Norges miljø- og biovitenskapelige universitet

Masteroppgave 2019 30 stp
Fakultet for realfag og teknologi

Endringsdeteksjon i skog med Sentinel-2

Change detection in forested areas with Sentinel-2

Jonas Njøten Sletmo
Geomatikk

Forord

Etter flere måneder med hard arbeid gjennom mange lange dager og netter markerer denne oppgaven slutten på min tid som student. Aller først vil jeg takke min biveileder, Floris Groesz, ved Blom for å stille opp og gi meg gode råd og tilbakemeldinger når det har vært nødvendig under arbeidet. Jeg vil også takke Stian Rostad ved Blom for all den gode hjelpen og rådene han har gitt under arbeidet. Jeg må også rette en takk til min hovedveileder, Ivar Maalen-Johansen, for den gode hjelpen han ga i oppstartfasen av oppgaven. En takk må også rettes til resten av Blom som tok svært godt imot meg, og fikk meg til å føle meg som en de av gjengen.

Til slutt vil jeg rette en stor takk til forelder og søsken som, med stor tålmodighet har støttet meg gjennom dette arbeidet.

Sammendrag

Sentinel-2 har vært operativt siden 2015 og har i perioden etter det blitt benyttet til jordovervåkning, klimaovervåkning og nødadministrasjonstjenester. Sentinel-2 med sin høye romlige og temporale oppløsning er et kraftfullt verktøy som har vært benyttet til å finne endringer på jordoverflaten over tid.

Denne oppgaven fokuserer på hvordan man kan bruke informasjon fra Sentinel-2 på best mulig måte for å finne endringer i skog (hogster). Jeg har benyttet to Sentinel-2 bilder, det første fra 2017 og det andre fra 2018. Disse bildene har jeg hentet fra Copernicus Open Access Hub. I oppgaven benyttes en nevral nettverksmodell til å finne de nevnte endringer. Det er benyttet tre datasett som er satt opp forskjellige måter. Det ene inneholder data fra 2018 og 2017. Det neste inneholder data fra 2018 og differansene mellom 2018 og 2017. Det tredje kun differanser mellom 2018 og 2017. For de tre datasettene har jeg sett på forskjellige kombinasjoner av spektralbånd.

Resultatene av mine tester viser at både valg av spektralbånd og oppsett av datasett har en innvirkning på hvor gode resultatene blir. Mengden av treningsdata har stor innvirkning på hvor godt en modell klarer å predikere resultatet. Resultatene viser også at et godt datagrunnlag er viktig for å oppnå et gode resultater. De beste resultatene fremkommer ved bruk av alle tilgjengelige 10 og 20 meters bånd, i tillegg gir testen med 10 meters bånd bedre resultater enn testen med 20 meters bånd.

Abstract

Sentinel-2 has been operational since 2015 and has since then been used for land monitoring, climate monitoring and emergency management. Sentinel-2 with its high spatial and temporal resolutions is a powerful tool that has been used to detect changes on the earth's surface over time.

This thesis focuses on how to use information from Sentinel-2 in the best possible way to find changes in forest (logging). I have used two Sentinel-2 images, the first was taken in 2017 and the second in 2018. I have obtained these images from the Copernicus Open Access Hub. I have uses a neural network model to find the aforementioned changes. I have used three different datasets that have been set up in different ways. One contains data from 2018 and 2017. The next contains data from 2018 and the differences between 2018 and 2017. The third only difference between 2018 and 2017. For the three datasets I have looked at different combinations of spectral bands.

The results of my tests show that both the choice of spectral bands and the set-up of data sets have an impact on how good the results become. The amount of training data has a great impact on how well a model manages to predict the result. The results also show that a good data base is important for achieving good results. The best results are obtained with the use of all available 10- and 20-meter bands, in addition, the test with 10 meters of band gives better results than the test with 20 meters band.

Innholdsfortegnelse

Figurer

Formler

Tabeller

1. Innledning

1.1 Bakgrunn

Norge er et land med mye skog. I følge NIBIO er så mye som 38 prosent av Norges landareal dekket av skog (NIBIO, u.å). Med så mye skog er det klart at skogen er en viktig arbeidsplass og eksportnæring for Norge. I 2017 var det ifølge SSB 17402 personer med positiv næringsinntekt på skogbruk (SSB, 2019). Det er ikke kun økonomiske interesser i skogen, men også viktig infrastruktur som kraftlinjer. I følge NVE er trær som faller på kraftledninger en viktig grunn til strømbrudd i Norge (NVE, 2016). Med alt dette i tankene er det viktig å overvåke hvilke endringer som skjer i skogen.

Sentinel-2 er et jordobservasjon satellittoppdrag fra Copernicus programmet til EU. Med sin høye temporale oppløsning, og multispektrale instrument med 13 kanaler i de synlige, nær infrarøde og kortbølge infrarøde delene av det elektromagnetiske spekteret, kan Sentinel-2 være et kraftig hjelpemiddel for å finne endringer i skog.
Derfor skal jeg med denne oppgaven besvare problemstillingen «Hvordan kan Sentinel-2 benyttes til å detektere endringer i skog?».

1.2 Oppgavens mål og hensikt

Målet med oppgaven er å undersøke hvordan Sentinel-2 kan brukes til å finne endringer i skog. I denne oppgaven er endringene jeg har sett på snauhogst. Detekteringen vil bli gjort med en nevral nettverksmodell laget med Keras i Python. For å svare på hvordan Sentinel-2 kan brukes til å finne endringer skal jeg se på forskjellige måter å presentere dataene mine på, samt forskjellige kombinasjoner av Sentinel-2 sine spektralbånd. Etter å ha sett på forskjellige datakombinasjoner skal jeg sammenligne resultatene for å komme frem til hvilke kombinasjoner av bånd og datasett som gir de beste resultatene.

1.3 Tidligere forskning

Sentinel-2 ble operativt i 2015 og det har derfor vært vanskelig å finne relevant forskning knyttet til endringsanalyse med Sentinel-2. Jeg har imidlertid funnet relevante artikler som omhandler endringsanalyse med satellitt, og artikler knyttet til klassifisering med sentinel-2.

Et forsøk på automatisk klassifisering av hogster og skogsskader med bilder fra Landsat 7 og 8 fra 2014 gir resultater for automatisk klassifisering med en total nøyaktighet på 82,68% og en kappa koeffisient på 0,20 ved bruk av en *Normalized Difference Infrared Index* (NDII). Og en total nøyaktighet på 88,19% og en kappa koeffisient på 0.24 ved bruk av en *Specific Leaf Area Vegetation Index* (SLAVI). I dette forsøket er det brukt to testområder. Et i Akershus hvor bildene ble tatt før og etter stormen Dagmar, og et i Midt-Norge hvor bildene ble tatt før og etter stormene Hilde og Ivar. Bildene som ble benyttet for testsett 1 ble tatt 12.10.2011 og 28.09.2012, bildene for testsett 2 ble tatt 05.09.2013 og 17.09.2014. For klassifisering av hogster oppnådde dette forsøket en presisjon på 62% og en recall på 5% (Solberg et al., 2014).

En doktoravhandling fra 2017 så på sammenligning og kombinasjon av Sentinel-2 og Landsat data for å finne mindre endringer i skog. Det ble sett på kombinasjoner av NDVI indekser for Sentinel-2 og Landsat data med 30 meters oppløsning, Landsat data med 30 meters

oppløsning, Sentinel-2 data med 30 meters oppløsning og Sentinel-2 data med 10 meters oppløsning. Avhandlingen fant at Sentinel-2 med sin 10 meters oppløsning utgjorde en betydelig forskjell når det kom til å kartlegge mindre endringer i skog. Den viste også at det er bedre med kombinasjoner av 30 meters data fra Landsat og Sentinel-2 enn med kun 30 meters data fra Landsat (Hamunyela, 2017).

1.4 Oppgavens oppbygning

Etter denne innledningen følger oppgavens resterende deler. I teoridelen skal jeg forklare hva Sentienl-2 er, og hvordan kunstige nevrale nettverk fungerer. Det er også en kort forklaring av de statistiske målene som er brukt i oppgaven. Etter dette følger en oversikt over programvare og filformater som har vært aktuelle for å løse denne oppgaven. Videre følger oppgaven framgangsmåte og metode, resultater og diskusjonsdel. I fremgangsmåte og metode delen vil trinnene i arbeidsgangen bli presentert, før resultatene blir presentert og diskutert i henholdsvis resultat og diskusjonsdelen av oppgaven. I diskusjonsdelen vil det også bli presentert forslag til videre arbeid. Til slutt kommer en konklusjon.

2. Teori

2.1 Seninel-2

Sentinel-2 er et europeisk multispektralt bilde oppdrag, som er en del av Copernicus programmet til ESA. Oppdraget består av to satellitter, S2A og S2B, som ble skutt opp i henholdsvis 2015 og 2017. De to satellittene følger samme omløpsbane med en faseforskyvning på 180°. Dette gir en høy temporal oppløsning med et besøk hver femte dag ved ekvator. Hver av satellittene er utstyrt med et multispektralt instrument (ESA, 2015).

Figur 2.1: Modell av sentinel-2 satellitt. Tilgjengelig fra: https://en.wikipedia.org/wiki/Sentinel-2#/media/File:Sentinel_2-IMG_5873-white_(crop).jpg. Creative commons lisens https://creativecommons.org/licenses/by-sa/2.0/fr/legalcode (lest 18.04.2019).

2.1.1 Instrument

Sentinel-2 sitt multispektrale instrument (MSI) er en push broom skanner med en brennbredde på 290 km. Instrumentet tar opp 13 spektralbånd, 4 med 10 meter, 6 med 20 meter og 3 med 60 meter spektral oppløsning. De fire båndene med 10 meters oppløsning er henholdsvis 3 bånd i det synlige spektret, rød, grønn og blå, og et nærinfrarødt bånd (NIR). De seks båndene med 20 meters oppløsning er fire *vegetation red edge* bånd og to kortbølge IR bånd (SWIR). De tre båndene med 60 meters oppløsning er atmosfæriske korreksjonsbånd (ESA, 2015).

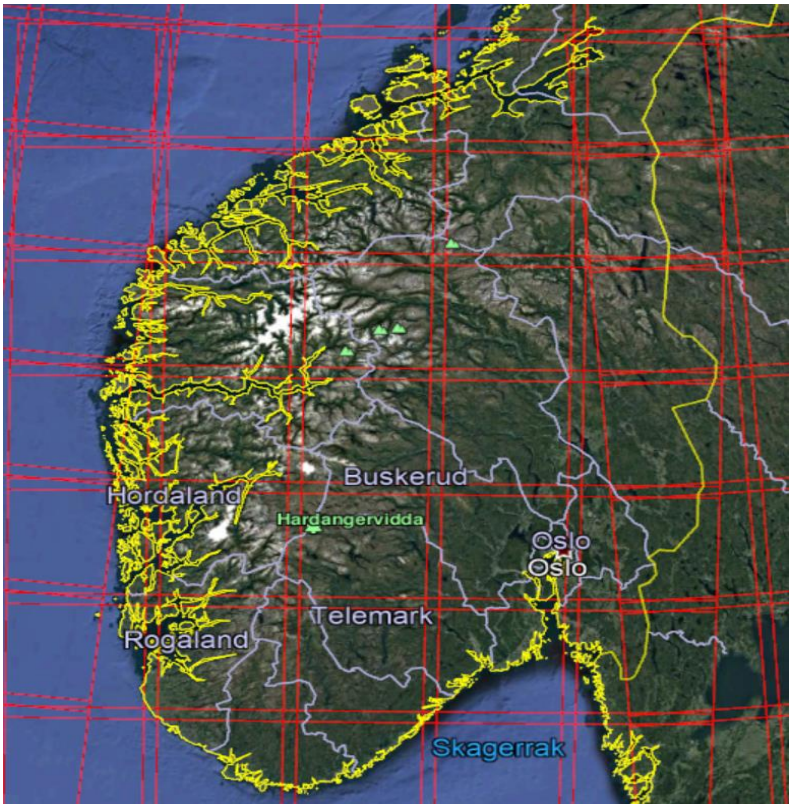
Bånd	Sentral bølgelengde S2A (nm)	Sentral bølgelengde S2B (nm)	Oppløsning (m)
B01	442,7	442,2	60
B02	492,4	492,1	10
B03	559,8	559,0	10
B04	664,6	664,9	10
B05	704,1	703,8	20
B06	740,5	739,1	20
B07	782,8	779,7	20
B08	832,8	832,9	10
B8A	864,7	864,0	20
B09	945,1	943,2	60
B10	1373,5	1376,9	60
B11	1613,7	1610,4	20
B12	2202,4	2185,7	20

Tabell 2.1: Sentral bølgelengde og oppløsning for båndene til Sentinel-2A og Sentinel-2B (ESA, u.å-c).

Sentinel-2 sine multispektrale sensorer er passive og virker ved å samle det reflekterte sollyset fra jordoverflaten. Det reflekterte sollyset blir splittet av et filter og fokusert inn på to forskjellige brennpunktsplan, ett for synlige og nær infrarøde bånd og ett for kortbølge infrarøde bånd. Den spektrale oppdelingen av bånd inn i individuelle bølgelengder blir utført av stripefiltre som er plassert foran detektorene(ESA, u.å-b).

2.1.2 Produkter

Det finnes to sentinel-2 produkter som er tilgjengelig for forbrukere; 1C og 2A. Disse produktene kommer i soner på 100 x 100 km kalt tiles. De er ortometriske bilder i UTM/WGS84 projeksjon (ESA, 2015).



Figur 2.2: Sentinel-2 soner over Sør-Norge i Google Earth

Produktene er tilgjengelig for nedlastning gratis fra https://scihub.copernicus.eu/ (ESA, u.å-a). Bildene kan inneholde kun delvis datadekning, dette forekommer hvis det aktuelle bildet befinner seg i kanten av en opptaksstripe.
1C er et Top Of Atmosphere (TOA) reflektans produkt. Hvert 100x100km bilde inneholder ca. 600 MB med data. Produktet blir systematisk generert og er tilgjengelig for nedlastning fra ESA. 2A er et Bottom Of Atmosphere (BOA) reflektans produkt. Størrelsen er ca. 800 MB per bilde. Produktet blir systematisk generert og kan lastes ned fra ESA, eller så kan det bli generert av brukere ved hjelp av ESA sin Sentinel-2 Toolbox. Ved bruker generering brukes det korresponderende 1C produktet som input (ESA, 2015).

2.2 Endringsanalyse

Endringsanalyse er i GIS definert som en prosess der man måler forskjeller i et områdes attributter over en tidsperiode. Dette er ofte gjort ved hjelp av å studere to eller flere satellitt- eller flybilder som er tatt opp ved forskjellige tidspunkt (Khandelwal et al., 2013).

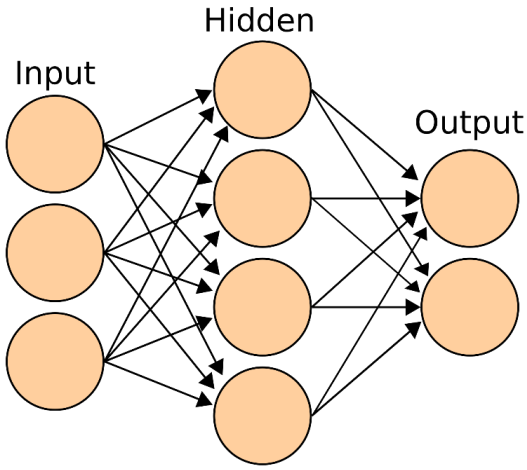
2.3 Nevrale nettverk

Kunstige nevrale nettverk er algoritmer som til en viss grad bygger på den menneskelige hjerne. De er designet for å gjenkjenne mønstre. Hovedoppgaven til et nevralt nettverk er å tolke data ved å enten klassifisere eller koble sammen lignende data. De kan brukes til både ikke-styrte og styrte klassifiseringer (Skymind, u.å).

Ved en ikke-styrt klassifisering kan de koble ikke klassifisert data sammen ved å se på likheter ved dataene. Noen eksempler på hva ikke-styrt klassifisering kan brukes til er søking etter lignende objekter, som å bruke et bilde til å finne andre bilder med lignende egenskaper. Ikke-styrt klassifisering kan også brukes til det motsatte, altså å finne avvik (Skymind, u.å).

Ved en styrt klassifisering kan uklassifisert data bli klassifisert ved hjelp av at man har et treningsdatasett med klassifiserte data til å trene nettverket med. Ved en styrt klassifisering er det derfor nødvendig at et menneske overfører sine kunnskaper til nettverket ved å konstruere et treningsdatasett. En styrt klassifisering kan brukes til blant annet å se etter spesifikke objekter i bilder, gjenkjenne ansikter eller ansiktsuttrykk i bilder, gjenkjenne stemmer eller lage transkripsjoner av samtaler og å bestemme om tekst er for eksempel spam i eposter (Skymind, u.å).

Et nevralt nettverk består av flere lag. Det starter med et input lag, her ligger de uklassifiserte dataene som ønskes klassifisert av modellen. Denne dataen passerer gjennom et eller flere skjulte lag før det kommer ut i andre enden som klassifisert data i output-laget.

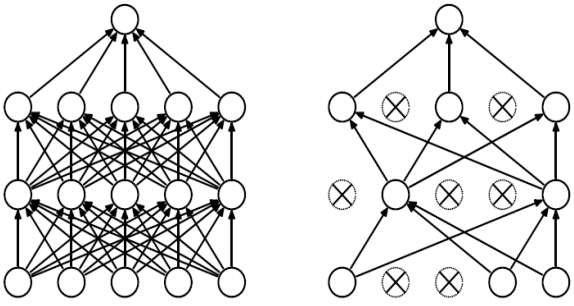


Figur 2.3: Modell av enkelt nevralt nettverk. Tilgjengelig fra: https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg. Creative commons lisens https://creativecommons.org/licenses/by-sa/3.0/legalcode (lest 06.05.2019)

Hvis modellen inneholder mer enn et skjult lag har man et dypt nevralt nettverk (Skymind, u.å). Hvert lag består av kunstige neuroner. Disse neuronene er matematiske funksjoner som tar en input, prosesserer, vekter og kjører den gjennom et aktiveringsfilter. Neuronet returnerer en aktivert output (Lagandula, 2018). Vektingen gjort av neuronet vil bestemme til hvilken grad outputen fra dette neuronet vil påvirke den endelige klassifiseringen av en input.

For dype nettverk med mer en ett skjult lag, vil hvert lag trene på et sett med egenskaper basert på de tidligere lagene. Det vil si at for hvert lag blir egenskapene neuronene kan gjenkjenne mer komplekse, siden de legger sammen egenskapene fra de tidligere lagene (Skymind, u.å).

Ved et standard dypt nevralt nettverk vil alle noder i alle lag kobles opp mot hverandre. Dette gjør at man kan få svært ekspressive modeller som kan lære seg svært komplekse forhold mellom input og output. I sett med lite treningsdata kan mange av disse komplekse forholdene skyldes støy i modellen (Srivastava et al., 2014). Dette kan føre til overtilpasning og burde derfor unngås. Overtilpasning oppstår når modellen er så tett tilpasset til treningsdataen at det er vanskelig å bruke modellen til å predikere for nye data. Det finnes flere måter å unngå dette på, den metoden som er brukt i denne oppgaven er dropout. Dropout ignorerer neuroner fra både synlige og skjulte lag i modellen, samt deres innkommende og utgående forbindelser (Srivastava et al., 2014). Det er tilfeldig hvilke lag som blir fjernet fra modellen. Den uttynnede modellen har mindre sjanse for å utvikle avhengighet mellom neuroner, og på den måten minsker det sjansen for overtilpasning (Srivastava et al., 2014).



Figur 2.4: Eksempel på et nevralt nettverk med og uten dropout (Srivastava et al., 2014).

2.4 Python

Python er et programmeringsspråk skapt av Guido van Rossum (van Rossum, u.å). Python ble gitt ut for første gang i 1991, og har siden vokst til et av de mest brukte programmeringsspråkene i verden (TIOBE, 2019; van Rossum, 2009). Python har åpen kildekode og er utviklet under en OSI godkjent open-source lisens. Lisensen administreres av The Python Software Foundation (PSF, u.å).

Python Software Foundation er en ikke profittbasert organisasjon som hoder de intellektuelle rettighetene til python, samt vedlikeholder python.org, The Python Package Index (PyPI) og Pythons dokumentasjon (PSF, u.å).

2.5 Statistiske mål

Når resultatene i denne oppgaven skal presenteres vil det bli presentert noen statistiske mål. En fin måte å presentere resultatene, og lage et godt grunnlag for de statistiske målene er en forvirringsmatrise. I en forvirringsmatrise blir predikerte verdier plottet mot faktiske verdier. Verdier som har blitt riktig klassifisert ligger i diagonalen, mens de feilklassifiserte ligger utenfor diagonalen.

	Predikert A	Predikert B
Faktisk A	Sann A	Falsk B
Faktisk B	Falsk A	Sann B

Figur2.5: Eksempel på en forvirringsmatrise

2.5.1 Total nøyaktighet

Total nøyaktighet er et mål på hvor god modellen har vært til å klassifisere riktig. Total nøyaktighet regnes ut ved å dele de riktig klassifiserte elementene med totalt antall elementer. I forvirringsmatrisen betyr det at du tar summen av elementene langs diagonalen og deler på summen av alle elementene i matrisen.

$$Total\ nøyaktighet\ = \frac{Riktig\ klassifisere\ elementer}{totalt\ antall\ elementer}$$

Formel 2.1: Formel for total nøyaktighet

2.5.2 Kappa koeffisient

Kappa koeffisient er et mål på hvor god en modell er til å predikere. Kappa vil bli et tall mellom 0 og 1. En kappa på 1 indikerer at modellen har en perfekt riktighet, mens en kappa på 0 indikerer at riktigheten kun er basert på tilfeldighet. For å regne ut kappa koeffisienten brukes formelen:

$$K = \frac{N\sum_{i=1}^{r} x_{ii} - \sum_{i=1}^{r}(x_{i+} * x_{+1})}{N^2 - \sum_{i=1}^{r}(x_{i+} * x_{+1})}$$

Formel 2.2: Formel for utregning av kappa koeffisient

Der N er det totale antall verdier i datasettet, $\sum_{i=1}^{r} x_{ii}$ er summen av verdiene i diagonalen av forvirringsmatrise og $\sum_{i=1}^{r}(x_{i+} * x_{+1})$ er summen av de faktiske verdiene i hver klasse multiplisert med de predikerte verdiene i hver klasse (Viera & Garrett, 2005).

For å tolke kappakoeffisienten brukes ofte en skala (Viera & Garrett, 2005):

Grad av riktighet	Svært dårlig	Dårlig	Svak	Moderat	Sterk	Nesten prefekt
Kappa	0,00	0,20	0,40	0,60	0,80	1

Tabell 2.2: Tabell for tolkning av kappa koeffisienter

2.5.3 Presisjon

Presisjon eller produsentnøyaktighet er et mål hvor mange av resultatene dine innenfor en klasse som er sanne positive. For forvirringsmatrisen betyr det at man tar for eksempel sann A delt på sann A + falsk A (Powers, 2007).

$$Presisjon = \frac{Sann\ A}{Sann\ A + Falsk\ A}$$

Formel 2.3: Formel for presisjon

2.5.4 Recall

Recall eller brukernøyaktighet er et mål på hvor mange instanser av en klasse som ble riktig klassifisert. For forvirringsmatrisen betyr det at man tar for eksempel sann A delt på sann A + falsk B (Powers, 2007).

$$Recall = \frac{Sann\ A}{Sann\ A + Falsk\ B}$$

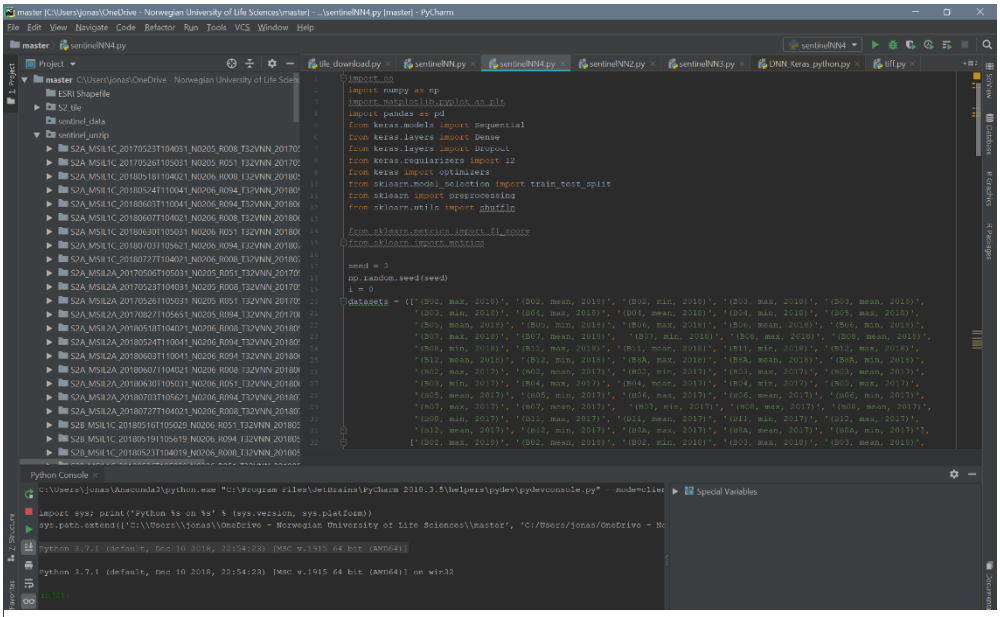
Formel 2.4: Formel for recall

3. Programvare og filformater

For å løse denne oppgaven har det vært nødvendig å bruke mange forskjellige programvarer og filformater. Alle de programvarene som er brukt i denne oppgaven har vært gratis tilgjengelige for meg, og kun en har krevd at jeg benytter meg av en studentlisens for å få gratis tilgang. I dette kapittelet skal jeg gi en kort oversikt over de dataprogrammene og filformatene som har blitt brukt til å løse denne oppgaven.

3.1 PyCharm

PyCharm er et integrert utviklingsmiljø (IDE) utviklet av det tsjekkiske selskapet JetBrains. PyCharm ble først gitt ut i februar 2010, den første fulle versjonen ble sluppet i oktober samme år (Jetbrains, u.å-a). PyCharm er tilgjengelig på operativsystemene Windows, MacOS og Linux. Det er to tilgjengelige versjoner av PyChram. Den første er PyChram Professional. Denne versjonen er lisensiert, og krever at man tegner et abonnement for å få tilgang. Det finnes flere forskjellige abonnementer og priser, og det er i tillegg en mulighet for studenter til å få et gratis abonnement på produktet (Jetbrains, u.å-b; Jetbrains, u.å-c). Den neste versjonen er PyCharm Community. Denne versjonen krever ingen abonnementer og er tilgjengelig gratis under en Apache 2 lisens (Jetbrains, u.å-c; Jetbrains, u.å-d). I denne oppgaven har jeg brukt PyCharm Professional for å laste ned sentinel-2 data med programtillegget Sentinelsat, generere 2A-produkter ved hjelp av sen2cor og til å lage nevrale nettverksmodeller med Keras.



Figur 3.1: Skjermdump av PyCharm

3.2 Anaconda

Anaconda er et åpent program for å håndtere pakker og arbeidsmiljøer i Python. Anaconda kommer med mer enn 200 forhåndsinstallerte pythonmoduler, og har i tillegg et bibliotek på mer enn 2000 pakker som kan lastes ned. Anaconda kan brukes både gjennom et grafisk brukergrensesnitt (GUI) som heter Anaconda Navigator eller på kommandolinje med Anaconda Prompt (Anaconda, u.å). I denne oppgaven har jeg brukt Anaconda til å installere de Pythonmodulene som jeg har trengt for å løse oppgaven

3.3 Pythonmoduler

Her følger en kort forklaring av de mest aktuelle pythonmodulene som har blitt brukt til å løse denne oppgaven.

3.3.1 Sentinelsat

Sentinelsat er en python-modul som er laget for å søke etter, laste ned og hente ut Sentinel-data fra Copernicus Open Access Hub. Modulen kan kjøres både på kommandolinje, eller med en API. Modulen har mulighet til å søke etter og laste ned enkeltprodukter, eller flere produkter om gangen. Man kan søke etter produkter både ved å avgrense et søkeområde, eller søke etter Sentinel tiles (seninelsat, u.å). I denne oppgaven er Sentinelsat brukt til å laste ned Sentinel-produkter i PyCharm.

3.3.2 Rasterstats

Rasterstats er en python-modul for å summere romlige raster-datasett basert på vektorgeometri. Modulen kan regne ut statistikk både på soner og på enkeltpunkt (Perry, 2015). Modulen er i denne oppgaven brukt til å regne ut maks, minimum og gjennomsnittlige pikselverdier i trenings og valideringsdata polygonene.

3.3.3 Keras

Keras er en nevral nettverks API for python som kan kjøres over Tensorflow, CNTK og Theano. Keras støtter både konvensjonelle og tilbakevendende nevrale nettverk og kan kjøres på både CPU og GPU (Keras, u.å). I denne oppgaven er Keras brukt til å lage nevrale nettverksmodeller med Tensorflow i PyCharm.

3.3.4 Tensorflow

Tensorflow er en åpen kildekodeplattform for maskinlæring. Tensorflow kan kjøres på flere plattformer og kan brukes opp mot mange APIer, som Keras (Tensorflow, u.å).

3.3.5 GDAL

GDAL er et oversetterbibliotek for geospatiale data formater. GDAL er et åpent program og er utviklet av Open Source Geospatial Foundation under en åpen kildekodelisens (GDAL, u.å). I denne oppgaven er GDAL brukt til å lese inn raster- og vektordata i PyCharm.

3.3.6 Pandas og Geopandas

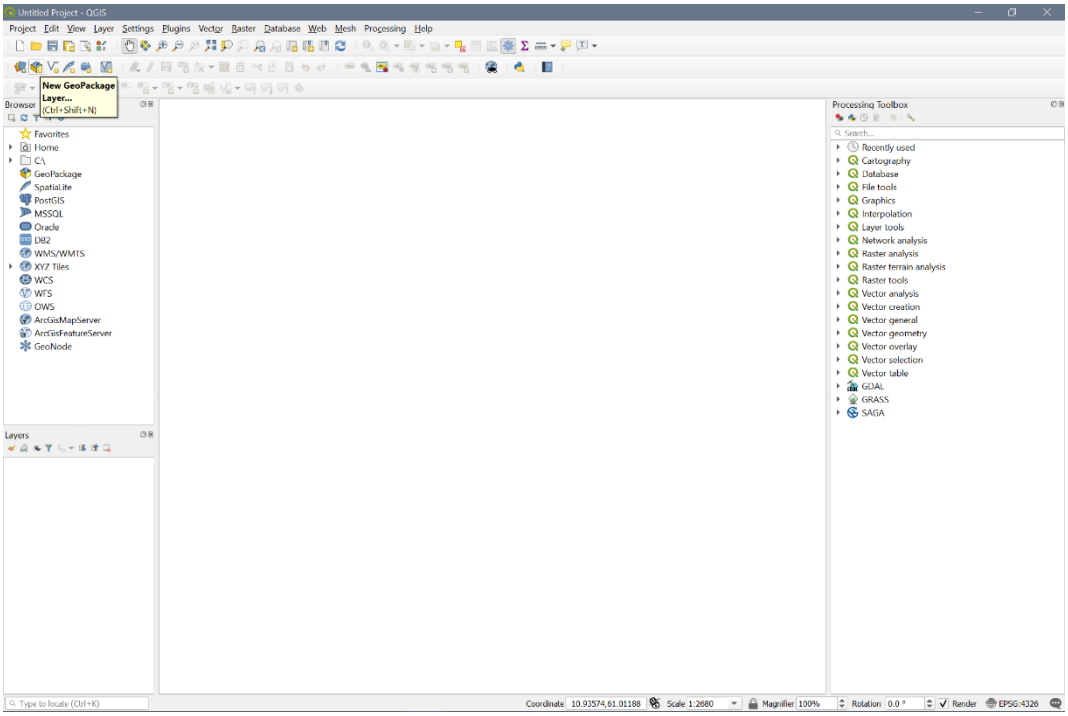
Pandas er et åpent python bibliotek utgitt under en BSD lisens. Pandas er designet for å gi raske, fleksible og uttrykksfulle datastrukturer, med et mål om å bli den fundamentale byggeblokken for å utføre praktisk dataanalyse i Python (Pandas, 2019).

Geopandas er en python-modul som bygger på Pandas. Pakken kombinerer evnene til Pandas og Shapley, for å gjøre det lettere å jobbe med geospatiale data i python. Geopandas er avhengig av python-modulen fiona for å lese filer, og modulene decartes og matplotlib for å plotte data (GeoPandas, u.å).

I denne oppgaven er Pandas og Geopandas brukt til å lage og håndtere datastrukturer.

3.4 QGIS

QGIS eller Quantum GIS er et åpent program for behandling av geografiske informasjonssystemer (GIS). Programmet er utviklet av Gary Sherman og blir i dag vedlikeholdt og oppdatert av frivillige. QGIS er et verktøy for arbeid med romlige data og kart. Man kan utføre analyse, prosessering og redigering av vektor og rasterbaserte data i QGIS (QGIS, u.å). I denne oppgaven er QGIS brukt til å opprette shape-polygoner for å lage trenings og valideringsdata til nevrale nettverk.



Figur 3.2: Skjermdump fra QGIS

3.5 sen2cor

Sen2Cor er et åpent program for generering av Sentinel-2 nivå 2A produkter. Programmet utfører de nødvendige atmosfæriske, terreng og cirrus korreksjonene som trengs for å generere et nivå 2A produkt fra et nivå 1C produkt (ESA, u.å-e). I denne oppgaven er Sen2Cor brukt til å generer 2A produkter fra nedlastede 1C Sentinel-2 produkter.

3.6 Filformater

Her følger en kort forklaring av ulike filformater som er aktuelle for denne oppgaven.

3.6.1 Shape

Shape er et filformat utviklet av ESRI. Shape-filer lagrer ikke-topografisk geometri og attributtinformasjon for romlige egenskaper i et datasett. Geometrien lagres som et sett med vektor-koordinater. Filformatet kan beskrive punkt, linje og areal egenskaper. Arealer blir representert som lukkede kreter som former koordinater (ESRI, 1998). I denne oppgaven er shape-formatet brukt i QGIS til å konstruere polygoner som er brukt som trenings og valideringsdata i nevrale nettverk.

3.6.2 CSV

CSV er et filformat brukt til å utveksle eller konvertere data mellom forskjellige regneark-program. Data lagres i rader og kolonner. Rader skilles ved linjeskift, mens data i kolonner kan skilles med komma, semikolon eller mellomrom. Dette må defineres ved import av data (Shafranovich, 2005). I denne oppgaven er CSV brukt til å lagre data fra shape-filer som skal brukes i PyCharm.

3.6.3 SENTINEL-SAFE

SENTINEL-SAFE er et filformat som brukes av ESA for distribusjon av sine Sentinel produkter. Formatet bygger på Standard Archive Format for Europe (SAFE) sine format spesifikasjoner. SENTINEL-SAFE formatet inneholder en mappe som inneholder bildedata på binærformat, og produktets metadata i XML format (ESA, u.å-d). I denne oppgaven er dette det formatet Sentinel-2 dataene er lagret i.

3.6.4 JPEG2000

JPEG2000 er en bildekompresjonstandard og kodesystem skapt av The Joint Photographic Experts Group i år 2000. JPEG2000 baserer seg på wavelets teknologi og er svært skalerbart (JPEG, u.å). Dette er filformatet bildedataen til Sentinel-2 er lagret i.

4. Framgangsmåte og metode

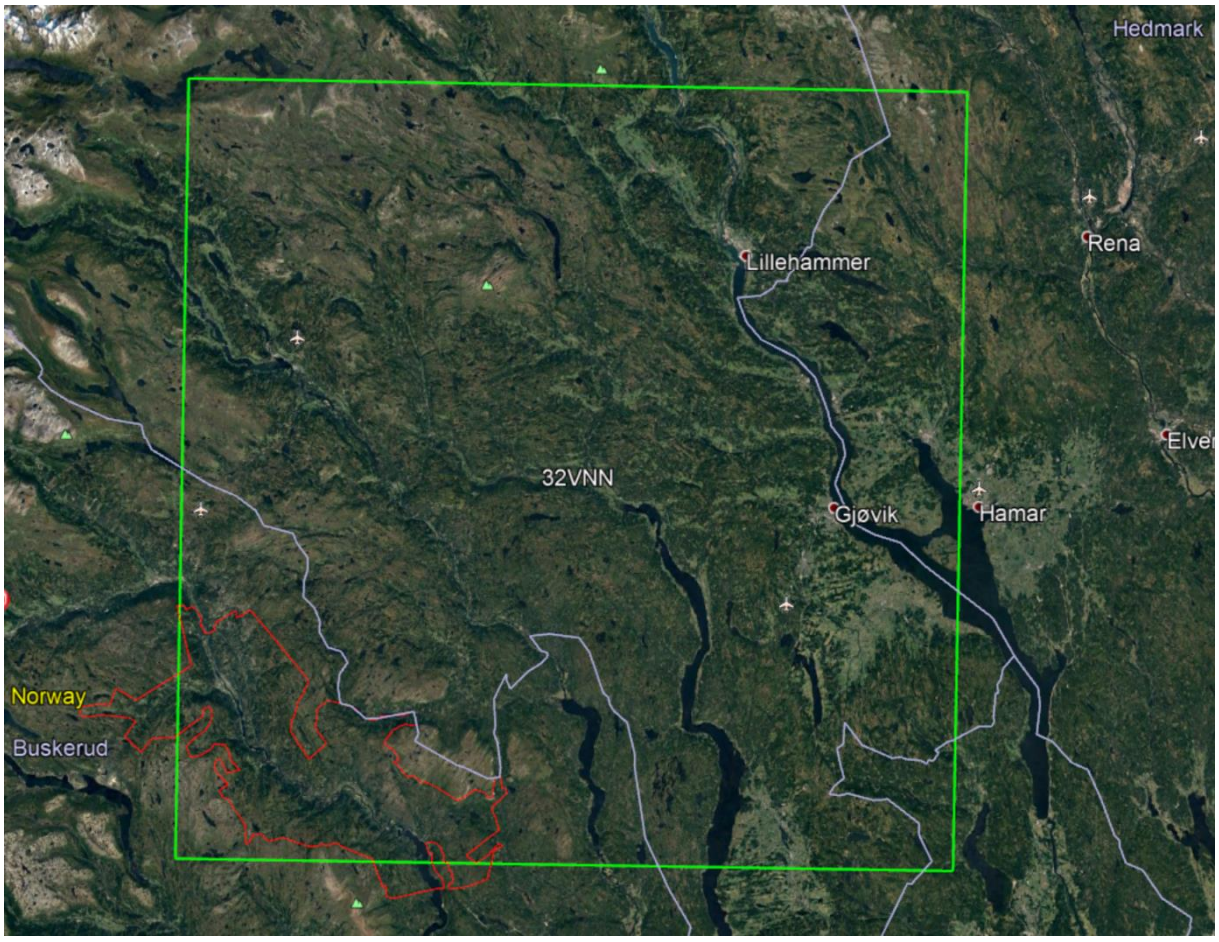
I dette kapittelet skal jeg gå gjennom og drøfte framgangsmåten og metoden som jeg har brukt for å løse denne oppgaven.

4.1 Valg av område

Ved valg av område i denne oppgaven var det to ting som var viktig; det første var at området innehold mye skog, og det andre var at det fantes et nyere laserdatasett på hoydedata.no, som kunne brukes som fasit for valideringsdataen. Etter å ha funnet aktuelle områder på hoydedata.no falt valget, etter anbefaling fra Floris Groesz ved Blom, på et område avgrenset av laserdatasettet *NDH Flå-Nes 5pkt 2018*. Datasettet avgrenser et område i kommunene Flå og Nes i Buskerud fylke, og ligger nesten utelukkende innenfor en sentinel-2 tile. Dette gjør nedlastningen enklere, fordi man da kun behøver å laste ned bilder fra en tile.

4.2 Innhenting av data

Etter å ha avgrenset det aktuelle området, lastet jeg ned sentinel-2 data. Siden nesten hele området ligger innenfor sentinel-2 sone 32VNN valgte jeg å kun laste ned bilder fra denne ene sonen og dermed bruke data som er avgrenset av laserdatasettet og Sentinel sonen til å lage valideringsdata. Figur 4.1 viser både laserdatasettets utstrekning og Sentinel-2 sone 32VNN.



Figur 4.1 Utstrekningen av testområde (rødt) og sentinel-2 sone 32VNN (grønn) vist i Google Earth

Sentinel data kan lastes ned gratis fra Copernicus Open Access Hub. I denne oppgaven har ikke data blitt lastet ned direkte fra Copernicus Open Access Hub, men ved hjelp av python modulen Sentinelsat. Med Sentinelsat lastet jeg ned produkter i sentinel-2 tile 32VNN som ble tatt opp mellom mai og august i 2017 og 2018. Alle produktene er av type 1C og har mindre enn 10% skydekke.

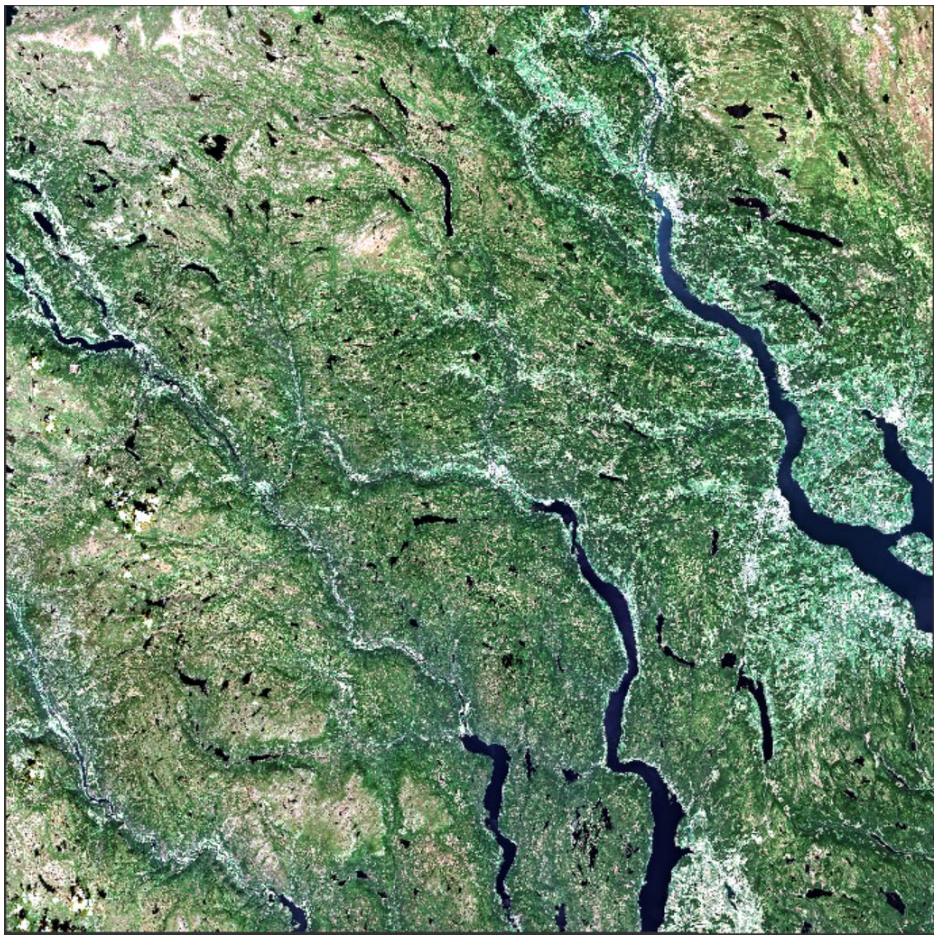
4.3 Bearbeiding av data

De nedlastede produktene var av type 1C og må derfor prosesseres for å skape 2A produkter. Prosesseringen gjøres med programmet sen2cor. Sen2cor kjøres på kommandolinje, og startes via PyCharm. For å gjøre det lettere å sammenlikne og kombinere bånd senere ble båndene med 20m romlig oppløsning resamplet til 10m. Dette blir gjort i PyCharm med funksjonen resize2min. (Se vedlegg B).

4.4 Trenings- og valideringsdata

For å kunne lage god trenings- og valideringsdata trengte jeg produkter som hadde lav skydekkeprosent og lite NoData. (NoData er et område hvor satellitten ikke har fanget opp informasjon). Jeg valgte derfor å se på endringer mellom to produkter som oppfyller disse kravene. Det ene er tatt opp den 26 mai 2017, se figur 4.2. Og det andre er tatt opp den 5 juli 2018, se figur 4.3.

Figur 4.2: RGB fremstilling av Sentinel-2 produkt fra 26.05.2017



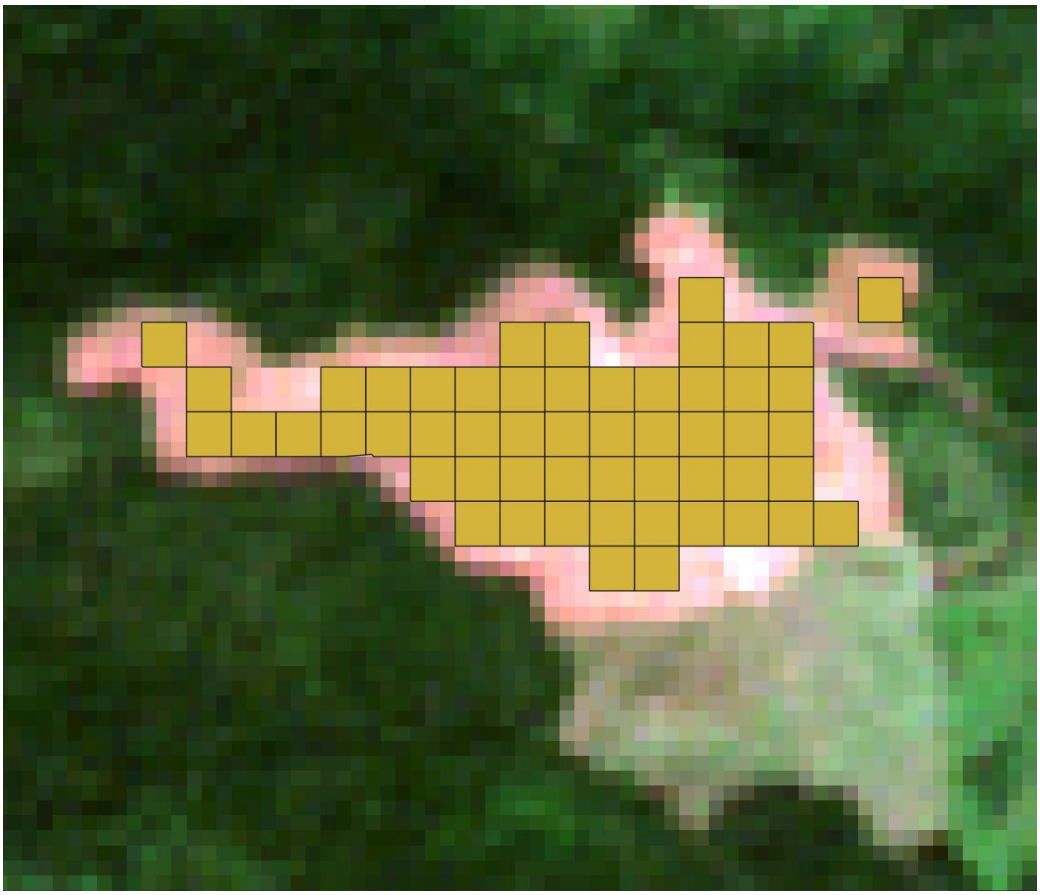
Figur 4.3: RGB fremstilling av Sentinel-2 produkt fra 05.07.2018

Etter at produkter var valgt begynte utvalget av treningsflater. Formålet med oppgaven var å finne hogstflater, men jeg hadde også lyst til å se hvilke terrengtyper som kunne enklest forveksles med hogster, jeg valgte derfor å lage fire arealklasser i mitt treningsdatasett. Arealklassene var; hogst, skog, dyrket mark og eng/gammel hogstflate. Treningsflatene er valgt ut ved visuell kontroll av mine to utvalgte produkter. For å gjennomføre denne kontrollen brukte jeg QGIS. I QGIS lastet jeg inn RGB båndene til hvert produkt og konstruerte et RGB-bilde med funksjonen Build Virtual Raster. For å finne hogster så jeg etter åpninger eller bare områder i skog på bildet fra 2018 for å så sammenligne det samme området med bildet fra 2017. Et eksempel kan sees i figur 4.4. Der det var en klar endring fra skog til klaring klassifiserte jeg det som en hogst.

Figur 4.4: Treningsflate av hogst, det øverste bildet viser 2017 og det nederst viser 2018

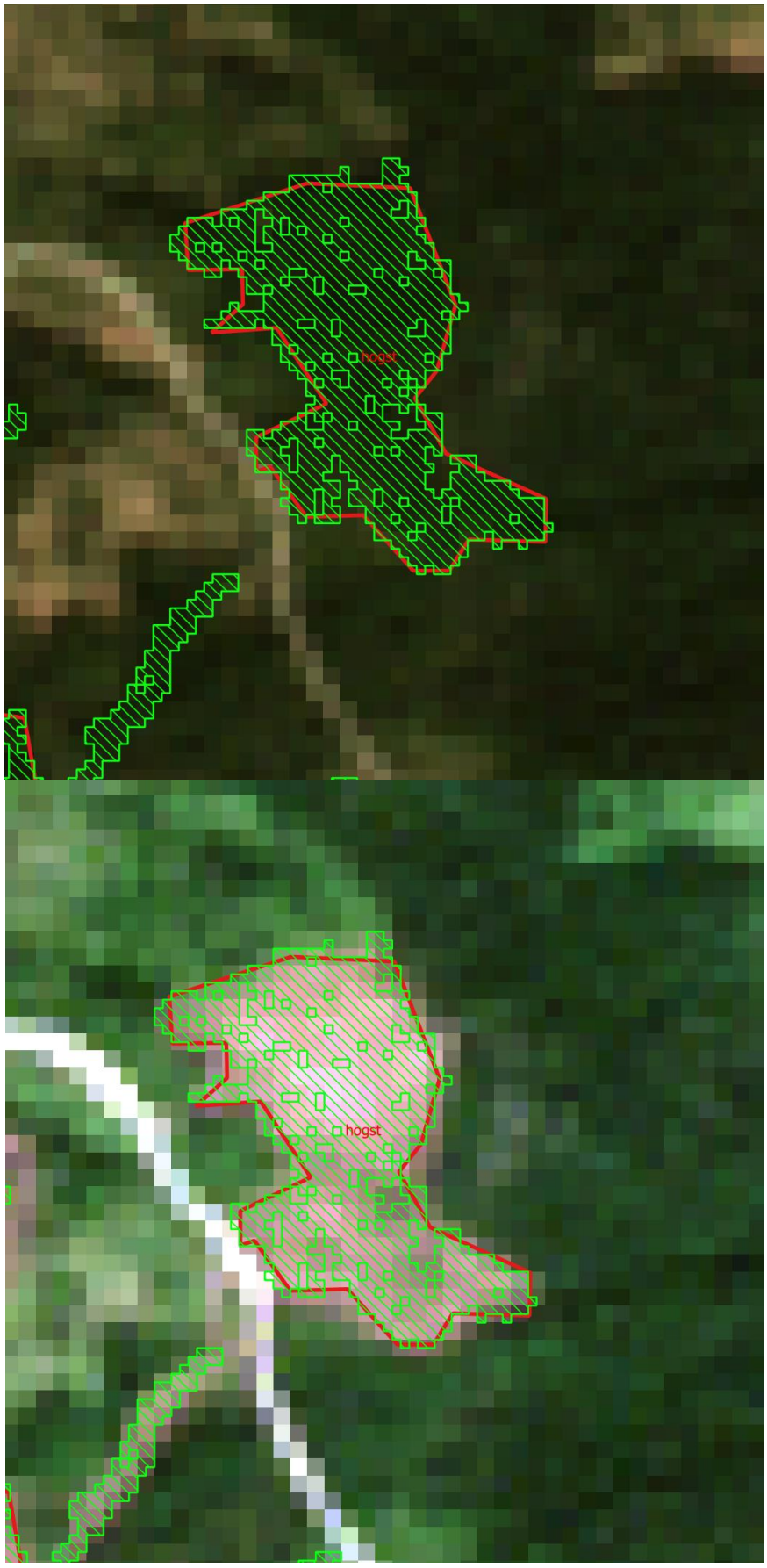
For de andre arealklassene var fremgangsmåten for det meste lik. Jeg fant aktuelle områder på bildet fra 2018 for å så sjekke at det aktuelle området var synlig i bildet fra 2017. For å velge ut skogsflater lette jeg etter skogsområder som forble uendret mellom bildene. For dyrket mark lette jeg etter åkere og for gamle hogster lette jeg etter åpninger i skog som var tilstede i begge bildene. Treningdataen ble så lagret som polygoner i en shape-fil. Treningsdatasettet inneholdt 29 hogstflater, og 10 flater av hver av de andre arealklassene.

Etter å ha valgt ut treningsflatene valgte jeg å splitte hver flate opp i 30x30 meter store rektangler for å skape mer treningsdata. Oppdelingen ble gjort ved å legge et 30x30 meter rutenett over polygonlaget, for å så klippe de lagene mot hverandre. Til slutt fjernet jeg de delene av polygonen som ikke utgjorde fulle kvadrater. Et eksempel på en treningsflate etter oppdeling kan sees i figur 4.5. Totalt sett var det 6183 treningsdataflater etter oppdelingen. Av de 6183 flatene var 1439 hogstflater, 2527 skogsflater, 1802 dyrket mark flater og 415 var gammel hogst/gressflater.



Figur 4.5: Treningsflater etter oppdeling

Da treningsdataene var på plass begynte jeg med valideringsdataene. Valideringsdataene ble valgt ut på samme måte som treningsdataene, og alle valideringsflatene kommer fra data som ligger innenfor området avgrenset av laserdatasettet *NDH Flå-Nes 5pkt. 2018*. Valideringsflatene ligger inni dette området for å kunne bruke laserdataene som en fasit på om det har skjedd en hogst eller ikke. Laserdataene brukes til å lage en høydeforskjellsmodell ved å sammenligne høydene på førstereturene fra laserdatasettet fra 2018 med et tidligere sett fra 2008 over det samme området. Denne modellen ble brukt til å skape en shape-fil som inneholder polygoner som indikerer hvor det har vært en endring i høydeprofilen. Ved hjelp av denne shape-filen og en visuell kontroll valgte jeg ut hogstfelter til å bruke i valideringsdatasettet mitt. Et eksempel på en valideringsflate og fasitdata kan sees i figur 4.6. Valideringsflater for de andre arealklassene ble valgt på samme måte som ved utvalg av treningsdata.



Figur4.6: Valideringssoner for hogst. Øverst fra 2017 og nederst fra 2018. Den lysegrønne flaten er laserdatasettet, og den røde er valgt valideringsflate

Figur 4.7: Alle valideringsflater innenfor testområde

Da alle trenings- og valideringsflater var utvalgt i QGIS måtte de kobles opp mot spektraldataen fra sentinel-2. Dette ble gjort i PyCharm med hjelp av modulen Rasterstats. Fra Rasterstats brukte jeg funksjonen zonalstatistics for å regne ut statistikk for hver av spektralklassene til sentinel-2 produktene inne i områdene avgrenset av polygonene mine. For hvert polygon regnet jeg ut maks, minimum og gjennomsnittlig pikselverdi. Denne dataen lagret jeg i en geopandas dataframe, og koblet den opp mot informasjonen fra polygonene mine fra QGIS. Etter å ha koblet sammen datasettene eksporterte jeg dataene ut på CSV format.

4.5 Keras modell

Etter at utvalget av data var gjennomført, begynte arbeidet med Keras modellen. Modellen som ble brukt i denne oppgaven er en sequential modell. (Sequential er en modell der lagene ligger lineært stablet). Modellen har 4 skjulte lag. Hvert lag er av typen dense, dette er et standard nevralt nettverkslag der alle input-nodene er koblet til alle output-nodene. Lagene inneholder også en regulizer for å gjøre modellens vektsmatriser mindre, dette blir gjort for å forhindre overtilpasning i modellen. Treningdataen ble splittet inn i trenings- og testdata som ble brukt til å skape modellen og en intern nøyaktighetstest. For hvert lag i modellen la jeg inn en dropout på 20%. Det vil si at 20% av neuronene ble ignorert under treningen. Modellen kjører 60 epoker, valgene er også denne gangen tatt for å forhindre at modellen lider av overtilpasning.

4.6 Test av valideringsdata mot modell

Med en ferdig modell på plass kunne nå arbeidet med å teste valideringsdataene starte. Det var totalt tre sett med valideringsdata som skulle testes. Alle valideringsdatasettene inneholdt maks, minimum og gjennomsnittsverdiene for hver valideringsflate, men denne dataen ble fremstilt på forskjellige måter i hvert av datasettene. Datasettene inneholdt følgende informasjon:

- Det første inneholdt data fra bildet tatt i 2018 og bildet tatt i 2017
- Det andre inneholdt data fra bildet fra 2018 og differansene i verdier mellom bildene fra 2018 og 2017
- Det tredje inneholdt kun differanseverdiene mellom bildene fra 2018 og 2017

For hvert av datasettene ble det kjørt tre tester;

- Test 1 hvor alle bånd med 10 og 20 meters romlig oppløsning ble brukt.
- Test 2 hvor kun de fire båndene med 10 meters romlig oppløsning ble brukt.
- Test 3 brukte fire av de tilgjengelige båndene med 20 meters romlig oppløsning. De fire båndene som ble brukt i denne testen var 5, 6, 7 og 8A.

Treningsdatasettene var delt inn på samme måte som valideringsdatasettene, og for hver test ble treningsdata med tilsvarende verdier som valideringsdataene brukt til å trene modellen. Resultatene fra disse testene vil bli presentert i neste kapittel.

5. Resultater

I dette kapittelet skal jeg presentere resultatene fra testene av valideringsdataene. Jeg utførte som beskrevet i kapittelet over, flere tester med forskjellige måter å sette opp dataene og med forskjellige kombinasjoner av bånd. Framgangsmåten for hvordan jeg kom fram til disse resultatene er beskrevet i kapittel 4. Det er med grunnlag i disse resultatene jeg skal besvare problemstillingen min. Resultatene vil bli presentert etter hvilket datasett de kommer fra. Det vil også bli presenter resultater fra en test der alle bånd ble kjørt separat for datasettet med data fra 2018 og 2017.

Dataene vil bli presentert i forvirringsmatriser der radene inneholder den faktiske arealklassen til valideringsflatene, og kolonnene vil vise hvilken arealklasse flatene ble klassifisert som. Den første vil vise klassifiseringen da testen ble kjørt med alle bånd med 10 og 20 meters oppløsning, den neste testen med alle bånd med 10 meters oppløsning og den siste testen med fire bånd med 20 meters oppløsning. Det vi også bli presentert statistiske mål for hver test. Statistikken som blir presentert er total nøyaktighet, kappa koeffisient og presisjon og recall for hver av arealklassene. Til slutt vil det være en tabell som viser kun hogstene for hver av testene. Denne tabellen vil vise om en hogst ble klassifisert riktig eller feil, samt hvor mange falske positive hogster det var i den aktuelle testen.

5.1 Datasett med data fra 2018 og 2017

Her vil resultatene fra datasettet som inneholder data fra 2018 og 2017 bli presentert.

Alle 10 og 20 meters bånd

Type	Klassifisert som			
	Hogst	Skog	Dyrket mark	Gress
Hogst	20	0	0	0
Skog	0	20	0	0
Dyrket mark	0	0	20	0
Gress	0	6	3	11

Tabell 5.1: Forvirringsmatrise for klassifisering av valideringsflater for test med alle 10 og 20 meters bånd fra datasett med data fra 2018 og 2017.

Total nøyaktighet	88,75%
Kappa koeffisient	0,85
Presisjon for hogst	100%
Recall for hogst	100%
Presisjon for skog	76,92%
Recall for skog	100%
Presisjon for dyrket mark	86,96%
Recall for dyrket mark	100%
Presisjon for gress	100%
Recall for gress	55%

Tabell 5.2: Statistiske mål for klassifisering av valideringsflater for test med alle 10 og 20 meters bånd fra datasett med data fra 2018 og 2017.

10 meters bånd (02, 03, 04 og 08)

Type	Klassifisert som			
	Hogst	Skog	Dyrket mark	Gress
Hogst	20	0	0	0
Skog	0	20	0	0
Dyrket mark	0	0	20	0
Gress	0	6	3	11

Tabell 5.3: Forvirringsmatrise for klassifisering av valideringsflater for test med alle 10 meters bånd fra datasett med data fra 2018 og 2017.

Total nøyaktighet	88,75%
Kappa koeffisient	0,85
Presisjon for hogst	100%
Recall for hogst	100%
Presisjon for skog	76,92%
Recall for skog	100%
Presisjon for dyrket mark	86,96%
Recall for dyrket mark	100%
Presisjon for gress	100%
Recall for gress	55%

Tabell 5.4: Statistiske mål for klassifisering av valideringsflater for test med alle 10 meters bånd fra datasett med data fra 2018 og 2017.

20 meters bånd (05, 06, 07 og 8A)

Type	Klassifisert som			
	Hogst	Skog	Dyrket mark	Gress
Hogst	20	0	0	0
Skog	0	20	0	0
Dyrket mark	1	0	15	4
Gress	1	6	4	9

Tabell 5.5: Forvirringsmatrise for klassifisering av valideringsflater for test med 20 meters bånd fra datasett med data fra 2018 og 2017.

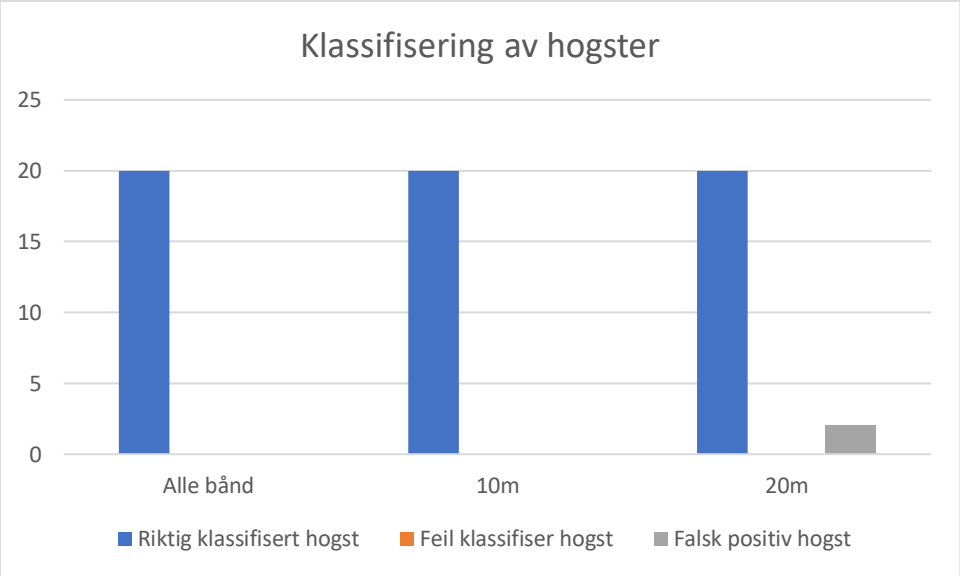
Total nøyaktighet	80%
Kappa koeffisient	0,733
Presisjon for hogst	90.91%
Recall for hogst	100%
Presisjon for skog	76,92%
Recall for skog	100%
Presisjon for dyrket mark	78,95%
Recall for dyrket mark	75%
Presisjon for gress	69,23%
Recall for gress	45%

Tabell 5.6: Statistiske mål for klassifisering av valideringsflater for test med 20 meters bånd fra datasett med data fra 2018 og 2017.

Hogster

	Riktig klassifisert hogst	Feil klassifiser hogst	Falsk positiv hogst
Alle bånd	20	0	0
10m	20	0	0
20m	20	0	2

Tabell 5.7: Klassifisering av hogstflater for datasett med data fra 2018 og 2017.



Figur 5.1: Grafisk fremstilling av hogstklassifiseringen for datasett med data fra 2018 og 2017

Resultatene fra testene på dette datasettet var veldig gode. På både testen med alle 10 og 20 meters bånd og testen med alle 10 meters bånd ble alle hogster, skogsflater og dyrket mark klassifisert riktig. Testen har derimot større problemer med å klassifisere gamle hogster/gress, men ingen av disse flatene ble klassifisert som hogster. Begge disse testene fikk en total nøyaktighet på 88,75% og en kappa på 0,85. På 20 meters båndene er det igjen ingen feilklassifiserte hogster, imidlertid ble det to falske positive hogster. Denne testen fikk en total nøyaktighet på 80%, og kappa på 0,733. Totalt ble det som vi da ser, ved dette datasettet, ingen feilklassifiserte hogster, og kun to falske positive hogster.

5.2 Datasett med data fra 2018 og differanser

Her vil resultatene fra datasettet som inneholder data fra 2018 og differanser bli presentert.

Alle 10 og 20 meters bånd

Type	Klassifisert som			
	Hogst	Skog	Dyrket mark	Gress
Hogst	20	0	0	0
Skog	0	19	0	1
Dyrket mark	1	0	19	0
Gress	0	4	4	12

Tabell 5.8: Forvirringsmatrise for klassifisering av valideringsflater for test med alle 10 og 20 meters bånd fra datasett med data fra 2018 og differanser.

Total nøyaktighet	87,5%
Kappa koeffisient	0,83
Presisjon for hogst	95,24%
Recall for hogst	100%
Presisjon for skog	82,61%
Recall for skog	95%
Presisjon for dyrket mark	82,61%
Recall for dyrket mark	95%
Presisjon for gress	92,31%
Recall for gress	60%

Tabell 5.9: Statistiske mål for klassifisering av valideringsflater for test med alle 10 og 20 meters bånd fra datasett med data fra 2018 og differanser.

10 meters bånd (02, 03, 04 og 08)

Type	Klassifisert som			
	Hogst	Skog	Dyrket mark	Gress
Hogst	20	0	0	0
Skog	0	19	0	1
Dyrket mark	1	0	19	0
Gress	0	5	2	13

Tabell 5.10: Forvirringsmatrise for klassifisering av valideringsflater for test med alle 10 meters bånd fra datasett med data fra 2018 og differanser.

Total nøyaktighet	88,75%
Kappa koeffisient	0,85
Presisjon for hogst	95,24%
Recall for hogst	100%
Presisjon for skog	79,17%
Recall for skog	95%
Presisjon for dyrket mark	90,48%
Recall for dyrket mark	95%
Presisjon for gress	92,86%
Recall for gress	65%

Tabell 5.11: Statistiske mål for klassifisering av valideringsflater for test med alle 10 meters bånd fra datasett med data fra 2018 og differanser.

20 meter bånd (05, 06, 07 og 8A)

Type	Klassifisert som			
	Hogst	Skog	Dyrket mark	Gress
Hogst	20	0	0	0
Skog	0	19	1	0
Dyrket mark	2	0	17	1
Gress	1	6	8	5

Tabell 5.12: Forvirringsmatrise for klassifisering av valideringsflater for test med 20 meters bånd fra datasett med data fra 2018 og differanser.

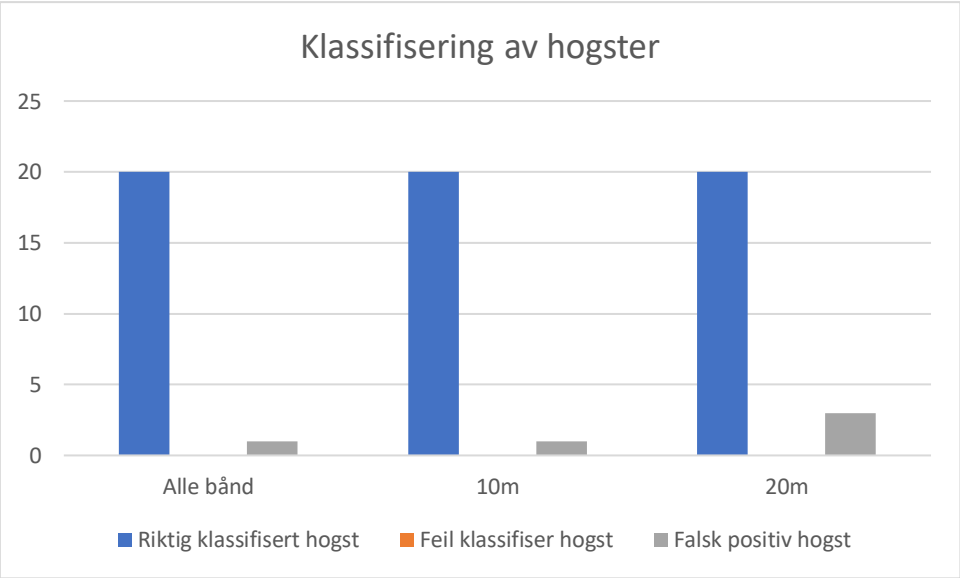
Total nøyaktighet	76,25%
Kappa koeffisient	0,68
Presisjon for hogst	86,96%
Recall for hogst	100%
Presisjon for skog	76%
Recall for skog	95%
Presisjon for dyrket mark	65,39%
Recall for dyrket mark	85%
Presisjon for gress	83,33%
Recall for gress	25%

Tabell 5.13: Statistiske mål for klassifisering av valideringsflater for test med 20 meters bånd fra datasett med data fra 2018 og differanser.

Hogster

	Riktig klassifisert hogst	Feil klassifiser hogst	Falsk positiv hogst
Alle bånd	20	0	1
10m	20	0	1
20m	20	0	3

Tabell 5.14: Klassifisering av hogstflater for datasett med data fra 2018 og differanser.



Figur 5.2: Grafisk fremstilling av hogstklassifiseringen for datasett med data fra 2018 og differanser

Igjen ble resultatene svært gode. Det er ingen feilklassifiserte hogster, men det er noen flere feil i de andre klassene. Viktigst er det at ved denne testen får jeg flere falske positive hogster. Det er en falsk positiv hogst i skogsflatene både i testen med alle 10 og 20 meters bånd, og to falske positive i testen med fire 20 meters bånd. I 20 meters testen er det også en falsk positiv hogst i gammel hogst/gress klassen. Dette gir totalt fem falske positive hogster i denne testen, men ingen feilklassifiserte hogster. Den totale nøyaktigheten til testen med alle 10 og 20 meters bånd ble 87,5% og kappa ble 0,83. For testen med 10 meters bånd ble total nøyaktighet 88,75% og kappa 0,85, og for testen med 20 meters bånd ble nøyaktigheten 76,25% og kappa 0,68.

5.3 Datasett som inneholder differansedata

Her vil resultatene fra datasettet som inneholder differansedata bli presentert.

Alle 10 og 20 meters bånd

Type	Klassifisert som			
	Hogst	Skog	Dyrket mark	Gress
Hogst	19	0	1	0
Skog	0	15	0	5
Dyrket mark	0	3	17	0
Gress	0	2	7	11

Tabell 5.15: Forvirringsmatrise for klassifisering av valideringsflater for test med alle 10 og 20 meters bånd fra datasett med differansedata.

Total nøyaktighet	77,5%
Kappa koeffisient	0,7
Presisjon for hogst	100%
Recall for hogst	95%
Presisjon for skog	75%
Recall for skog	75%
Presisjon for dyrket mark	68%
Recall for dyrket mark	85%
Presisjon for gress	68,75%
Recall for gress	55%

Tabell 5.16: Statistiske mål for klassifisering av valideringsflater for test med alle 10 og 20 meters bånd fra datasett med differansedata.

10 meters bånd (02, 03, 04 og 08)

Type	Klassifisert som			
	Hogst	Skog	Dyrket mark	Gress
Hogst	19	1	0	0
Skog	0	15	1	4
Dyrket mark	3	6	10	1
Gress	0	3	9	8

Tabell 5.17: Forvirringsmatrise for klassifisering av valideringsflater for test med alle 10 meters bånd fra datasett med differansedata.

Total nøyaktighet	65%
Kappa koeffisient	0,53
Presisjon for hogst	86,36%
Recall for hogst	95%
Presisjon for skog	60%
Recall for skog	75%
Presisjon for dyrket mark	50%
Recall for dyrket mark	50%
Presisjon for gress	61,54%
Recall for gress	40%

Tabell 5.18: Statistiske mål for klassifisering av valideringsflater for test med alle 10 meters bånd fra datasett med differansedata.

20 meter bånd (05, 06, 07 og 8A)

Type	Klassifisert som			
	Hogst	Skog	Dyrket mark	Gress
Hogst	19	0	1	0
Skog	0	15	0	5
Dyrket mark	1	2	16	1
Gress	0	6	7	7

Tabell 5.19: Forvirringsmatrise for klassifisering av valideringsflater for test med 20 meters bånd fra datasett med differansedata.

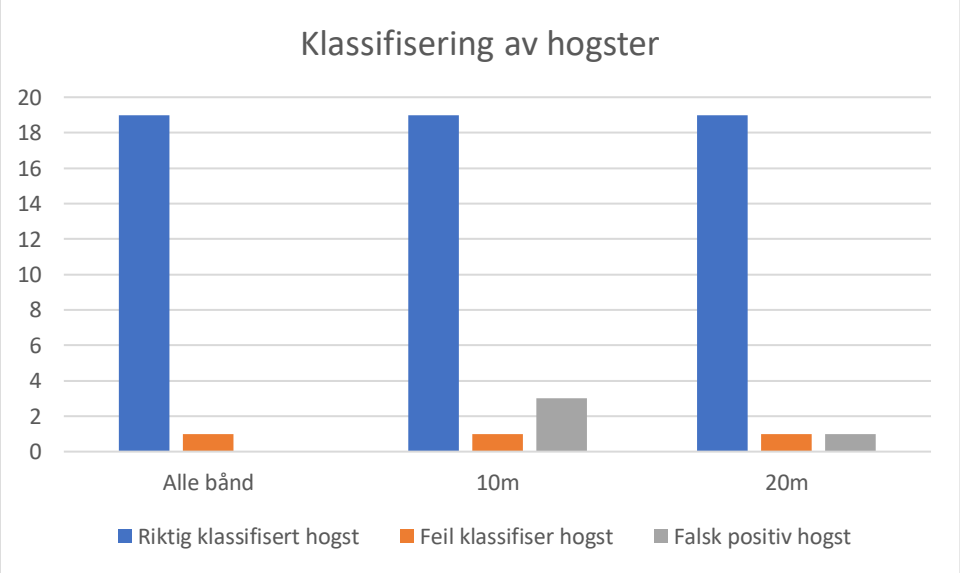
Total nøyaktighet	71,25%
Kappa koeffisient	0,62
Presisjon for hogst	95%
Recall for hogst	95%
Presisjon for skog	65,22%
Recall for skog	75%
Presisjon for dyrket mark	66,67%
Recall for dyrket mark	80%
Presisjon for gress	53,85%
Recall for gress	35%

Tabell 5.20: Statistiske mål for klassifisering av valideringsflater for test med 20 meters bånd fra datasett med differansedata.

Hogster

	Riktig klassifisert hogst	Feil klassifiser hogst	Falsk positiv hogst
Alle bånd	19	1	0
10m	19	1	3
20m	19	1	1

Tabell 5.21: Klassifisering av hogstflater for datasett med differansedata.



Figur 5.3: Grafisk fremstilling av hogstklassifiseringen for datasett med differansedata.

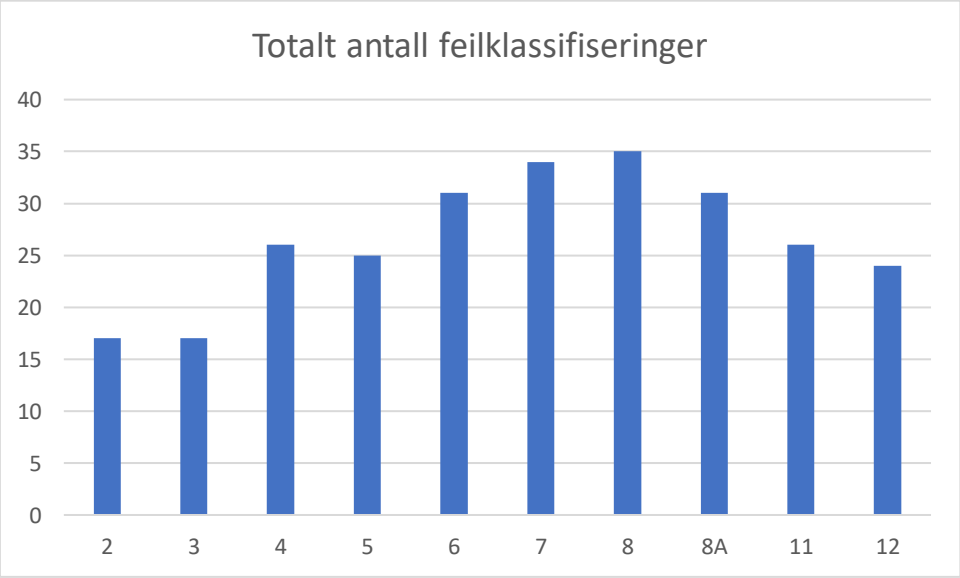
Med disse resultatene har det for første gang dukket opp feilklassifiserte hogster, det gjør det ved alle tre datasettene. Ved datasettet med alle 10 og 20 meters bånd og datasettet med fire 20 meters bånd ble en hogst klassifisert som dyrket mark, og ved datasettet med alle 10 meters bånd ble en hogst klassifisert som skog. Ved testen med alle 10 meters bånd er det også tre instanser av dyrket mark som har blitt klassifiser som hogst, det er også en instans i testen med 20 meters bånd som ble klassifisert som hogst. Dette gjør at for dette datasettet er det totalt tre feilklassifiserte hogster og fire falske positive. I dette datasettet fikk testen med 10 og 20 meters bånd en total nøyaktighet på 77,5 og en kappa på 0,70. Testen med 10 meters bånd fikk en total nøyaktighet 65% på og en kappa på 0,53. Testen med 20 meters bånd fikk en total nøyaktighet på 71,245 og en kappa på 0,62. Dette er det eneste datasettet der testen med 20 meters data fikk et bedre resultat enn testen med 10 meters bånd.

5.4 Enkeltbånd

Fram til nå har jeg sett på senarioer der det brukes flere bånd om gangen. Men det kan også være en fordel å se på hvordan modellen fungerer på enkeltbånd. Jeg har derfor tatt for meg datasettet som inneholder data fra både 2018 og 2017 og kjørt modellen på nytt for hvert av de 10 båndene jeg har brukt i denne oppgaven. Jeg valgte å bruke dette datasettet fordi det har gitt best resultater på de foregående testene.

Bånd	Totalt antall feilklassifiseringer
02	17
03	17
04	26
05	25
06	31
07	34
08	35
8A	31
11	26
12	24

Tabell 5.22: Totalt antall feilklassifiseringer for alle bånd.



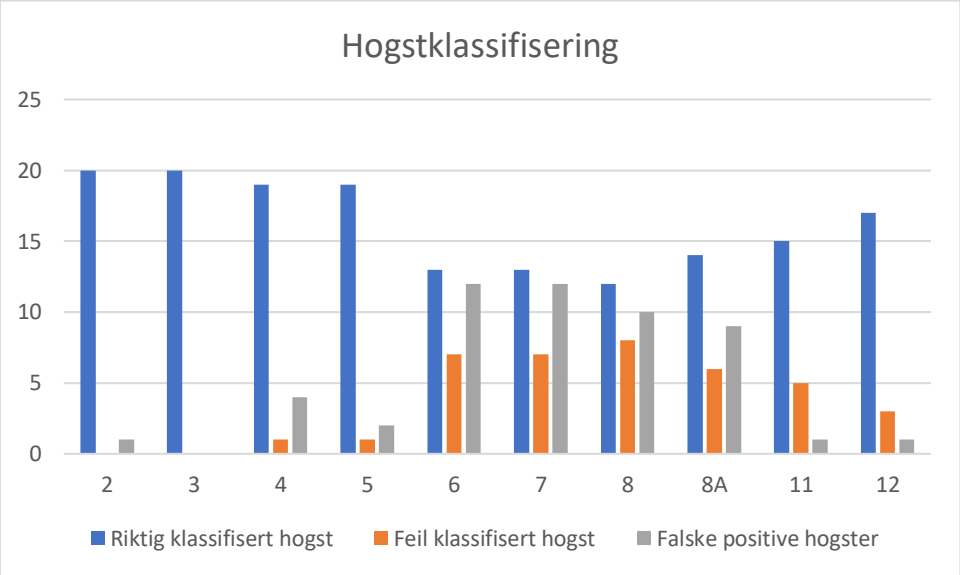
Figur 5.4: Grafisk fremstilling av feilklassifiseringer for alle bånd

Ut fra tabell 5.22 kan man se at det er bånd 2 og 3 som gir de aller beste resultatene. Disse båndene ligger i henholdsvis den blå og grønne delen av spekteret, og har begge en oppløsning på 10 meter. Bånd 4 som er båndet for den røde delen av spektret gir ikke et like godt resultat som de to andre for synlig lys, men er fortsatt blant de bedre med totalt 26 feilklassifiseringer. Noe overraskende er det at bånd 8, det siste båndet med 10 meter oppløsning har aller flest feilklassifiseringer med 35. Videre er det verdt å merke seg at bånd 5, 11 og 12, som alle har oppløsning på 20 meter, gir jevnt like gode resultater som bånd 4.

Ut fra tabellene kan man se at den største delen av feilklassifiseringene stammer fra klassen gammel hogst/gress. Siden det er hogster som er det viktigste i denne oppgaven, kan det være en fordel å se på hvilke bånd som gjør det best på klassifisering av hogster.

Bånd	Riktig klassifisert hogst	Feil klassifisert hogst	Falske positive hogster
02	20	0	1
03	20	0	0
04	19	1	4
05	19	1	2
06	13	7	12
07	13	7	12
08	12	8	10
8A	14	6	9
11	15	5	1
12	17	3	1

Tabell 5.23: Hogstklassifisering for alle bånd.



Figur 5.5: Grafisk fremstilling av hogstklassifiseringer.

Som forventet ut fra antall feilklassifiseringer er det bånd 2 og 3 som kommer best ut. Begge har ingen feilklassifiserte hogster, mens bånd 2 har en falsk positiv mot ingen for bånd 3. Igjen er det bånd 4, 5, 11 og 12 som følger i puljen bak, og det er igjen verdt å merke seg at bånd 8 med ti meters oppløsning er blant de dårligste.

6. Diskusjon

I dette kapittelet skal jeg drøfte resultatene fra forrige kapittel opp mot oppgavens problemstilling. Jeg vil også se på hva jeg mener vil være aktuelt å arbeide videre med for å utdype og utvikle funnene i denne oppgaven.

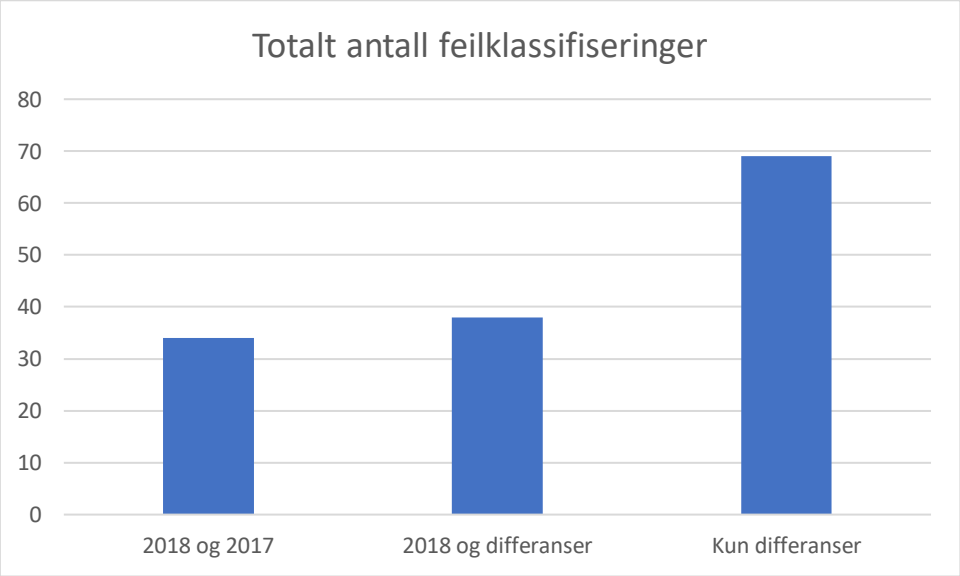
Vi startet med problemstillingen «*Hvordan kan sentinel-2 brukes til å detektere endringer i skog?»*. Jeg har sett på flere måter å sette opp dataen fra sentinel-2 bildene og forskjellige kombinasjoner av bånd å kjøre modellen min på. Jeg skal nå se på hva som påvirker resultatet, hvilke valideringsflater som gir feil utslag og hvordan få best mulige resultater.

6.1 Diskusjon av resultater

Jeg vil starte med å se på de forskjellige datasettene. Jeg hadde totalt tre datasett med trenings og valideringsdata som jeg brukte i denne oppgaven. Det første inneholdt maks, minimum og gjennomsnitt pikselverdiene til trenings og valideringsflatene fra bildet tatt 07.05.2017 0g bildet tatt 23.07.2018. Det andre inneholdt maks, minimum og gjennomsnittverdiene for bildet fra 2018 og differansene i maks, minimum og gjennomsnitt pikselverdiene mellom bildet fra 2018 og 2017. Det siste datasettet inneholdt kun differansene mellom bildene. Det første jeg ønsker å gjøre er å sammenligne totalt antall feilklassifiseringer for hver av datasettene for alle tre testene som er kjørt på hvert av dem.

Datasett	Totalt antall feilklassifiseringer
2018 og 2017	34
2018 og differanser	38
Kun differanser	69

Tabell 6.1: Totalt antall feilklassifiseringer for alle datasett



Figur 6.1: Grafisk fremstilling av totalt antall feilklassifiseringer fra alle datasett

I tabellen ser vi at det er relativt jevnt mellom datasettet med data fra 2018 og 2017 og datasettet med 2018 og differanser, mens datasettet med kun differanser har rundt dobbelt

så mange feilklassifiseringer som de to andre. Dette kan skyldes datamengden i datasettene. Der de to første datasettene inneholder maks, minimum og gjennomsnittverdiene for hvert spektralbånd to ganger, det vil si 60 datapunkter for hver klassifiserings og treningsflate, inneholder det siste datasettet kun verdiene en gang for hver flate. Det kan se ut som om dette har en stor innvirkning på hvor godt modellen klarer å klassifisere riktig.

Til nå har jeg sett på totalt antall klassifiseringsfeil, men hvis vi ser på resultatfilene i forrige kapittel ser vi at de aller fleste stammer fra klassen gammel hogst/gress og veldig få fra hogst klassen. Det er hogster som er mest interessante for denne oppgaven, så det kan lønne seg å se på kun hogster.

	Riktig klassifiserte hogster	Feil klassifiserte hogster	Falske positive hogster
2018 og 2017	60	0	2
2018 og differanser	60	0	5
Kun differanser	57	3	4

Tabell 6.2: Klassifisering av hogster for alle datasett

Fra tabellen kan man se at det ikke er et like klart skille mellom datasettet med kun differanser og datasettet med 2018 og differanser, når man kun ser på hogster. Det er fortsatt datasettet med 2018 og 2017 som kommer best ut, og ved dette tilfellet vil jeg si at denne testen kommer klart best ut. Etter å ha sett på både alle klassifiseringer og kun hogster ser det ut til at hvordan man setter opp datasettet sitt har en klar innvirkning på hvor nøyaktige resultater man kan forvente av en klassifisering. I dette tilfellet kom datasettet med data fra både 2018 og 2017 best ut i begge tilfellene jeg har sett på.

For hvert datasett kjørte jeg tre tester, en hvor jeg brukte alle tilgjengelige bånd med 10 og 20 meters oppløsning, en med alle fire båndene med 10 meters oppløsning og en hvor jeg brukte fire av de tilgjengelige båndene med 20 meters oppløsning. Nå skal jeg se på hvilken av disse testene som ga de beste resultatene, og fra hvilket datasett testen med de beste resultatene kommer fra.

Test	Totalt antall feilklassifiseringer
2018 og 2017	
Alle bånd	9
10m bånd	9
20m bånd	16
2018 og differanser	
Alle bånd	10
10m bånd	9
20m bånd	19
Kun differanser	
Alle bånd	18
10m bånd	28
20m bånd	23

Tabell 6.3: Totalt antall feilklassifiseringer for alle tester

Fra tabell 6.3 kan man se at det totalt sett er testen med alle bånd som gjør det best, etterfulgt av testen med 10 meters bånd og til slutt kommer testen med 20 meters bånd. Det er vært å merke seg at testen med 20 meters bånd gjør det betydelig dårligere enn testen med 10 meters bånd for datasettene med data fra 2018 og 2017 og 2018 og differanser, men mye bedre for datasettet med kun differanse data. Dette kan skyldes at dette datasettet generelt gjør det dårligere, og det er mer støy i denne modellen.

Videre kan det igjen være lurt å se på kun hogster for å få en bedre indikasjon på hvor bra hver test fungerte for vårt formål.

Test	Riktig klassifisert hogst	Feil klassifisert hogst	Falsk positiv hogst
2018 og 2017			
Alle bånd	20	0	0
10m bånd	20	0	0
20m bånd	20	0	2
2018 og differanser			
Alle bånd	20	0	1
10m bånd	20	0	1
20m bånd	20	0	3
Kun differanser			
Alle bånd	19	1	0
10m bånd	19	1	3
20m bånd	19	1	1

Tabell 6.4: Klassifisering av hogster for alle tester

Fra tabell 6.4 kan vi igjen se at det er testene med alle bånd som gir de beste resultatene, etterfulgt av testen med 10 meters bånd og igjen gjør testen med 20 meters bånd det jevnt over dårligst. Det er igjen vært å merke seg at 20 meters bånd gjør det dårligere for de to første datasettene, men bedre for det siste.

Totalt sett vil jeg si at det datasettet som jeg ville anbefale å benytte videre, er datasettet med data fra både 2018 og 2017. Jeg vil også si at 10 meters båndene presterer bedre enn 20 meters båndene. Jeg vil anbefale å bruke alle de tilgjengelige båndene for å få best mulig resultat. Hvis det er ønskelig å senke datamengden er det et godt alternativ og bruke kun 10 meters båndene.

Jeg skal nå se på egenskapene til de forskjellige båndene. Her skal jeg se på enkeltbånd og se på om det er båndenes romlige oppløsning eller spektrale oppløsning som har mest å si for hvor bra de presterer i testen. Fra tabell 5.23 og 5.24 kan man se at det er bånd 2 og 3 som gir aller best resultater. Begge disse båndene har en romlig oppløsning på 10 meter og en spektral oppløsning i det synlige spekteret, henholdsvis for blått og grønt lys. Etter disse følger båndene 5 og 12. Bånd 5 er et *vegetation red edge* bånd med romlig oppløsning på 20 meter. Bånd 12 er et kortbølge IR bånd med romlig oppløsning på 20 meter. Videre følger bånd 4 og 11. Bånd 4 har en romlig oppløsning på 10 meter og er båndet for rødt lys. Bånd 11 har en romlig oppløsning på 20 meter og er et kortbølge IR bånd. De fire dårligste båndene er bånd 6, 7, 8 og 8A. Bånd 6, 7 og 8A er alle *vegetation red edge* bånd og har en romlig oppløsning på 20 meter, mens bånd 8 er et nær infrarødt bånd og har en romlig oppløsning på 10 meter. Det er vært å merke seg at båndene som presterer best har en lignende spektral oppløsning, det gjelder også de fire båndene som presterer dårligst i denne testen. Jeg vil derfor si at i denne testen er det vanskelig å bedømme om det er den spektrale eller romlige oppløsningen som betyr mest for hvor bra båndene presterer. Det virker som om det er en kombinasjon av de to oppløsningene som bestemmer hvor godt et bånd fungerer. Det er da interessant å se på at blant de båndene som presterer best er båndene for synlig lys. Det kan tilsi at man kan få svært gode resultater fra Sentinel-2 kun ved å se på RGB bilder.

Har skal jeg gå nærmere innpå hvilke valideringsflater som gir feilklassifiserte hogster. Jeg vil se på testene med alle bånd, 10 meters bånd og 20 meters bånd. Valideringsflatene er nummerert fra 1 til 80, der 1-20 er hogster, 21-40 er skogsflater, 41-60 er dyrket mark og 61-80 er gamle hogster/gressflater. Ved feilklassifiserte hogster mener jeg her både hogstflater som er feilklassifisert og andre flater som er klassifisert som hogster.

Test	Feilklassifiseringer	Flatenr.
2018 og 2017		
Alle bånd	0	
10m bånd	0	
20m bånd	2	51 og 79
2018 og differanser		
Alle bånd	1	58
10m bånd	1	60
20m bånd	3	55, 60 og 79
Kun differanser		
Alle bånd	1	20
10m bånd	4	10, 43, 52, 60
20m bånd	2	20 og 60

Tabell 6.5: Antall feilklassifiserte hogster for hver test

Det første man kan trekk ut er at det er klassen dyrket mark, nr. 41-60 som oftest blir feilklassifisert som hogster. Det er også verdt å merke seg at ingen skogsflater blir feilklassifisert som hogster. Det er kun for datasettet med kun differanser det er hogstflater som blir feilklassifisert. Ser man på alle feilklassifiseringer i oppgaven ser man at modellen har størst problemer med å klassifisere gammel hogst/gressflate klassen, og som nevnt er det flest dyrket mark flater som blir feilklassifisert som hogster. Det kan tyde på at det kunne vært behov for å forbedre treningsdataen for disse klassene.

6.2 Sammenligning med tidligere forskning

I det etterfølgende sammenligner jeg mine funn med det jeg har funnet av tidligere forskning på feltet. Forsøket på automatisk klassifisering av hogst og skogsskader fra 2014 med Landsat 7 og 8 ga total treffsikkerhet på 82,68% og 88,19% for sine to forsøk. Kappa koeffisientene var 0,20 og 0,24. For hogster fikk man en presisjon på 62%, og en recall på 5%. For å sammenligne det med mine beste resultater fikk jeg en total nøyaktighet på 88,75% og en kappa koeffisient på 0,85. Min presisjon og recall for hogstflater ble begge 100%. Det er flere forskjeller på mine eksperimenter og de som er gjennomført i 2014. Forsøket fra 2014 er pikselstyrt, mens mitt forsøk er basert på minimum, maksimum og gjennomsnittverdier for flater. Førsøket fra 2014 bruker vegetasjonsindekser, mens jeg kun bruker rene båndkombinasjoner. Mest fremtredende er det at det er forskjellige oppløsninger på Landsat satellittene og Sentinel-2. Der Landsat dataene her en romlig oppløsning på 30 meter, mens Sentinel-2 dataene ha en romlig oppløsning på 10 eller 20 meter. Det kan derfor være grunnlag for å si at den betydelig høyere romlige oppløsningen til Sentinel-2 har en sterk innvirkning på kvaliteten til resultatene.
Konklusjonene i doktoravhandlingen fra 2017 som sammenlignet og kombinerte Sentinel-2 og Landsat data for å finne mindre endringer i skog, underbygger at det er en betydelig forbedring i klassifiseringen ved bruk av 10 meters data.

6.3 Forslag til videre forskning

Etter å ha arbeidet med denne oppgaven har jeg nå noen forslag til hvordan man kan videreutvikle resultatene. Det første som kan gjøres er å teste ut tidsserier istedenfor å kun se på endringer mellom to bilder, som er det jeg har gjort i denne oppgaven. Det kan også sees på om en pikselstyrt klassifisering kan oppnå lignende eller bedre resultater enn jeg har gjort i denne oppgaven. Det kan forskes mer på om andre båndkombinasjoner eller indekser kan forbedre resultatet ytterligere. I mitt arbeide er den eneste typen endringer som er sett på snauhogster. Det kan derfor være en ide for fremtiden å se på om det er andre typer endringer som også kan oppdages ved bruk av Sentinel-2 data. Eksempel kan det sees på om man kan oppdage tynningshogster.

7. Konklusjon

Med bakgrunn i de testene som er kjørt i denne oppgaven skal jeg nå besvare problemstillingen fra innledningen. Problemstillingen var: «Hvordan kan Sentinel-2 benyttes til å detektere endringer i skog?».

Det var klart fra starten av mitt arbeide at Sentinel-2 kan benyttes til å finne endringer i skog. Min forskning viser at både hvordan man setter opp data, og hvilke spektralbånd som benyttes påvirker resultatene. Totalt sett var det datasettet med data fra både 2018 og 2017 som ga best resultater. Av testene som ble kjørt var det testen med alle tilgjengelige 10 og 20 meters bånd som ga de beste resultatene. Videre ga testen med 10 meters bånd bedre resultater enn testen med 20 meters bånd. Jeg vil anbefale å bruke alle de tilgjengelige båndene for å få best mulig resultat. Hvis det er ønskelig å senke datamengden er det et godt alternativ og bruke kun 10 meters båndene.

Videre er det verdt å merke seg at funnene har gitt generelt svært gode resultater. Årsaken til dette kan sees i sammenheng med de relativt enkle egenskapene ved spektralklassifisering av hogst, med dette mener jeg at hogst er enkelt å visuelt klassifisere. Videre kan de gode resultatene tyde på at datagrunnlaget for valideringsdata er lite variert. Årsaken til dette kan være at dataene er hentet fra et lite geografisk område med få variasjoner.

Et vesentlig funn i mitt arbeide, er at høyere romlig oppløsning gir bedre resultater. Dette samsvarer med funnene i doktoravhandlingen fra 2017 hvor bruk av 10 meters Sentinel-2 data gir bedre resultater enn bruk av 30 meters Landsat data.

Selv med funnene av at høyere romlig oppløsning gir bedre resultater, er det noe overraskende at RGB båndene gir såpass mye bedre resultater enn mange av 20 meters båndene. Dette kan tyde på at det ved tester med Sentinel-2 kun er nødvendig å bruke RGB bilder.

Ved bruk av nevrale nettverk har mine funn vist at dataoppsett for trenings- og valideringsdata påvirker hvor gode resultater testene gir. Det er til dels store forskjeller mellom mine datasett i denne testen. Mengden treningsdata har også en stor innvirkning på hvor godt en modell klarer å predikere resultater. Dette er tydelig ut fra mine tester der arealklassen gammel hogst/gress, som er trent på en god del færre flater enn de andre klassene, har betydelig flere feilklassifiseringer enn de andre klassene.

Sentunel-2 er et godt verktøy når man arbeider med å finne endringer i skog. Det viser seg at et godt datagrunnlag er viktig for å oppnå nøyaktige resultater når man jobber med endringsanalyse.

Referanseliste

Anaconda. (u.å). *Anaconda Documentation*. Tilgjengelig fra: https://docs.anaconda.com/ (lest 29.04.2019).

ESA. (2015). Sentinel-2 User Handbook. 64.

ESA. (u.å-a). *Copernicus Open Access Hub*. Tilgjengelig fra: https://scihub.copernicus.eu/ (lest 17.04.2019).

ESA. (u.å-b). *MultiSpectral Instrument (MSI)*. Tilgjengelig fra: https://sentinel.esa.int/web/sentinel/missions/sentinel-2/instrument-payload (lest 16.04.2019).

ESA. (u.å-c). *MultiSpectral Instrument (MSI) Overview*. Tilgjengelig fra: https://sentinel.esa.int/web/sentinel/technical-guides/sentinel-2-msi/msi-instrument (lest 09.05.2019).

ESA. (u.å-d). *SAFE spesification*. Tilgjengelig fra: https://sentinel.esa.int/web/sentinel/user-guides/sentinel-1-sar/data-formats/safe-specification (lest 26.04.2019).

ESA. (u.å-e). *Sen2Cor*. Tilgjengelig fra: http://step.esa.int/main/third-party-plugins-2/sen2cor/ (lest 20.04.2019).

ESRI. (1998). ESRI Shapefile Technical Description. 34.

GDAL. (u.å). *GDAL - Geospatial Data Abstraction Library*. Tilgjengelig fra: https://www.gdal.org/ (lest 29.04.2019).

GeoPandas. (u.å). *GeoPandas 0.4.0*. Tilgjengelig fra: http://geopandas.org/ (lest 29.04.2019).

Hamunyela, E. (2017). *Space-time monitoring of tropical changes using observations from multiple satellites*. Doktoravhandling. Wageningen: Wageningen University.

Jetbrains. (u.å-a). *The drive to develop*. Tilgjengelig fra: https://www.jetbrains.com/company/?fromMenu (lest 24.04.2019).

Jetbrains. (u.å-b). *Free individual licenses for students and faculty members*. Tilgjengelig fra: https://www.jetbrains.com/student/ (lest 24.04.2019).

Jetbrains. (u.å-c). *PyCharm*. Tilgjengelig fra: https://www.jetbrains.com/pycharm/?fromMenu (lest 24.04.2019).

Jetbrains. (u.å-d). *PyCharm Community Edition*. Tilgjengelig fra: https://github.com/JetBrains/intellij-community/tree/master/python (lest 24.04.2019).

JPEG. (u.å). *Overview of JPEG 2000* Tilgjengelig fra: https://jpeg.org/jpeg2000/index.html (lest 26.04.2019).

Keras. (u.å). *Keras: The python deep learning library*. Tilgjengelig fra: https://keras.io/ (lest 26.04.2019).

Khandelwal, P., Singh, K. K., Singh, B. K. & Merotra, A. (2013). Unsupervised Change Detection of Multispectral Images using Wavelet Fusion and Kohonen Clustering Network. *International Journal of Engineering and Technology*, 5 (2): 6.

Lagandula, A. C. (2018). *Perceptron: The Artificial Neuron (An Essential Upgrade To The McCulloch-Pitts Neuron)*. Tilgjengelig fra: https://towardsdatascience.com/perceptron-the-artificial-neuron-4d8c70d5cc8d (lest 06.05.2019).

NIBIO. (u.å). *Nøkkeltall 2013-2017*. Tilgjengelig fra: https://nibio.no/tema/skog/skog-og-miljoinformasjon-fra-landsskogtakseringen/nokkeltall (lest 06.05.2019).

NVE. (2016). *Ny veileder om skogrydding under kraftledninger*. Tilgjengelig fra: https://www.nve.no/nytt-fra-nve/nyheter-sikkerhet-og-energiforsyningsberedskap/ny-veileder-om-skogrydding-under-kraftledninger/ (lest 06.05.2019).

Pandas. (2019). *pandas: powerful Python data analysis toolkit*. Tilgjengelig fra: http://pandas.pydata.org/pandas-docs/stable/ (lest 29.04.2019).

Perry, M. T. (2015). *rasterstats*. Tilgjengelig fra: https://pythonhosted.org/rasterstats/ (lest 24.04.2019).

Powers, D. M. W. (2007). *Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation* Adelaide: School of Informatics and Engineering Flinders University.
PSF. (u.å). *Python*. Tilgjengelig fra: https://www.python.org/about/ (lest 23.04.2019).
QGIS. (u.å). *QGIS User Guide*. Tilgjengelig fra: https://docs.qgis.org/3.4/en/docs/user_manual/ (lest 19.04.2019).
seninelsat. (u.å). *Sentinelsat*. Tilgjengelig fra: https://sentinelsat.readthedocs.io/en/stable/index.html (lest 26.04.2019).
Shafranovich, Y. (2005). *Common format and MINE type of comma-separated values (CSV) files*. Tilgjengelig fra: https://tools.ietf.org/html/rfc4180 (lest 26.04.2019).
Skymind. (u.å). *A beginner's guide to neural netwoks and deep learning* Tilgjengelig fra: https://skymind.ai/wiki/neural-network (lest 06.05.2019).
Solberg, S., Bjørkelo, K., Gjertsen, A., May, J., Pierzchala, M., Timmermann, V. & Solheim, H. (2014). *Satellittkartlegging av hogst og skogsskader*. Oppdragsrapport fra Skog og landskap. Ås: Skog os landskap.
Srivastava, N., Hinton, G., Krizhevsky, A. & Salakhtdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15.
SSB. (2019). *Skogeiernes inntekt*. Tilgjengelig fra: https://www.ssb.no/jord-skog-jakt-og-fiskeri/statistikker/skoginnt (lest 06.05.2019).
Tensorflow. (u.å). *Why tensorflow*. Tilgjengelig fra: https://www.tensorflow.org/about (lest 29.04.2019).
TIOBE. (2019). *TIOBE index for May 2019*. Tilgjengelig fra: https://www.tiobe.com/tiobe-index/ (lest 10.05.2019).
van Rossum, G. (2009). *A Brief Timeline of Python*. Tilgjengelig fra: http://python-history.blogspot.com/2009/01/brief-timeline-of-python.html (lest 23.04.2019).
van Rossum, G. (u.å). *Guido van Rossum - Brief Bio*. Tilgjengelig fra: https://gvanrossum.github.io//bio.html (lest 27.04.2019).
Viera, A. J. & Garrett, J. M. (2005). Understanding Interobserver Agreement: The Kappa Statistic. *Family Medicine*, 37 (5): 4.

Vedlegg A – Forvirringsmatriser for enkeltbåndstester

Bånd 02

Type	Klassifisert som			
	Hogst	Skog	Dyrket mark	Gress
Hogst	20	0	0	0
Skog	0	20	0	0
Dyrket mark	1	0	19	0
Gress	0	8	8	4

Tabell A.1: Forvirringsmatrise for klassifisering av valideringsflater for test med bånd 02.

Bånd 03

Type	Klassifisert som			
	Hogst	Skog	Dyrket mark	Gress
Hogst	20	0	0	0
Skog	0	20	0	0
Dyrket mark	0	0	20	0
Gress	0	7	10	3

Tabell A.2: Forvirringsmatrise for klassifisering av valideringsflater for test med bånd 03.

Bånd 04

Type	Klassifisert som			
	Hogst	Skog	Dyrket mark	Gress
Hogst	19	0	1	0
Skog	0	19	1	0
Dyrket mark	4	0	16	0
Gress	0	8	12	0

Tabell A.3: Forvirringsmatrise for klassifisering av valideringsflater for test med bånd 04.

Bånd 05

Type	Klassifisert som			
	Hogst	Skog	Dyrket mark	Gress
Hogst	19	0	1	0
Skog	0	19	1	0
Dyrket mark	2	0	17	1
Gress	0	7	13	0

Tabell A.4: Forvirringsmatrise for klassifisering av valideringsflater for test med bånd 05.

Bånd 06

Type	Klassifisert som			
	Hogst	Skog	Dyrket mark	Gress
Hogst	13	6	1	0
Skog	6	14	0	0
Dyrket mark	0	0	20	0
Gress	6	1	11	2

Tabell A.5: Forvirringsmatrise for klassifisering av valideringsflater for test med bånd 06.

Bånd 07

Type	Klassifisert som			
	Hogst	Skog	Dyrket mark	Gress
Hogst	13	7	0	0
Skog	7	13	0	0
Dyrket mark	0	0	20	0
Gress	5	3	12	0

Tabell A.6: Forvirringsmatrise for klassifisering av valideringsflater for test med bånd 07.

Bånd 08

Type	Klassifisert som			
	Hogst	Skog	Dyrket mark	Gress
Hogst	12	8	0	0
Skog	7	13	0	0
Dyrket mark	0	0	20	0
Gress	3	2	15	0

Tabell A.7: Forvirringsmatrise for klassifisering av valideringsflater for test med bånd 08.

Bånd 8A

Type	Klassifisert som			
	Hogst	Skog	Dyrket mark	Gress
Hogst	14	5	1	0
Skog	5	15	0	0
Dyrket mark	0	0	20	0
Gress	4	4	12	0

Tabell A.8: Forvirringsmatrise for klassifisering av valideringsflater for test med bånd 8A.

Bånd 11

Type	Klassifisert som			
	Hogst	Skog	Dyrket mark	Gress
Hogst	15	0	5	0
Skog	0	19	1	0
Dyrket mark	1	0	19	0
Gress	0	8	11	1

Tabell A.9: Forvirringsmatrise for klassifisering av valideringsflater for test med bånd 11.

Bånd 12

Type	Klassifisert som			
	Hogst	Skog	Dyrket mark	Gress
Hogst	17	0	3	0
Skog	0	19	0	1
Dyrket mark	1	0	19	0
Gress	0	10	9	1

Tabell A.10: Forvirringsmatrise for klassifisering av valideringsflater for test med bånd 12.

Vedlegg B – Python script for nedlastning og bearbeiding av Sentinel-2 data

```
# -*- coding: utf-8 -*-

__author__ = 'Jonas Njøten Sletmo'
__email__ = 'josletmo@nmbu.no'

from sentinelsat import SentinelAPI, read_geojson, geojson_to_wkt
from collections import OrderedDict
from glob import glob
import zipfile
import os
from osgeo import gdal
import matplotlib.pyplot as plt
import pandas as pd
import geopandas as gpd
import numpy as np
import rasterio
from rasterstats import zonal_stats
api = SentinelAPI('josletmo', 'jns93ONE',
                  'https://colhub.met.no/#/home')

def download(tiles, date, platformname='Sentinel-2', producttype='S2MSI1C',
             cloudcoverpercentage=(0, 10), save_dir='sentinel_data'):
    """
    Function for downloading Sentinel data by tile
    """
    query_kwargs = {'platformname': platformname,
                    'producttype': producttype,
                    'date': date,
                    'cloudcoverpercentage': cloudcoverpercentage,
                    'filename': 'NOT DTERRENGDATA'}

    products = OrderedDict()
    for tile in tiles:
        kw = query_kwargs.copy()
        kw['tileid'] = tile
        pp = api.query(**kw)
        products.update(pp)

    api.download_all(products, save_dir)


def unzip(zipfile_dir, unzip_dir='sentinel_unzip'):
    """
    Unzips all products in directory
    """
    zip_files = glob(zipfile_dir + '/*.zip')
    for file in zip_files:
        zip_ref = zipfile.ZipFile(file, 'r')
        zip_ref.extractall(unzip_dir)
        zip_ref.close()


def sen2cor(file_dir):
    """
    Runs sen2cor on all products in directory
    """
    sen2cor_path = 'C:/Sen2Cor-02.05.05-win64'
    s2a_1c = glob(file_dir + '/*MSIL1C*.SAFE')
    for _1c in s2a_1c:
        cmd = sen2cor_path + '/L2A_Process ' + _1c
        os.system(cmd)


def resize2min(input_rast, output):
    """
    resizes 20m products to 10m
    """
    in_ds = input_rast
    rast = gdal.Open (in_ds)
    in_band = rast.GetRasterBand (1)
    out_rows = in_band.YSize * 2
    out_columns = in_band.XSize * 2
    driver = gdal.GetDriverByName ('GTiff')
    out_ds = driver.Create (output, out_columns, out_rows, 1, gdal.GDT_UInt16)
    out_ds.SetProjection (rast.GetProjection ())
    geotransform = list (rast.GetGeoTransform ())
    geotransform[1] /= 2
    geotransform[5] /= 2
    out_ds.SetGeoTransform (geotransform)
    data = in_band.ReadAsArray (buf_xsize=out_columns,
                                buf_ysize=out_rows)  # Specify a larger buffer size when reading data
    out_band = out_ds.GetRasterBand (1)
    out_band.WriteArray (data)

    out_band.FlushCache ()
    out_band.ComputeStatistics (False)
    out_ds.BuildOverviews ('average', [2, 4, 8, 16, 32, 64])


def image_import(product, b, res):
    """
    imports image to array
    """
    input_folder = glob (product + '/GRANULE/*')
    input_path = glob (input_folder[0] + '/IMG_DATA/R10m/*' + b + '_' + res + '*.jp2')[0]
    input_rast = gdal.Open (input_path)
```

```
    input_array = input_rast.ReadAsArray ()

    return input_array


def resize_all(path):
    """
    uses resize2min on all products in directory
    """
    products = glob(path + '/*MSIL2A*.SAFE')
    num_products = len(products)
    product_num = 0
    for product in products:
        product_num += 1
        print('Product {0} of {1}'.format(product_num, num_products))
        input_folder = glob(glob(product +
                                 '/GRANULE/*')[0] + '/IMG_DATA/R20m/*.jp2')
        for _input in input_folder:
            input_name = _input
            output_name = input_name.replace('20m', '10m')
            if not os.path.isfile(output_name):
                print('Creating {0}'.format(output_name))
                resize2min(input_name, output_name)
            else:
                print('{0} already exists'.format(output_name))


def import_all(path):
    """
    uses import_image on all products un directory
    """
    products = glob(path + '/*MSIL2A*.SAFE')
    bands = ['B02', 'B03', 'B04', 'B05', 'B06', 'B07', 'B08', 'B8A', 'B11', 'B12']
    products_allbands = {}
    for product in products:
        for band in bands:
            globals()[band] = image_import(product, band, '10')


if __name__ == '__main__':
    download(['32VNN'], ('20180501', '20180831'))
    download(['32VNN'], ('20170501', '20170831'))
    unzip('sentinel_data')
    sen2cor('sentinel_unzip')
    resize_all('sentinel_unzip')
```

Vedlegg C – Python script for Keras modell kjørt på datasett med data fra 2018 og 2017

```
import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.regularizers import l2
from keras import optimizers
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.utils import shuffle


seed = 3
np.random.seed(seed)
i = 0
datasets = (['(B02, max, 2018)', '(B02, mean, 2018)', '(B02, min, 2018)', '(B03, max, 2018)', '(B03, mean, 2018)',
             '(B03, min, 2018)', '(B04, max, 2018)', '(B04, mean, 2018)', '(B04, min, 2018)', '(B05, max, 2018)',
             '(B05, mean, 2018)', '(B05, min, 2018)', '(B06, max, 2018)', '(B06, mean, 2018)', '(B06, min, 2018)',
             '(B07, max, 2018)', '(B07, mean, 2018)',  '(B07, min, 2018)', '(B08, max, 2018)', '(B08, mean, 2018)',
             '(B08, min, 2018)', '(B11, max, 2018)', '(B11, mean, 2018)', '(B11, min, 2018)', '(B12, max, 2018)',
             '(B12, mean, 2018)', '(B12, min, 2018)', '(B8A, max, 2018)', '(B8A, mean, 2018)', '(B8A, min, 2018)',
             '(B02, max, 2017)', '(B02, mean, 2017)', '(B02, min, 2017)', '(B03, max, 2017)', '(B03, mean, 2017)',
             '(B03, min, 2017)', '(B04, max, 2017)', '(B04, mean, 2017)', '(B04, min, 2017)', '(B05, max, 2017)',
             '(B05, mean, 2017)', '(B05, min, 2017)', '(B06, max, 2017)', '(B06, mean, 2017)', '(B06, min, 2017)',
             '(B07, max, 2017)', '(B07, mean, 2017)',  '(B07, min, 2017)', '(B08, max, 2017)', '(B08, mean, 2017)',
             '(B08, min, 2017)', '(B11, max, 2017)', '(B11, mean, 2017)', '(B11, min, 2017)', '(B12, max, 2017)',
             '(B12, mean, 2017)', '(B12, min, 2017)', '(B8A, max, 2017)', '(B8A, mean, 2017)', '(B8A, min, 2017)'],
            ['(B02, max, 2018)', '(B02, mean, 2018)', '(B02, min, 2018)', '(B03, max, 2018)', '(B03, mean, 2018)',
             '(B03, min, 2018)', '(B04, max, 2018)', '(B04, mean, 2018)', '(B04, min, 2018)', '(B08, max, 2018)',
             '(B08, mean, 2018)', '(B08, min, 2018)', '(B02, max, 2017)', '(B02, mean, 2017)', '(B02, min, 2017)',
             '(B03, max, 2017)', '(B03, mean, 2017)', '(B03, min, 2017)', '(B04, max, 2017)', '(B04, mean, 2017)',
             '(B04, min, 2017)', '(B08, max, 2017)', '(B08, mean, 2017)', '(B08, min, 2017)'],
            ['(B05, max, 2018)', '(B05, mean, 2018)', '(B05, min, 2018)', '(B06, max, 2018)', '(B06, mean, 2018)',
             '(B06, min, 2018)', '(B07, max, 2018)', '(B07, mean, 2018)', '(B07, min, 2018)', '(B8A, max, 2018)',
             '(B8A, mean, 2018)', '(B8A, min, 2018)', '(B05, max, 2017)', '(B05, mean, 2017)', '(B05, min, 2017)',
             '(B06, max, 2017)', '(B06, mean, 2017)', '(B06, min, 2017)', '(B07, max, 2017)', '(B07, mean, 2017)',
             '(B07, min, 2017)', '(B8A, max, 2017)', '(B8A, mean, 2017)', '(B8A, min, 2017)'],
            ['(B02, max, 2018)', '(B02, mean, 2018)', '(B02, min, 2018)',
             '(B02, max, 2017)', '(B02, mean, 2017)', '(B02, min, 2017)'],
            ['(B03, max, 2018)', '(B03, mean, 2018)', '(B03, min, 2018)',
             '(B03, max, 2017)', '(B03, mean, 2017)', '(B03, min, 2017)'],
            ['(B04, max, 2018)', '(B04, mean, 2018)', '(B04, min, 2018)',
             '(B04, max, 2017)', '(B04, mean, 2017)', '(B04, min, 2017)'],
            ['(B05, max, 2018)', '(B05, mean, 2018)', '(B05, min, 2018)',
             '(B05, max, 2017)', '(B05, mean, 2017)', '(B05, min, 2017)'],
            ['(B06, max, 2018)', '(B06, mean, 2018)', '(B06, min, 2018)',
             '(B06, max, 2017)', '(B06, mean, 2017)', '(B06, min, 2017)'],
            ['(B07, max, 2018)', '(B07, mean, 2018)', '(B07, min, 2018)',
             '(B07, max, 2017)', '(B07, mean, 2017)', '(B07, min, 2017)'],
            ['(B08, max, 2018)', '(B08, mean, 2018)', '(B08, min, 2018)',
             '(B08, max, 2017)', '(B08, mean, 2017)', '(B08, min, 2017)'],
            ['(B8A, max, 2018)', '(B8A, mean, 2018)', '(B8A, min, 2018)',
             '(B8A, max, 2017)', '(B8A, mean, 2017)', '(B8A, min, 2017)'],
            ['(B11, max, 2018)', '(B11, mean, 2018)', '(B11, min, 2018)',
             '(B11, max, 2017)', '(B11, mean, 2017)', '(B11, min, 2017)'],
            ['(B12, max, 2018)', '(B12, mean, 2018)', '(B12, min, 2018)',
             '(B12, max, 2017)', '(B12, mean, 2017)', '(B12, min, 2017)'])
name_list = ['all', '10m', '20m', 'B02', 'B03', 'B04', 'B05', 'B06', 'B07', 'B08', 'B8A', 'B11', 'B12']

for data in datasets:
    filename = 'training_2018_2017.csv'
    dataset = pd.read_csv(filename)
    dataset = shuffle(dataset)
    Class_ID = dataset[['Class_ID']]
    X = dataset[data]

    Y = np.ravel(Class_ID)
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.20, random_state=42)
    X_train = preprocessing.scale(X_train)
    X_test = preprocessing.scale(X_test)

    input_num_units = len(data)
    hidden1_num_units = 200
    hidden2_num_units = 200
    hidden3_num_units = 200
    hidden4_num_units = 200
    output_num_units = 5

    model = Sequential([
        Dense(output_dim=hidden1_num_units, input_dim=input_num_units, kernel_regularizer=l2(0.0001),
              activation='relu'),
        Dropout(0.2),
        Dense(output_dim=hidden2_num_units, input_dim=hidden1_num_units, kernel_regularizer=l2(0.0001),
              activation='relu'),
        Dropout(0.2),
        Dense(output_dim=hidden3_num_units, input_dim=hidden2_num_units,  kernel_regularizer=l2(0.0001),
              activation='relu'),
        Dropout(0.2),
        Dense(output_dim=hidden4_num_units, input_dim=hidden3_num_units,  kernel_regularizer=l2(0.0001),
              activation='relu'),
        Dropout(0.2),
        Dense(output_dim=output_num_units, input_dim=hidden4_num_units, activation='softmax'), ])
    model.summary()

    sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
```

```
    model.compile(loss='sparse_categorical_crossentropy',
                  optimizer='sgd',
                  metrics=['accuracy'])
    history = model.fit(X_train,
                        Y_train,
                        epochs=60,
                        batch_size=100,
                        validation_split=0.2,
                        verbose=2,)

    print(history.history.keys())

    val = 'validation_2018_2017.csv'
    val_point = pd.read_csv(val)

    val_data = val_point[data]
    val_class = val_point[['id', 'type']]

    val_data = preprocessing.scale(val_data)

    validated = pd.DataFrame(model.predict_classes(val_data))

    validated_2 = pd.concat([val_class, validated], axis=1, join_axes=[val_class.index])

    # Rename predicted class column to Class_ID
    validated_2.columns.values[2] = 'Class_ID'
    validated_2.to_csv('results20182017_{0}.csv'.format(name_list[i]))
    i += 1
```

Vedlegg D – Python script for Keras modell kjørt på datasett med data fra 2018 og differanser

```
import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.regularizers import l2
from keras import optimizers
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.utils import shuffle

seed = 3
np.random.seed(seed)
i = 0
datasets = (['(B02, max, 2018)', '(B02, mean, 2018)', '(B02, min, 2018)', '(B03, max, 2018)', '(B03, mean, 2018)',
             '(B03, min, 2018)', '(B04, max, 2018)', '(B04, mean, 2018)', '(B04, min, 2018)', '(B05, max, 2018)',
             '(B05, mean, 2018)', '(B05, min, 2018)', '(B06, max, 2018)', '(B06, mean, 2018)', '(B06, min, 2018)',
             '(B07, max, 2018)', '(B07, mean, 2018)',  '(B07, min, 2018)', '(B08, max, 2018)', '(B08, mean, 2018)',
             '(B08, min, 2018)', '(B11, max, 2018)', '(B11, mean, 2018)', '(B11, min, 2018)', '(B12, max, 2018)',
             '(B12, mean, 2018)', '(B12, min, 2018)', '(B8A, max, 2018)', '(B8A, mean, 2018)', '(B8A, min, 2018)',
             '(B02, max, diff)', '(B02, mean, diff)', '(B02, min, diff)', '(B03, max, diff)', '(B03, mean, diff)',
             '(B03, min, diff)', '(B04, max, diff)', '(B04, mean, diff)', '(B04, min, diff)', '(B05, max, diff)',
             '(B05, mean, diff)', '(B05, min, diff)', '(B06, max, diff)', '(B06, mean, diff)', '(B06, min, diff)',
             '(B07, max, diff)', '(B07, mean, diff)',  '(B07, min, diff)', '(B08, max, diff)', '(B08, mean, diff)',
             '(B08, min, diff)', '(B11, max, diff)', '(B11, mean, diff)', '(B11, min, diff)', '(B12, max, diff)',
             '(B12, mean, diff)', '(B12, min, diff)', '(B8A, max, diff)', '(B8A, mean, diff)', '(B8A, min, diff)'],
            ['(B02, max, 2018)', '(B02, mean, 2018)', '(B02, min, 2018)', '(B03, max, 2018)', '(B03, mean, 2018)',
             '(B03, min, 2018)', '(B04, max, 2018)', '(B04, mean, 2018)', '(B04, min, 2018)', '(B08, max, 2018)',
             '(B08, mean, 2018)', '(B08, min, 2018)', '(B02, max, diff)', '(B02, mean, diff)', '(B02, min, diff)',
             '(B03, max, diff)', '(B03, mean, diff)', '(B03, min, diff)', '(B04, max, diff)', '(B04, mean, diff)',
             '(B04, min, diff)', '(B08, max, diff)', '(B08, mean, diff)', '(B08, min, diff)'],
            ['(B05, max, 2018)', '(B05, mean, 2018)', '(B05, min, 2018)', '(B06, max, 2018)', '(B06, mean, 2018)',
             '(B06, min, 2018)', '(B07, max, 2018)', '(B07, mean, 2018)', '(B07, min, 2018)', '(B8A, max, 2018)',
             '(B8A, mean, 2018)', '(B8A, min, 2018)', '(B05, max, diff)', '(B05, mean, diff)', '(B05, min, diff)',
             '(B06, max, diff)', '(B06, mean, diff)', '(B06, min, diff)', '(B07, max, diff)', '(B07, mean, diff)',
             '(B07, min, diff)', '(B8A, max, diff)', '(B8A, mean, diff)', '(B8A, min, diff)'],
            ['(B02, max, 2018)', '(B02, mean, 2018)', '(B02, min, 2018)',
             '(B02, max, diff)', '(B02, mean, diff)', '(B02, min, diff)'],
            ['(B03, max, 2018)', '(B03, mean, 2018)', '(B03, min, 2018)',
             '(B03, max, diff)', '(B03, mean, diff)', '(B03, min, diff)'],
            ['(B04, max, 2018)', '(B04, mean, 2018)', '(B04, min, 2018)',
             '(B04, max, diff)', '(B04, mean, diff)', '(B04, min, diff)'],
            ['(B05, max, 2018)', '(B05, mean, 2018)', '(B05, min, 2018)',
             '(B05, max, diff)', '(B05, mean, diff)', '(B05, min, diff)'],
            ['(B06, max, 2018)', '(B06, mean, 2018)', '(B06, min, 2018)',
             '(B06, max, diff)', '(B06, mean, diff)', '(B06, min, diff)'],
            ['(B07, max, 2018)', '(B07, mean, 2018)', '(B07, min, 2018)',
             '(B07, max, diff)', '(B07, mean, diff)', '(B07, min, diff)'],
            ['(B08, max, 2018)', '(B08, mean, 2018)', '(B08, min, 2018)',
             '(B08, max, diff)', '(B08, mean, diff)', '(B08, min, diff)'],
            ['(B8A, max, 2018)', '(B8A, mean, 2018)', '(B8A, min, 2018)',
             '(B8A, max, diff)', '(B8A, mean, diff)', '(B8A, min, diff)'],
            ['(B11, max, 2018)', '(B11, mean, 2018)', '(B11, min, 2018)',
             '(B11, max, diff)', '(B11, mean, diff)', '(B11, min, diff)'],
            ['(B12, max, 2018)', '(B12, mean, 2018)', '(B12, min, 2018)',
             '(B12, max, diff)', '(B12, mean, diff)', '(B12, min, diff)'])
name_list = ['all', '10m', '20m', 'B02', 'B03', 'B04', 'B05', 'B06', 'B07', 'B08', 'B8A', 'B11', 'B12']

for data in datasets:
    filename = 'training_2018_diff.csv'
    dataset = pd.read_csv(filename)
    dataset = shuffle(dataset)
    Class_ID = dataset[['Class_ID']]
    X = dataset[data]

    Y = np.ravel(Class_ID)
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.20, random_state=42)
    X_train = preprocessing.scale(X_train)
    X_test = preprocessing.scale(X_test)

    input_num_units = len(data)
    hidden1_num_units = 200
    hidden2_num_units = 200
    hidden3_num_units = 200
    hidden4_num_units = 200
    output_num_units = 5

    model = Sequential([
        Dense(output_dim=hidden1_num_units, input_dim=input_num_units, kernel_regularizer=l2(0.0001),
              activation='relu'),
        Dropout(0.2),
        Dense(output_dim=hidden2_num_units, input_dim=hidden1_num_units, kernel_regularizer=l2(0.0001),
              activation='relu'),
        Dropout(0.2),
        Dense(output_dim=hidden3_num_units, input_dim=hidden2_num_units,  kernel_regularizer=l2(0.0001),
              activation='relu'),
        Dropout(0.2),
        Dense(output_dim=hidden4_num_units, input_dim=hidden3_num_units,  kernel_regularizer=l2(0.0001),
              activation='relu'),
        Dropout(0.2),
        Dense(output_dim=output_num_units, input_dim=hidden4_num_units, activation='softmax'), ])
    model.summary()

    sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
    model.compile(loss='sparse_categorical_crossentropy',
```

```
                optimizer='sgd',
                metrics=['accuracy'])
    history = model.fit(X_train,
                        Y_train,
                        epochs=60,
                        batch_size=100,
                        validation_split=0.2,
                        verbose=2,)

    print(history.history.keys())

    val = 'validation_2018_diff.csv'
    val_point = pd.read_csv(val)

    val_data = val_point[data]
    val_class = val_point[['id', 'type']]

    val_data = preprocessing.scale(val_data)

    validated = pd.DataFrame(model.predict_classes(val_data))

    validated_2 = pd.concat([val_class, validated], axis=1, join_axes=[val_class.index])

    # Rename predicted class column to Class_ID
    validated_2.columns.values[2] = 'Class_ID'
    validated_2.to_csv('results2018diff_{0}.csv'.format(name_list[i]))
    i += 1
```

Vedlegg E – Python script for Keras modell kjørt på datasett med kun differansedata

```
import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.regularizers import l2
from keras import optimizers
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.utils import shuffle

seed = 3
np.random.seed(seed)
i = 0
datasets = (['(B02, max, diff)', '(B02, mean, diff)', '(B02, min, diff)', '(B03, max, diff)', '(B03, mean, diff)',
             '(B03, min, diff)', '(B04, max, diff)', '(B04, mean, diff)', '(B04, min, diff)', '(B05, max, diff)',
             '(B05, mean, diff)', '(B05, min, diff)', '(B06, max, diff)', '(B06, mean, diff)', '(B06, min, diff)',
             '(B07, max, diff)', '(B07, mean, diff)',  '(B07, min, diff)', '(B08, max, diff)', '(B08, mean, diff)',
             '(B08, min, diff)', '(B11, max, diff)', '(B11, mean, diff)', '(B11, min, diff)', '(B12, max, diff)',
             '(B12, mean, diff)', '(B12, min, diff)', '(B8A, max, diff)', '(B8A, mean, diff)', '(B8A, min, diff)'],
            ['(B02, max, diff)', '(B02, mean, diff)', '(B02, min, diff)',
             '(B03, max, diff)', '(B03, mean, diff)', '(B03, min, diff)', '(B04, max, diff)', '(B04, mean, diff)',
             '(B04, min, diff)', '(B08, max, diff)', '(B08, mean, diff)', '(B08, min, diff)'],
            ['(B05, max, diff)', '(B05, mean, diff)', '(B05, min, diff)',
             '(B06, max, diff)', '(B06, mean, diff)', '(B06, min, diff)', '(B07, max, diff)', '(B07, mean, diff)',
             '(B07, min, diff)', '(B8A, max, diff)', '(B8A, mean, diff)', '(B8A, min, diff)'],
            ['(B02, max, diff)', '(B02, mean, diff)', '(B02, min, diff)'],
            ['(B03, max, diff)', '(B03, mean, diff)', '(B03, min, diff)'],
            ['(B04, max, diff)', '(B04, mean, diff)', '(B04, min, diff)'],
            ['(B05, max, diff)', '(B05, mean, diff)', '(B05, min, diff)'],
            ['(B06, max, diff)', '(B06, mean, diff)', '(B06, min, diff)'],
            ['(B07, max, diff)', '(B07, mean, diff)', '(B07, min, diff)'],
            ['(B08, max, diff)', '(B08, mean, diff)', '(B08, min, diff)'],
            ['(B8A, max, diff)', '(B8A, mean, diff)', '(B8A, min, diff)'],
            ['(B11, max, diff)', '(B11, mean, diff)', '(B11, min, diff)'],
            ['(B12, max, diff)', '(B12, mean, diff)', '(B12, min, diff)'])
name_list = ['all', '10m', '20m', 'B02', 'B03', 'B04', 'B05', 'B06', 'B07', 'B08', 'B8A', 'B11', 'B12']

for data in datasets:
    filename = 'training_2018-2017.csv'
    dataset = pd.read_csv(filename)
    dataset = shuffle(dataset)
    Class_ID = dataset[['Class_ID']]
    X = dataset[data]

    Y = np.ravel(Class_ID)
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.20, random_state=42)
    X_train = preprocessing.scale(X_train)
    X_test = preprocessing.scale(X_test)

    input_num_units = len(data)
    hidden1_num_units = 200
    hidden2_num_units = 200
    hidden3_num_units = 200
    hidden4_num_units = 200
    output_num_units = 5

    model = Sequential([
        Dense(output_dim=hidden1_num_units, input_dim=input_num_units, kernel_regularizer=l2(0.0001),
              activation='relu'),
        Dropout(0.2),
        Dense(output_dim=hidden2_num_units, input_dim=hidden1_num_units, kernel_regularizer=l2(0.0001),
              activation='relu'),
        Dropout(0.2),
        Dense(output_dim=hidden3_num_units, input_dim=hidden2_num_units,  kernel_regularizer=l2(0.0001),
              activation='relu'),
        Dropout(0.2),
        Dense(output_dim=hidden4_num_units, input_dim=hidden3_num_units,  kernel_regularizer=l2(0.0001),
              activation='relu'),
        Dropout(0.2),
        Dense(output_dim=output_num_units, input_dim=hidden4_num_units, activation='softmax'), ])
    model.summary()

    sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
    model.compile(loss='sparse_categorical_crossentropy',
                  optimizer='sgd',
                  metrics=['accuracy'])
    history = model.fit(X_train,
                        Y_train,
                        epochs=60,
                        batch_size=100,
                        validation_split=0.2,
                        verbose=2,)

    print(history.history.keys())

    val = 'validation_2018-2017.csv'
    val_point = pd.read_csv(val)

    val_data = val_point[data]
    val_class = val_point[['id', 'type']]

    val_data = preprocessing.scale(val_data)
```

```
    validated = pd.DataFrame(model.predict_classes(val_data))

    validated_2 = pd.concat([val_class, validated], axis=1, join_axes=[val_class.index])

    # Rename predicted class column to Class_ID
    validated_2.columns.values[2] = 'Class_ID'
    validated_2.to_csv('resultsdiff_{0}.csv'.format(name_list[i]))
    i += 1
```

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

Postboks 5003
NO-1432 Ås
Norway