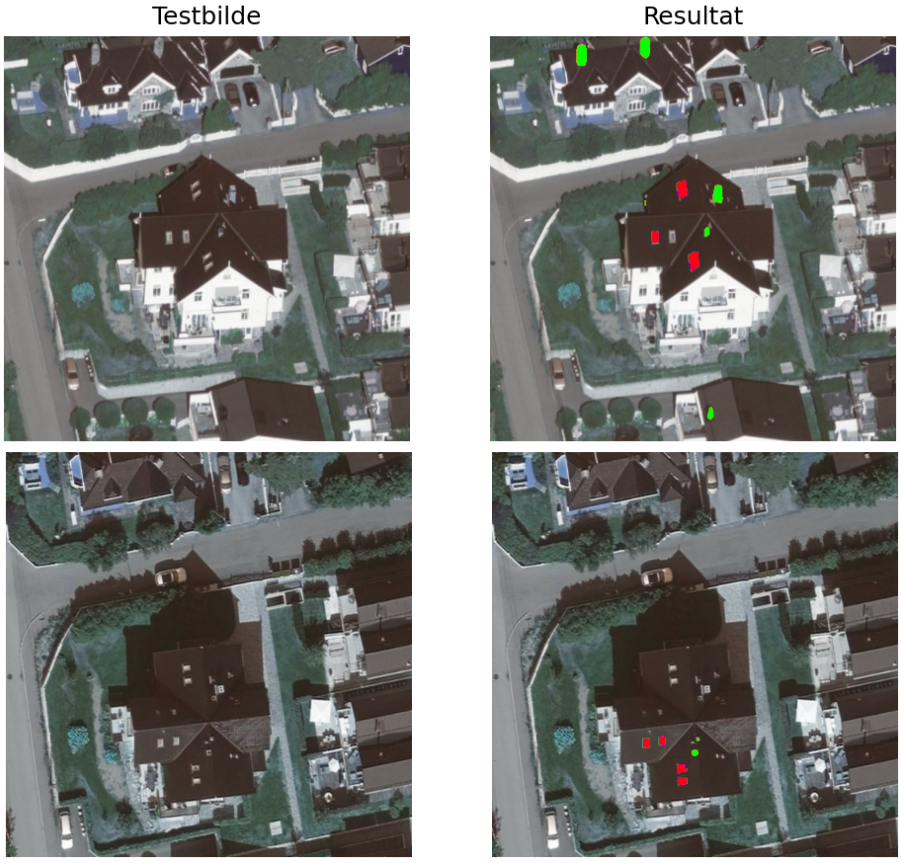
Figur 4.5: Resultat - God segmentering av ortofoto

Figur 4.6: Resultat - Skygger og svarte tak kan være et problem.

Figurene 4.5 og 4.6 viser at også ortofoto-modellen sin segmentering av ukjente bilder kan være noe inkonsekvent. I figur 4.5 ser vi at modellen finner alle objektene i bildet, mens i figur 4.6 finner den ingen av objektene på det svarte taket. Kombinasjonen av at taket er mørkt og at alle objektene ligger i skyggen blir for vanskelig for modellen. Vedlagt i tillegg A ligger en samling predikerte bilder fra begge modellene.

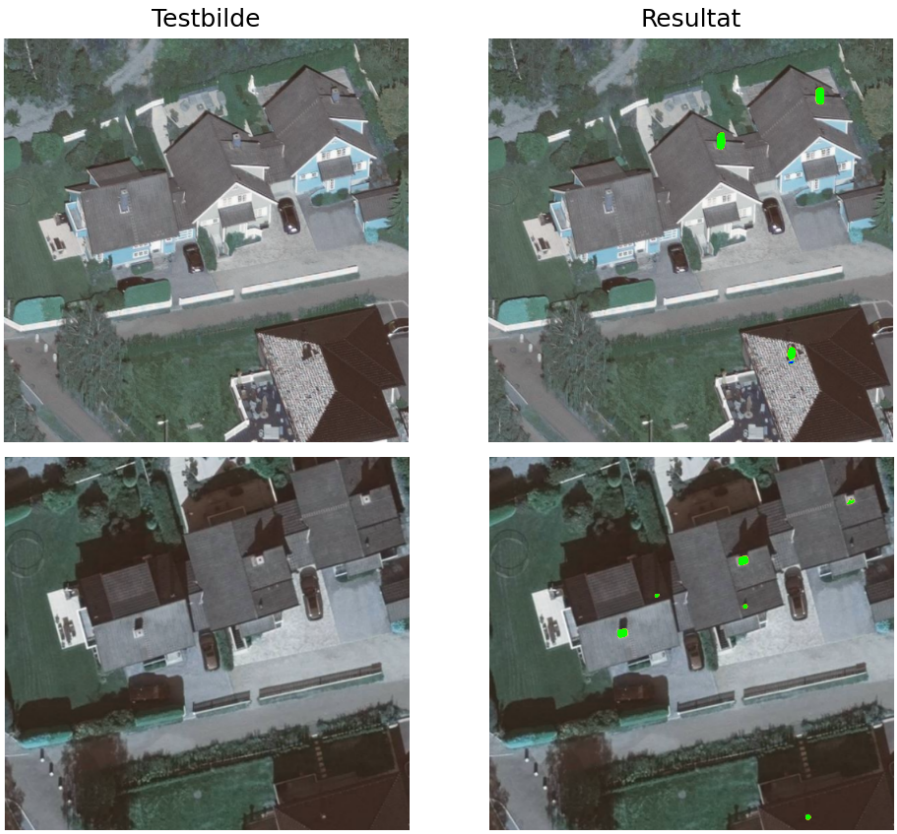
4.3.1 Sammenlikning

Videre følger et par eksempler der de to modellen har blitt testet på bilder av samme bygg, både i skråbilder og ortofoto. Dette er for å vise hvordan modellene presterer når de settes opp mot hverandre. På denne måten får man se mer spesifikke eksempler på hva som er modellenes styrker og svakheter.

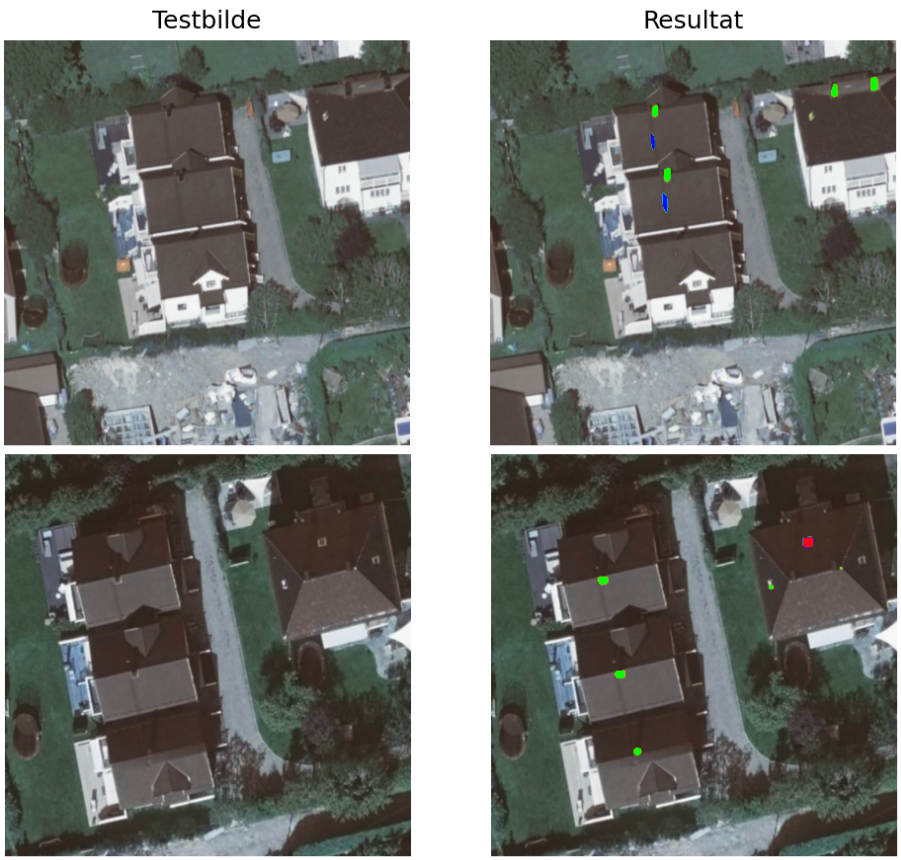


Figur 4.7: Sammenlikning 1

Figur 4.7 , 4.8 og 4.9 illustrerer noen fordeler og ulemper med de to metodene. Fra figur 4.7 kan det virke som at modellen som er trent på skråfoto presterer best, og at ortofoto-modellen kan være noe ustabil. I denne sammenlikningen kommer skråbilde-modellen best ut, men hvis man ser på figur 4.8 er det ortofoto-modellen som gjør det best. Hvordan modellene presterer kan virke noe varierende, avhengig av lysforhold, taktyper, m.m. Det er sannsynlig at mer treningsdata ville lært modellene å takle disse variasjonene bedre.



Figur 4.8: Sammenlikning 2



Figur 4.9: Sammenlikning 3

Figur 4.9 viser at ortofoto har en fordel over skråfoto når det kommer til dekning. I ortofotoet finner modellen takvinduet på huset til høyre, men skråfotomodellen finner ikke det samme vinduet fordi det heller vekk fra kameraposisjonen.

Kapittel 5

Diskusjon

5.1 Diskusjon - Datasettene

5.1.1 Området

All data i datasettene er laget fra bilder av det samme boligområdet i Oslo. Selv om området er valgt med tanke på generalisering av modellen er det ikke sikkert man får nok variasjon til at modellen vil prestere like godt i helt andre områder. Lysforhold og taktyper kan variere og introdusere trender modellen ikke har sett under trening. Prestasjonen kan dermed være noe annerledes om man bruker modellen på et nabolag i for eksempel en annen by eller et annet land. Modellen vil heller ikke fungere i bymiljø eller på industritak, da disse ikke forekommer i denne oppgavens datasett.

5.1.2 Produksjon av datasett

Datasettene i dette prosjektet har blitt produsert manuelt ut ifra bilder. I mange andre liknende prosjekter har det blitt produsert datasett med utgangspunkt i andre tilgjengelig datakilder. I sin masteroppgave fra 2018 produserte Mällberg og Rolfsen[26] treningsdata for arealsegmentering med utgangspunkt i FKB-data. Det samme gjorde Brekke i 2020[27] da hun klassifiserte arealtyper i satellittbilder, også med treningsgrunnlag fra FKB-data.

Manuell produksjon av datasett har både fordeler og ulemper:

I dette prosjektet har det tatt lang tid å produsere treningsdata, og med utgangspunkt i andre datakilder kan man lage mye mer treningsdata på kortere tid. Modeller kan som vist settes opp for å takle problemet med små datasett, men den beste måten å forbedre resultater på vil være å øke datamengden.

En stor fordel med å bygge datasett fra grunnen er at man i større grad kan velge hva slags objekter man vil detektere. Baserer man produksjonen av datasettet på for eksempel FKB er det veldig begrenset hva slags klasser man kan segmentere seg frem til. Følger man metoden i denne oppgaven er man mye mindre bundet når man skal velge klasser. Så lenge man har nok bilder, og kan se objektene man vil detektere i bildene, kan man segmentere

hva man vil enten det er piper og vinduer eller biler og kumlokk.

En annen ulempe er at manuell produksjon introduserer muligheten for feil av operatøren når objektene markeres. Hvis man ikke konsentrer seg er det fort gjort at forekomster blir feilklassifisert eller oversett. Dette vil forvirre modellen og bremse læringsprosessen.

5.1.3 Pikselfordeling

Antallet piksler er i de to datasettene like, men fordelingen er veldig forskjellig. Det første man legger merke til når man sammenlikner diagrammene i figur 3.4 er at skråbilde-datasettet har langt flere pipepiksler enn flyfoto-datasettet. Dette gir mening, da arealet til en pipe i et bilde blir større når man ser den fra en skrå vinkel.

I ortofoto-datasettet er det flere takstige-piksler. Dette tyder på at takstiger er lettere å markere i ortofoto enn skråfoto. En skulle trodd at takvindu-klassen ville hatt samme forskjellen i antall piksler, men dette gjelder ikke. En mulig grunn til dette kan være at det er mer vanlig at takvinduer er rettet mot sør.

Det er uansett flest pipe-piksler i begge datasettene.

5.2 Diskusjon - Nettverket

Det nevrale nettet i denne oppgaven er et U-net med ResNet34 som grunnlag, forhåndstrent med bilder fra imagenet-datasettet. Resnet har blitt brukt effektivt til semantisk segmentering tidligere. Qiu, Yuhang og Chang klarte i 2019 å klassifisere hjerneblødninger i CAT-bilder med høy presisjon ved bruk av ResNet som grunnlag til et U-net.[28]. Resnet34 kom også godt ut i en test gjort av Zhang, Rongyu og Du der de sammenliknet flere nettverk etter trening på det samme datasettet.[29]. Det kunne vært interessant å gjøre den samme testen med datasettene i denne oppgaven for å bekrefte eller avkrefte om ResNet34 var det rette valget. ResNet34 er et utrolig dypt nettverk, med en stor mengde trenbare parametere. Uten tilgang til Google Colab hadde det vært vanskelig å få trent et så stort nett

Tilfeldig inndeling av datasettene i trenings- og valideringsdeler introduserer utfordringer knyttet til validering av effektiviteten til modellene. Dette er løst ved å gjenta trenings- og valideringsprosessen 10 ganger, for så å beregne gjennomsnittet av resultatene. En annen løsning kunne vært å sikre lik pikselfordeling i trenings- og valideringsdelene til hvert datasett.

Det kunne også vært nyttig å få gjennomført et 'grid search' for å systematisk finne de optimale hyperparameterene.

5.3 Diskusjon - Resultatene

5.3.1 Forvirringsmatriser

Begge forvirringsmatrisene viser at modellene har lært mye under treningen. De diagonale verdiene som beskriver riktig klassifisering kommer tydelig frem, og det er veldig lite

feilklassifisering mellom klassene. Feilklassifiseringen som skjer er mellom hver klasse og bakgrunnen.

De to modellene produserer veldig like forvirringsmatriser. I begge tilfellene kommer pipe-klassen best ut, noe som kan forklares delvis ved å se på oversikten over pikselfordelingen i datasettene(3.4). Pipe-klassen i begge datasettene har klart flest piksler, noe som blidrar til at modellene lærer mye om piper under treningsprosessen. En annen grunn til at piper klassifiserer lettest kan være at de stikker tydeligere ut i bildet og lager mønstre som er lette for modellene å kjenne igjen.

Antallet piksler i hver klasse som er riktig klassifisert, altså de tydelige verdiene diagonalt i matrisene, er marginalt høyere for ortofoto-modellen. Denne forskjellen er veldig liten, men kan tyde på at ortofoto egner seg litt bedre til bruken i denne oppgaven.

5.3.2 Intersection over Union - IoU

Tabell 4.1 viser hvordan modellene måles opp mot hverandre når man beregner Intersection over Union for hver klasse. Også her ser vi at piper blir klassifisert lettest, og at modellene sliter noe mer med de andre klassene. Det er veldig lite som skiller de to modellene fra hverandre, men igjen så heller alle klassene så vidt i ortofoto-modellen sin favør.

Alle IoU-beregninger ligger mellom 0.35 og 0.5. Sammenliknet med IuO-beregninger gjort i liknende sammenhenger er dette ganske lavt, og kan ha med at datasettene er mindre enn ønsket. I 2016 oppnådde Audebert [30] IoU-beregninger på over 0.6 da de detekterte biler i flybilder. En grunn til dette er et mer solid datagrunnlag enn hva som ble produsert for denne oppgaven.

5.4 Diskusjon - Eksempler

Ved å se på figurene i delkapittel 4.3 får man et mer spesifikt inntrykk av modellens styrker og svakheter som ikke kommer ordentlig frem i forvirringsmatriser eller IoU-beregninger. Modellene viser at de kan finne takobjekter selv om de ikke stikker tydelig ut i bildet. Samtidig er modellene inkonsekvente, da det er eksempler der fremtredende objekter ikke blir markert. Dette gjør det vanskelig å avgjøre hvilken modell som presterer best ved visuell sammenlikning av predikerte bilder.

5.5 Forslag til videre arbeid

Lite treningsdata er en begrensende faktor i denne oppgaven. I en videre studie hadde det vært interessant å se hvor mye en økning av data hadde hjulpet resultatet. For å bekrefte eller avkrefte at mer treningsdata hadde forbedret resultatet kunne et mulig forsøk vært å holde igjen deler av datasettene for å se om resultatet blir påvirkes som forventet.

Skråbilder presenterer også muligheten for å produsere treningsdata fra samme område i flere vinkler enn sør-nord. En sammenlikning av modeller trent med treningsdata fra skråbilder med forskjellig himmelretning kunne vist om noen himmelretninger tilbyr mer effektivt læringsmateriale enn andre.

Skråbilder muliggjør også effektivisering av produksjonsprosessen til trenigsdataene. Dersom skråbildene av et område er nøyaktig georeferert kan det gå an å markere objekter i alle fire skråbilder av samme området samtidig. Altså ville man kunne markert en pipe i et skråbilde rettet mot nord, og den samme pipen vil bli markert i bilder av samme området i de tre andre himmelretningene. I denne oppgavens tilfelle ville dette resultert i 1000 segmenterte bilder i stedet for 250 for hvert datasett. Det gjenstår å se om georeferering av strukturer i skråbildene kan gjøres nøyaktig nok til at markering i flere bilder på en gang er mulig.

I denne oppgaven er det valgt tre klasser som skal detekteres. Med mer tid og ressursser kunne man undersøkt hvilke muligheter som finnes når det gjelder valg av klasser, og ikke minst antallet klasser. Med det datagrunnlaget som finnes er det få begrensinger for hva som kan detekteres.

Kapittel 6

Konklusjon

I denne oppgaven er det vist et eksempel på hva som kan oppnås når man kombinerer tilgang til store mengder fjernmålinsdata med dyp læring. Oppgavens metode kan lett gjentas og brukes til detektering av nye klasser. Resultatene viser at modellene har lært mye, men at det er rom for forbedringer. Spesielt ved å legge til mer treningsdata i datasettene vil man sannsynligvis se en forbedring i IoU-beregninger, som er relativt svake i denne oppgaven. Forvirringsmatriser og IoU-beregninger for modellene er veldig like, men den lille forskjellen som finnes heller i Ortofoto-modellen sin favør.

Modellene har blitt brukt til å predikere bilder av det samme området, og disse prediksjonene har blitt sammenliknet. Ingen av modellene har markert seg som best i denne testen, men fordelen med perspektivet man får med ortofoto blir tydeligere.

Basert på resultatene i denne oppgaven kan det konkluderes med at skrå- og ortofoto danner et solid grunnlag for generell objektdetektering med dyp læring. Data produsert ut ifra ortofoto har vist seg som mest effektiv, selv om denne fordelen over skråfoto-data er marginal.

Bibliografi

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[3] Umberto Michelucci. *Advanced Applied Deep Learning: Convolutional Neural Networks and Object Detection*. Apress, USA, 1st edition, 2019.

[4] Bruno Artacho and Andreas Savakis. Waterfall atrous spatial pooling architecture for efficient semantic segmentation. *Sensors*, 19(24), 2019.

[5] Josh Patterson and Adam Gibson. *Deep Learning: A Practitioner's Approach*. O'Reilly, Beijing, 2017.

[6] Roger Birkeland. Fjernmåling. `https://snl.no/fjernmåling`. Lest 29/4/2022.

[7] Sentral felles kartdatabase. `https://www.kartverket.no/geodataarbeid/sfkb`.

[8] Lars Mæhlum. Ortofoto. `https://snl.no/ortofoto`. Lest 21/4/2022.

[9] Øystein B. Dick. Skråbilder. `https://snl.no/skråfoto`. lest 5/4/2022.

[10] Sebastian Raschka and Vahid Mirjalili. *Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-Learn, and TensorFlow, 2nd Edition*. Packt Publishing, 2nd edition, 2017.

[11] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *towards data science*, 6(12):310–316, 2017.

[12] Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. Dice loss for data-imbalanced nlp tasks, 2019.

[13] Shruti Jadon. A survey of loss functions for semantic segmentation. oct 2020.

[14] Saad Albawi, Tareq Abed Mohammed, and Saad ALZAWI. Understanding of a convolutional neural network. 08 2017.

[15] Zitao Zeng, Weihao Xie, Yunzhe Zhang, and Yao Lu. Ric-unet: An improved neural network based on unet for nuclei segmentation in histology images. *IEEE Access*, 7:21420–21428, 2019.

[16] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[18] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6:107–116, 04 1998.

[19] Mahmoud Hassaballah and Khalid Hosny. *Recent Advances in Computer Vision: Theories and Applications*. 01 2019.

[20] Floris van Beers, Arvid Lindström, Emmanuel Okafor, and Marco A Wiering. Deep neural networks with intersection over union loss for binary image segmentation. In *ICPRAM*, pages 438–445, 2019.

[21] Dimitrios Marmanis, Mihai Datcu, Thomas Esch, and Uwe Stilla. Deep learning earth observation classification using imagenet pretrained networks. *IEEE Geoscience and Remote Sensing Letters*, 13(1):105–109, 2016.

[22] Padraig Cunningham, Matthieu Cord, and Sarah Delany. Supervised learning. pages 21–49, 01 2008.

[23] Rui Ma, Pin Tao, and Huiyun Tang. Optimizing data augmentation for semantic segmentation on small-scale dataset. page 77–81, 2019.

[24] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, Jul 2019.

[25] Blom norway as. `https://www.blom.no/om-oss/`.

[26] Ruben Schmidt Mällberg and Valdemar Edvard Sandal Rolfsen. Map creation from semantic segmentation of aerial images using deep convolutional neural networks-utilizing publicly available spatial data to make an aerial image labeling dataset. Master's thesis, NTNU, 2018.

[27] Rebecca Seim Brekke. Klassifisering av arealtyper i satellittbilder ved bruk av konvolusjonelle nevrale nettverk. Master's thesis, Norwegian University of Life Sciences, Ås, 2020.

[28] Yuhang Qiu, Chia Chang, Jiun-Lin Yan, Li Ko, and Tian Chang. Semantic segmentation of intracranial hemorrhages in head ct scans. pages 112–115, 10 2019.

[29] Rongyu Zhang, Lixuan Du, Qi Xiao, and Jiaming Liu. Comparison of backbones for semantic segmentation network. *Journal of Physics: Conference Series*, 1544:012196, 05 2020.

[30] Nicolas Audebert, Bertrand Le Saux, and Sébastien Lefèvre. Segment-before-detect: Vehicle detection and classification through semantic segmentation of aerial images. *Remote Sensing*, 9(4), 2017.

Tillegg A

Samling predikerte bilder

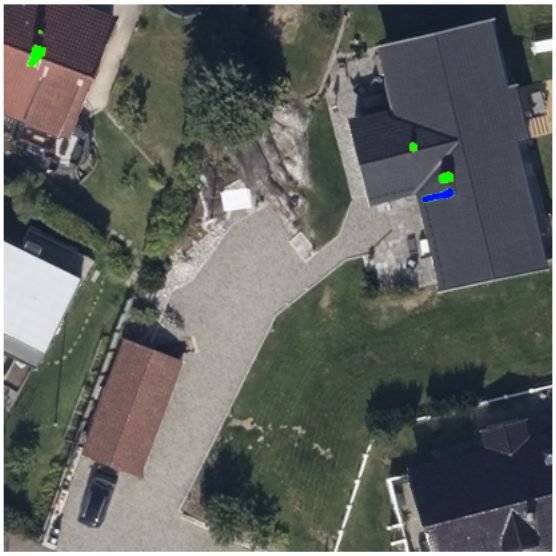


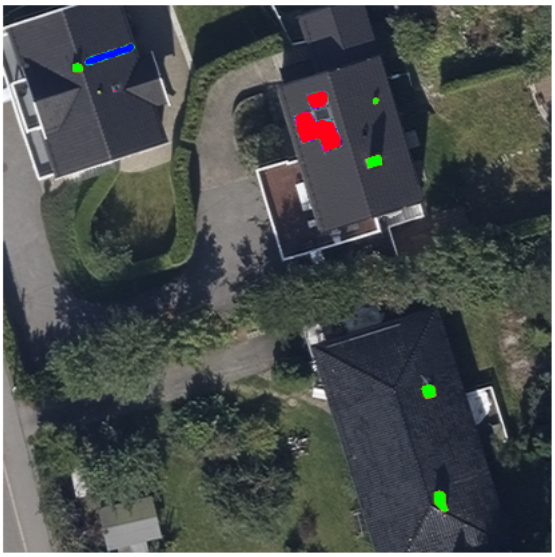


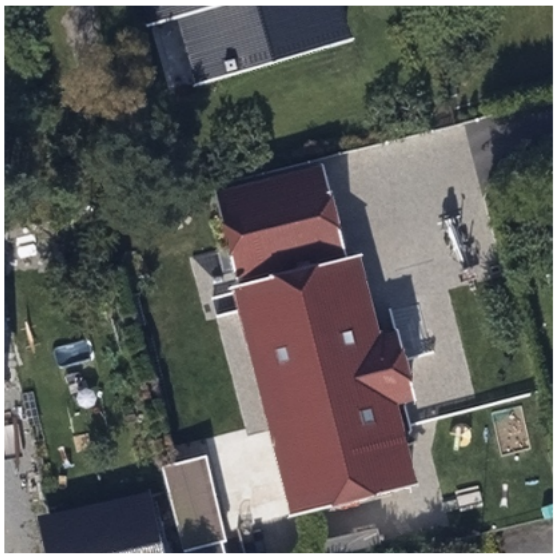
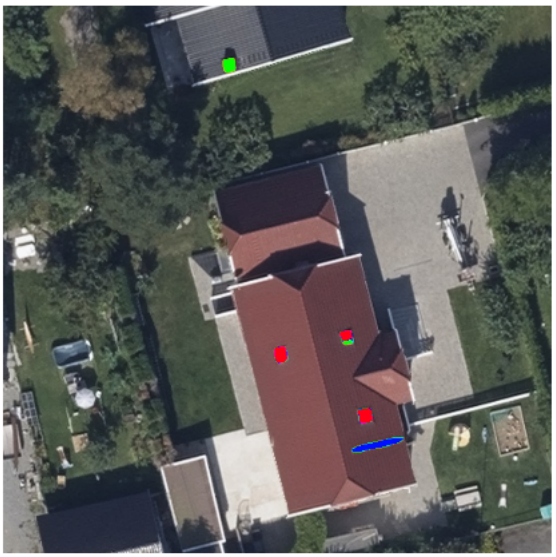




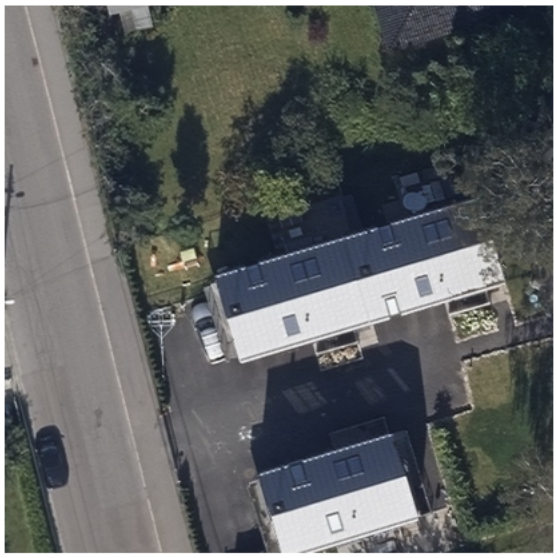


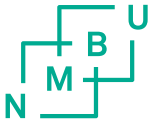








Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway