Norges miljø- og biovitenskapelige universitet

Masteroppgave 2021 30 stp
Fakultet for realfag og teknologi

Segmentering av hode- og halskreft i PET/CT-bilder ved bruk av dype nevrale nettverk

Segmentation of head and neck cancer in PET/CT images using deep neural networks

Malene Elise Gjengedal
Miljøfysikk og fornybar energi

Forord

Denne oppgaven markerer min avslutning på en femårig mastergrad i Miljøfysikk og fornybar energi, og er skrevet ved Fakultet for realfag og teknologi ved Norges miljø- og biovitenskapelige universitet (NMBU) våren 2021.

Jeg vil først og fremst rette en stor takk til min hovedveileder Professor Cecilia Marie Futsæther, for svært god veiledning gjennom hele skriveprosessen. Jeg er veldig takknemlig for alle de grundige og konstruktive tilbakemeldingene, og den fantastiske oppfølgingen jeg har fått.

Takk til førsteamanuensis Oliver Tomic og gruppen til Professor Kathrine Røe Redalen ved NTNU for interessante møter og spennende diskusjoner, og til Professor Eirik Malinen ved Universitetet i Oslo for å ha bidratt med datasettet brukt i denne oppgaven.

Videre vil jeg takke Ph.d-stipendiat Bao Ngoc Huynh for veiledning i bruk av rammeverket *deoxys*, og for å ha programmert mye av koden som ble brukt i denne masteroppgaven. Jeg vil også takke Ph.d-stipendiat Aurora Rosvoll Grøndahl for alltid å ha tatt seg tid til å hjelpe hvis jeg har stått fast. Begge har bidratt med gode svar på spørsmål og verdifulle tilbakemeldinger, som har vært til stor hjelp.

Til slutt vil jeg takke familie og venner, som har støttet og oppmuntret meg hele veien. En spesielt stor takk rettes til Maria Ødegaard og Sofie Roko Krogstie for tett samarbeid under masteren, og gjennom hele studiet. Tusen takk til alle mine medstudenter, og spesielt det kjære kollektivet mitt, for fem fantastisk fine år på NMBU. Det har vært en sann glede å dele studiehverdagen med dere.

Ås, 01.06.2021

_____

Malene Elise Gjengedal

Sammendrag

Bakgrunn

Manuell segmentering av krefttumorer er en tidkrevende prosess som kan føre til stor inter- og intravariabilitet. Den lange inntegningstiden og usikkerheten i inntegningene, kan påvirke forløpet og utfallet av pasientens behandling. Formålet med denne masteroppgaven er å optimalisere og evaluere en automatisk segmenteringsmodell kalt 2D Dense-Net til segmentering av hode- og halskreft fra PET/CT-bilder. Modellen sammenlignes deretter med U-Net, som er mye brukt på området. Bruken av automatiske segmenteringsmodeller som støtteverktøy, kan potensielt frigi tid og bidra til nøyaktigere inntegninger som kan forbedre behandlingsforløpet til pasienten.

Metode

Fire forskjellige parametere ble justert for å optimalisere 2D Dense-Net i denne oppgaven. Det utgjorde 36 eksperimentmodeller, med forskjellige kombinasjoner av parameterne batchnormalisering, utelatelsesrate, læringsrate og antall filtre i første lag. Modellene ble trent, validert og testet ved hjelp av rammeverket *deoxys* på et datasett bestående av PET/CT-bilder fra 197 pasienter med hode- og halskreft fra Oslo universitetssykehus (OUS). Ytelsen ble evaluert ved ytelsesmålet Dice-score, og statistiske tester ble benyttet for å undersøke effekten av parameterne. Kombinasjonen av parameternivåene som bidro mest til økt Dice-score, ble definert som den beste modellen. Denne modellen ble videre testet med augmentering på validerings- og testsettet fra OUS, og på et eksternt testsett fra Maastro-klinikken i Nederland. Til slutt ble ytelsen til 2D Dense-Net sammenlignet med ytelsen til U-Net.

Resultat

Modellen som ble definert som den beste hadde batchnormalisering, en læringsrate på 10^{-4}, 32 filtre i første lag og 0,5 i utelatelsesrate. Den oppnådde en gjennomsnittlig Dice-score per pasient på 0,73 uten augmentering og 0,74 med augmentering på OUS-testsettet. Prestasjonen lå dermed på linje med U-Nettet på samme datasett. På Maastro-testsettet ble gjennomsnittlig Dice-score per pasient 0,64 både med og uten augmentering. Det var tydelig at augmentering hadde en positiv effekt på ytelsen ved bruk på OUS-datasettet, men på Maastro-datasettet var effekten marginal.

Konklusjon

Ved sammenligning av 2D Dense-Net og U-Net, var det tydelig at 2D Dense-Net har potensiale til å være like anvendelig eller bedre som automatisk segmenteringsverktøy for hode- og halskreft i PET/CT-bilder som U-Net. Selv om modellen viste en god prestasjonsevne, gjorde den fortsatt noen grove feilpredikeringer. Dette belyser viktigheten av at formålet til segmenteringsmodellen er å være et støtteverktøy, ikke en erstatning av manuell inntegning. Ved videre arbeid kan det være hensiktsmessig å undersøke flere nivåer av de allerede vurderte parameterne, samt andre parametere og interaksjonen mellom dem. Videre arbeid bør også innebære flere kjøringer for mer data til de statistiske testene, og et større og mer variert datasett til trening og validering.

Abstract

Purpose

Manual tumour segmentation is a time-demanding process that can lead to large inter and intra-observer variability. The time required and the uncertainty associated with the segmentation can affect the outcome of the patient's treatment. The purpose of this thesis is to optimise and evaluate an automatic delineation model known as 2D Dense-Net for segmentation of head and neck cancer from PET/CT images. The model is then compared with U-Net, which is an often-used model in the field of semantic segmentation. The use of auto-delineation models could potentially save time and contribute to more accurate delineations, which can improve treatment outcome.

Method

Four different parameters were adjusted in this thesis to optimize the 2D Dense-Net. This amounted to 36 experiments with different combinations of the parameters batch normalization, dropout rate, learning rate and the number of filters in the first layer. The models were trained, validated, and tested using the framework *deoxys*, on a dataset consisting of PET/CT images from 197 patients with head and neck cancer from Oslo university hospital (OUS). The performance was evaluated with the metric Dice score, and statistical tests were used to examine the effect of the parameters. The combination of the parameter levels that contributed the most to increased Dice score was defined as the best model. This model was further tested with augmentation on the validation and test set from OUS and on an external test set from the Maastro clinic in the Netherlands. Finally, 2D Dense-Net's performance was compared to the performance of the U-Net on the same patient cohort.

Results

The model defined as the best model included batch normalization, 10^{-4} as learning rate, 32 filters in the first layer and a dropout rate of 0.5. The model achieved a mean Dice score per patient of 0.73 without augmentation and 0.74 with augmentation on the OUS test set. The performance was similar to U-Net results obtained on the same test set. The mean Dice score per patient on the Maastro test set was 0.64, both with and without augmentation. Augmentation had a positive effect when used on the OUS dataset, but the effect was marginal on the Maastro data.

Conclusion

Comparing 2D Dense-Net with U-Net, it is clear that 2D Dense-Net has the potential to be just as useful, or even better than U-Net for auto-delineation of head and neck cancer in PET/CT images. Even though the model showed an acceptable performance, it still made some significant errors. This highlights the importance of the purpose of the segmentation model, which is to be a support tool and not a replacement of the manual delineation. It is recommended to examine more levels of the already assessed parameters, as well as other parameters and the interaction between them. Further research should also involve more runs of the models to achieve more data for the statistical tests, and a larger and more varied dataset.

Innholdsfortegnelse

Lister over forkortelser

Adam:	Adaptive Moment Estimation
AI:	Artificial intelligence / Kunstig intelligens
ANOVA:	Analysis of Variance
CIGENE:	Centre for Interactive Genetics
CNN:	Convolutional Neural Network / Konvolusjonsnettverk
CPU:	Central Processing Unit
CT:	Computertomografi
CTV:	Clinical Target Volume
DBMS:	Database Management System / Databasehåndteringssystem
DNA:	Deoksyribonukleinsyre
EU:	Den europeiske union
FCN:	Fully Convolutional Network / Fullt konvolusjonsnettverk
FDG:	Fluorodeoksyglukose
FN:	Falsk negativ
FP:	Falsk positiv
GPU:	Graphics Processing Unit
GTV:	Gross Tumour Volume
GTV-N:	Gross Tumour Volume - Node
GTV-T:	Gross Tumour Volume - Tumour
HD:	Hausdorff-distanse
HD_{95}:	95. persentil Hausdorff-distanse
HDF5:	Hierarchical Data Format version 5
HECKTOR:	HEad and neCK TumOR
HNC:	Head and Neck Cancer
HPV:	Humant papillomavirus
HU:	Houndsfield Unit
JSON:	JavaScript Object Notation
LOR:	Line of Response

Lr:	Læringsraten
MICCAI:	Medical Image Computation and Computer Assisted Intervention
MR:	Magnetisk resonans
MSD:	Median Surface Distance / Median overflatedistanse
MSE:	Mean Squared Error / Midlere kvadratisk feil
NMBU:	Norges miljø- og biovitenskapelige universitet
OUS:	Oslo universitetssykehus
PET:	Positronemisjonstomografi
PPV:	Positive Predicted Value/ Presisjon
QQ-plott:	Quantile-Quantile plot
RAM:	Random Access Memory
REK:	Den regionale komitéen for medisinsk og helsefaglig forskning
ReLU:	Rectified Linear Unit
SSH:	Secure Shell
SUV:	Standardized Uptake Value
Tanh:	Tangens hyperbolikus
TN:	True Negative/ Sann negativ
TNM:	Tumor, node, metastase
TNR:	True Negative Rate/ Spesifisitet
TP:	True Positive/ Sann positiv
TPR:	True Positive Rate/ Sensitivitet
WL:	Window Level / Vindussentrum
WW:	Window Width / Vindusbredden

Kapittel 1: Innledning

1.1 Motivasjon

Kreft er samlebegrepet på en gruppe sykdommer med ukontrollert celledeling, der cellene ikke lenger utfører oppgavene som friske celler ville gjort [1]. Det er den ledende årsaken til død i verden, og tok i 2020 livet av 10 millioner mennesker [2]. I 2018 døde 11 049 av kreft i Norge, og i 2019 fikk 34 979 mennesker påvist kreft [1]. Hode- og halskreft («head and neck cancer», HNC) utgjør ca. 2,5 % av alle nye krefttilfeller i Norge [3, 4], og man ser en høyere forekomst av denne krefttypen blant menn enn kvinner [5].

Behandlingsformen av hode- og halskreft er avhengig av tumortype, utbredelse og plassering [6]. I noen tilfeller kan kirurgi bli for omfattende, da er strålebehandling foretrukket [7]. Ved strålebehandling er det kritisk at lokasjonen til kreftsvulsten er kjent, slik at stråledosen konsentreres til det syke vevet og ødelegger minst mulig friskt vev. Den nåværende prosedyren er at omrisset av kreftsvulsten og de nærliggende lymfeknutene tegnes inn på medisinske bilder av onkologer og radiologer. Dette er en svært tidkrevende prosess, som kan resultere i stor intra- og intervariabilitet [8, 9]. Det vil si variasjon mellom inntegningene til forskjellige leger, og variasjon hos en lege som tegner inn samme svulst flere ganger [8].

Den lange inntegningstiden, og usikkerheten i inntegningene, kan påvirke forløpet og utfallet av behandlingen til pasienten [9]. En feilaktig inntegning av svulsten vil kunne føre til en systematisk feil gjennom hele strålebehandlingen. Underdosering på visse deler av svulstvolumet vil dermed kunne gjøre det vanskelig å få svulsten under kontroll [9]. Dette kan påvirke behandlingstiden, og livskvaliteten til pasienten etter behandling. Den manuelle inntegningen kan ses på som en flaskehals innen planleggingen av strålebehandlingen. Derfor er en metode for å effektivisere og forbedre inntegningskvaliteten ettertraktet [10].

1.2 Maskinlæring

Nyere forskning har vist at maskinlæring har potensialet til å bidra positivt på dette feltet [10-12]. Maskinlæring er et underfelt innen kunstig intelligens, som handler om å la maskiner lære mønstre og sammenhenger ved hjelp av store mengder data [13]. Dette har blitt et mer relevant fagområde i løpet av de siste årene, ettersom det har vært en stor fremgang innen beregningskraft og rask datalagring. Denne fremgangen gjør det lettere å analysere store datamengder [14]. Store selskaper, som Huawei og Apple, bruker blant annet maskinlæring til segmentering av objekter ved bildebehandlingsoppgaver [15].

Et av målene med maskinlæringsforskning innen helsesektoren er å lage modeller som kan segmentere ut kreftsvulster på medisinske bilder, og bli brukt som et støtteverktøy for radiologer og onkologer som driver med kreftsvulstinntegning. Dette vil kunne minke flaskehals-effekten av den manuelle inntegningen, og effektivisere og forbedre svulstinntegningene. Forskning rundt automatisk svulstinntegning er omdiskutert, grunnet frykt for feilinntegninger og at maskiner skal overta legenes arbeidsoppgaver [16]. Dette er dog ikke målet med denne forskningen. Hovedmålet er å kunne utvikle et støtteverktøy som vil frigi tid slik at legene kan bruke mer tid på andre viktige oppgaver. Målet er altså å bruke dyplæringsmodeller som supplement til legene, ikke som en erstatning av dem.

1.3 Tidligere arbeid

Dype konvolusjonelle nevrale nettverk er en underkategori innen maskinlæring, og brukes ofte til segmentering av objekter i bilder. Tidligere forsking har vist at flere av disse nettverkene gir gode resultater innen segmentering av kreftsvulster i hode- og halsregionen [10-12, 17]. Lin et al. [17] konkluderte med at konvolusjonelle nevrale nettverk som hjelpemiddel til svulstinntegning kan redusere intra- og intervariabiliteten hos radiologer og onkologer betraktelig. I tillegg viste hjelpemiddelet seg å være tidsbesparende for legene [17]. Guo et al. [11] undersøkte bruken av et 3D Dense-Net til segmentering av hode- og halskreft, og konkluderte med at denne arkitekturen har et stort potensial som hjelpemiddel innen svulstsegmentering. I 2020 ble HECKTOR-utfordringen («Head and neCK TumOR») arrangert på den internasjonale konferansen MICCAI («Medical Image Computation and Computer Assisted Intervention») [18]. Oppgaven handlet om å utnytte bi-modal informasjon i sammenheng med segmentering av hode- og halskreft [18]. Utfordringen viste at interessen for automatisk segmentering av hode- og halskreft var høy, og et stort antall av bidragene var U-Net-modeller.

Moe et al. [10] og Groendahl et al. [12] testet et U-Net på data fra en pasientkohort med hode- og halskreft fra Oslo Universitetssykehus (OUS), og oppnådde lovende resultater. Bao Ngoc Huynh utforsket samme arkitektur og pasientgruppe i sin masteroppgave [19] og utviklet et rammeverk kalt *deoxys* for automatisk svulstinntegning. *Deoxys* gir brukeren mulighet til å opprette og trene dype nevrale nettverk [19]. Dette rammeverket kan benyttes til å utvikle modellarkitekturer og trene modeller til segmentering av svulster på medisinske bilder [19].

1.4 Mål med oppgaven

Målet med denne oppgaven var å trene, optimalisere og evaluere dyplæringsmodellen 2D Dense-Net, utviklet av Guo et al. [11], for å undersøke om den kan brukes som hjelpemiddel til svulstinntegning. 2D Dense-Net-arkitekturen var implementert i rammeverket *deoxys*, som ble brukt i denne masteroppgaven. PET/CT-bilder av pasienter med hode- og halskreft, behandlet ved Oslo Universitetssykehus, ble brukt under modelltrening og testing. Et tilsvarende datasett fra Maastro-klinikken i Nederland ble brukt som ekstern validering av modellen.

Modellen ble optimalisert ved justering av fire ulike parametere – batchnormalisering («batch normalization»), utelatelsesraten («dropout»), læringsraten og antall filtre. Rammeverket *deoxys* ble brukt til trening og testing av modellen med forskjellige kombinasjoner av de nevnte parameterne. Ytelsen ble til slutt sammenlignet med prestasjonen til U-Net-modellen testet av Moe et al. [10], Groendahl et al. [12] og Huynh [19] på samme pasientkohort.

1.5 Organisering av oppgaven

I denne masteroppgaven vil Kapittel 2 gi grunnleggende informasjon om kreft og medisinsk avbildning ved hjelp av PET og CT, samt informasjon om kunstig intelligens. Under kunstig intelligens inkluderes undergruppen dyp læring, relevante dyplæringsarkitekturer og mål for ytelse. Kapittel 3 tar for seg datasettet og rammeverket *deoxys*. I tillegg forklares parameterne som justeres, oppsettet av eksperimenter, og statistikken som brukes til vurdering av eksperimentresultatene. I Kapittel 4 presenteres resultatene fra eksperimentene. Resultatene diskuteres i Kapittel 5, og i Kapittel 6 framlegges konklusjonene.

Kapittel 2: Teori

2.1 Kreft

2.1.1 Kreft generelt

Som tidligere nevnt, betegnes kreft som ukontrollert celledeling [1]. Naturlig fornyer cellene seg hele tiden, og dette skjer ved at en celle dobler sitt DNA, og deler seg i to. Ved mutasjoner i DNA-molekylet kan cellene begynne å dele seg ukontrollert. Det vil da etter hvert oppstå en opphopning av disse cellene, og slik dannes en kreftsvulst [20]. Kreften kan spres i kroppen ved at kreftcellene løsrives og fraktes rundt ved hjelp av blod- og lymfebaner [1, 20]. De ulike krefttypene opptrer forskjellig fra hverandre, og sykdomsforløpet varierer av type kreft [1].

2.1.2 Hode- og halskreft

Det finnes mange ulike diagnoser under hode- og halskreft, men fellesbetegnelsen er ondartede svulster i nese og bihuler, leppe, munnhule, svelg, strupehode eller spyttkjertler [21]. Som tidligere nevnt, utgjør hode- og halskreft cirka 2,5 % av alle nye krefttilfeller i Norge [3]. Per dags dato er overlevelsesraten for hode- og halskreft etter fem år 68,6 % for menn og 75,7 % for kvinner [5]. De viktigste risikofaktorene for hode- og halskreft er alkohol og tobakk (gjelder særlig for munnhulekreft) [22], Humant papillomavirus (HPV), samt fedme og eksponering for kjemiske stoffer [5].

2.1.3 Behandling

For pasienter med hode- og halskreft vil behandlingen være cellegift, operasjon, strålebehandling eller en kombinasjon av disse [21]. Cellegift innebærer medisiner som virker hemmende på celledelingen [23], mens ved en operasjon fjernes svulsten fysisk ved kirurgi [21]. Ved strålebehandling får pasienten en stråledose med stor nok energi til å drepe kreftcellene, men samtidig må den gi minst mulig skade til det friske vevet rundt [21]. Det er derfor avgjørende at kreftsvulsten lokaliseres så nøyaktig som mulig før behandling for å begrense stråledosen til vevet rundt svulsten. Ved lokalisering av svulsten brukes begrepene *gross tumor volume* (GTV) og *clinical target volume* (CTV). GTV definerer tumorens omfang, mens CTV inkluderer spredning og andre mistenkte berørte områder, som for eksempel lymfeknuter, i tillegg til GTV [9]. Radiologer benytter seg av medisinske bilder fra blant annet PET/CT-skanning for å tegne inn disse lokasjonene.

2.2 Medisinske bilder

2.2.1 Introduksjon til medisinske bilder

I diagnostisk medisin er muligheten til å kunne ta bilder av forskjellige organer, vev og beinstrukturer svært nyttig [24]. Dette gir medisinsk personell mulighet til å vurdere den medisinske tilstanden til menneskekroppen, uten å utføre invasiv behandling først [25]. Effektiv og enkel billedtakning med god kvalitet, er derfor essensielt for å kunne diagnostisere og behandle pasienter.

2.2.2 Introduksjon til CT og PET

Allerede tidlig på 1900-tallet ble røntgen tatt i bruk til diagnostisering og forskning [24]. Senere kom computertomografi (CT) som baserer seg på konseptene bak røntgenteknologi, og positronemisjonstomografi (PET). Disse ble tatt i bruk på 1960-1970-tallet, og har blitt brukt hyppig i moderne tid på grunn av deres evne til å visualisere anatomien og metabolismen i menneskekroppen [24]. CT og PET er derfor viktige verktøy for diagnostisering og lokalisering av kreftsvulster den dag i dag.

2.3 Computertomografi

2.3.1 Kort om CT

Computertomografi (CT) er en diagnostisk bildetakningsprosedyre der røntgenstråling brukes for å visualisere et tverrsnitt av pasienten [26]. Røntgenstråling sendes inn i pasienten fra flere vinkler i samme plan, og transmisjonen av røntgenstrålene måles ved hjelp av en detektor på motsatt side. På denne måten kan den anatomiske informasjonen rekonstrueres digitalt, og man vil minimere hindringen en struktur utgjør for en annen slik at flere typer vev kan detekteres [26]. Computertomografi kan visualisere blodårer, mykt vev og bein.

2.3.2 Røntgen

Røntgenstråling er en blanding av bremsestråling og karakteristisk stråling, som dannes ved at en stråle med elektroner akselereres fra en katode mot en anode i et røntgenrør [26]. Bremsestråling dannes når elektronene bremses av de positivt ladde kjernene i anoden, og karakteristisk stråling dannes når elektronene som ble sparket ut av de innerste lagene i atomene i anoden blir erstattet med elektroner fra andre skall [26].

Når en røntgenstråle sendes gjennom et medium, vil intensiteten endres basert på attenuasjonsevnen til mediet. Intensiteten til strålen vil følge likning 2.1.

$$I = I_0 e^{\mu x} \tag{2.1}$$

Her er I sluttintensiteten til strålen, I_0 er startintensiteten, μ er attenueringskoeffisienten og x er tykkelsen til mediet strålen passerer gjennom. Attenueringskoeffisienten er et mål på transmisjonen av røntgenstråler i mediet, altså hvordan intensiteten til strålen svekkes på grunn av absorbering, spredning eller refleksjon [27, 28].

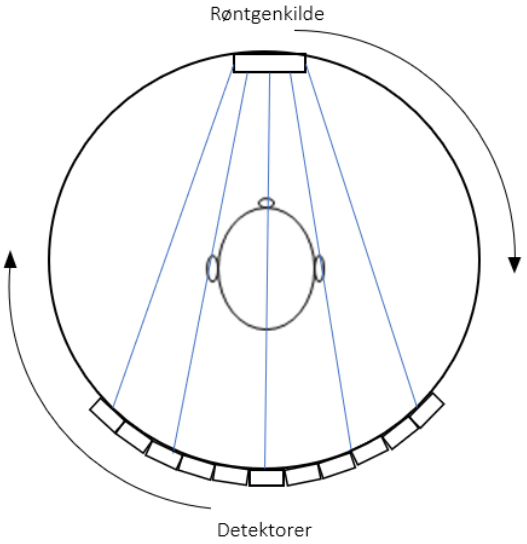
Måten røntgenstrålene blir absorbert eller spredt på, er ved fotoelektrisk absorpsjon eller comptonspredning [25]. Ved fotoelektrisk absorpsjon overføres all energi fra røntgenfotonet til ett av elektronene i et atom. Hvis energien til røntgenfotonet er stor nok, vil elektronet sparkes ut av banen sin og bevege seg fritt før det vanligvis fanges opp av et annet atom. Hullet som etterlates etter elektronet, fylles fort av et annet elektron. Energien som da frigis kalles karakteristisk stråling [25]. Ved høyere energi er sannsynligheten større for comptonspredning. Noe av energien vil da overføres til elektronet som blir sparket ut av banen sin, mens resten blir hos røntgenfotonet. På grunn av bevaring av energi vil røntgenfotonet fortsette videre i en annen retning enn utgangspunktet, med mindre energi og en annen bølgelengde [25]. Røntgenstrålene som ikke blir spredt ved comptonspredning eller absorbert ved fotoelektrisk absorpsjon, blir transmittert. De transmitterte strålene fanges opp av en detektor på motsatt side av pasienten [26]. Resultatet er en projeksjon av objektene som har absorbert røntgenstrålene på veien, der intensiteten vil variere basert på absorpsjonsevnen til mediet strålene har passert [25]. På bildet vises høy absorpsjon som lyse områder, mens høy transmisjon gir mørke [28].

2.3.3 Fra røntgen til CT

Med vanlig røntgen mangler man dybdeinformasjon. Ettersom objekter vil overlappe hverandre på bildet, er det umulig å si om et objekt ligger foran eller bak et annet langs røntgenbanen [25]. Dette utgjør en utfordring når vevet har varierende tetthet langs røntgenbanen. CT-skannere eksponerer pasienten med røntgenstråling fra flere forskjellige vinkler [26]. Dette gir et tverrsnitt av pasienten, og man vil kunne skille objekter og vev fra hverandre. Disse tverrsnittene kan legges sammen og rekonstruere en 3D-framstilling av pasientens anatomi [25]. Dette kan særlig være nyttig for områder med kompleks anatomi og varierende struktur, som hode- og halsregionen [28].

2.3.4 CT-maskinens oppbygning

CT-maskinen er formet som en ring, der pasienten plasseres på et bord som kan skyves inn i åpningen i midten [25]. Maskinen inneholder en roterende røntgenkilde som sender røntgenstråler gjennom pasienten i det transversale planet [26]. På motsatt side av røntgenkilden er det flere detektorer som måler strålefluksen gjennom pasienten. Kilden og detektorene beveger seg rundt pasienten synkront, vinkelrett på stråleretningen, og målinger blir gjort med jevne mellomrom langs banen, som vist i Figur 2.1 [26]. Etter en runde rundt et snitt flyttes kilden og detektorene til et nytt plan. Målingene blir lagret, og blir så matematisk rekonstruert til et tverrsnitt av pasienten [25].



Figur 2.1. Her vises CT-maskinens oppbygning, med røntgenkilde og detektorer som roterer rundt pasienten. Illustrasjonen er tegnet etter inspirasjon fra figur i Introduction to Physics in Modern Medicine av S. A. Kane og B. A. Gelman (2020) [25].

2.3.5 Rekonstruering av bildene

Rekonstrueringen av tverrsnittet baserer seg på prinsippet om at tettheten til mediet kan måles ved hjelp av attenueringskoeffisienten [29]. Konseptet handler om å beregne attenueringskoeffisienten langs forskjellige røntgenbaner tatt i forskjellige vinkler, som kalles projeksjoner [30]. Tverrsnittet av pasienten rekonstrueres basert på data fra disse projeksjonene. For å kunne rekonstruere disse dataene til et bilde, bruker maskinen matematiske algoritmer [29].

Algoritmene bygger på den matematiske oppdagelsen av at en todimensjonal funksjon bestemmes av projeksjonen av den i alle retninger. En samling av projeksjoner fra likt fordelte

vinkler kan da gi en rekonstruksjon av funksjonen [26]. På denne måten kan man få ut en bildematrise fra projeksjonene fra et snitt av pasienten. Hver rute i matrisen representerer et volumelement, også kalt voxel. Voxelens dybde avgjøres av tverrsnittets bredde, og hver voxel har en verdi basert på attenueringskoeffisienten til mediet i dette volumet [25, 29]. Den matematiske algoritmen som rekonstruerer projeksjonene kan deles inn i to generelle klasser: iterativ og analytisk [26].

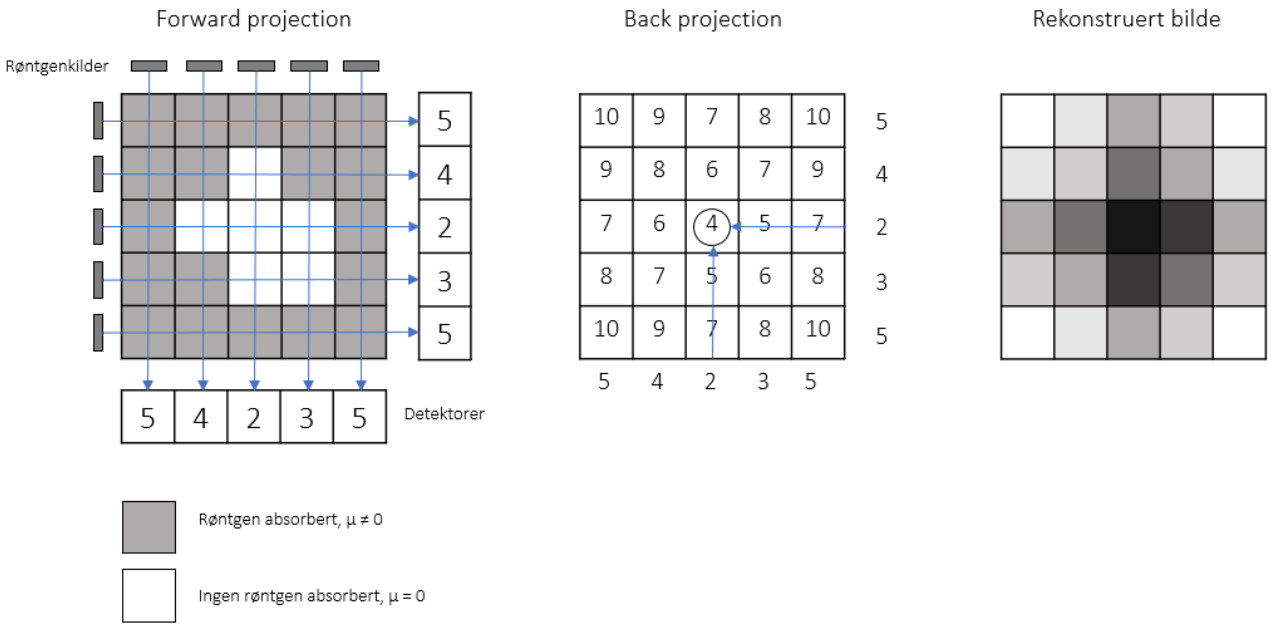
2.3.6 Iterativ algoritme

Den iterative algoritmen starter med en vilkårlig gjetning av pikslenes attenueringsverdi i det todimensjonale bildet. Projeksjonsdataene blir deretter beregnet, og sammenlignet med gjetningen. Justeringer gjøres helt til avvikene mellom predikerte og beregnede verdier er så små som mulig [26].

2.3.7 Analytisk algoritme

Den analytiske metoden rekonstruerer direkte fra projeksjonsdataen uten å sammenligne med en gjetning først. Dette er den mest anvendte teknikken for å konstruere et bilde fra projeksjonsdata [24]. Den analytiske metoden bruker en metode som kalles *back projection*.

Når pasienten blir skannet kalles det lagrede resultatet *forward projection* [25]. Intensiteten til hver stråle måles, og intensiteten i hvert punkt på denne linja kan regnes ut. Etter at *forward projection* er gjennomført fra forskjellige vinkler kan *back projection* starte [25]. Ved *back projection* regnes det ut en absorpsjonsverdi til alle punktene langs linjene fra hver detektor. Disse numrene kalles CT-numre, og er videre forklart i neste avsnitt. Resultatet blir da en matrise med absorpsjonsverdier, ofte visualisert som et gråskalabilde. Før *back projection* finner sted gjøres ofte en prosess kalt *filtering* for å unngå artefakter som for eksempel at jevne overflater blir taggete [25]. Metoden heter da *filtered back projection*, og er visualisert i Figur 2.2.



Figur 2.2. Figuren viser output fra forward projection og back projection, samt det rekonstruerte bildet. I det rekonstruerte bildet har svart verdien 4, mens hvit har verdien 10. Illustrasjonen er tegnet med inspirasjon fra figur i Introduction to Physics in Modern Medicine av S. A. Kane og B. A. Gelman (2020) [25].

2.3.8 CT-nummeret

CT-skannere representerer røntgenabsorbsjon ved en størrelse som kalles CT-numre [25]. CT-numre er definert som prosentforholdet mellom en voxels attenueringskoeffisient og attenueringskoeffisienten til vann. Nummeret måles i Hounsfield-enhet (HU), og er gitt i likning 2.2 [25].

$$CT - nummer = \frac{\mu_{vev} - \mu_{vann}}{\mu_{vann}} * 1000 \qquad (2.2)$$

μ_{vev} angir attenueringskoeffisienten til vevet, mens μ_{vann} er attenueringskoeffisienten til vann.

2.3.9 Kontrastmiddel

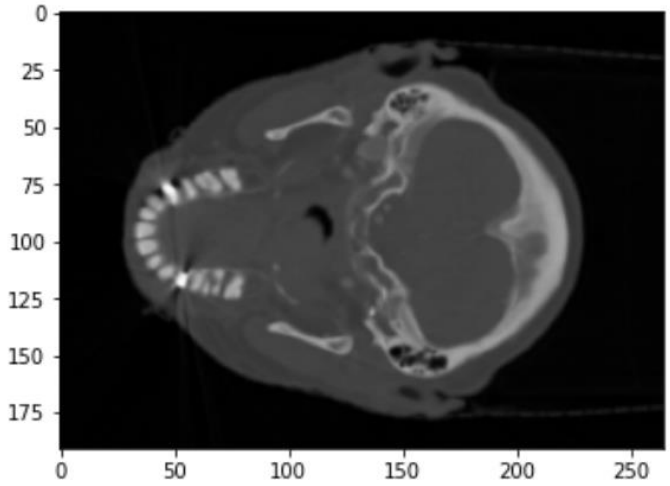
Ulike typer vev kan ha lignende røntgenabsorpsjon, og vil dermed se likt ut på røntgenbilder. Dette kan skape problemer med å skille de ulike vevstypene. For å unngå dette problemet kan kontrastmiddel brukes [25]. Kontrastmiddel består av stoffer med høy røntgenabsorberingsevne, som Barium og Jod. Kontrastmiddelets høye

røntgenabsorpsjonsevne vil skape en forsterking av kontraster, og anatomien i området som ble introdusert for kontrastmiddelet vil dermed komme tydelig fram på røntgenbildene [25].

2.3.10 Intensitetsvindu

Hvis et bilde inneholder et stort spenn av CT-numre, kan man bruke et intensitetsvindu («windowing») for å gjøre det enklere å forstå bildet [25]. Et intensitetsvindu lages ved at gråskalaen i CT-bilder manipuleres via CT-numrene [31]. Ved å endre dette, vil utseende til bildet forandres. Det kan for eksempel gjøres lysere eller få forsterkede kontraster. Det er to parametere som kan endres, vindusbredden («window width», WW) og vindussentrum («window level», WL). Endringer i vindusbredden justerer kontrasten i bildet, mens endringer i vindussentrum justerer lysstyrken [31].

Vindusbredden angir hvilke CT-numre som bildet kan inneholde. Et bredt vindu vil være nyttig å bruke der det er vev med mange forskjellige attenueringskonstanter, for eksempel i lungene der man har luft og mykt vev om hverandre. Et smalt vindu kan være nyttig der det ikke er en stor variasjon i attenueringskonstanter i vevet, som for eksempel i mykt vev [31]. Vindussentrum angir midtpunktet i vindusbredden. Et høyere midtpunkt vil gi høyere lysstyrke, og motsatt [31]. Figur 2.3 viser et CT-bilde der et intensitetsvindu er tatt i bruk.



Figur 2.3. CT-bilde tatt med et intensitetsvindu der WL er lik 70 HU og WW er lik 200 HU.

2.4 Positronemisjonstomografi

2.4.1 Kort om PET

Positronemisjonstomografi (PET) er en medisinsk avbildningsmetode som benytter positroners emittering fra radioaktive markører. Positronene som emitteres annihilerer med elektroner, og former to fotoner. Disse fotonene fanges opp av en ring med detektorer, og transformeres til et digitalt bilde [28].

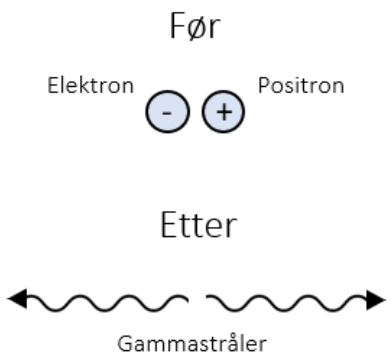
2.4.2 Kjernefysikken bak

En atomkjerne består av nøytroner og protoner, med fellesbetegnelsen nukleoner [28]. Antallet protoner kalles atomnummeret *Z* og angir hvilket kjemisk element nuklidet er. Antallet nøytroner og protoner til sammen er massetallet, og noen kombinasjoner av nøytroner og protoner kan føre til ustabile nuklider, som kalles radionuklider [28]. Radionuklider er radioaktive og vil desintegrere og frigi energi i form av stråling. Desintegreringen til et radionuklid blir ofte beskrevet med halveringstid, som er tiden det tar før halve mengden av stoffet har desintegrert [28].

Radionuklider som er ustabile grunnet overtall av protoner, kan desintegrere ved å sende ut et positron. Et positron er en elementærpartikkel som har like stor masse som et elektron. Et elektron har negativ ladning, mens et positron har en positiv ladning av samme størrelse [32] . Når et positron emitteres, vil det miste energi ved å ionisere og eksitere nærliggende atomer. Når det har mistet mye av energien sin, vil det interagere med en nærliggende antipartikkel, altså et elektron [28]. Positronet og elektronet vil da trekkes til hverandre fordi de har motsatte ladninger, og de vil til slutt annihileres [25]. Massen til positronet og elektronet vil da konverteres til energi i form av to gammafotoner med en energi på 0,511 MeV hver [24, 25], som vist i Figur 2.4. Energien kan regnes ut fra likning 2.3, som er Einsteins energilikning [25].

$$E = mc^2 = m_e c^2 + m_p c^2 \tag{2.3}$$

Der m_e er massen til elektronet, m_p er massen til protonet og c er lyshastigheten i vakuum. På grunn av bevaring av energi og bevegelsesmengde, så vil disse fotonene bevege seg i motsatt retning langs en rett linje fra annihileringspunktet [28]. PET benytter seg av denne egenskapen for å lokalisere kilden til strålingen [26].

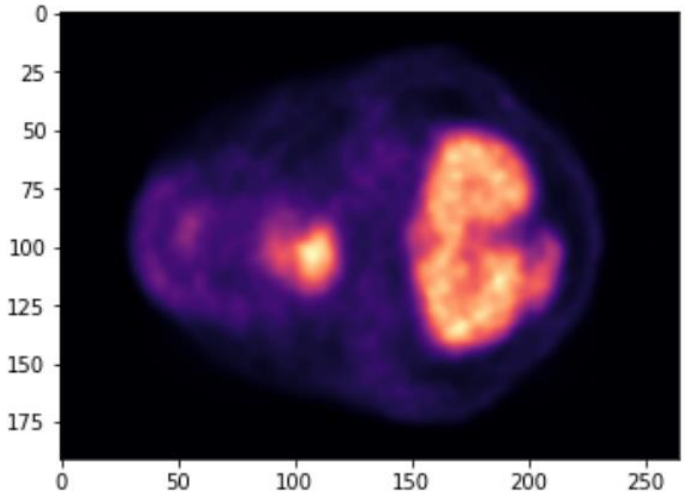


Figur 2.4. Annihilering av elektron og positron. Illustrasjonen er tegnet etter inspirasjon fra figur i Introduction to Physics in Modern Medicine av S. A. Kane og B. A. Gelman (2020) [25].

2.4.3 Fluorodeoksyglukose

Pasienten som skal undersøkes, injiseres intravenøst med en radioaktiv markør. Den inneholder en positronemitterende isotop som er bundet til en organisk ligand [33]. Markørens ligandkomponent vil reagere med visse proteiner i kroppen. En av de mest brukte radiomarkørene er Fluorodeoksyglukose (2-deoxy-2-[F-18]-fluoro-D-glucose, FDG) [24]. Den består av en fluorine-18-isotop som er bundet til sukkermolekylet glukose [33]. Fluorine-18 har en halveringstid på 109 minutter og 97 % av degraderingen skjer ved positronemisjon [34].

Ettersom glukose gir celler energi, vil dette kobles kjemisk til områder i kroppen med høy metabolisme [33]. Svulstceller trenger mer energi enn andre celler og har derfor en høyere metabolsk aktivitet. Konsentrasjonen av radiomarkøren vil derfor akkumuleres her [35]. Man vil på denne måten kunne kartlegge svulster og andre områder med høy metabolsk aktivitet i kroppen, som vist på Figur 2.5.



Figur 2.5. PET-bilde. Områder med høy metabolisme lyser opp ved bruk av PET.

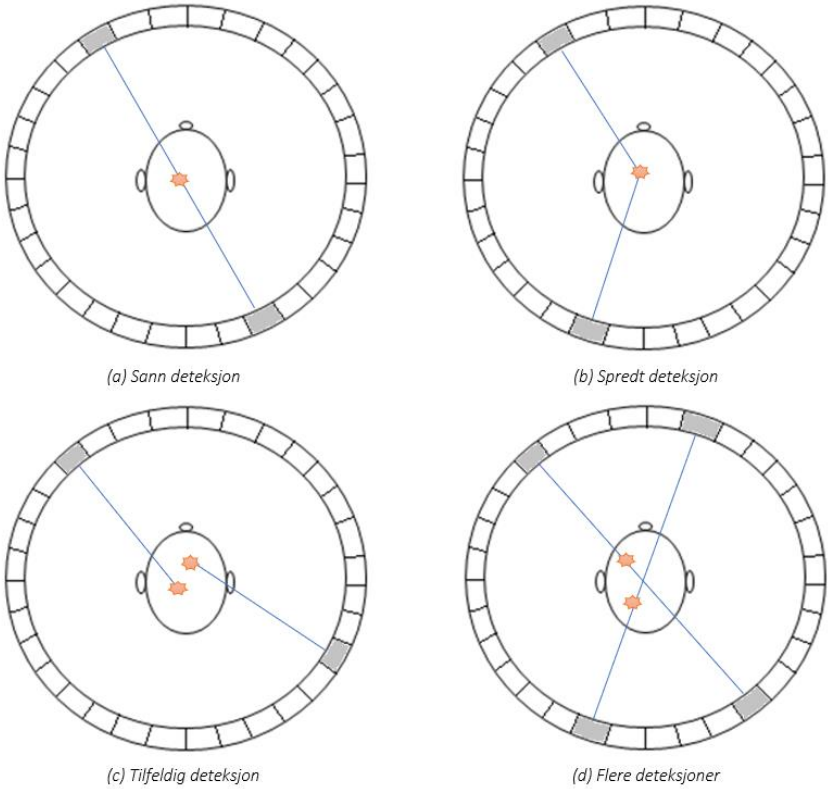
2.4.4 PET-maskinens oppbygning

Utsendelsen av to fotoner i hver sin retning gir informasjon om annihileringens lokasjon til PET-skanneren. Skanneren består av en ring med detektorer som fanger opp fotonene på hver sin side av ringen [24]. Når to detektorer treffes samtidig, indikerer dette at det har foregått en annihilering langs en rett linje mellom dem. Denne linja er ofte referert til som *Line of Response* (LOR) [28]. Hvis mange LOR krysser et område indikerer det at det er mye aktivitet der. Detektorene vil kun fange opp treffene hvis de forekommer svært nære i tid, slik at det er sannsynlig at de kommer fra samme annihilering. Tidsvinduet strålene må treffe innen, er 10 til 25 nanosekunder [24]. Dette kalles *coincidence detection* [25].

Det finnes to typer bakgrunnsstråling som også kan detekteres, og det er spredt og tilfeldig stråling [28]. Spredt deteksjon er vist i Figur 2.6b, og tilfeldig deteksjon er vist i Figur 2.6c. Dette kan skje hvis ett eller flere av fotonene blir spredt på grunn av comptonspredning før de treffer detektorene, eller hvis to detekterte fotoner definerer en LOR som ikke inneholder annihileringspunktet [28]. Hvis dette fanges opp, kan de detekterte fotonene angi feil annihileringspunkt og gi et unøyaktig bilde av den metabolske aktiviteten i pasienten. I tillegg kan flere fotoner detekteres samtidig, som vist i Figur 2.6d.

For en gitt LOR er det korrigert for tilfeldige treff og treff fra spredning, slik at flest mulig sanne treff blir registrert [28]. Sanne treff er vist i Figur 2.6a. Attenueringsfaktoren til hver LOR beregnes [24]. Til slutt samles informasjonen, og *filtered back projection* brukes slik

som i CT til å lage et bildetverrsnitt av den skannede regionen [24, 25]. Tverrsnittet viser en oversikt over metabolismeaktiviteten til det skannede planet av pasienten.



Figur 2.6. Illustrasjon av sann deteksjon (a), spredt deteksjon (b), tilfeldig deteksjon (c) og flere deteksjoner (d). Illustrert etter inspirasjon fra figur i PET physics, instrumentation, and scanners av M. E. Phelps (2006) [36].

2.4.5 SUV

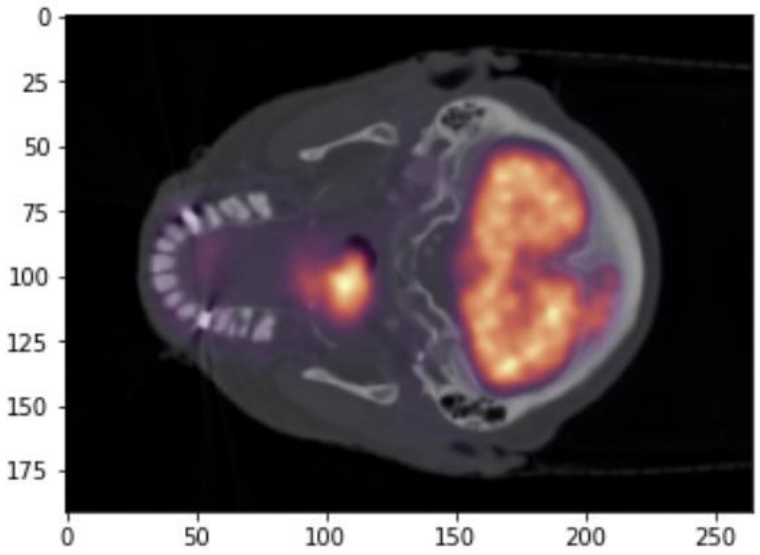
Opptaket av FDG vil variere, hovedsakelig basert på mengden markør som injiseres i pasienten og pasientens størrelse [37]. For å kompensere for denne variasjonen brukes SUV som et mål på relativt opptak av FDG. SUV står for *Standardized Uptake Value*, og regnes ut ved hjelp av likning 2.4.

$$SUV = \frac{r}{\left(\frac{a'}{w}\right)} \tag{2.4}$$

Her står *r* for konsentrasjonen av radioaktivitet som måles av PET-skanneren, *a'* er mengden FDG som er korrigert for henfall, og *w* er vekten til pasienten. Fordelen ved å bruke SUV er blant annet at man kan sammenligne pasienter [37].

2.5 PET/CT

Kombinasjonen av PET og CT er en av de mest avanserte billedundersøkelsene som finnes [38]. I dette kombinerte apparatet utføres gjerne CT-skanningen først, etterfulgt av PET-skanning rett etterpå. Bildene blir fusjonert, og man kan dermed utnytte fordelene til både PET og CT. PET gir informasjon om opphopningen av celler med høy metabolsk aktivitet, mens CT vil gi en anatomisk kartlegging over hvor opphopningen finner sted [38]. I tillegg til å gi en anatomisk kartlegging kan CT-bildene brukes til korreksjon av attenueringsfaktoren som regnes ut ved PET [28]. Et eksempel på et bilde tatt med det kombinerte apparatet er gitt i Figur 2.7.

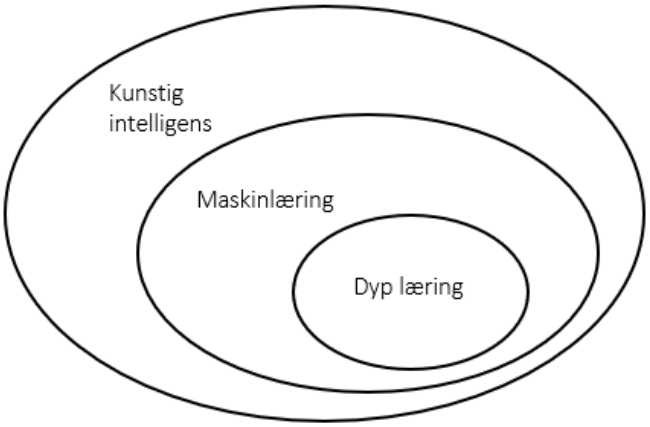


Figur 2.7. Bilde tatt med det kombinerte PET/CT-apparatet.

2.6 Kunstig intelligens

Kunstig intelligens (AI) betegnes som muligheten til å automatisere intellektuelle oppgaver som normalt gjennomføres av mennesker, og er et generelt felt som innebefatter maskinlæring og dyp læring [13]. Siden kunstig intelligens oppstod på 1950-tallet, har det vært utsatt for perioder med tvil og optimisme, før det ble en viktig del av teknologien rundt 2010. Kunstig intelligens kan grovt skilles inn i to grupper: systemer som forstår regler ved at reglene er programmert inn på forhånd, og systemer som lærer ved å bli eksponert for store mengder

data [13]. Den siste gruppen kalles maskinlæring, og i nyere tid har maskinlæring vært det mest suksessfulle og populære underfeltet innen kunstig intelligens [13]. Figur 2.8 viser en oversikt over undergruppene innen kunstig intelligens, som introduseres og forklares i de kommende kapitlene.



Figur 2.8. Oversikt over undertemaene innen kunstig intelligens. Illustrert etter inspirasjon fra figur i Deep Learning With Python av F. Chollet (2018) [13].

2.7 Maskinlæring

2.7.1 Hva er maskinlæring

Maskinlæring stammer fra ideen om at en datamaskin skal kunne lære på egenhånd [13]. Den skal lære hvordan en spesiell oppgave skal løses ved å se på data, og tilpasse løsningsmetoden til nye data. Maskinlæring tar dermed steget videre fra kunstig intelligens ved at maskinen får data som input, og gir regler ut [13]. Disse reglene kan brukes på ny data, og gir ut svar. Det sies derfor at en maskinlæringsmodell trenes, ikke programmeres [13].

2.7.2 Data

For at en maskinlæringsmodell skal kunne trenes, trenger den data. Kvaliteten på dataen er en avgjørende faktor for hvor god maskinlæringsalgoritmen kan bli [39]. Det er derfor viktig at dataen er så gunstig for trening som mulig. Det vil si at den inneholder informasjon som maskinlæringsmodellen klarer å utnytte [39]. For å oppnå dette undersøkes og preprosesseres datasettet før det mates inn til modellen.

Før preprosesseringen begynner, må datasettet undersøkes og videre bearbeiding gjøres på grunnlag av hva slags data man har. Preprosessering kan innebære å standardisere eller normalisere tallverdier for å få de på samme skala, fjerne utliggere («outliers») eller tildele

ord tallverdier som datamaskinen kan forstå [39]. Det kan også innebære å fjerne data som ikke har betydning for treningen, eller endre dimensjonen. Hvis man har for lite eller for ensformig data i utgangspunktet, kan man benytte augmentering («augmentation»). Da justerer man deler av datasettet og legger den justerte dataen til det originale. Dette gjøres ofte med bildedata, og kan bidra til at modellen blir mer robust ved at den får trent på et mer variert datasett.

Dataen som mates inn til en veiledet læringsmodell, som forklares nærmere i neste avsnitt, deles ofte inn i flere sett. Et treningssett, et valideringssett og et testsett. Treningssettet brukes til trening og optimalisering av modellen. Når modellen er trent, testes den på valideringssettet. Resultatet blir evaluert, og modellens konfigurasjon endres etter dette. Konfigurasjonen endres ved å justere såkalte hyperparametere i modellen. Hyperparametere er parametere som kan endres, og kan for eksempel være antall lag eller størrelsen til lagene i nettverket [13]. Her ligger det en risiko for overtilpasning, som forklares i delkapittel 2.8.2, hvis modellen trenes for mye. Når modellen er ferdig trent blir den testet på testsettet, og man får ut den endelige ytelsen til modellen [13].

2.7.3 Veiledet og ikke-veiledet læring

Maskinlæringsteknikker kan grovt sett deles inn i to klasser: veiledet læring og ikke-veiledet læring [40]. Veiledede læringsalgoritmer bruker data som allerede er klassifisert. Det vil si at hvert dataeksempel i treningssettet er klassifisert på forhånd. Klassifiseringene kan enten være diskrete eller kontinuerlige variabler. Det betyr at betegnelsen av hvert eksempel enten er en klasse som representerer den gitte dataen, eller en tallverdi. Hvis betegnelsene tilsvarer klasser, får vi klassifiseringsproblemer. Er de tallverdier, får vi regresjonsproblemer. Maskinlæringsalgoritmen vil dermed prøve å klassifisere eller predikere dataen basert på hva den er klassifisert eller predikert til tidligere [40]. Ved ikke-veiledet læring benyttes data uten forhåndsklassifisering, og maskinlæringsalgoritmen konkluderer på egenhånd. Klyngeanalyse er en av de mest kjente ikke-veiledede læringsteknikkene. Teknikken grupperer inputdataen til undergrupper som har så liten intravariabilitet og stor intervariabilitet som mulig [40].

2.8 Dyp læring

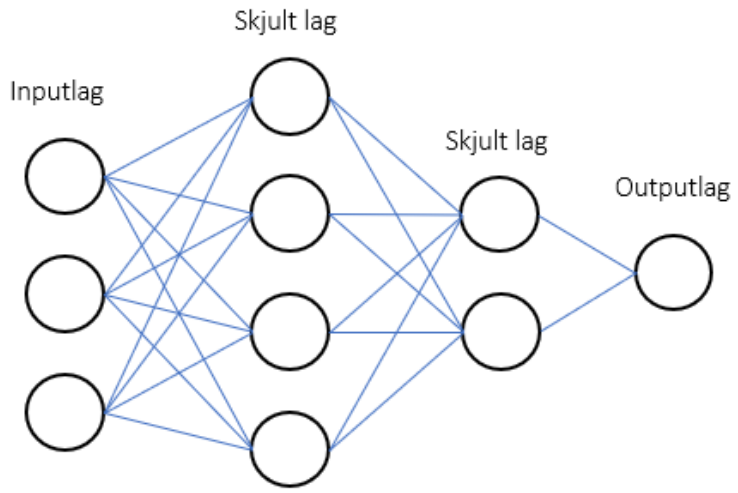
2.8.1 Hva er dyp læring

Dyp læring har lenge vært et undertema innen maskinlæring, men de siste årene har dyp læring revolusjonert feltet innen bildeanalyse, talegjenkjenning, håndskriftsoversetting og

mye mer [13]. Konseptet er det samme som ved vanlig maskinlæring, men ved dyp læring kan maskinen lære fra enda større og mer ustrukturerte datamengder. Modellen prosesserer informasjon gjennom hierarkiske lag, der kompleksiteten øker for hvert lag [40].

2.8.2 Nevrale nettverk

Nevrale nettverk er maskinlæringsalgoritmer som er inspirert av oppbygningen til det nevrale nettverket i hjernen [40]. De er bygd opp som lag med noder, der nodene i ett lag har koblinger til nodene i neste lag, som vist i Figur 2.9. Lagene består av inputlag, skjulte lag og outputlag. Inputlaget er der dataene mates inn, mens outputlaget gir klassifiseringer eller prediksjoner ut. Antall noder i outputlaget avgjøres av hvor mange klasser nettverket skal klassifisere. Er problemet binært, består outputlaget av ett eller to noder. Med flere klassifiseringer vil det være flere noder [40]. Mellom input og output ligger de skjulte lagene. De består av flere noder, og gjør operasjoner på dataene. Antall skjulte lag og antall noder i lagene, varierer mellom nettverk. Flere skjulte lag gir en mer komplisert modell som vil ta lenger tid å kjøre, men som samtidig vil kunne håndtere flere kompliserte problemstillinger [39].



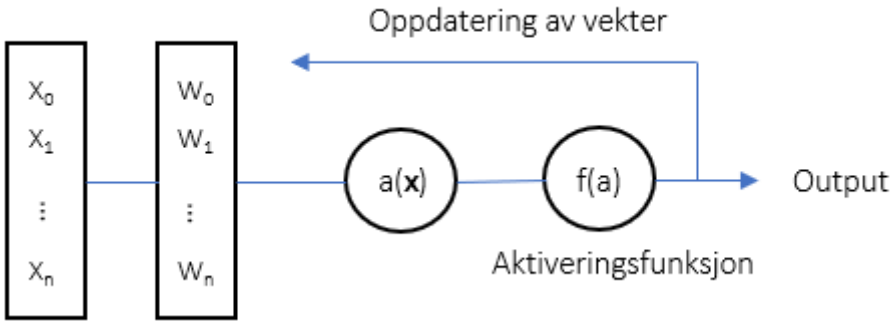
Figur 2.9. Illustrasjon av et nevralt nettverk. Sirklene representerer nodene, og de blå linjene representerer de vektede koblingene mellom dem. Illustrert etter inspirasjon fra figur i Python Deep Learning av Zocca et al. [40, 41].

Nevroner

Nodene i hvert lag får en input fra forrige lag og gir en output videre. Denne delen av nettverket kalles et nevron [40]. Koblingen mellom nevronene får tilgitt en styrke i form av et vekttall, og disse vektene vil oppdateres når algoritmen lærer. Koblingen mellom nevronene,

antall nevroner i hvert lag, og totalt antall lag, angir arkitekturen til modellen [40]. Arkitekturen vil variere mellom forskjellige modeller.

Et nevron kan illustreres slik som i Figur 2.10. Hvert nevron får inn inputverdier fra vektor ***x*** og korresponderende vekter fra vektor ***w*** fra forrige lag. Disse verdiene kombineres i en funksjon som angir aktiveringsverdien *a(**x**)*, til nevronet [39]. Aktiveringsverdien brukes videre i aktiveringsfunksjonen som frembringer en output. Vektene blir så justert ut fra aktiveringsfunksjonens output [39].



*Figur 2.10. Illustrasjon av et nevron. Inputverdiene og de opprinnelige vektene gir aktiveringsverdien a(**x**) til hvert nevron, og denne verdien brukes videre i aktiveringsfunksjonen, f(a). Vektene blir oppdatert basert på aktiveringsfunksjonen. Illustrert etter inspirasjon fra figur i Python Deep Learning Zocca et al. [40, 41].*

Aktiveringsverdi

Aktiveringsverdien i hvert nevron beskriver nevronets indre tilstand [40], og den beskriver dermed hvordan nevronet virker. Aktiveringsverdien bestemmer også om signalet skal sendes videre til neste nevron eller ikke, og den er gitt ved likning 2.5.

$$a(\boldsymbol{x}) = \sum_i w_i x_i \tag{2.5}$$

Her er x_i inputverdien til nevronet fra nevron i i forrige lag, og w_i er en vektkoeffisient som er verdien til denne koblingen [40]. Aktiveringsverdien kan defineres som prikkproduktet mellom vektor ***w*** og vektor ***x***. Vektor ***w*** inneholder alle vektene til koblingene fra forrige lag, og ***x*** inneholder alle inputverdiene. Vektor ***x*** ligger vinkelrett på vektor ***w*** hvis $\langle \boldsymbol{x}, \boldsymbol{w} \rangle = 0$. Alle vektorer ***x*** som gir dette vil derfor utgjøre et hyperplan i $\mathbf{R}^d$, der d er dimensjonen til ***x***. Dermed vil enhver vektor ***x*** som gir $\langle \boldsymbol{w}, \boldsymbol{x} \rangle > 0$ ligge på siden av planet [40]. På denne måten kan et nevron fungere som en klassifikator [40]. Hyperplanet kan settes til andre verdier enn

null, for å tilpasses dataen bedre. Man må da introdusere en bias *b* som justerer planet [40]. Uttrykket for aktiveringsverdien vil da være gitt ved likning 2.6, der *b* er biasen.

$$a(\boldsymbol{x}) = \sum_i w_i x_i + b \tag{2.6}$$

Aktiveringsfunksjon

I hvert nevron blir output bestemt av en aktiveringsfunksjon som baserer seg på aktiveringsverdien ***a(x)*** til nevronet [40]. Det finnes mange forskjellige aktiveringsfunksjoner, og videre omtales et utvalg av de mest vanlige [40]. Den enkleste kalles Identitetsfunksjonen og er gitt ved likning 2.7.

$$f(a) = a \tag{2.7}$$

Dette er en lineær funksjon som lar aktiveringsverdien til nevronet bli output videre. En annen enkel funksjon er Terskelfunksjonen, som er gitt ved likning 2.8.

$$f(a) = \begin{cases} 1 \; if \; a \geq 0 \\ 0 \; if \; a < 0 \end{cases} \tag{2.8}$$

Denne funksjonen sender signalet videre kun hvis aktiveringsverdien er over terskelverdien 0. Signalet som sendes videre er satt til 1, og terskelverdien kan endres til andre verdier enn 0 [40]. Neste aktiveringsfunksjon kalles Tangens hyperbolikus (Tanh), og er gitt ved likning 2.9.

$$f(a) = \frac{\exp(a) - \exp(-a)}{\exp(a) + \exp(-a)} = \frac{1 - \exp(-2a)}{1 + \exp(-2a)} \tag{2.9}$$

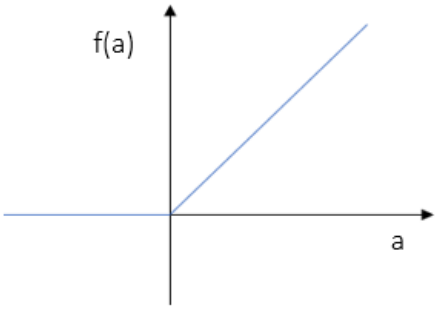
Denne grafen spenner seg mellom -1 og 1. En annen funksjon som er mye brukt, kalles Logistikkfunksjonen («Logistic function») eller Sigmoidfunksjonen. Den er gitt ved likning 2.10.

$$f(a) = \frac{1}{1 + \exp(-a)} \tag{2.10}$$

Funksjonen likner på Tangens hyperbolikus. Forskjellen er at Sigmoidfunksjonen spennes mellom 0 og 1, og ikke -1 og 1 slik som Tanh. Funksjonen kan brukes som sannsynligheten for at et nevron aktiveres, ettersom output ligger mellom 0 og 1. En siste aktiveringsfunksjon som også er mye brukt heter *Rectified Linear Unit* (ReLU), og er vist i Figur 2.11. ReLU er en blanding av Identitetsfunksjonen og Terskelfunksjonen, og er gitt ved likning 2.11.

$$f(a) = \begin{cases} a \; if \; a \geq 0 \\ 0 \; if \; a < 0 \end{cases} \tag{2.11}$$

Her sendes aktiveringsverdien videre hvis den ligger over 0. Ved tilbakepropagering, som forklares senere i oppgaven, brukes den deriverte av aktiveringsfunksjonen til å oppdatere vektene for hvert lag bakover i modellen [39]. For visse aktiveringsfunksjoner, som Sigmoidfunksjonen eller Tanh, vil den deriverte nærme seg null hvis modellen har mange lag. Informasjonen vil da ikke propageres videre bakover, og de tidlige vektene i nettverket vil ikke bli oppdatert. Dette problemet kalles forsvinnende gradient-problem, og unngås ved bruk av ReLU som aktiveringsfunksjon [39]. Dette er en av grunnene til at ReLU er hyppig brukt.



Figur 2.11. Aktiveringsfunksjonen Rectified Linear Unit.

Hvilke aktiveringsfunksjoner som lønner seg å bruke, varierer imidlertid fra problem til problem. Et nevralt nettverk kan bruke flere forskjellige aktiveringsfunksjoner. Alle nevronene i ett lag har ofte samme aktiveringsfunksjon, men forskjellige lag kan ha forskjellige aktiveringsfunksjoner [40].

Vekter

Hver node i ett lag er koblet med noen av, eller alle, nodene i neste lag via en vektkoeffisient [39]. Vektkoeffisienten er den samme som brukes i ligningen for aktiveringsverdien, og betegnes som styrken til koblingen mellom nodene i de forskjellige lagene. Vektene er et sett med tall, og kalles i noen tilfeller for de trenbare parameterne til laget [13]. Konseptet med at maskinen lærer, handler egentlig om å justere verdiene til disse vektene slik at modellen gir så riktige output som mulig [13]. I starten settes vektene til små tilfeldige tall, før de blir oppdatert etter hvert som maskinen lærer.

Kostfunksjon

Et nevralt nettverk er en tilnærming til en funksjon, men ettersom det kun er en approksimasjon vil denne funksjonen avvike fra den ønskede funksjonen [40]. Dette avviket kan beskrives ved kostfunksjonen, som er en funksjon av vektene i vektor $\boldsymbol{w}$ [40]. Kostfunksjonen gir en indikasjon på hvor god prediksjonen til modellen er ved å angi feilen [13]. Det er derfor ønskelig at kostfunksjonen skal ha en så lav verdi som mulig for å minimere feilen, og optimalisere modellen.

Valg av kostfunksjon vil variere basert på hvilken problemstilling nettverket skal løse [13]. Hvis problemet er klassifisering, brukes ofte kryssentropi («Cross entropy»). Binær kryssentropi brukes for binære klassifiseringsproblemer, og kategorisk kryssentropi for klassifiseringsproblemer med flere klasser [13]. Kryssentropifunksjonen er generelt definert som i likning 2.12.

$$J(\boldsymbol{w}) = -\sum_{i=1}^{n} y_i \ln(p_i) \tag{2.12}$$

Der y_i er de observerte verdiene, og p_i er den predikerte sannsynligheten for den i-ende klassen. For regresjonsproblemer brukes ofte midlere kvadratisk feil («mean squared error», MSE) [13, 39], som er gitt ved likning 2.13.

$$J(\boldsymbol{w}) = \frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)^2 \tag{2.13}$$

Her er y_i de observerte verdiene, mens $\hat{y}_i$ er de predikerte verdiene fra nettverket.

En kostfunksjon som brukes mye til binære segmenteringsproblemer er DiceLoss, som ble introdusert av Milletari et al. [42]. Denne funksjonen er basert på Dice-score, som introduseres i delkapittel 2.9.2. DiceLoss er gitt ved likning 2.14.

$$J_{DiceLoss} = 1 - \frac{2 * TP}{\sum_i p_i^2 + \sum_i g_i^2} \tag{2.14}$$

Her står *TP* for sann positiv, og beskriver antall positivt klassifiserte piksler som er riktige, p_i beskriver den predikerte sannsynligheten for at piksel i hører til den positive klassen, og g_i beskriver den sanne verdien til piksel i.

For hver iterasjon blir vektene justert etter kostfunksjonen, slik at verdien av den blir mindre. Når kostfunksjonen er så liten som mulig, er vektene optimalisert og nettverket er ferdig trent [40].

Optimalisering

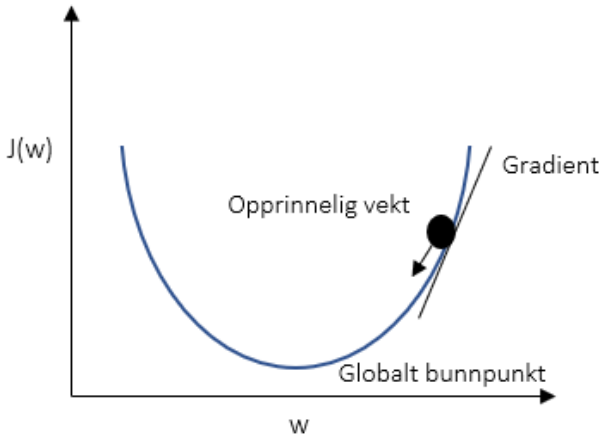
Kostfunksjonen kan minimaliseres med forskjellige algoritmer, slik at vektene optimaliseres. En av de vanligste og mest populære algoritmene er gradient-nedstigning [43]. Hovedideen bak gradient-nedstigning er å klatre ned en bakke til bunnpunktet nås. For hver iterasjon tar man et steg i motsatt retning av stigningsgraden til kostfunksjonen [39]. Oppdateringen av vektene skjer for hver iterasjon, og kan skrives slik som i likning 2.15.

$$\boldsymbol{w} = \boldsymbol{w} + \Delta \boldsymbol{w} \tag{2.15}$$

Der $\boldsymbol{w}$ er de opprinnelige vektene, og $\Delta \boldsymbol{w}$ er endringen av vektene [39]. $\Delta \boldsymbol{w}$ beregnes ved likning 2.16.

$$\Delta \boldsymbol{w} = -\eta \nabla J(\boldsymbol{w}) \tag{2.16}$$

Her er η læringsraten, som beskriver hvor mye vektene skal endres for hver iterasjon [40]. Læringsraten er vanligvis et tall som er mye mindre enn 1, og kan ses på som størrelsen til stegene mot bunnpunktet [40]. $J(\boldsymbol{w})$ er kostfunksjonen, og ∇ angir at kostfunksjonen er derivert med hensyn på hver vekt for å finne stigningsgraden. Gradient-nedstigning kan dermed ses på som en ball som ruller ned til bunnpunktet av kostfunksjonen, som vist i Figur 2.12 [43].



Figur 2.12. Illustrasjon av gradient-nedstigning, tegnet etter inspirasjon fra figur i Python Machine Learning av S. Raschka og V. Mirjalili (2019) [39].

Det kan være utfordrende å avgjøre hvor stor læringsraten skal være ved gradient-nedstigning. For stor læringsrate kan hindre konvergens, og lar heller vektene fluktuere over bunnpunktet [43]. For liten læringsrate vil finne bunnpunktet etter hvert, men det kan ta lang tid. I tillegg har vanlig gradient-nedstigning en tendens til å havne i lokale bunnpunkt, og ikke nå fram til det globale bunnpunktet [43]. For å håndtere disse utfordringene har gradient-nedstigning blitt oppgradert ved forskjellige metoder. En metode som er mye brukt, er *Adaptive Moment Estimation* (Adam). Adam gir tilpassede læringsrater for hver parameter, og lagrer gjennomsnittet og variansen fra tidligere gradienter. Denne metoden gir gode resultater sammenlignet med andre optimaliseringsmetoder [43].

Tilbakepropagering

For et nevralt nettverk med kun ett lag, er optimalisering enkelt ettersom man vet hva slags output man vil at alle nodene skal gi. Lineær eller logistisk regresjon kan da brukes direkte for å oppdatere vektene samtidig [40]. Dette er vanskeligere for nettverk med flere lag, ettersom man ikke vet hva slags output som er ønskelig fra nevronene i andre lag enn det siste [40]. Ved flere lag må man dermed regne ut feilen i det bakerste skjulte laget, og propagere feilen fram til det første laget [40]. Dette konseptet kalles tilbakepropagering.

Algoritmene bak tilbakepropagering er komplekse, og vil ikke bli forklart i dybden i denne masteroppgaven. Men konseptet bygger på differensialregning og kjerneregelen [40].

Overtilpasning og undertilpasning

En viktig del av maskinlæring handler om å finne balansen mellom optimalisering og generalisering [13]. Optimalisering går ut på å justere vektene slik at kostfunksjonen blir minst mulig, og ytelsen til modellen blir så god som mulig på treningsdataen. Generalisering derimot, handler om hvor god modellen er på ny og usett data [13]. Det er viktig at modellen ikke blir for optimalisert, såkalt overtilpasset.

Overtilpasning skjer når modellen trenes for mye på treningsdataen [13]. I starten vil modellen bli både mer optimalisert og mer generalisert når den trenes. Hvis treningen fortsetter vil modellen etter hvert nå et punkt hvor den ikke generaliseres mer, selv om den fortsetter å optimaliseres. Modellen vil da lære seg mønstre som er spesielle for treningsdataen, men som ikke nødvendigvis gjelder for ny data [13].

Det finnes flere metoder for å unngå overtilpasning, men den beste metoden er å tilføre mer treningsdata [13]. En større mengde data vil naturlig gi en bedre generalisering. Hvis dette ikke er mulig, kan man kontrollere hvor mye informasjon modellen kan lagre fra

treningsdataen, slik at nettverket bare husker et lite antall mønstre. Det vil da fokusere på de mest dominerende mønstrene, og modellen vil forbli mer generell [13]. Denne metoden kalles regularisering.

Noen av de mest vanlige regulariseringsmåtene er å minke antallet trenbare parametere i modellen, legge på vektregularisering eller å bruke utelatelse. Å minke antall trenbare parametere i modellen er ekvivalent med å senke modellens hukommelseskapasitet [13]. Her er det viktig å ikke senke hukommelseskapasiteten for mye slik at modellen blir undertilpasset. Ved vektregularisering hindrer man at nettverket blir for komplekst ved å tvinge vektene til å kun ha små verdier [13]. Vi har to typer vektregularisering, L1 og L2, der hovedpoenget med begge er å legge på et ledd på kostfunksjonen som gir en ekstra «kostnad» ved å ha store vekter. Utelatelse er den mest brukte regulariseringsmetoden ettersom den er mest effektiv [13]. Utelatelsen legges på lag i nettverket, og dropper dermed et tilfeldig antall noder fra laget under treningsprosessen. Utelatelsesraten er andelen noder som skal droppes fra laget, og settes ofte mellom 0,2 og 0,5 [13].

2.8.3 Konvolusjonsnettverk

Dyp læring er en klasse maskinlæringsalgoritmer som alle prosesserer informasjon gjennom hierarkiske lag, der kompleksiteten øker for hvert lag [40]. Konvolusjonsnettverk («Convolutional Neural Network», CNN) er et eksempel på en slik algoritme, og brukes spesielt mye til bildeklassifiseringsoppgaver [39].

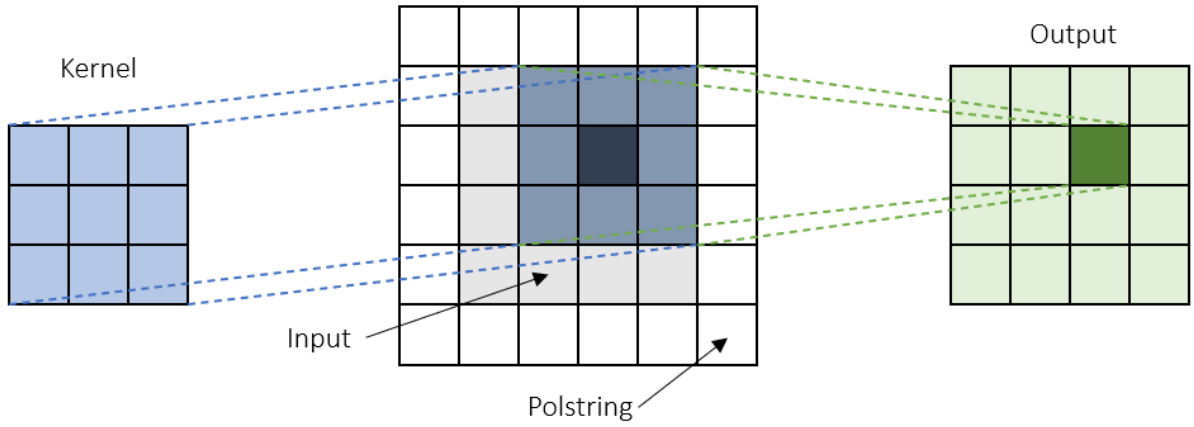
Konvolusjonsnettverk er dype nevrale nettverk som er inspirert av hvordan menneskehjernens visuelle korteks fungerer når den skal kjenne igjen objekter [39]. De tidligste lagene i nettverket ekstraherer ut hovedtrekkene i bildet, mens de senere lagene bruker disse trekkene til å predikere en verdi eller en klasse til bildet. På denne måten kan de tidligste lagene finne enkle strukturer i bilder, som kanter eller andre geometriske former, og de senere lagene kan kombinere denne informasjonen for å finne ut hvilket objekt som er avbildet [39].

Konvolusjonsnettverk består vanligvis av tre typer lag: konvolusjonslag, subsamplingslag og dense-lag [39]. Hvert lag får inn et input-egenskapskart fra forrige lag, og gir et output-egenskapskart til neste. Et slikt egenskapskart er ofte en 3D vektor, med høyde, bredde og dybde [13]. Dybden blir ofte referert til som kanaler, og kan for eksempel være fargene i bildet eller forskjellige fremhevede trekk [13].

Konvolusjonslag

Konvolusjonslag er et lag som framhever visse trekk i bildet [40]. Hvert nevron i laget får kun informasjon fra nabonevronene på et lite område, og lærer dermed små, lokale mønstre [13]. Disse mønstrene kan kjennes igjen flere steder i bildet, og kan settes sammen til større mønstre lenger ut i nettverket [13]. Konvolusjonslag fungerer ved å flytte et lite vindu (ofte 3×3 eller 5×5) rundt på input-egenskapskartet, slik at alle pikslene etter hvert dekkes. Forflytningen skjer med en steglengde, som gir informasjon om hvor langt vinduet skal flytte seg av gangen. Alle de små vinduene transformeres via en kernel, som er en vektmatrise [13]. Vektmatrisen inneholder vekttall, og ganges med vinduet for å trekke ut relevant informasjon. Transformeringene settes sammen til et output-egenskapskart, der alle lokasjonene korresponderer med de samme lokasjonene i input-egenskapskartet [13], som vist i Figur 2.13.

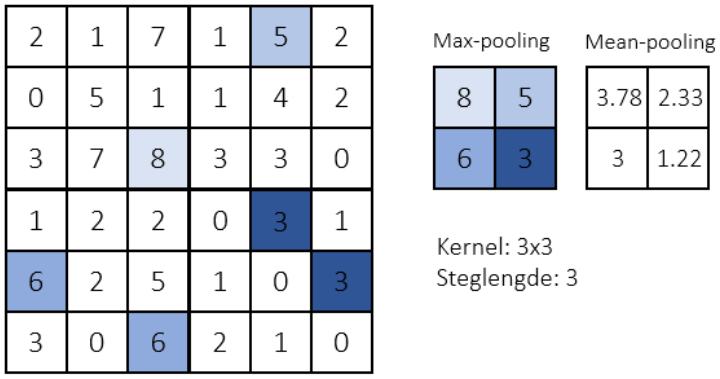
Høyden og bredden til output-egenskapskartet kan variere fra input-egenskapskartet [13]. Variasjonen kan skyldes to grunner: kanteffekter og bruken av steglengde [13]. Kanteffekter handler om at de ytterste pikslene i input-egenskapskartet ikke blir med videre fordi vektmatrisen ikke når helt ut i kantene. Legger man på en polstring i form av tomme piksler rundt bildet, vil de ytterste pikslene også bli med, som vist i Figur 2.13. På denne måten kan man forhindre at dimensjonen til output-egenskapskartet endrer seg. Størrelsen på polstringen varierer ut fra størrelsen til vektmatrisen og steglengden. Om steglengden er større enn én, hopper vektmatrisen rundt på input-egenskapskartet med større avstander av gangen, og alle pikslene i bildet får ikke blitt med i output-egenskapskartet. Dermed vil output-egenskapskartet bli mindre.



Figur 2.13. Konvolusjon med polstring. De ytterste rutene i input-egenskapskartet er polstring i form av tomme piksler. Illustrert etter inspirasjon fra figur i Python Machine Learning S. Raschka og V. Mirjalili (2019) [39].

Subsamplingslag

Subsamplingslag, også kalt samlelag, endrer størrelsen til output-egenskapskartet, men holder dybden lik [40]. Dette kan gjøres på flere måter, men de vanligste er ved *max-pooling* eller *mean-pooling* [40]. Da flyttes en kernel (f.eks. 2×2) rundt på input-egenskapskartet, slik som i et konvolusjonslag. Ved *max-pooling* overføres pikselen med den høyeste verdien til output-egenskapskartet, mens ved *mean-pooling* overføres gjennomsnittet av alle pikslene [40]. Output-egenskapskartet vil da bli mindre enn input-egenskapskartet, som vist i Figur 2.14.



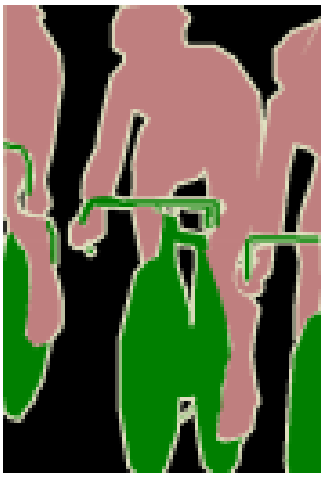
Figur 2.14. Pooling-eksempel med max-pooling og mean-pooling. Illustrert etter inspirasjon fra figur i Python Machine Learning av S. Raschka og V. Mirjalili (2019) [39].

Dense-lag

Dense-lag kalles også fullt tilkoblede lag, og kjennetegnes ved at alle nodene i et lag er koblet med alle nodene i neste lag [39]. Nodene i laget får dermed en input fra samtlige noder i forrige lag. Fordelen med Dense-lag er at de lærer fra alle kombinasjonene av egenskaper i forrige lag. Ulempen med dette er at beregningen er mer tidkrevende enn ved et konvolusjonslag.

Semantisk segmentering

Semantisk segmentering handler om å segmentere ut objekter i bilder. Det kan for eksempel være å skille ut syklister og sykler fra bakgrunnen, som vist i Figur 2.15. I den medisinske verden kan semantisk segmentering blant annet brukes til å segmentere ut svulster og lymfekjertler fra medisinske bilder. Ved denne typen segmentering må hver piksel i bildet klassifiseres [44]. Output-bildet vil derfor som oftest ha samme størrelse, altså samme antall piksler, som input-bildet.



Figur 2.15. Semantisk segmentering av syklist og sykkel fra bakgrunnen. Bildet er hentet fra Fully Convolutional Networks for Semantic Segmentation © [2015] IEEE [44].

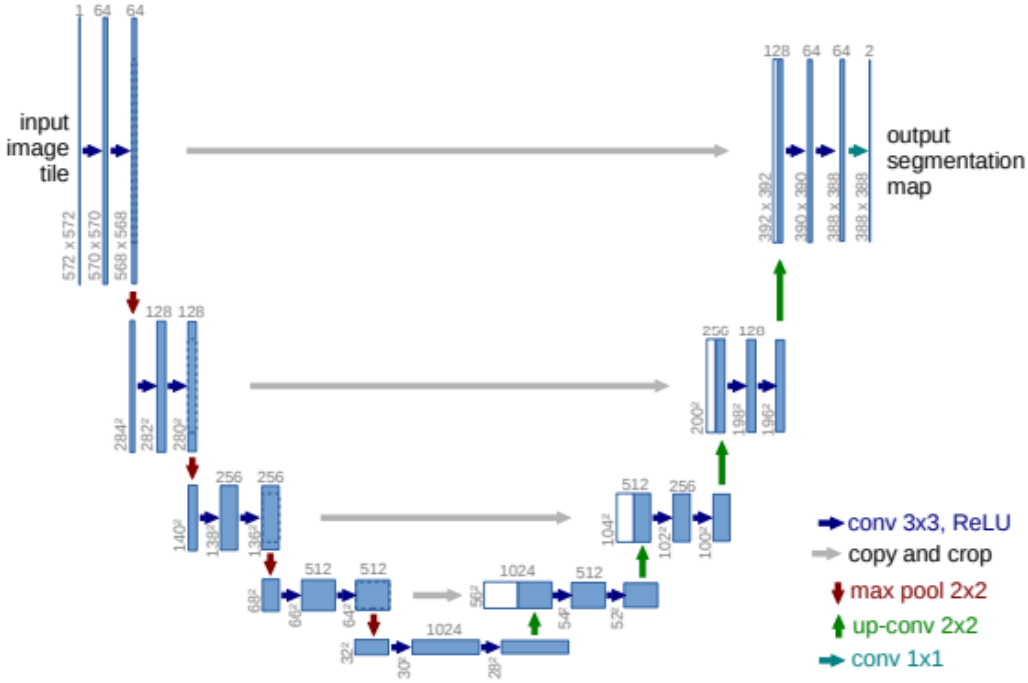
Konvolusjonsnettverk har vist seg å gi gode resultater på semantisk segmentering [45], særlig Fullt konvolusjonsnettverk («Fully Convolutional Networks», FCN) som kun består av konvolusjonslag og subsamplingslag. Ronneberger et al. [46] viste at en FCN-arkitektur kalt U-Net gjorde det særlig bra på segmentering av nevronstrukturer og celler på mikroskopiske bilder.

U-Net

U-Net består av en forminskende vei og en forstørrende vei [46]. Dette gir arkitekturen en U-struktur som vist i Figur 2.16, derav navnet U-Net. Den forminskende veien repeterer to 3×3 konvolusjonslag med ReLU som aktiveringsfunksjon, og en 2×2 *max-pooling*-operasjon med steglengde 2 som nedsampling [46]. For hvert steg dobles antall kanaler i egenskapskartet. Dette gjøres til et bunnpunkt nås, og den forstørrende veien begynner [46]. Den forstørrende veien gjør det samme som den forminskende, bare motsatt. Hvert steg består av en oppsampling etterfulgt av to 2×2 konvolusjonslag [46]. For hvert steg halveres antall kanaler i egenskapskartet.

Mellom den forminskende og den forstørrende veien finnes det såkalte lange hoppforbindelser («skip connections»), som hjelper til med å bedre den romlige oppløsningen som mistes ved nedskalering [47]. Disse hoppforbindelsene kobler sammen hvert steg i den forminskende veien med den forstørrende veien, og legger til kompleksitet og ikke-linearitet [47]. Som vist i Figur 2.16 blir input-egenskapskartet til det første nedsamplingslaget koblet med output-egenskapskartet fra det siste oppsamplingslaget, og input-egenskapskartet til det andre

nedsamplingslaget koblet med output-egenskapskartet til det nest siste oppsamplingslaget og så videre.



Figur 2.16. Oversikt over arkitekturen til U-Net. De blå boksene representerer egenskapskart, der tallet over angir antall filtre. Pilene er operasjoner som gjøres på egenskapskartene, der typen operasjon er spesifisert nederst til høyre i bildet. Gjengitt med tillatelse fra Olaf Ronneberger [46].

Moe et al. sitt U-Net

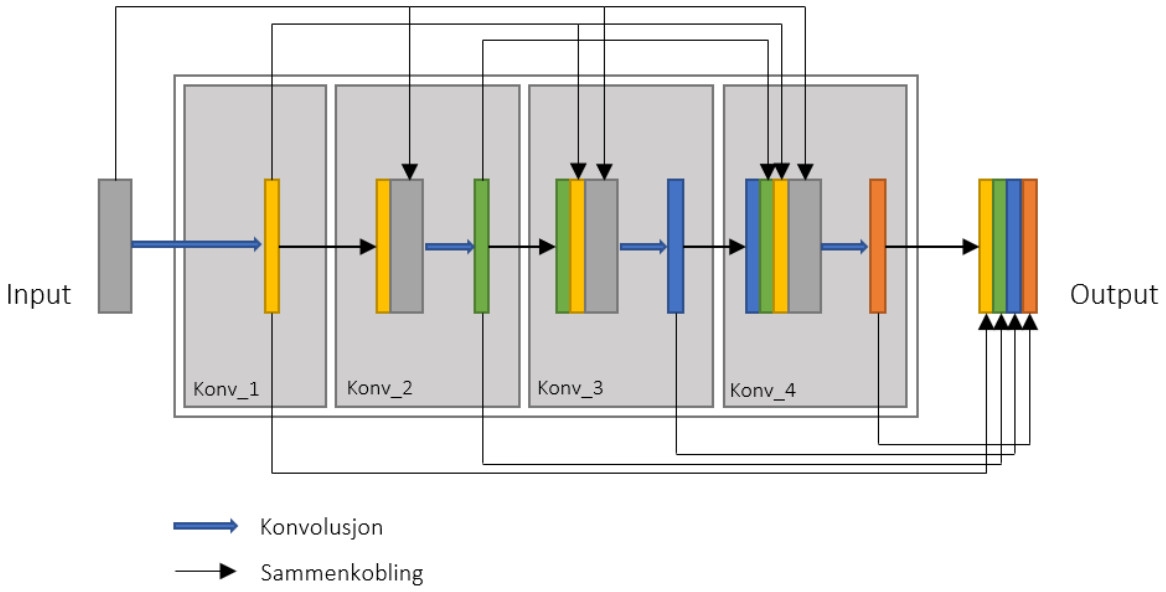
Moe et al. [10] undersøkte bruken av et U-Net til segmentering av hode- og halskreft på medisinske bilder tatt med PET og CT. Modellarkitekturen er den samme som forklart over, og består av fire nedsamplingslag i den forminskende veien, og fire oppsamplingslag i den forstørrende [10]. Det skiller seg imidlertid fra det originale U-Nettet til Ronneberger et al. [46] ved at det er tilført polstring i konvolusjonene, og at bildene ikke beskjæres før *max-pooling*-lagene [10]. I tillegg ble det brukt 3×3-konvolusjoner i den forstørrende veien istedenfor 2×2, og den siste konvolusjonen er av størrelse 1 [10]. I likhet med det originale U-Nettet, starter det første laget med 64 filtre, der antall filtre dobles for hvert forminskende lag og halveres for hvert forstørrende lag videre [10]. Antall filtre vil dermed følge denne rekken: 64 - 128 - 256 - 512 - 1024 - 512 - 256 - 128 - 64. En oversikt over arkitekturen vises i Vedlegg A. Kostfunksjonen som ble brukt er DiceLoss [42], ettersom den har vist seg å gi høye Dice-scorer for tumor-segmenteringsmodeller [10].

Moe et al. [10] testet modellen på tre forskjellige modaliteter: PET, CT og PET/CT. Det vil si at en modell ble trent og testet på bilder tatt med PET (en kanal), en på bilder tatt med kontrastforsterket CT (en kanal), og en på bilder der både PET og CT ble brukt (to kanaler). Ut fra prestasjonen til modellen ble det konkludert med at PET/CT-modaliteten gav høyest ytelse [10].

Dense-Net

Guo et al. [11] foreslår å bruke en arkitektur kalt Dense-Net til semantisk segmentering. I likhet med U-Net, består et Dense-Net av en forminskende og en forstørrende vei. Den forminskende veien trekker ut kontekstuelle trekk, mens den forstørrende gjenoppretter detaljene i bildet [11]. Mellom den forminskende og den forstørrende veien, er det hoppforbindelser som vist i Figur 2.18. Disse hoppforbindelsene overfører informasjon ved å koble lagene sammen. Det forbedrer flyten av informasjon gjennom nettverket og gjør det dermed lettere å trene [48]. Dette gjør at nettverket kan lages dypere og gi nøyaktigere svar [11, 48]. Hovedforskjellen fra U-Net er at mange av konvolusjonslagene er satt sammen til såkalte Dense-blokker.

Dense-blokker er blokker bestående av flere konvolusjonslag, med hoppforbindelser mellom alle konvolusjonslagene, som illustrert i Figur 2.17. Som vist på figuren er input-egenskapskartet og output-egenskapskartet til det første konvolusjonslaget integrert sammen og brukt som input til det andre konvolusjonslaget. Slik fortsetter det gjennom hele Dense-blokken, og til slutt er egenskapskartene fra alle konvolusjonslagene satt sammen til output-egenskapskartet til selve Dense-blokken.



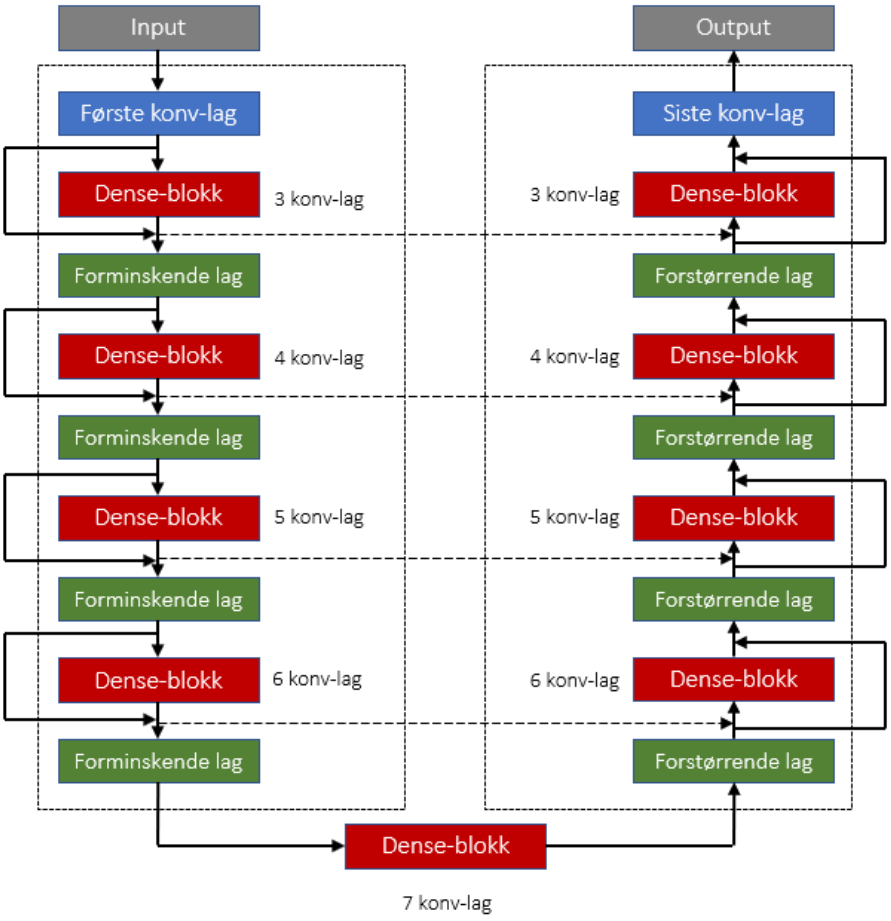
Figur 2.17. Illustrasjon av en Dense-blokk. Pilene i figuren representerer hoppforbindelser, og de fargede boksene representerer output-egenskapskart. Illustrert etter inspirasjon fra figur i Gross tumor volume segmentation for head and neck cancer radiotherapy using deep dense multi-modality network av Guo et al. [11].

Guo et al. sitt Dense-Net

Guo et al. [11] evaluerte bruken av et 3D Dense-Net til segmentering av svulster på PET/CT-bilder fra hode- og halskreftpasienter. Bildene ble preprosessert ved hjelp av normalisering og beskjæring [11]. CT-bildene ble normalisert med et intensitetsvindu, med vindussentrum i 0 HU og med en vindusbredde på 400 HU [11]. PET-bildene ble normalisert i forhold til *standardized uptake value* (SUV). Beskjæringen reduserte bildene til 3D-volum med 128×128×48 piksler, og ble foretatt for å minimere minnebruken. I tillegg ble bildene preprosessert ved hjelp av augmentering [11].

Ytelsen til nettverket ble blant annet evaluert med Dice-score, som introduseres i delkapittel 2.9.2, og ble sammenlignet med Dense-Net med enkeltmodalitet (kun PET eller CT) og 3D U-Net med multimodalitet (PET/CT) [11]. De fikk tilfredsstillende resultater på et uavhengig testsett, med en gjennomsnittlig Dice-score på 0,71 for Dense-Nettet med PET/CT-modaliteten [11]. Til sammenligning fikk Dense-Nettet med PET-modaliteten en Dice-score på 0,64, Dense-Nettet med CT-modaliteten fikk 0,31, og 3D U-Nettet med multimodalitet fikk en Dice-score på 0,69 på det samme testsettet [11]. Dermed konkluderte de med at 3D Dense-Net kan måle seg med andre ledende modeller med færre trenbare parametere, og har derfor et potensial til bruk innen segmentering av svulster på medisinske bilder [11].

Nettverksarkitekturen brukt av Guo et al. [11] inneholder 9 Dense-blokker, 4 forminskende lag og 4 forstørrende lag, som vist i Figur 2.18. Hver Dense-blokk består av et varierende antall konvolusjonslag, som starter på 3 og øker med 1 for hver forminskende overgang og minker med 1 for hver forstørrende. Modellen starter med 48 filtre, og øker med 16 for hvert forminskende steg og minker med 16 for hvert forstørrende steg [11]. Aktiveringsfunksjonen ReLU blir brukt etter alle lagene bortsett fra det siste, og batchnormalisering blir brukt som normaliseringsmetode etter konvolusjonslagene [11]. Det siste laget er en 1×1×1-konvolusjon med en sigmoid aktiveringsfunksjon, og en binær terskelverdi satt til 0,5. Utelatelse ble ikke brukt. Nettverket ble trent ved 100 epoker, og Adam ble brukt som optimaliseringsalgoritme [11].



Figur 2.18. Oversikt over arkitekturen til Dense-Nettet. Arkitekturen har ni Dense-blokker, fire forminskende lag og fire forstørrende lag. De stiplede linjene representerer lange hoppforbindelser. Illustrert etter inspirasjon fra figur i Gross tumor volume segmentation for head and neck cancer radiotherapy using deep dense multi-modality network av Guo et al. [11].

2.9 Prestasjon

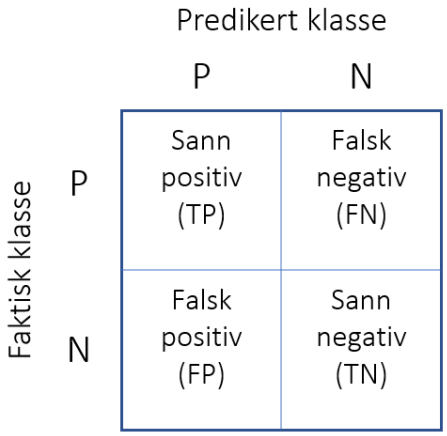
Hvordan prestasjonen til en modell måles, er avhengig av hvilken oppgave den er ment for å utføre [39]. For problemer der klassene er balansert, og hver klasse er like sannsynlig, er *accuracy* en vanlig måte å måle suksess [13]. For problemer der klassene er ubalansert, kan man for eksempel bruke *precision* eller *recall* [13]. Det finnes mange forskjellige måter å måle ytelse på, og det er heller ikke uvanlig å definere en egen måte basert på et spesielt problem [13]. De fleste ytelsesmålene ved binære problemer baserer seg imidlertid på en forvirringsmatrise [39].

2.9.1 Forvirringsmatrise

En forvirringsmatrise er en matrise som gir antall sanne positive (TP), sanne negative (TN), falske positive (FP) og falske negative (FN) klassifiseringer, som vist i Figur 2.19 [39]. De sanne positive og sanne negative er prøvene som henholdsvis ble klassifisert positivt eller negativt og faktisk var det, mens de falske positive og falske negative ble klassifisert feil. *Accuracy* kan for eksempel regnes ut ved bruk av en forvirringsmatrise, som vist i likning 2.17 [39].

$$accuracy = \frac{TP + TN}{FP + FN + TP + TN} \tag{2.17}$$

Formelen gir antall riktig klassifiserte dividert med totalt antall klassifiserte. Forvirringsmatrisen kan også brukes til å regne ut alle overlappsmål, som er en undergruppe innen prestasjonsmål [49].



Figur 2.19. Illustrasjon av en forvirringsmatrise.

2.9.2 Overlappsmål

Overlappsmål er evalueringsmetoder som bygger på overlappingen av de predikerte klassifiseringene og de sanne klassifiseringene. Forvirringsmatrisen er grunnlaget for utregningen av alle overlappsmål [49]. *True positive rate* (TPR) og *True negative rate* (TNR), også kalt Sensitivitet og Spesifisitet, er eksempler på overlappsmål [49]. De er henholdsvis gitt i likning 2.18 og 2.19.

$$TPR = Sensitivitet = \frac{TP}{TP + FN} \tag{2.18}$$

$$TNR = Spesifisitet = \frac{TN}{TN + FP} \tag{2.19}$$

Disse metodene er imidlertid ikke vanlige å bruke til validering av svulstinntegning på medisinske bilder, ettersom de vektlegger feil i små segmenter mer enn i store segmenter [49]. En annen valideringsmetode er *positive predicted value* (PPV), også kalt Presisjon. Den er gitt ved likning 2.20.

$$PPV = Presisjon = \frac{TP}{TP + FP} \tag{2.20}$$

Valideringsmetoden brukes vanligvis ikke så mye til validering av svulstinntegning på medisinske bilder direkte, men brukes sammen med *TPR* i utregningen av Dice-score, som er svært relevant på området [49].

Dice-score

Dice-score er det mest brukte prestasjonsmålet innen segmentering av volum på medisinske bilder [49]. Dice-score går ofte under navnet F1-score, og er en kombinasjon av *PPV* og *TPR* [39]. Kombinasjonen balanserer de positive og negative sidene med å optimalisere *PPV* og *TPR*. Optimaliserer man *TPR* vil man redusere sjansen for uoppdaget kreft i pasienten, men dette kan igjen øke risikoen for å predikere kreftsvulster som ikke finnes [39]. Optimaliserer man *PPV* øker man sjansen for at det som er predikert som kreft stemmer, men man øker samtidig sjansen for å gå glipp av noen prediksjoner som skulle være positive [39]. Derfor brukes Dice-score, som er gitt ved likning 2.21.

$$Dice = F1 = 2\frac{PPV * TPR}{PPV + TPR} \tag{2.21}$$

Dice-score egner seg til segmentering av svulster på medisinske bilder ettersom det tar hensyn til datasettets klassebalanse, som vil være ujevnt fordelt ved at andelen ikke-berørte piksler er betydelig større enn andelen berørte piksler. Ytelsesmålet tar hensyn til denne fordelingen ved at antall sanne negative ikke involveres i utregningen av ytelsen. Dette er en fordel ettersom modellen enkelt kan oppnå et stort antall sanne negative, men det er antall sanne positive som er av interesse.

En ulempe med Dice-score er at ytelsesmålet ikke tar hensyn til anatomisk plassering [11]. Modellen kan gjøre en inntegning i nærheten av den sanne inntegningen, men likevel oppnå en Dice-score lik null hvis de ikke overlapper. I slike tilfeller kunne prediksjonen gitt verdifull informasjon om lokasjonen til svulsten, til tross for den dårlige ytelsesscoren. Dice-score tar altså ikke hensyn til alvorlighetsgraden til feilpredikeringen, ettersom ytelsesmålet ikke angir avstanden mellom predikert og sann inntegning.

Fβ-score

F1-scoren er en spesiell versjon av den mer generelle Fβ-scoren. Man kan nemlig variere hvor mye *PPV* og *TPR* skal vektlegges ved hjelp av variabelen β. Så når $\beta = 1$, slik som i F1-scoren, vektlegges de like mye [49]. Fβ-scoren er gitt ved likning 2.22.

$$F\beta = \frac{(\beta^2 + 1) * PPV * TPR}{\beta^2 * PPV + TPR} \tag{2.22}$$

2.9.3 Avstandsbaserte ytelsesmål

Avstandsbaserte ytelsesmål brukes ofte som supplement til Dice-score i sammenheng med semantisk segmentering, da de tar hensyn til den romlige posisjonen til voxelene, og omgår dermed noen av utfordringene knyttet til Dice-score [49]. Et eksempel på et avstandsmål er Hausdorff-distanse (HD). Hausdorff-distanse er definert som den største avstanden mellom predikert inntegning og sann inntegning, og er gitt ved likning 2.23 [49]. I likningen er avstanden definert mellom punktene *A* og *B*.

$$HD\ (A, B) = \max\big(h(A, B), h(B, A)\big) \tag{2.23}$$

Der $h(A, B)$ er gitt ved

$$h(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\| \tag{2.24}$$

Her er $\|a - b\|$ en Euklidsk avstand mellom punkt *a* og *b* [49]. Ettersom HD er sensitiv for utliggere, er det anbefalt å bruke 95. persentil HD (HD_{95}), som ekskluderer de mest ekstreme utliggerne [12, 49].

Median surface distance (MSD) er et annet avstandsbasert ytelsesmål som ligner HD_{95}, men som angir medianen istedenfor 95. persentil av avstandene fra predikert inntegning til sann inntegning. Både HD_{95} og MSD angis med benevningen mm, og bør være så små som mulig for en god inntegning.

Kapittel 3: Metode

3.1 Datasettet

I denne oppgaven brukes samme datasett som ble brukt i Moe et al. [10]. Det er et datasett som består av 3D PET/CT-bilder og segmenteringsmasker fra 197 pasienter med hode- og halskreft. Pasientene ble behandlet ved Oslo Universitetssykehus (OUS) - Radiumhospitalet [50], og dataene ble samlet i tidsrommet 2007-2013 [51]. Studien ble godkjent av den regionale komitéen for medisinsk og helsefaglig forskningsetikk (REK) [12]. Dette er en forutsetning for å kunne gjøre medisinsk og helsefaglig forskning på mennesker og personlige helsedata i Norge [52].

3.1.1 Pasienter

Alle pasientene som ble inkludert i studien hadde hode- og halskreft som plateepitelkreft i munnhulen, munnsvelg («oropharynx»), strupesvelg («hypopharynx») eller i strupehodet («larynx») [51]. Strålebehandlingen av kreften ble planlagt på grunnlag av FDG PET/CT [51]. Informasjon om de 197 pasientene ble samlet inn før behandling, og finnes i Tabell 3.1. Alle pasientene er av-identifisert i ettertid [12].

Tabell 3.1. Pasientinformasjon om kjønn, gjennomsnittsalder, svulst- og lymfestadiet, posisjon og størrelse på svulstene og lymfeknutene (GTV-T og GTV-N). Tabellen er hentet fra Moe et al. [10].

Characteristic[a]	All patients ($n = 197$)	Train ($n = 142$)	Validation ($n = 15$)	Test ($n = 40$)
Age [years]				
Mean	60.3	60.7	58.8	59.4
Range	39.9–79.1	39.9–79.1	43.2–73.7	43.0–77.0
Sex				
Female	24.9 %	25.4 %	13.3 %	27.5 %
Male	75.1 %	74.7 %	86.7 %	72.5 %
TNM[b]				
T1	9.1 %	9.2 %	6.7 %	10.0 %
T2	39.6 %	39.4 %	40.0 %	40.0 %
T3	23.4 %	23.9 %	20.0 %	22.5 %
T4	27.9 %	27.5 %	33.3 %	27.5 %
N0	23.9 %	25.4 %	6.7 %	25.0 %
N1	11.7 %	12.0 %	13.3 %	10.0 %
N2	60.9 %	58.5 %	80 %	62.5 %
N3	3.6 %	4.2 %	0 %	2.5 %
AJCC/UICC[b] stage				
I	1.0 %	1.4 %	0 %	0 %
II	8.6 %	9.2 %	0 %	10.0 %
III	19.8 %	19.7 %	20.0 %	20.0 %
IV	70.1 %	69.0 %	80.0 %	70.0 %
Tumour site				
Oral cavity	8.6 %	7.0 %	26.7 %	7.5 %
Oropharynx	72.6 %	73.2 %	60.0 %	75.0 %
Hypopharynx	8.1 %	9.2 %	13.3 %	2.5 %
Larynx	10.7 %	10.1 %	0 %	15.0 %
GTV-T[c] [cm^3]				
Mean	25.0	23.9	37.3	24.3
Range	0.8–285.0	0.8–285.0	2.6–247.2	1.4–157.6
GTV-N[d] [cm^3]				
Mean	19.3	26.6	37.4	19.5
Range	0.5–276.7	0.5–276.7	2.6–247.2	0.5–76.4

[a] Percentages may not sum to exactly 100 due to rounding.
[b] 7th edition
[c] Gross primary tumour volume
[d] Involved nodal volume (for patients with nodal stage $\geq$ N1)

3.1.2 Innsamling av data

En Siemens Biograph 16 ble brukt til å utføre FDG PET/CT-skanning med kontrastforsterket CT av pasientene [51]. Etterpå tegnet en erfaren nukleærmedisinsk spesialist inn tumorvolumene (GTV-T) og de affiserte lymfeknute-volumene (GTV-N) på bildene ved hjelp av FDG-opptaket [51]. Senere tegnet onkologer inn GTV-T og GTV-N basert på kontrastforbedrede CT-bilder, klinisk informasjon og FDG-opptak i PET-bildene [51]. Inntegningene ble gjennomgått av en erfaren onkolog til slutt [12]. De affiserte lymfeknutene skulle ha samme strålingsdose som primærsvulstene ifølge dagens praksis, og ble derfor inkludert [10]. Unionen av inntegningene ble brukt til behandling av pasientene og som sann inntegning («ground truth») ved trening og evaluering av auto-segmenteringsmodeller [12, 50, 51].

3.1.3 Splitting av data

Datasettet ble delt opp i trenings-, validerings- og testsett, slik som beskrevet i Moe et al. [10]. Splittingen ble stratifisert for å sikre at fordelingen av pasienter, i henhold til TNM-klassifiseringens primærsvulst (T) sitt stadium, var tilnærmet lik mellom de forskjellige settene [53]. TNM-klassifisering er en anerkjent standard for klassifisering av omfanget og spredningen av kreftsvulster, der T står for primærtumor, N for involverte lymfeknuter og M for metastase [53]. Splittingen er beskrevet i Tabell 3.1. I tillegg ble datasettet preprosessert ved å legge til et intensitetsvindu i CT-bildene, som forklart under delkapittel 2.3.10 [10]. Sentrum på vinduet ble satt til 70 HU, og vindusbredden ble satt til 200 HU [10]. Dette vindussentrumet korresponderte med medianen til HU-verdiene innen GTV-T og GTV-N i treningssettet [10]. Til forskjell fra datasettet som ble brukt av Moe et al. [10], ble datasettet i denne oppgaven normalisert. Pikslene i bildene fikk dermed verdier mellom 0 og 1. Normalisering hjelper til med å generalisere modellen, og gjør at treningsprosessen kan konvergere raskere ved at all inputdata har samme pikselfordeling [54].

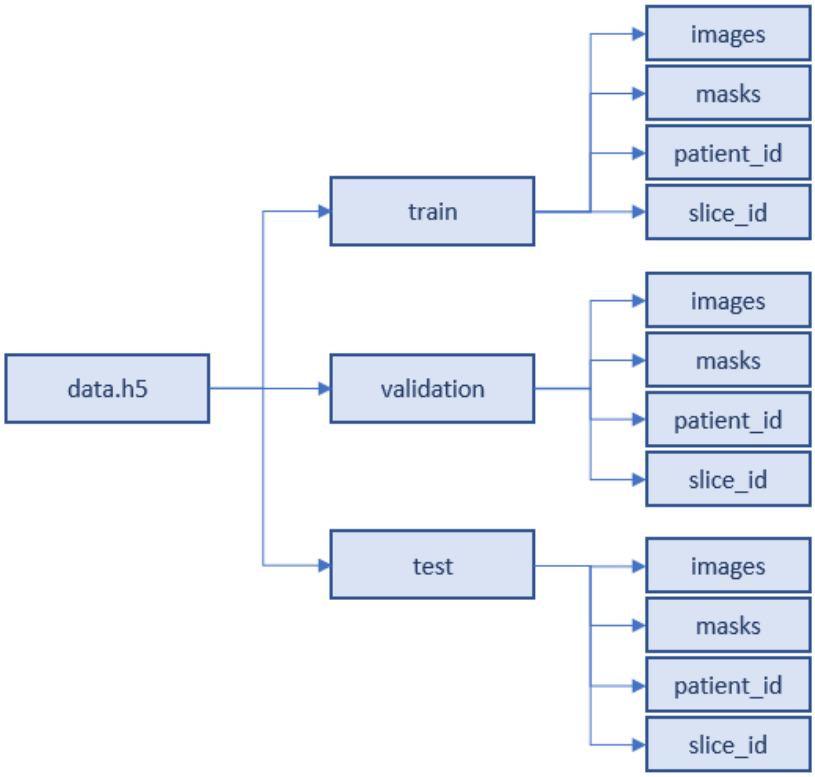
Datasettet består av 3D-bilder som blir delt opp i 2D-tverrsnitt, der hvert tverrsnitt har $x \times y$ piksler og c kanaler. I denne masteroppgaven hadde bildene i datasettet dimensjonene 191×256×2. Bildene inneholder to grupper piksler, friskt vev og berørt vev [50]. For hvert bildetverrsnitt er det m segmenteringsmasker, der segmenteringsmaskene beskriver den sanne inntegningen fra spesialistene [50]. I dette tilfellet er antall kanaler lik 2, ettersom det kun er PET og kontrastforsterket CT som brukes. Antall segmenteringsmasker er 1 fordi det er en binær segmentering. Altså er det kun berørt vev som er tegnet inn som sann inntegning.

3.1.4 Dataformat - HDF5

Datasettet ble lagret på *Hierarchical Data Format version 5* (HDF5). Dette er en praktisk måte å lagre store mengder numerisk data på [55]. HDF5 består av tre hovedelementer: datasett, grupper og attributter. Datasettene er array-liknende objekter som lagrer dataen på disken, mens grupper er hierarkiske grupperinger av disse datasettene. Attributter er metadata som forklarer gruppene eller datasettene [55].

Datasettene lagres på disken, og leses kun når de skal brukes. Man kan dermed enkelt få tilgang til en del av datasettet, uten å måtte laste inn alt til minnet først. Denne egenskapen gjør det mulig å håndtere store datasett, som i utgangspunktet ikke får plass i datamaskinens RAM («Random Access Memory») [55]. Med HDF5 kan man lese og skrive filer fra nesten alle mulige plattformer, noe som også gjør HDF5-formatet praktisk [55].

En annen fordel med HDF5 er at data kan organiseres hierarkisk, og brukeren kan organisere dataene på en måte som passer med problemet [55]. I denne oppgaven ble HDF5-filen hierarkisk strukturert ved tre grupper: *train*, *validation* og *test*. Innad i disse gruppene ligger datasettene: *images*, *masks*, *patient_id* og *slice_id*, som er vist i Figur 3.1. Innholdet i datasettene er spesifisert i Tabell 3.2.



Figur 3.1. HDF5-filens hierarkiske struktur. Train tilsvarer trenigssettet, validation tilsvarer valideringssettet og test er testsettet. Alle settene består av fire undergrupper: images, masks, patient_id og slice_id.

Tabell 3.2. Beskrivelse av datasettets format. Laget etter inspirasjon fra tabell i masteroppgaven til Moe [50].

Datasett	**Størrelse**	**Innhold**
images	[n_images, *x*, *y*, *c*]	Inputbildene.
masks	[n_images, *x*, *y*, *m*]	Segmenteringsmaskene.
patient_id		Pasientens ID-nummer.
slice_id		Gjeldende bildesnitt-nummer fra pasientens 3D-bilde.

3.1.5 Maastro-datasettet

Etter at trening og testing var gjennomført på datasettet fra Oslo universitetssykehus, skulle modellen definert som best bli testet på et eksternt datasett for å se hvordan den presterte på helt usett data. Dette datasettet inneholder PET/CT-bilder av 114 pasienter med samme diagnoser som datasettet fra OUS, og pasientene mottok tilsvarende behandling. Det eksterne datasettet er fra Maastro-klinikken i Nederland [56].

3.2 *deoxys*

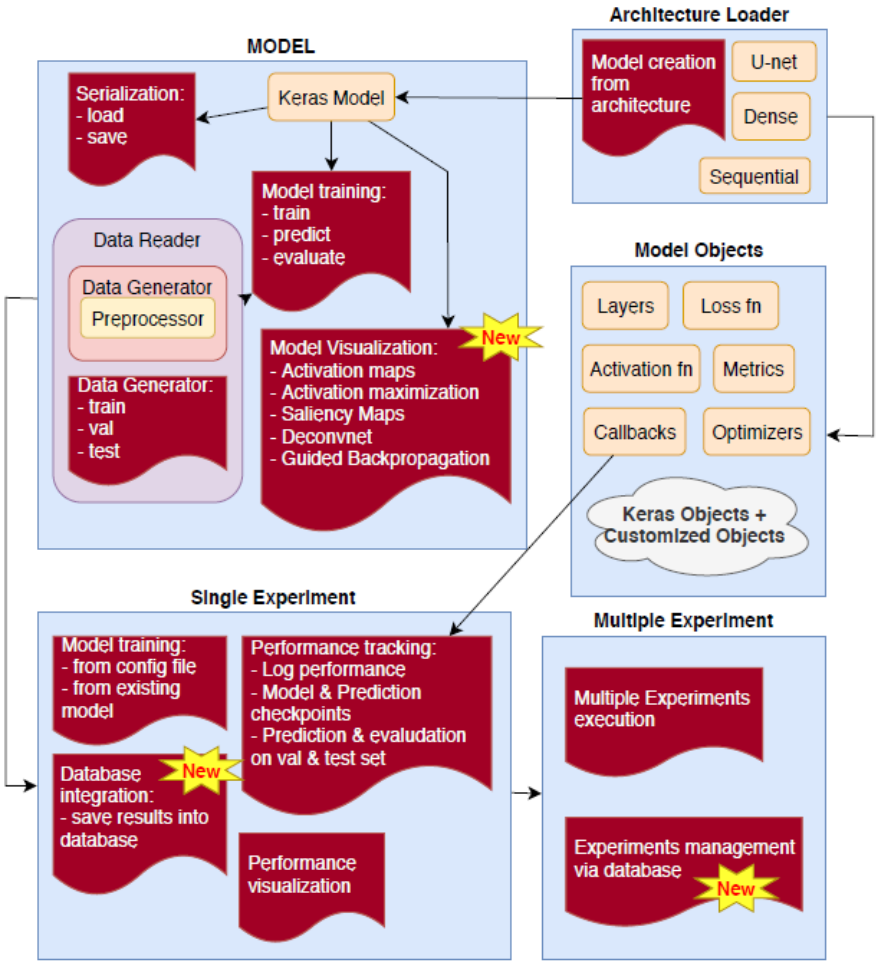
Deoxys er et rammeverk som ble utviklet av Bao Ngoc Huynh [19], og ble brukt i denne masteroppgaven. Rammeverket lar brukere lage konvolusjonsnettverk, og trene disse på et sett med bilder. Etterpå kan brukerne visualisere prestasjonen til den trente modellen [19]. Konvolusjonsnettverkene kan defineres av brukerne ved hjelp av konfigurerbare JSON-filer («JavaScript Object Notation») [19]. Detaljert beskrivelse av *deoxys* er tilgjengelig på GitHub-siden https://github.com/huynhngoc/deoxys.

3.2.1 Om rammeverket

Deoxys er Python-basert, og krever at brukeren har versjonene Python 3.7[1] og Keras 2.3.0[2] eller nyere [19]. *Deoxys* baserer seg på Keras, som er et Python-bibliotek som tilbyr byggesteiner til utvikling av dyplæringsmodeller. Byggesteinene kan for eksempel være forskjellige lag, aktiveringsfunksjoner eller optimaliseringsalgoritmer [19]. Keras fungerer som et høynivå programmeringsgrensesnitt som kommuniserer godt med andre dyplærings-serverdeler, som for eksempel TensorFlow [13, 57].

Hovedkomponentene i *deoxys* er: dataleseren, innlastningsmodulene, modellene og eksperimentene [19]. Dataleseren leser inn bildedataene og mater de til dyplæringsmodellen som skal trenes [19]. Innlastningsmodulene brukes for å bygge dyplæringsmodellen ved å laste inn konfigureringsfilene, og eksperimentene brukes til trening av modellene med forskjellige parametere [19]. Komponentene er vist i Figur 3.2.

[1] https://www.python.org/
[2] https://keras.io/



Figur 3.2. Strukturen til deoxys. Data reader representerer dataleseren, Architecture Loader og Model Objects representerer innlastningsmodulene, MODEL representerer modellene, og Single Experiment og Multiple Experiment representerer eksperimentene. Figuren er gjengitt med tillatelse fra Bao Ngoc Huynh [19].

3.2.2 Dataleseren

Dataleseren brukes til å lese data fra disken, deretter splitte dataen til trenings-, valideringss- og testsett. Den mater deretter modellen med små partier av data gjennom trenings-, valideringss- og testprosessen [19]. Dataleseren prosesserer data fra en HDF5-fil, ettersom *deoxys*-rammeverket kun støtter filer i dette formatet. Ettersom HDF5-filen i denne oppgaven allerede er gruppert inn i trenings-, valideringss- og testsett, blir det lettere for dataleseren å dele dataen inn i de samme settene.

3.2.3 Innlastningsmodulene

Innlastningsmodulene består av modulene *Architechture loader* og *Model objects*, som er vist i Figur 3.2. De lager modeller som brukeren definerer, eller fra allerede definerte arkitekturer som for eksempel U-Net eller Dense-Net [19]. Modulene kan laste inn en konfigurerbar

JSON-fil for å lage en Keras-modell basert på konfigurasjonen [19]. Innlastningsmodulene lar brukeren velge mellom forskjellige parametere. Eksempler på parametere er lag, aktiveringsfunksjoner, kostfunksjoner, optimaliseringsalgoritmer, regulariseringsmåter, ytelsesmål osv.

3.2.4 Modellene

Modeller kan lages fra JSON-konfigureringsfiler ved hjelp av innlastningsmodulene. Modellen skal inneholde metodene til en Keras-modell. Den skal kunne laste inn en modell og lagre den, tilpasse modellen til dataen, predikere et output og til slutt evaluere modellens ytelse [19]. Dataen som brukes til trening og validering av modellen, tilføres ved hjelp av dataleseren.

3.2.5 Eksperimentene

Eksperimentene gir brukerne mulighet til å trene forskjellige nettverk med forskjellige arkitekturer og parametere [19]. For hver iterasjon logges og visualiseres ytelsen til eksperimentet [19]. Man kan ha enkle og multiple eksperimenter. De enkle eksperimentene blir utført en av gangen, og bruker Keras-tilbakekallinger («Keras callbacks») for å logge ytelsen til modellen for hver epoke. Ved multiple eksperimenter, blir flere enkle eksperimenter utført samtidig, og den med høyest ytelse blir evaluert [19].

3.2.6 Databasehåndteringssystem

For å kunne organisere dataen fra eksperimentene som kjøres, lagde utvikleren av *deoxys* et databasehåndteringssystem (DBMS) [19]. *Deoxys* kommuniserer med databasen, slik at resultater fra flere eksperimenter kan lagres. DBMS administrerer disse gruppene: eksperimenter, eksperimentkjøringer, lagrede modeller, lagrede prediksjoner og ytelseslogger [19]. Alle eksperimentene som DBMS håndterer har et navn bestemt av brukeren, og inneholder informasjon om konfigurasjonen brukt i eksperimentet [19].

3.2.7 Kjøring av eksperimenter i *deoxys*

For å kjøre eksperimenter med *deoxys* må en konfigurasjonsfil lages. Konfigurasjonsfilen beskriver modellen som skal testes, og lages ved hjelp av et Python-skript som kjøres i Spyder[3]. Før konfigurasjonsfilen lages, må *deoxys* importeres. Når konfigurasjonsfilen er opprettet, lastes den opp til kode-plattformen GitHub[4] til brukerens lagringslokasjon. Derfra lastes den ned igjen til Orion, som beskrives i delkapittel 3.3, der eksperimentet kan kjøres.

[3] https://www.spyder-ide.org/
[4] https://github.com/

Resultatene av eksperimentet lagres i databasehåndteringssystemet beskrevet ovenfor, og kan evalueres i ettertid. Et eksempel på en konfigurasjonsfil som ble brukt i denne oppgaven finnes i Vedlegg B. Alle konfigurasjonsfilene som ble brukt i oppgaven kan finnes på GitHub-siden https://github.com/malenegjeng/cnn-template.

3.3 Orion

Ettersom medisinske bilder inneholder mye informasjon, er behandlingen av de svært tid- og ressurskrevende. For å unngå dette problemet ble Orion[5] brukt til kjøring av eksperimenter i denne masteroppgaven. Orion er en beregningsklynge som er styrt av *Centre for Interactive Genetics* (CIGENE), og er tilgjengelig for masterstudenter og ansatte ved NMBU [58]. Det er en SLURM-basert Linux-klynge, med 580 CPU-er, 7 TB RAM og 550 TB lagringsplass [58]. Det har også 4 GPU-er, der hver GPU har 3 grafikkort med 256 GB RAM hver [59]. SLURM er et klyngestyrings- og jobbplanleggingssystem med åpen kildekode, for Linux-klynger [60]. Den kraftige prosesseringsevnen og det store minnet til Orion øker kapasiteten på regnearbeidet, slik at beregningene utføres raskere.

For å kommunisere med Orion, må brukeren ha en Orion-bruker og laste ned en SSH-klient («Secure-Shell») som etablerer en sikker tilkobling til serveren [61]. Ved bruk av Windows som operativsystem er MobaXterm[6] et eksempel på en SSH-klient som kan brukes, og det benyttes i denne masteroppgaven. Når forbindelsen er klar har brukeren tilgang til sin hjemmekatalog, og kan redigere skripter, administrere filer og sende inn jobber som skal utføres av Orion [61]. For å kjøre eksperimenter i Orion, må kildekoden, datasettet og JSON-konfigurasjonsfilene ligge i MobaXterm.

3.4 Sammenligningsmodellen U-Net

I denne oppgaven ble resultatene målt mot en sammenligningsmodell for å undersøke om 2D Dense-Net kan gi like gode, eller bedre, automatiske inntegninger. Sammenligningsmodellen tilsvarer U-Net-modellen brukt i Moe et al. [10], Groendahl et al. [12] og i masteroppgaven til Bao Ngoc Huynh [19]. Groendahl et al. [12] og Huynh [19] benyttet den samme modellen som i Moe et al. [10], som er beskrevet i delkapittel 2.8.3. Modellen har vist seg å være lovende for bruk innen segmentering av berørt vev i medisinske bilder.

[5] https://nmbu-orion-support.readthedocs.io/en/latest/index.html
[6] https://mobaxterm.mobatek.net/

3.4.1 Parametere i sammenligningsmodellen

Kostfunksjonen som ble brukt i sammenligningsmodellen var DiceLoss, som ble beskrevet i delkapittel 2.8.2. PET/CT-modaliteten ble brukt som sammenligningsgrunnlag i denne oppgaven, ettersom den gav høyest ytelse for Moe et al. [10]. Sammenligningsmodellen brukte Adam som optimaliseringsalgoritme, som er beskrevet i delkapittel 2.8.2. Moe et al. [10] og Huynh [19] brukte en læringsrate på 10^{-4}, mens Groendahl et al. [12] brukte en læringsrate på 10^{-5}. Aktiveringsfunksjonen ReLU ble brukt i alle lagene bortsett fra det siste, der Sigmoid ble brukt for å bestemme om en piksel var en del av en tumor eller ikke [19]. I tillegg var batchnormalisering, som forklares nærmere under delkapittel 3.5.2, lagt til som normaliseringsmetode etter hvert konvolusjonslag [10, 19]. Utelatelse ble ikke brukt.

3.5 2D Dense-Net

Modellen som ble målt mot sammenligningsmodellen i denne oppgaven var 2D Dense-Net. Dense-Net er en CNN-arkitektur som likner U-Net, og er beskrevet under delkapittel 2.8.3.

3.5.1 Dense-Net brukt i denne oppgaven

Guo et al. [11] sitt 3D Dense-Net, som beskrives under delkapittel 2.8.3, ble brukt som utgangspunkt for 2D Dense-Nettet som ble testet i denne masteroppgaven. 2D Dense-Nettet skiller seg fra 3D Dense-Nettet ved at det segmenterer svulster i et 2D-tverrsnitt istedenfor 3D-volum. 2D Dense-Nettet som er definert i *deoxys* har samme struktur som Guo et al. [11] sitt, der den eneste forskjellen er at det er brukt 2D-konvolusjonslag istedenfor 3D-konvolusjonslag [54]. Dette er standardmodellen i *deoxys*, men parameterne i arkitekturen kan endres slik brukeren ønsker. 2D Dense-Nettet som er definert i *deoxys* er ikke testet ut tidligere, og ble dermed testet ut for første gang i denne oppgaven.

3.5.2 Parametere

For å kunne optimalisere 2D Dense-Nettet, ble fire parametere valgt ut til justering. Disse fire parameterne var antall filtre, læringsraten, batchnormalisering og utelatelsesraten. Nivåene som ble testet hos de forskjellige parameterne i forskjellige kombinasjoner vises i Tabell 3.3. Det første tallet ved antall filtre (32, 48 eller 64) angir antall filtre i første lag, og «+16» indikerer at 16 adderes for hver forminskende overgang, og subtraheres for hver forstørrende overgang.

Tabell 3.3. Parameterne med tilhørende verdier som ble utforsket i denne oppgaven.

Parameter	Verdier
Antall filtre	32 + 16, 48 + 16, 64 + 16
Læringsrate	10^{-3}, 10^{-4}, 10^{-5}
Batchnormalisering	Ja, Nei
Utelatelsesrate	0, 0,5

Batchnormalisering

Normalisering er en metode for å generalisere data, slik at ulike typer data ser mer like ut for en maskinlæringsmodell [13]. Dette hjelper maskinen med å lære og generalisere ny data bedre [13]. Batchnormalisering er et lag som normaliserer dataen underveis i treningsprosessen, og laget legges ofte etter konvolusjonslag eller dense-lag i nettverket [13]. Uten batchnormalisering endres fordelingen av inputdataen til hvert lag, og dette sakker ned treningsprosessen og krever en lavere læringsrate [62]. Ettersom batchnormalisering generaliserer dataen underveis som modellen trenes, gir dette rom for en høyere læringsrate og en dypere modell [62]. Det kan også fungere som en regulariseringsmetode, og kan dermed fjerne behovet for utelatelse [62]. I denne oppgaven ble modellen testet med og uten batchnormalisering etter konvolusjonslagene.

Utelatelsesraten

Utelatelse er en regulariseringsteknikk, som kort forklares under delkapittel 2.8.2. Utelatelsesraten angir hvor stor andel av nevronene og deres tilhørende input- og outputverdier, som skal utelates i et lag under treningsprosessen [40]. Hver node har da en sannsynlighet for å utelates fra laget, og den sannsynligheten er lik utelatelsesraten [40]. Utelatelse sørger dermed for at ingen nevroner avhenger for mye av andre nevroner, slik at alle delene av det nevrale nettverket bidrar til læringen [40]. Utelatelsesrate-verdiene som ble testet i denne oppgaven var 0 og 0,5. Det vil si at ingen nevroner ble droppet, og at halvparten av nevronene ble droppet i laget.

Læringsraten

Læringsraten angir størrelsen til stegene mot bunnpunktet av kostfunksjonen, som forklart i delkapittel 2.8.2. Læringsraten er typisk mye mindre enn 1, og ofte mindre enn 0,1 [40]. En stor læringsrate kan bidra til raskere konvergering mot bunnpunktet. Blir læringsraten for stor, risikerer man å hoppe over bunnpunktet. En mindre læringsrate kan hjelpe med å unngå dette

problemet, men vil sannsynligvis sakke ned prosessen betydelig. Tre forskjellige læringsrater ble testet i denne oppgaven, og det var 10^{-3}, 10^{-4} og 10^{-5}.

Antall filtre

Filtre er inkludert i konvolusjonslag og oppdager romlige mønstre i bilder. Et eksempel er detektering av kanter ved å registrere endring i intensitet. Filtrene detekterer altså egenskaper i bildet. Antall filtre i et lag angir antallet kanaler i output-egenskapskartet til laget. Flere filtre kan gi en mer nøyaktig modell, men samtidig kan for mange filtre gjøre modellen overtilpasset [13]. I denne oppgaven ble tre forskjellige filterantall testet i første lag: 32, 48 og 64. Antall filtre økte med 16 i de forminskende overgangene, og minket med 16 i de forstørrende overgangene.

Andre parametere

Det finnes flere parametere i 2D Dense-Net som kan justeres i tillegg til disse fire, men de ble satt til standardverdiene i *deoxys*. Disse verdiene ble bestemt basert på tidligere forskning, inkludert modellen til Guo et al. [11]. Adam ble brukt som optimaliseringsalgoritme, ettersom den er kjent for å gi gode resultater og være rask [43]. Adam gav også høyest Dice-score i [50]. I tillegg ble kostfunksjonen DiceLoss brukt, ettersom den gav gode resultater for Moe et al. [10] og Huynh [19]. Aktiveringsfunksjonen ReLU viste seg å gi god effekt for Moe et al. [10], og ble dermed brukt i denne oppgaven.

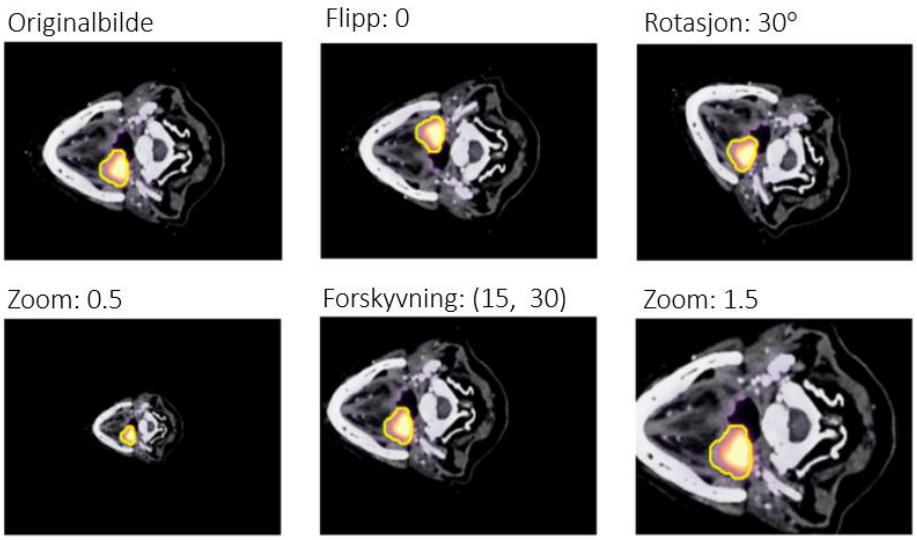
3.5.3 Augmentering

Augmentering ble undersøkt for å vurdere om dette kunne øke modellytelsen. Augmentering er, som nevnt under delkapittel 2.7.2, å legge til endrede deler av datasettet slik at variasjonen i datasettet utvides. Augmenteringsteknikker kan deles inn i to grupper: affine transformasjoner og filter- og punktoperasjoner. Affine transformasjoner beskriver geometriske endringer i bildet, mens filter- og punktoperasjoner endrer pikslenes intensitetsverdier [63].

Kombinasjonen av augmenteringskomponenter som ble brukt i denne oppgaven, var basert på resultatene fra Ødegaard sin masteroppgave [64] som gav høyest Dice-score. Disse resultatene utgjorde kun affine transformasjoner, og verdiene er vist og forklart i Tabell 3.4. Figur 3.3 viser en visualisering av de forskjellige augmenteringsteknikkene.

Tabell 3.4. Kombinasjon av augmenteringskomponenter som gav høyest Dice-score for 2D U-Nettet brukt på hode- og halskreftdatasettet i Ødegaard sin masteroppgave [64], med beskrivelse av hva de gjør med de originale bildene [63].

Augmentering	Verdi	Beskrivelse
Rotasjon	90	Modellen velger et tilfeldig tall mellom -90° og 90°, som angir rotasjonsvinkelen til bildet.
Zoom	0,5 – 1,5	Bildet kan bli zoomet inn eller ut med et tilfeldig tall mellom 0,5 og 1,5.
Forskyvning	10, 10	Bildet kan bli forskjøvet med maksimalt 10 og minimalt -10 antall piksler på x-aksen og y-aksen.
Flipp	0	Bildet speiles, og kan entes speiles horisontalt eller vertikalt. Verdien 0 gir en vertikal speiling om x-aksen, mens verdien 1 gir en horisontal speiling om y-aksen.



Figur 3.3. Visualisering av eksempler på de forskjellige augmenteringsteknikkene. Både PET- og CT-kanalen vises i alle bildene.

3.6 Eksperimentplan

3.6.1 Eksperimentoppsett

For å optimalisere 2D Dense-Nettet, ble ulike kombinasjoner av parameterne i Tabell 3.3 testet. Til sammen utgjorde dette 36 eksperimentmodeller, som er vist i Vedlegg A. De fleste parameternivåene ble valgt ut på forhånd, men læringsraten (lr) ble vurdert underveis i kjøringen av modellene. Modeller med lr lik 10^{-3} og lr lik 10^{-4} ble først testet, for så å avgjøre om lr lik 10^{-2} eller lr lik 10^{-5} skulle testes videre. Beslutningen ble tatt basert på om

læringsraten lr lik 10^{-3} (valg lr lik 10^{-2}) eller lr lik 10^{-4} (valg lr lik 10^{-5}) gav den gjennomsnittlig høyeste Dice-scoren per tverrsnitt. Alle modellene var programmert til kjøring i 200 epoker, med en partistørrelse på 16. Det ble imidlertid lagt til en kode som sørget for å stoppe kjøringen hvis ytelsen ikke hadde forbedret seg de siste 30 epokene. Dette ble gjort for å spare tid og lagringsplass ved kjøring av eksperimentene i Orion.

Parameterne ble undersøkt med statistiske tester for å undersøke hvilke nivåer av parameterne som bidro til høyest Dice-score. Kombinasjonen av parameternivåene som scoret høyest i de statistiske testene, var modellen som ble valgt for uttesting med augmentering. Deretter ble modellen testet på testsettet, for til slutt å testes på det eksterne datasettet fra Maastro-klinikken. Dette var for å vurdere hvordan den håndterte ny og usett data. I tillegg ble ytelsesmålene HD_{95} og MSD, som er beskrevet i delkapittel 2.9.3, utregnet for modellen på OUS- og Maastro-testsettet. Dette var for å gi ytterligere informasjon om kvaliteten til inntegningene.

3.6.2 Trening og test

Treningen og valideringen av hver modell tok mellom 4 til 48 timer i Orion. Følgelig ble det kjørt om lag to modeller hver dag i en periode på rundt fire uker. Orion har et begrenset minne per bruker, så det var kun resultatdata fra epoken med høyest Dice-score fra hver modell som ble lagret. Det ble også tatt sikkerhetskopier av resultatfilene på en NMBU-server underveis.

3.6.3 Statistikk

Justering av resultatdata

Orion gav opprinnelig ut 36 gjennomsnittlige Dice-scorer per tverrsnitt fra de 36 modellene som ble kjørt. På grunnlag av resultatfilene, ble Dice-scoren til hvert tverrsnitt for alle pasientene i de 36 modellene beregnet. I tillegg ble den gjennomsnittlige Dice-scoren per pasient for alle modellene beregnet. Disse resultatene ble samlet i et datasett per modell med 1033 Dice-scorer per tverrsnitt («Dice_per_slice») og et datasett med 15 gjennomsnittlige Dice-scorer per pasient («Dice_per_patient»). 1033 er antallet bildetverrsnitt i valideringssettet, og 15 er antall pasienter i valideringssettet. Alle datasettene med Dice-scorer per tverrsnitt fra de 36 modellene ble samlet i et felles resultatdatasett med 37 188 Dice-scorer per tverrsnitt. Disse justeringene ble utført i Spyder.

Evaluering av resultatdata

Videre skulle disse resultatsettene brukes for å evaluere hvilken modell som hadde størst potensiale til å foreta en nøyaktig svulstinntegning. Den planlagte evalueringsmetoden var

den statistiske testen N-veis ANOVA [65]. Denne testen kan imidlertid kun brukes hvis dataen oppfyller visse krav, og ett av disse kravene er at residualene til dataen må være normalfordelt. Ettersom det på forhånd var usikkerhet rundt oppfyllingen av disse kravene, var det planlagt tre ulike scenarioer for hvordan videre evaluering av resultatene skulle foregå. Diagnoseplott og en Anderson-Darling-test ble brukt for å undersøke normalitetskravet [66]. Hypotesene ble angitt slik:

H_0: Residualene er normalfordelt

H_1: Residualene er ikke normalfordelt

Scenario 1 skulle utføres dersom kravene var oppfylt og residualene var normalfordelt. Da skulle N-veis ANOVA brukes for å vurdere forskjellene mellom parameterne og interaksjoner mellom dem [65]. En post-hoc-test skulle videre brukes for å evaluere hvilken modell som var best egnet til segmenteringsoppgavene.

Scenario 2 skulle utføres dersom residualene ikke var normalfordelt. Da skulle det testes om transformering av dataen kunne bidra til normalfordeling. Transformeringsmetodene som skulle testes var log-, kvadratrot-, invers- og boxcox-transformering [67]. Hvis dette gav normalfordelte residualer skulle N-veis ANOVA gjennomføres etterfulgt av post-hoc, slik som scenario 1 beskriver.

Scenario 3 skulle utføres dersom transformeringen ikke gav normalfordelte resultater. Da skulle det brukes ikke-parametriske statistiske tester som ikke krever normalfordeling. Valget falt da på Friedmantest [68] med tosidig Nemenyi post-hoc-test [69] for vurdering av parameterne med tre eller flere nivåer (antall filtre og læringsraten), og tosidig Wilcoxon signed rank-test [69] for vurdering av parameterne med to nivåer (batchnormalisering og utelatelsesraten). Hypotesene som skulle testes var:

H_0: Valg av parameternivå utgjør ingen forskjell for ytelsen til modellen

H_1: Valg av parameternivå utgjør en forskjell for ytelsen til modellen

I tillegg skulle boksplott brukes for å visualisere resultatet fra testene, og avgjøre hvilke nivåer som bidro mest til en økning i Dice-score. Dette er nødvendig ettersom testene kun gir informasjon om det finnes en signifikant forskjell mellom nivåene, men ikke hvilket nivå som gir høyest Dice-score. Modellen med kombinasjonen av de parameterne som bidro mest til at Dice-scoren økte skulle dermed bli definert som den beste modellen.

Dersom det siste scenarioet ble utfallet, måtte datasettet justeres ytterligere. Friedmantesten krever data som ikke inneholder replikasjoner [68]. Dermed måtte et datasett på formen *unreplicated complete block design* lages. Dette kunne gjøres ved at radene per eksperiment ble tildelt en indeksverdi i en kolonne kalt Teller. Den gjennomsnittlige Dice-scoren måtte regnes ut for alle tverrsnittene med samme Teller-verdi og samme parameternivå. Dette kunne dermed settes sammen til fire forskjellige datasett, ett for hver parameter.

Datavisualiseringen og de statistiske beregningene ble gjennomført i programmet RStudio[7]. Skriptene finnes i Vedlegg C. Statistisk signifikans ble akseptert når p-verdien < 0,05.

[7] https://www.r-project.org/

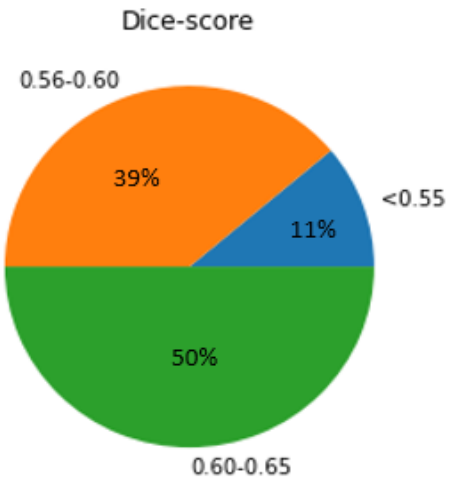
Kapittel 4: Resultater

4.1 Optimalisering av 2D Dense-Net

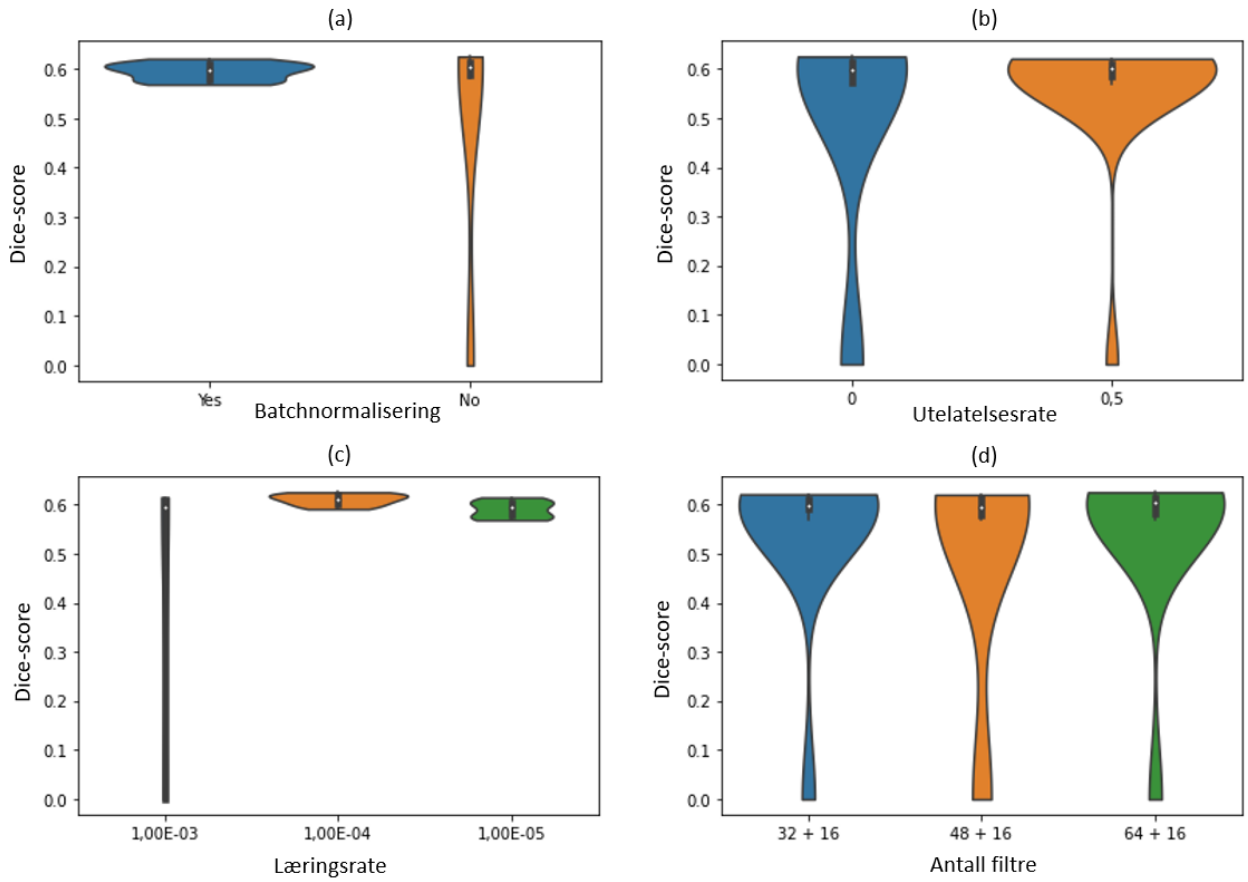
For å optimalisere 2D Dense-Net-modellen, ble hyperparameterne batchnormalisering, utelatelse, læringsrate og antall filtre justert. Effekten disse parameterne hadde på modellytelsen blir beskrevet i dette kapittelet. Totalt ble 36 modeller kjørt, som beskrevet i Tabell B.1 i Vedlegg B. Modellen bestående av parameterne som bidro mest til økning i Dice-score ble definert som best, og ble kjørt med augmentering til slutt. I tillegg ble denne modellen, med og uten augmentering, kjørt på OUS-testsettet og Maastro-datasettet.

4.2 Resultater fra eksperimentene

En gjennomsnittlig Dice-score per tverrsnitt for hver eksperimentmodell som ble kjørt på valideringssettet ble beregnet, og er visualisert ved et sektordiagram som vises i Figur 4.1. Tabellen med resultatene ligger i Vedlegg B. De fleste modellene hadde en Dice-score over 0,55, og majoriteten av disse lå mellom 0,60 og 0,63. Noen modeller (modell 2, 4, 14 og 26) oppnådde en Dice-score tilnærmet null. Dette antyder i gjennomsnitt ingen overlapp mellom predikert inntegning av Dense-Net-modellen og sann inntegning. Det som kjennetegnet disse feil-modellene, var at ingen benyttet batchnormalisering og alle brukte 10^{-3} som læringsrate. Videre benyttet tre av de en utelatelsesrate lik null, og to av modellene brukte 48 filtre. Fiolinplottene vist i Figur 4.2 ble brukt for å visualisere effekten av de forskjellige parameterne på modellytelsen.



Figur 4.1. Sektordiagram som viser fordelingen av gjennomsnittlig Dice-score per tverrsnitt for de 36 modellene med forskjellige parameterkombinasjoner.



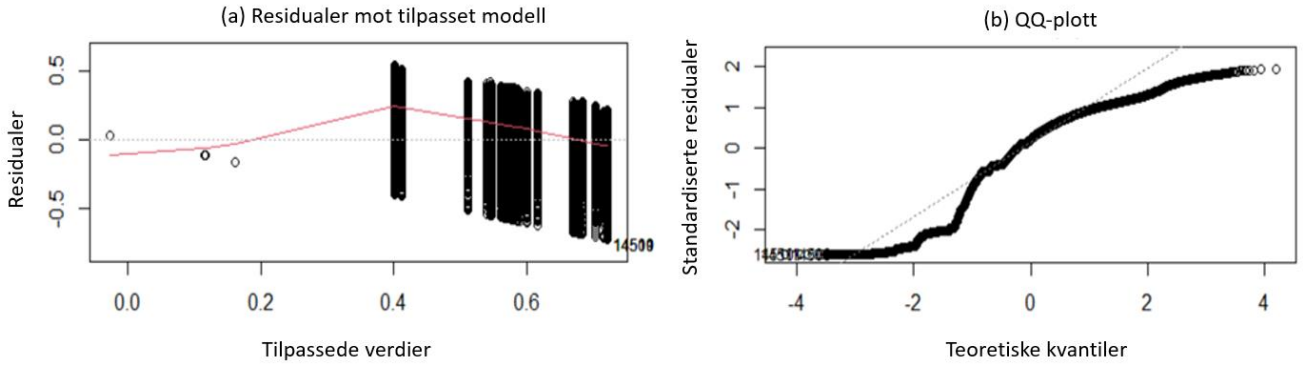
Figur 4.2. Fiolinplott som viser parameternivåene batchnormalisering (a), utelatelse (b), læringsrate (c) og antall filtre (d) mot gjennomsnittlig Dice-score per tverrsnitt for alle modellene på valideringssettet. Formen til fiolinene representerer fordelingen av Dice-score per parameternivå, og den hvite prikken angir medianen. Bredden representerer antallet modeller med den gitte Dice-scoreverdien, og lengden representerer variansen i Dice-score til modellene med det gitte parameternivået.

Fra Figur 4.2 er det tydelig at bruk av batchnormalisering i gjennomsnitt gav en betydelig høyere Dice-score, og lavere standardavvik enn uten. Uten batchnormalisering var ytelsen variabel, og den spenner over en Dice-score i området 0,0 til ca. 0,6. Det samme gjelder for læringsraten, der en læringsrate på 10^{-4} og 10^{-5} gav en høyere Dice-score og et lavere standardavvik enn 10^{-3}. Det sistnevnte parameternivået gav en variabel ytelse som også spenner over en Dice-score fra 0,0 til ca. 0,6. Effekten av utelatelsesraten og antall filtre var ikke tydelige. Alle parameternivåene var variable, og spenner også her over en Dice-score fra 0,0 til ca. 0,6. En utelatelsesrate på 0,5, og antall filtre på 32 eller 64 så imidlertid ut til å gi flere modeller med en Dice-score nærme 0,6.

4.3 Statistiske tester

4.3.1 Normalfordeling

Som beskrevet i delkapittel 3.6.3 ble det undersøkt om dataen oppfylte kravene for N-veis ANOVA, ettersom det i utgangspunktet var planlagt å bruke denne testen til evaluering av effekten av parameterne på modellytelsen. Dette ble undersøkt ved hjelp av diagnoseplott, som viser om residualene til datasettet er normalfordelt. Datasettet med Dice-score per tverrsnitt for alle tverrsnittene i valideringssettet ble brukt for å lage plottene. De to diagnoseplottene som ble laget var et plott som viste residualer mot tilpasset modell, og et QQ-plott, som vist i Figur 4.3. Er resultatene normalfordelt, vil den røde kurven følge den horisontale aksen i plottet med residualer mot tilpasset modell. I QQ-plottet vil en normalfordeling vises ved at punktene følger den rette stiplede linjen. Residualene var dermed ikke normalfordelt basert på Figur 4.3.

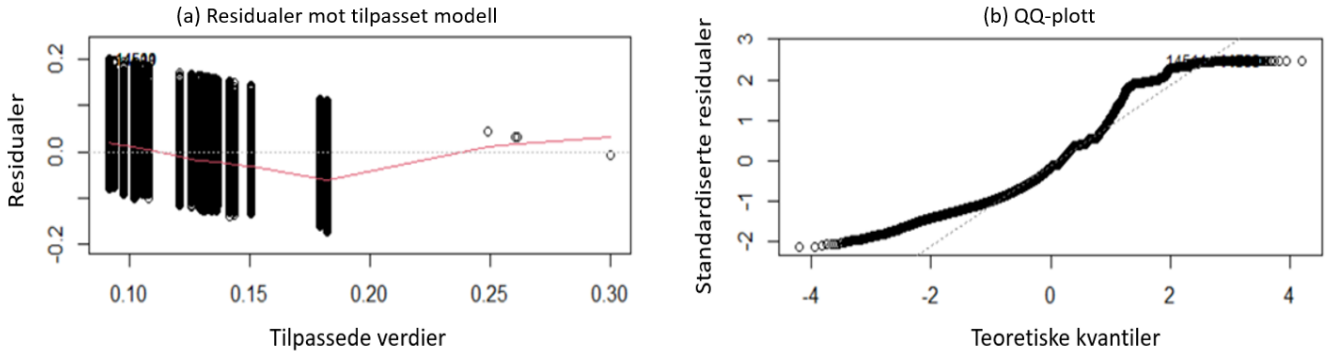


Figur 4.3. Her vises residualer mot tilpasset modell (a) og QQ-plott (b) for datasettet med Dice-score per tverrsnitt for alle tverrsnittene i valideringssettet.

En Anderson-Darling-test ble benyttet for å bekrefte at residualene ikke var normalfordelt. [66]. Denne testen gav en p-verdi på $p < 0{,}001$, som indikerte at nullhypotesen kunne forkastes. Residualene var dermed ikke normalfordelte og ANOVA kunne ikke brukes. Det ble forsøkt å transformere dataen slik at normalitetskravet ble oppfylt. Hverken log-, kvadratrot-, invers- eller boxcox-transformering gav normalfordelte residualer. Diagnoseplottene etter log-transformering vises i Figur 4.4, resten av plottene ligger i Vedlegg C. Disse plottene viser det samme som Figur 4.4, at residualene ikke fulgte en normalfordeling. Tabell 4.1 viser at Anderson-Darling-testen gav en p-verdi på $p < 0{,}001$ for alle tilfellene. Dette betyr at ANOVA ikke kunne brukes for å analysere disse resultatene.

Tabell 4.1. Resultatet av Anderson-Darling-testen etter de forskjellige transformasjonene.

Transformering	p-verdi
Log	< 0,001
Kvadratrot	< 0,001
Invers	< 0,001
Boxcox	< 0,001



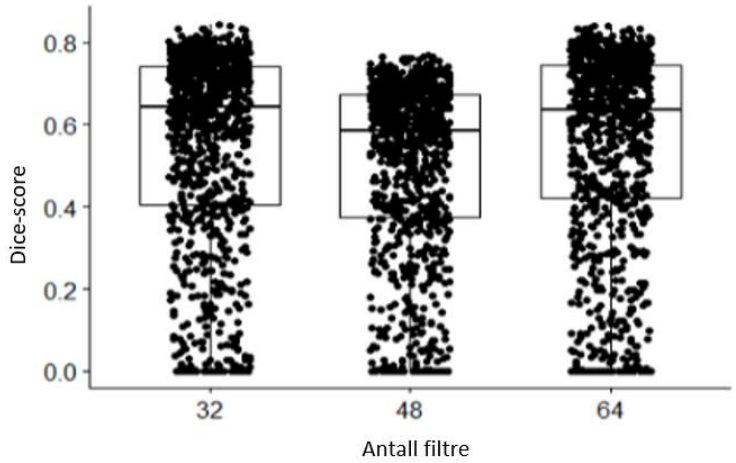
Figur 4.4. Her vises residualer mot tilpasset modell (a) og QQ-plott (b) for datasettet med Dice-score per tverrsnitt for alle modellene etter log-transformeringen.

4.3.2 Effekt av parameterne

Ettersom residualene ikke var normalfordelte kunne ikke N-veis ANOVA gjennomføres. Derfor ble en Friedmantest etterfulgt av en Nemenyi post-hoc-test gjennomført på parameterne med tre eller flere nivåer (antall filtre og læringsraten). Wilcoxon signed rank-test ble brukt på parameterne med to nivåer (batchnormalisering og utelatelsesraten).

Antall filtre

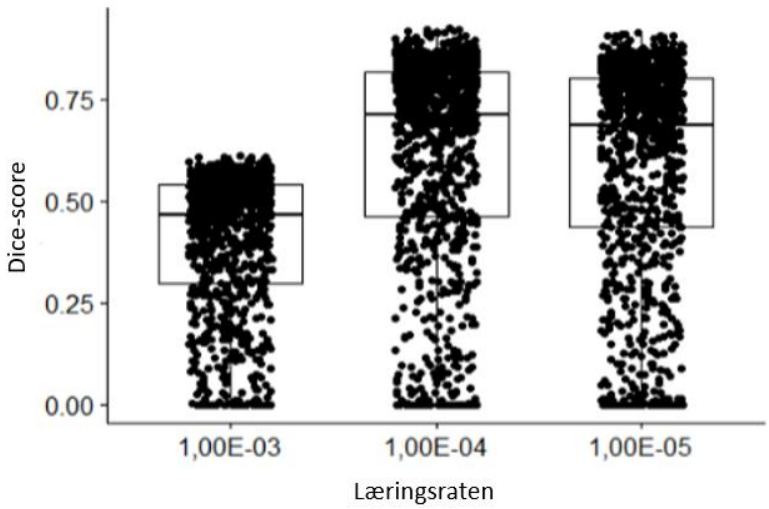
Friedmantesten gav p < 0,001 for antall filtre, som indikerte at nullhypotesen kunne forkastes. Dette antyder at antall filtre hadde en påvirkning på ytelsen. Videre gav Nemenyitesten p < 0,001 mellom alle nivåene. For å undersøke forskjellene nærmere ble et boksplott laget, som vist i Figur 4.5. Plottet viser at både 32 filtre og 64 filtre gav en noe høyere Dice-score enn 48. Testen om effektstørrelse gav parameteren en moderat effekt.



Figur 4.5. Boksplott som angir Dice-score mot parameteren antall filtre. Prikkene tilsvarer gjennomsnittlige Dice-score for ett tverrsnitt, midlet over alle modellene. Den horisontale streken i boksene viser medianen til Dice-scorene per tverrsnitt for det gitte parameternivået.

Læringsraten

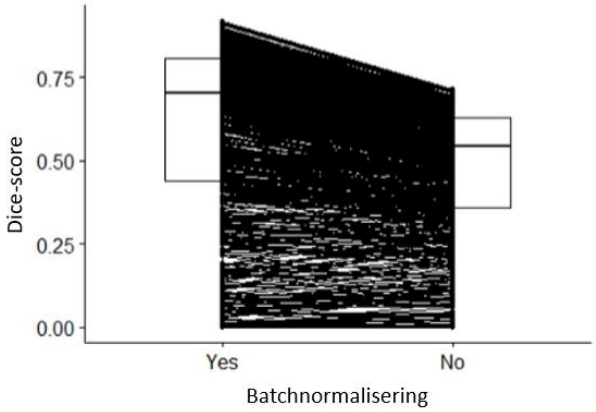
Friedmantesten gav $p < 0{,}001$ for læringsraten, som indikerte at nullhypotesen kunne forkastes. Nemenyitesten gav $p < 0{,}001$ mellom alle nivåene. Boksplottet gitt i Figur 4.6 viser at læringsraten 10^{-4} og 10^{-5} gav høyest Dice-score. I tillegg bekreftet effekttesten at parameternivået hadde en høy effekt.



Figur 4.6. Boksplott som angir Dice-score mot parameteren læringsraten. Prikkene tilsvarer gjennomsnittlige Dice-score for ett tverrsnitt, midlet over alle modellene. Den horisontale streken i boksene viser medianen Dice-scorene per tverrsnitt for det gitte parameternivået.

Batchnormalisering

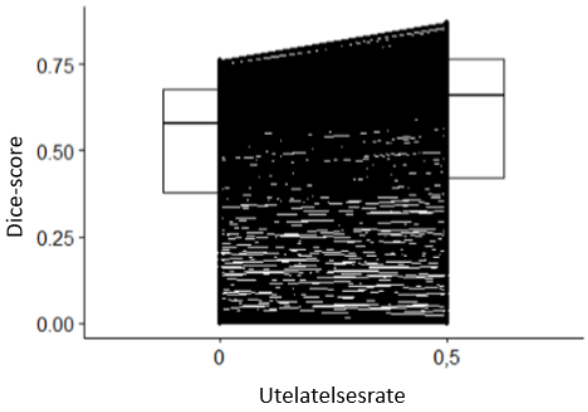
Wilcoxon signed rank-testen gav $p < 0{,}001$, som indikerte at nullhypotesen kunne forkastes og at forskjellen var signifikant. Effekttesten bekreftet også at parameteren hadde en høy effekt. Basert på boksplottet i Figur 4.7, er det tydelig at modellene med batchnormalisering gav en høyere Dice-score enn de uten.



Figur 4.7. Boksplott som angir Dice-score mot parameteren batchnormalisering. Strekene tilsvarer sammenhengen mellom Dice-scoren til samme tverrsnitt til samme pasient, men med ulikt parameternivå. Den horisontale streken i boksene viser medianen til Dice-scorene per tverrsnitt for det gitte parameternivået.

Utelatelsesraten

Wilcoxon signed rank-testen gav $p < 0{,}001$, som indikerte at nullhypotesen kunne forkastes og at forskjellen var signifikant. Effekttesten bekreftet også her at effekten av parameteren var høy. Basert på boksplottet i Figur 4.8, er det tydelig at utelatelsesraten på 0,5 gav høyere Dice-score enn ingen utelatelsesrate.



Figur 4.8. Boksplott som angir Dice-score mot parameteren utelatelse. Strekene tilsvarer sammenhengen mellom Dice-scoren til samme tverrsnitt til samme pasient, men med ulikt parameternivå. Den horisontale streken i boksene viser medianen til Dice-scorene per tverrsnitt for det gitte parameternivået.

4.4 Beste modell

Den beste modellen ble definert til å være modellen med kombinasjonen av parameterne som gav den høyeste Dice-scoren. Dermed ble modellen med 32 filtre i første lag, 10^{-4} som læringsrate, med batchnormalisering og med 0,5 i utelatelsesrate valgt som beste modell. Den beste modellens parametere er vist i Tabell 4.2. Basert på Friedmantesten og Nemenyitesten kunne 64 filtre og en læringsrate på 10^{-5} også blitt valgt, men ettersom 32 filtre gir en enklere modell og treningen går saktere med en mindre læringsrate så falt valget på disse parameternivåene.

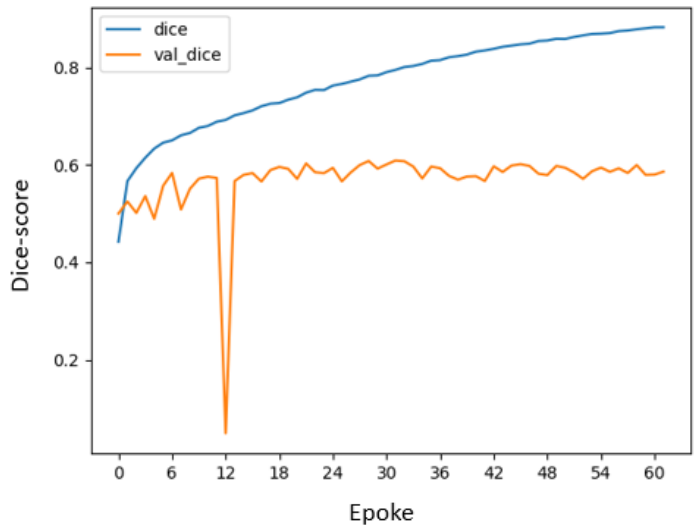
Tabell 4.2. Parameternivåene til den beste 2D Dense-Net-modellen.

Parameter	Verdier
Antall filtre	32
Læringsrate	10^{-4}
Batchnormalisering	Yes
Utelatelsesrate	0,5

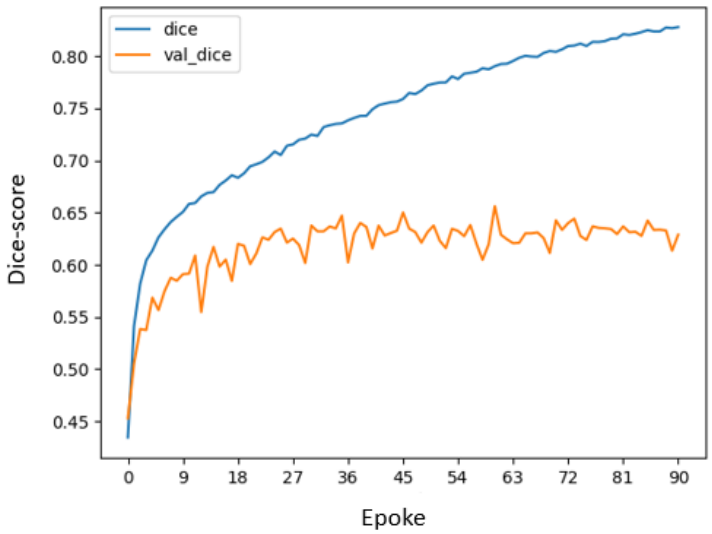
4.4.1 Modellytelse på valideringssettet

Trenings- og valideringskurven vises i Figur 4.9 og Figur 4.10 for den beste modellen, henholdsvis uten og med augmentering. Den blå kurven viser Dice-score per epoke under treningen, mens den oransje viser Dice-score per epoke på valideringssettet. Disse grafene kan gi et innblikk i om modellen er undertilpasset eller overtilpasset. I Figur 4.9 viker kurvene fra hverandre i økende grad for hver epoke. Treningskurven øker jo mer modellen trenes, og når en Dice-score rundt 0,90. Valideringskurven forholder seg relativt flat på rundt 0,60 i Dice-score per tverrsnitt. Den oppnår sin høyeste score rundt epoke 30 før den synker gradvis. Det betyr at modellen var godt tilpasset treningssettet, men at den slet med å generalisere valideringsdataen. Denne trenden kan tyde på at modellen var overtilpasset. For Figur 4.10 ser trenden noe annerledes ut, ettersom den oransje kurven følger den blå kurven noe tydeligere. Den blå kurven øker jo mer modellen trenes, og når en Dice-score på rundt 0,90. Den oransje stabiliserer seg på en Dice-score per tverrsnitt mellom 0,60 og 0,65. Avstanden mellom kurvene er imidlertid stor her også. Dette kan tyde på at modellen med augmentering også var noe overtilpasset, men mindre enn modellen uten. Utstrekningen på x-aksen varierer i de to

plottene grunnet koden som sørget for tidlig stopping. Modellen uten augmentering stoppet etter epoke 60, og modellen med augmentering stoppet etter epoke 90.



Figur 4.9. Treningskurven og valideringskurven til modellen uten augmentering. Plottet viser Dice-score per tverrsnitt mot epoke. Den blå grafen representerer ytelsen til treningssettet per epoke, mens den oransje representerer det samme for valideringssettet.



Figur 4.10. Treningskurven og valideringskurven til modellen med augmentering. Plottet viser Dice-score per tverrsnitt mot epoke. Den blå grafen representerer ytelsen til treningssettet per epoke, mens den oransje representerer valideringssettet.

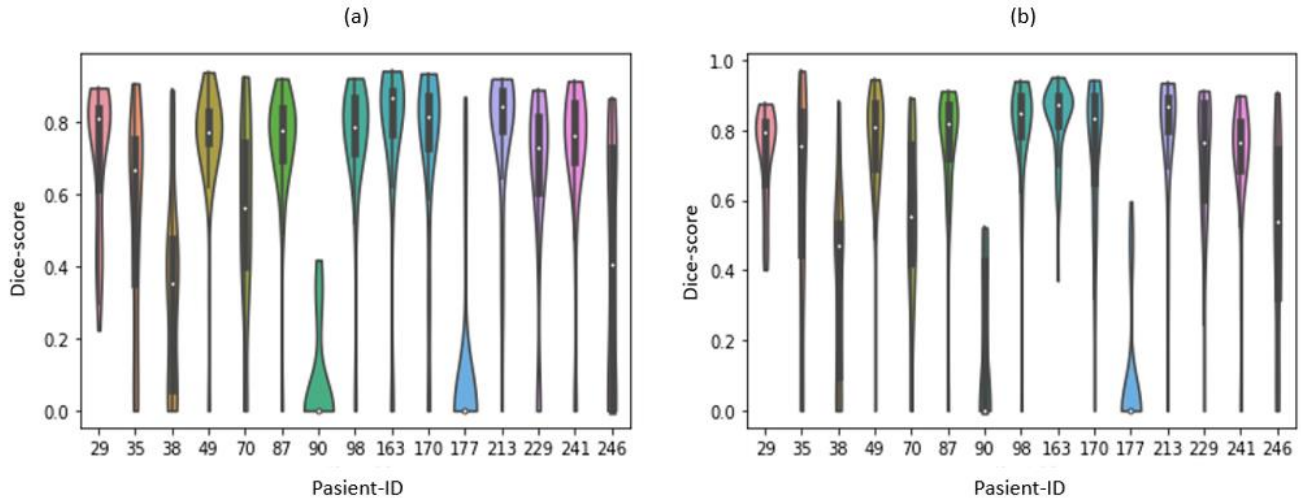
Modellen fikk en gjennomsnittlig Dice-score per tverrsnitt på valideringssettet på 0,609 ± 0,304. Den ble testet med augmentering, og fikk da en gjennomsnittlig Dice-score per tverrsnitt på 0,650 ± 0,281 på valideringssettet. Gjennomsnittlig Dice-score per pasient ble

regnet ut for begge modellene, og ble 0,635 ± 0,259 uten augmentering, og 0,666 ± 0,243 med augmentering.

En oversikt over modellens ytelse per pasient på valideringssettet er gitt i Tabell 4.3. Resultatene tyder på at augmentering hadde en positiv effekt på ytelsen. Modellen oppnådde en høyere gjennomsnittlig Dice-score på alle pasientene, bortsett fra på pasient 177 og pasient 241. For pasient 177 sank ytelsen ved bruk av augmentering, mens for pasient 241 holdt den seg til samme verdi. Modellen presterte imidlertid svært dårlig på noen av pasientene. Modellen oppnådde en ytelse under 0,45 på de tre pasientene 38, 90 og 177. Derimot presterte modellen svært godt med en ytelse på over 0,80 på pasient 163, 170 og 213. Inntegningene utført på disse pasientene er visualisert i Figur 4.13 og Figur 4.14. Modellens ytelse på valideringssettet, uten og med augmentering, er visualisert i Figur 4.11.

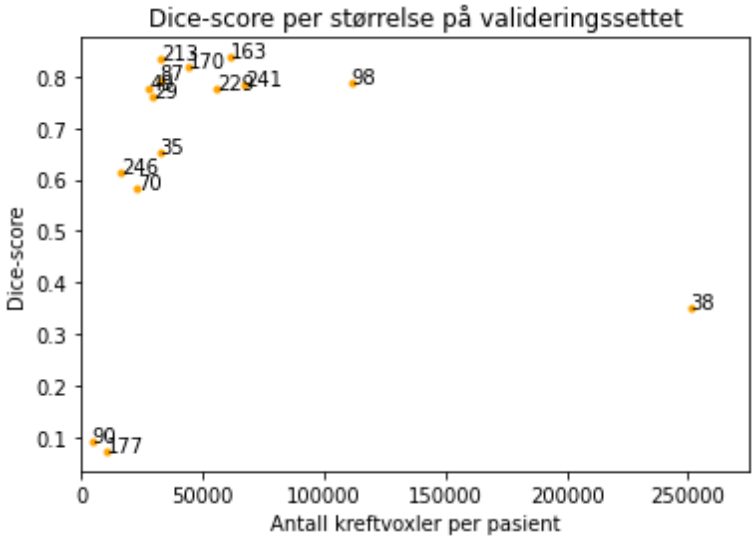
Tabell 4.3. Modellens ytelse gitt i Dice-score per pasient på valideringssettet, med og uten augmentering.

Beste modell på valideringssettet		
Pasientnummer	Dice-score	Dice-score med augmentering
29	0,762	0,772
35	0,654	0,724
38	0,349	0,425
49	0,776	0,782
70	0,583	0,584
87	0,793	0,803
90	0,090	0,222
98	0,787	0,831
163	0,838	0,855
170	0,819	0,821
177	0,072	0,065
213	0,833	0,852
229	0,775	0,809
241	0,785	0,785
246	0,613	0,655

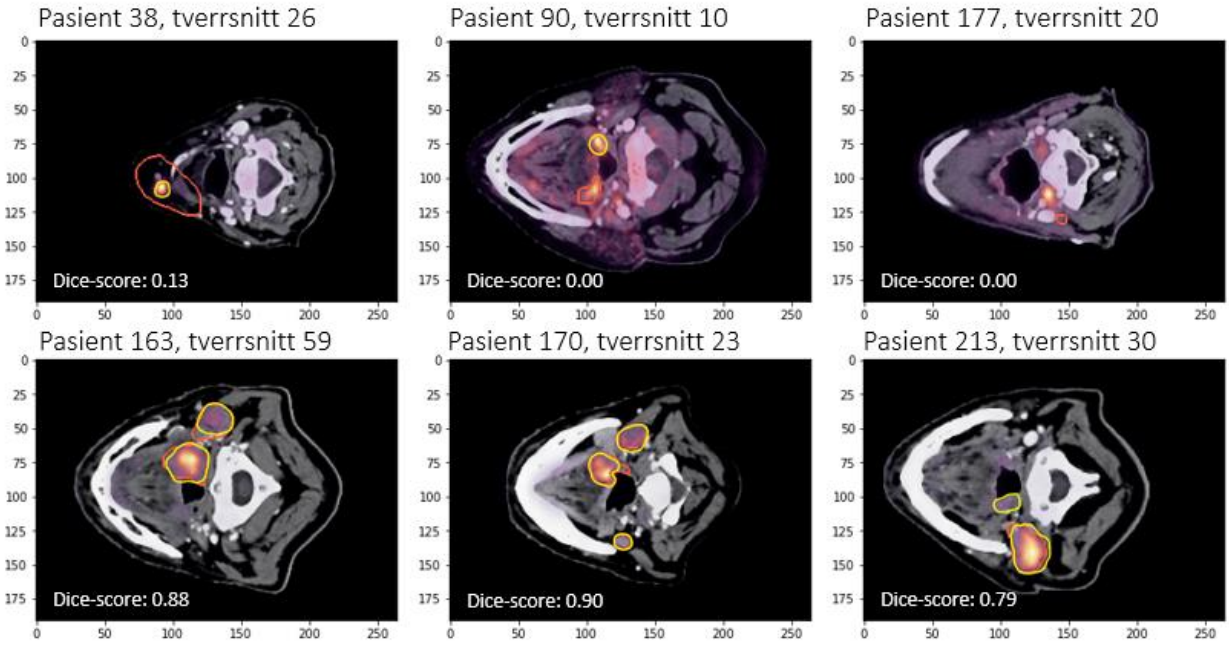


Figur 4.11. Dice-score per tverrsnitt plottet for alle pasientene i valideringssettet for den beste modellen uten augmentering (a) og med augmentering (b). Formen til fiolinplottene representerer fordelingen av Dice-score per tverrsnitt, og den hvite prikken angir medianen. Bredden representerer antallet tverrsnitt med den gitte Dice-scoreverdien, og lengden representerer variansen til Dice-score per tverrsnitt for pasienten. Verdier over eller under den vertikale streken i fiolinene kan ses på som utliggere, og den svarte boksen indikerer observasjonene som ligger mellom første og tredje kvartil.

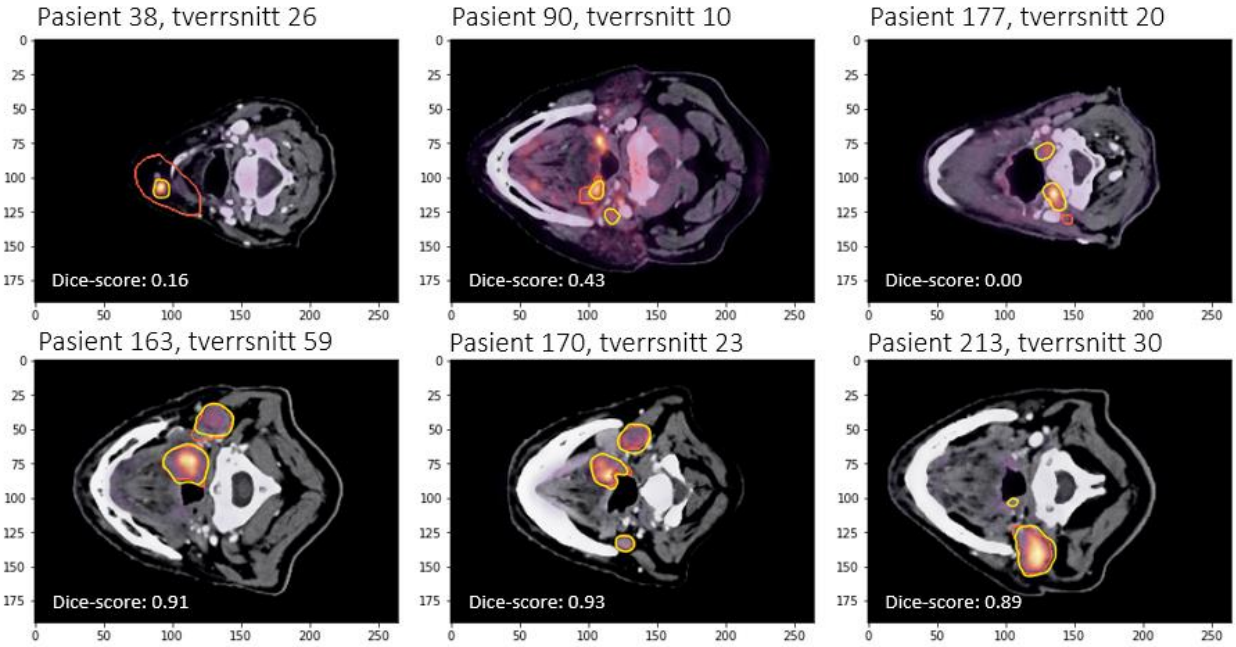
Figur 4.12 viser en oversikt over sammenhengen mellom Dice-score og antall voxler av klassen berørt vev fra de sanne inntegningene hos pasientene i valideringssettet. Plottet viser at de tre pasientene som fikk under 0,45 i Dice-score har berørt vev med et relativt lite eller stort volum.



Figur 4.12. Dice-score per antall voxler av klassen berørt vev (kreftvoxler) for modellen uten augmentering på OUS-valideringssettet. En voxel har dimensjon 1×1×1 mm^3.



Figur 4.13. Inntegninger gjort av den beste modellen uten augmentering på OUS-valideringssettet. Den røde inntegningen tilsvarer sann inntegning, mens den gule er prediksjonen til modellen. De tre øverste viser pasientene modellen scoret lavest på, med gjennomsnittlig Dice-score under 0,45. De tre nederste viser pasientene modellen scoret høyest på, med gjennomsnittlig Dice-score over 0,80. Nederst til venstre i alle bildene vises Dice-scoren til det viste tverrsnittet.



Figur 4.14. Inntegninger gjort av den beste modellen med augmentering på OUS-valideringssettet. Den røde inntegningen tilsvarer sann inntegning, mens den gule er prediksjonen til modellen. De tre øverste viser pasientene modellen scoret lavest på, med gjennomsnittlig Dice-score under 0,45. De tre nederste viser pasientene modellen scoret høyest på, med gjennomsnittlig Dice-score over 0,80. Nederst til venstre i alle bildene vises Dice-scoren til det viste tverrsnittet.

4.4.2 Modellytelse på testsettet

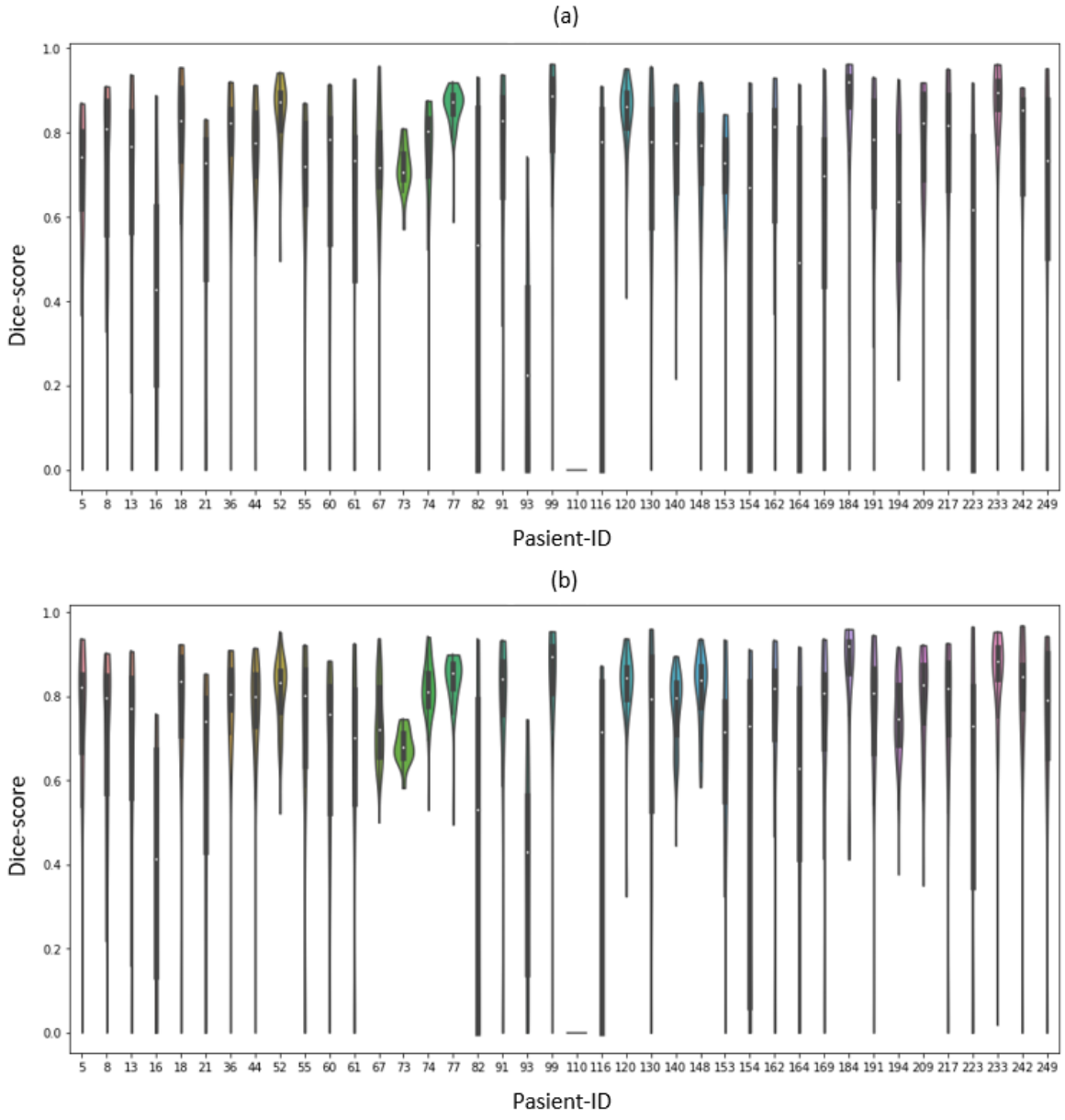
På OUS-testsettet fikk modellen en gjennomsnittlig Dice-score per tverrsnitt på 0,658 ± 0,294, og med augmentering fikk den en gjennomsnittlig score på 0,690 ± 0,267. Modellens gjennomsnittlige ytelsesscorer per pasient med og uten augmentering er gitt i Tabell 4.4. Resultatene er gitt i Dice-score, HD_{95} og MSD per pasient. Figur 4.15 viser Dice-score per tverrsnitt for hver pasient i testsettet, for den beste modellen uten og med augmentering. Basert på Figur 4.15 og Tabell D.1 i Vedlegg D kan man observere at modellen, både med og uten augmentering, gjorde det generelt bedre på testsettet enn valideringssettet. Forskjellen mellom modellen uten og med augmentering var ikke like stor for testsettet som valideringssettet, men den sistnevnte oppnådde noe bedre resultater.

Modellene scoret svært lavt på noen pasienter, blant annet pasient 110 som fikk null i Dice-score for begge modellene. HD_{95}-verdiene og MSD-verdiene for denne pasienten lå på henholdsvis 43,9 mm og 24,0 mm for modellen uten augmentering, og 42,8 mm og 26,2 mm for modellen med augmentering. Dette tyder på en gjennomsnittlig stor avstand mellom den predikerte og den sanne inntegningen. Selv om testsettet består av 40 pasienter, som er større enn valideringssettet på 15 pasienter, var det likevel kun tre pasienter i testsettet som fikk en gjennomsnittlig Dice-score under 0,450 av begge modellene. Disse pasientene var 16, 93 og 110. De samme pasientene oppnådde også en HD_{95}– score over 20,0 mm, som er over gjennomsnittet. De hvite stripene i PET/CT-bildet i Figur 4.17 og Figur 4.18 tyder på artefakter i bildet, som kan være grunnen til at modellen scoret så lavt på pasient 93.

Modellen uten augmentering oppnådde en gjennomsnittlig Dice-score over 0,800 på 13 pasienter, mens modellen med augmentering oppnådde det samme på 16 pasienter i testsettet. De tre pasientene begge modellene gjorde det best på, fikk over 0,870 i gjennomsnittlig Dice-score og var pasient 99, 184 og 233. HD_{95}-scoren for disse pasientene varierte. Pasient 99 oppnådde en HD_{95}-score under 4,00 mm for modellen med og uten augmentering. Pasient 184 fikk imidlertid en høyere score, der modellen uten augmentering fikk en HD_{95}-score på 10,9 mm og modellen med augmentering fikk en score på 35,6 mm. Modellen oppnådde en HD_{95}-score på 9,22 mm uten augmentering og 13,9 mm med augmentering på pasient 233.

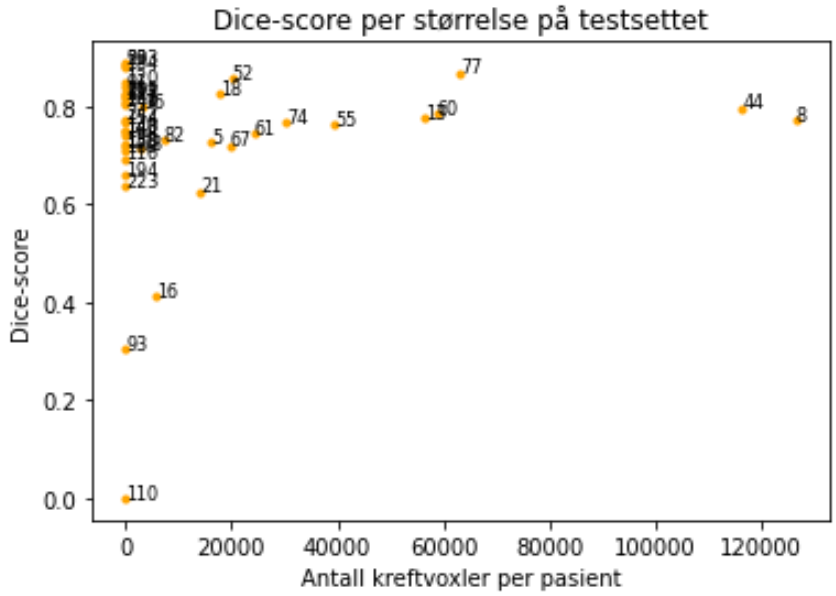
Tabell 4.4. Ytelsesscorer per pasient for den beste modellen uten og med augmentering på OUS-testsettet. Ytelsesmålene Dice-score, HD_{95} og MSD er oppgitt på formen gjennomsnitt ± standardavvik.

	Dice-score	HD_{95} [mm]	MSD [mm]
Uten augmentering	0,734 ± 0,259	17,4 ± 15,0	1,32 ± 3,81
Med augmentering	0,742 ± 0,243	19,0 ± 16,0	1,39 ± 4,13



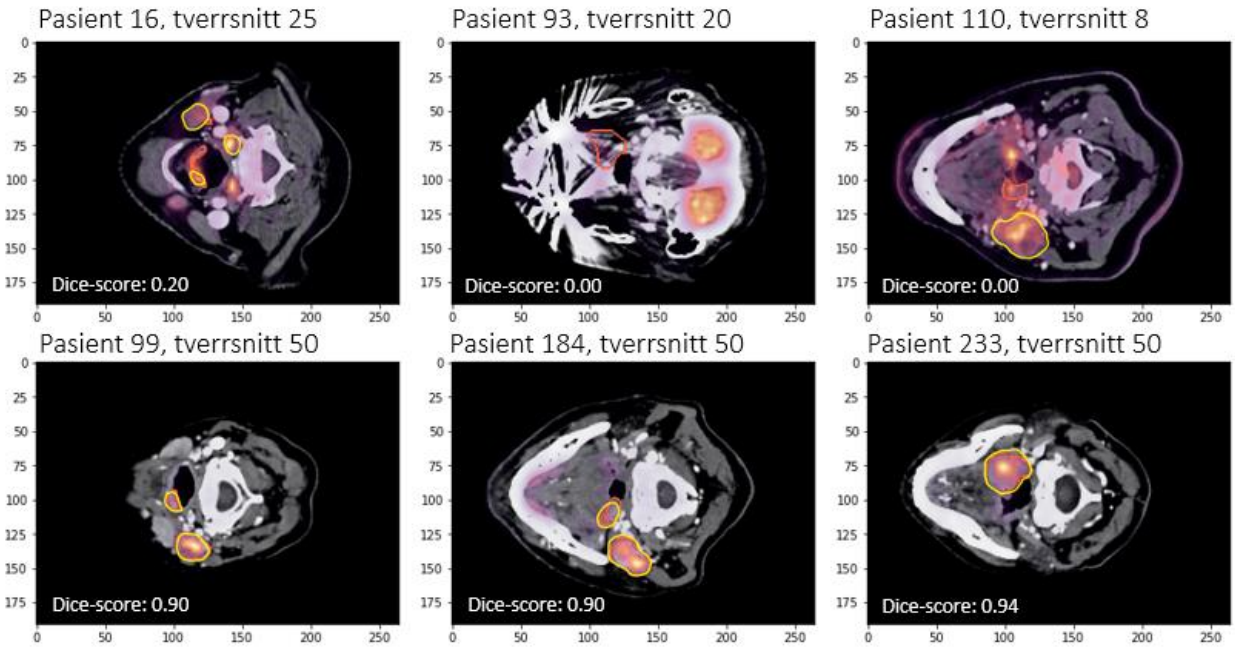
Figur 4.15. Fiolinplott som viser Dice-score per tverrsnitt, plottet for alle pasientene for den beste modellen uten augmentering (a) og med augmentering (b) kjørt på OUS-testsettet. Formen til fiolinene representerer fordelingen av Dice-score per tverrsnitt, og den hvite prikken angir medianen. Bredden representerer antallet tverrsnitt med den gitte Dice-scoreverdien, og lengden representerer variansen til Dice-score per tverrsnitt for pasienten. Verdier over eller under den vertikale streken i fiolinene kan ses på som utliggere, og den svarte boksen indikerer observasjonene som ligger mellom første og tredje kvartil.

Figur 4.16 viser en oversikt over sammenhengen mellom Dice-score og antall voxler av klassen berørt vev fra de sanne inntegningene hos pasientene i testsettet. Plottet viser at alle de tre pasientene (16, 93, 110) modellen gjorde det dårligst på hadde et lite volum. Likevel var det mange pasienter med tilsvarende volum som modellen klarte å predikere nøyaktig. Det var altså en høy varians i Dice-score for pasientene med små volum av berørt vev. Pasient 8 og 44 hadde de største volumene med berørt vev i testsettet. Volumene var likevel rundt halve størrelsen til pasient 38 i Figur 4.12, som hadde det største volumet i valideringssettet.

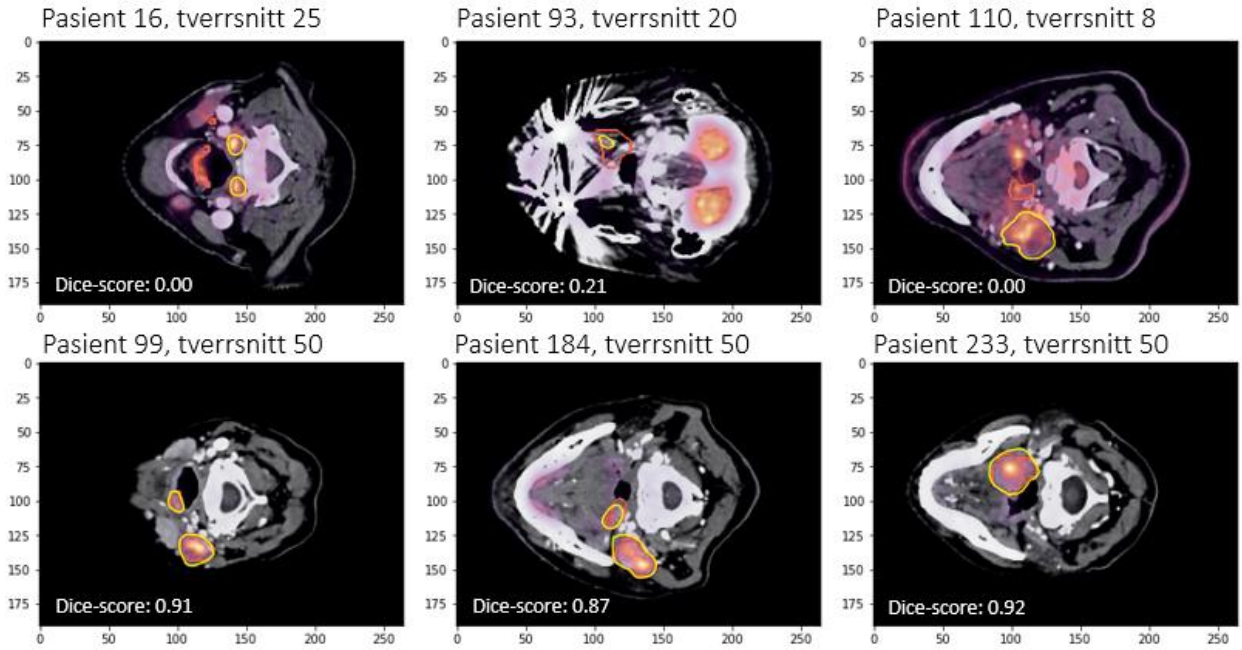


Figur 4.16. Dice-score per antall voxler av klassen berørt vev (kreftvoxler) for modellen uten augmentering på OUS-testsettet. En voxel har dimensjon 1×1×1 mm^3.

Figur 4.17 viser inntegninger gjort av den beste modellen uten augmentering på OUS-testsettet. De tre øverste bildene er inntegninger fra de tre pasientene modellen scoret lavest på, som fikk en Dice-score per pasient under 0,45. På disse pasientene bommet modellen slik at det ble delvis eller ingen overlapp, og på tverrsnitt 20 fra pasient 93 tegnet ikke modellen inn noe. Hos pasient 16 og 110 tegnet modellen inn noen falske positive områder. De tre nederste bildene er fra de tre pasientene modellen scoret best på, som fikk en Dice-score per pasient over 0,87. Figur 4.18 viser det samme, men for modellen med augmentering. Her har imidlertid modellen predikert et område med delvis overlapp for tverrsnitt 20 hos pasient 93.



Figur 4.17. Inntegninger gjort av den beste modellen uten augmentering på OUS-testsettet. Den røde inntegningen tilsvarer sann inntegning, mens den gule er prediksjonen til modellen. De tre øverste viser pasientene modellen scoret lavest på, med gjennomsnittlig Dice-score under 0,45. De tre nederste viser pasientene modellen scoret høyest på, med gjennomsnittlig Dice-score over 0,87. Nederst til venstre i alle bildene vises Dice-scoren til det viste tverrsnittet.



Figur 4.18. Inntegninger gjort av den beste modellen med augmentering på OUS-testsettet. Den røde inntegningen tilsvarer sann inntegning, mens den gule er prediksjonen til modellen. De tre øverste viser pasientene modellen scoret lavest på, med gjennomsnittlig Dice-score under 0,45. De tre nederste viser pasientene modellen scoret høyest på, med gjennomsnittlig Dice-score over 0,87. Nederst til venstre i alle bildene vises Dice-scoren til det viste tverrsnittet.

4.4.3 Modellytelse på Maastro-datasettet

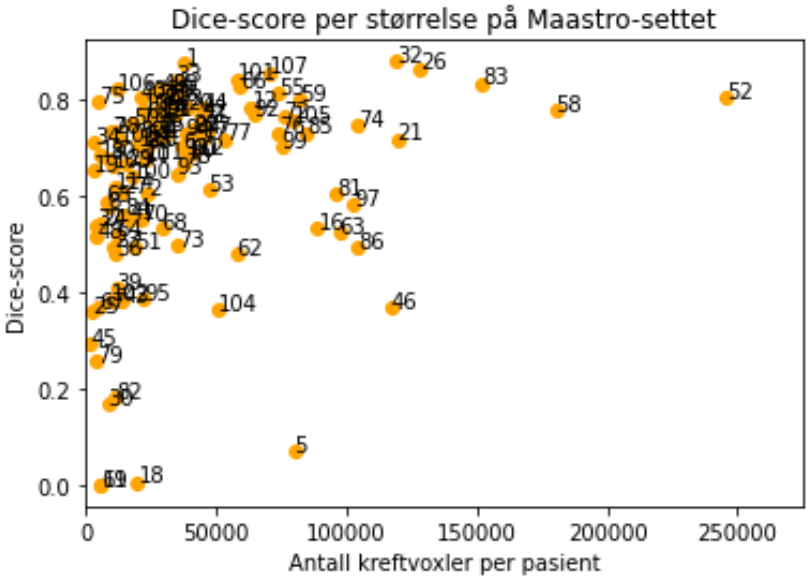
Ved kjøring på det eksterne datasettet fra Maastro-klinikken i Nederland, oppnådde modellen uten augmentering en gjennomsnittlig Dice-score per tverrsnitt på 0,541, mens modellen med augmentering oppnådde en gjennomsnittlig Dice-score per tverrsnitt på 0,550. Modellens gjennomsnittlige ytelse per pasient med og uten augmentering er gitt i Tabell 4.5. Modellenes ytelse per pasient finnes i Vedlegg D.

Tabell 4.5. Ytelsesscores per pasient på den beste modellen uten og med augmentering på Maastro-testsettet. Ytelsesmålene Dice-score, HD_{95} og MSD er oppgitt på formen median/gjennomsnitt ± standardavvik.

	Dice-score	HD_{95} [mm]	MSD [mm]
Uten augmentering	0,707/ 0,637 ± 0,193	14,0/ 19,1 ± 17,2	1,00/ 2,12 ± 6,23
Med augmentering	0,710/ 0,637 ± 0,210	16,5/ 21,8 ± 19,6	1,00/ 2,99 ± 10,6

Ytelsesscorene i Tabell 4.5 gir samme gjennomsnittlig Dice-score per pasient for modellen uten og med augmentering. Medianen og gjennomsnittet til HD_{95} viser seg å være noe større for modellen med augmentering enn uten. Det samme gjelder gjennomsnittet til MSD, men medianen er lik for begge. Standardavviket er noe større for modellen med augmentering for alle ytelsesscorene.

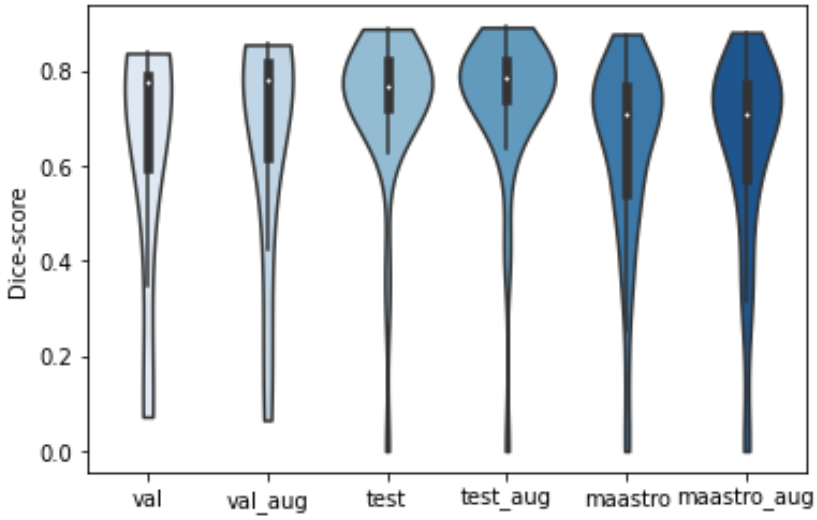
Modellens ytelse på Maastro-datasettet i forhold til test- og valideringssettet fra OUS er vist i Figur 4.20. Størrelsen til fiolinene varierer fra de forskjellige datasettene, ettersom de inneholder et ulikt antall pasienter. Modellene oppnådde en noe lavere ytelse med en lavere median og en større varians når de ble kjørt på det eksterne testsettet, sammenlignet med validerings- og testsettet fra OUS. Gjennomsnittlig HD_{95} og MSD var imidlertid noe lavere ved kjøring på Maastro-datasettet i forhold til OUS-testsettet. Figur D.1 og Figur D.2 i Vedlegg D viser Dice-score per tverrsnitt for hver pasient i Maastro-datasettet for modellen henholdsvis uten og med augmentering. Figur 4.19 viser sammenhengen mellom den gjennomsnittlige Dice-scoren til pasientene og volumet til pasientenes berørte vev. Fra figuren er det tydelig at et lite volum gir en mer variabel Dice-score enn et større volum.



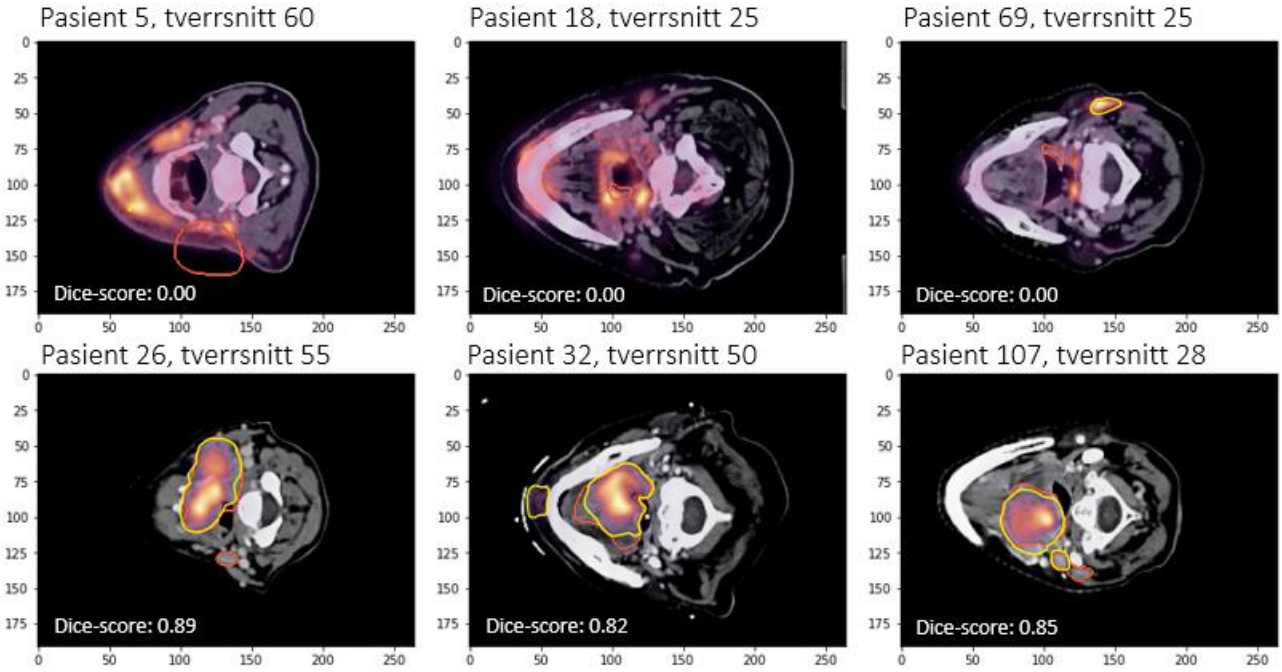
Figur 4.19. Dice-score per antall voxler av klassen berørt vev (kreftvoxler) for modellen uten augmentering på Maastro-testsettet. En voxel har dimensjon 1×1×1 mm^3.

I likhet med OUS-valideringssettet og -testsettet er det også her noen pasienter modellen presterte dårlig på. De tre pasientene modellen med og uten augmentering scoret lavest på var pasient 5, 18 og 69. Alle disse pasientene fikk en gjennomsnittlig Dice-score under 0,070 for begge modellene. To av de tre pasientene med lavest Dice-score har også et lite volum, på mindre enn 25 000 mm^3. De tre pasientene modellen presterte best på var pasient 26, 32 og 107. Disse svulstene hadde et volum mellom 75 000 mm^3 og 150 000 mm^3, som var betraktelig større enn de tre pasientene med lavest Dice-score. Modellen med og uten augmentering oppnådde en gjennomsnittlig Dice-score over 0,853 på disse pasientene.

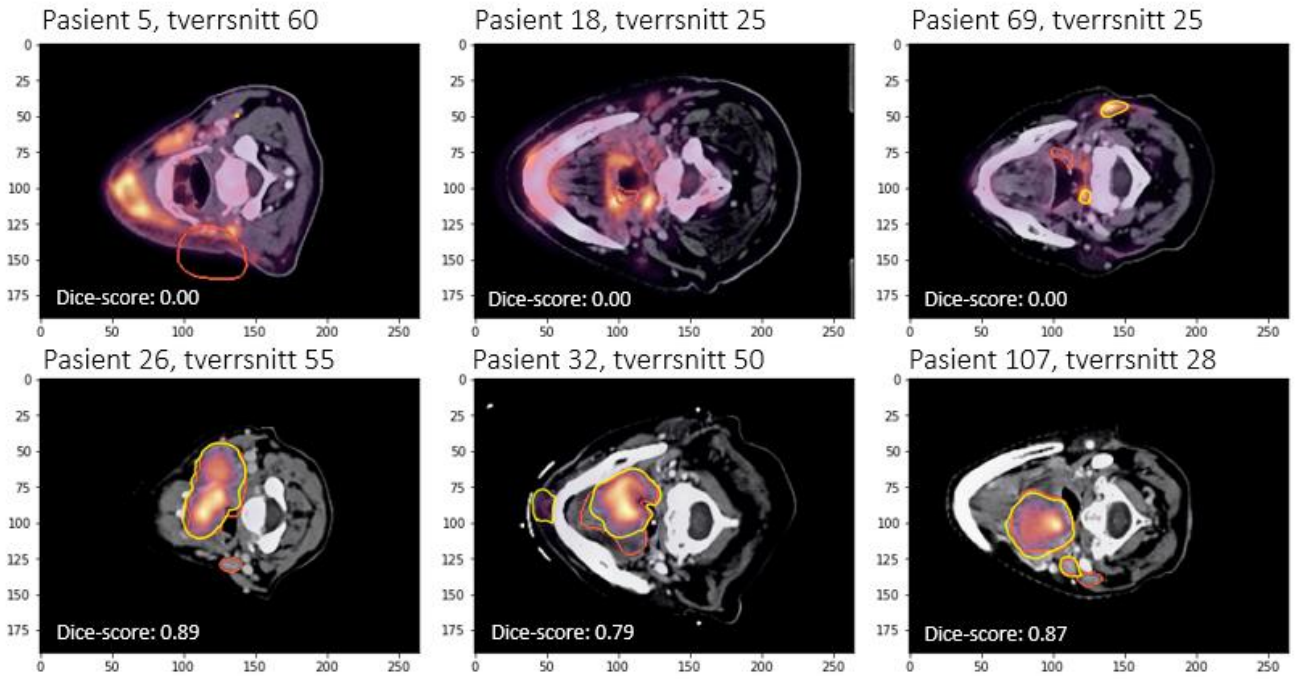
Eksempler på tverrsnitt med inntegninger fra de tre pasientene med lavest og høyest gjennomsnittlig Dice-score er gitt i Figur 4.21 og Figur 4.22. På pasient 69 har modellen bommet, og tegnet inn falske positive områder der FDG-opptaket er høyt. Modellen tegnet ikke inn noe for de viste tverrsnittene til pasient 5 og 18. For pasient 26, 32 og 107 har modellen med og uten augmentering predikert områder som i høy grad overlapper med den sanne inntegningen.



Figur 4.20. Fiolinplott som viser ytelsen per pasient for den beste modellen uten og med augmentering, kjørt på OUS-valideringssettet, OUS-testsettet og Maastro-testsettet. Formen til fiolinene representerer fordelingen av Dice-score per tverrsnitt, og den hvite prikken angir medianen. Bredden representerer antallet tverrsnitt med den gitte Dice-scoreverdien, og lengden representerer variansen til Dice-score per tverrsnitt for pasienten. Verdier over eller under den vertikale streken i fiolinene kan ses på som utliggere, og den svarte boksen indikerer observasjonene som ligger mellom første og tredje kvartil.



Figur 4.21. Eksempler på inntegninger gjort av modellen uten augmentering på Maastro-datasettet. Den røde inntegningen er den sanne inntegningen, mens den gule er predikert av modellen. De øverste er fra de tre pasientene med lavest Dice-score, mens de nederste er fra de tre pasientene med høyest Dice-score. Nederst til venstre i alle bildene vises Dice-scoren til det viste tverrsnittet.



Figur 4.22. Eksempler på inntegninger gjort av modellen med augmentering på Maastro-datasettet. Den røde inntegningen er den sanne inntegningen, mens den gule er predikert av modellen. De øverste er fra de tre pasientene med lavest Dice-score, mens de nederste er fra de tre pasientene med høyest Dice-score. Nederst til venstre i alle bildene vises Dice-scoren til det viste tverrsnittet.

Kapittel 5: Diskusjon

5.1 Målet med oppgaven

Målet med denne oppgaven var å vurdere bruken av et 2D Dense-Net til semantisk segmentering av hode- og halskreft på PET/CT-bilder. Modellen skulle optimaliseres ved justering av fire forskjellige parametere. Til slutt skulle modellen kjøres med bildeaugmentering. Videre skulle ytelsen måles mot sammenligningsmodellen U-Net som ble brukt av Moe et al. [10], Groendahl et al. [12] og Huynh [19] til samme formål på samme pasientkohort. For å sammenligne ytelsen ble overlappsmålet Dice-score og de avstandsbaserte målene HD_{95} og MSD brukt.

Modellen oppnådde en gjennomsnittlig Dice-score per pasient på 0,734 uten augmentering, og 0,742 med augmentering på OUS-testsettet. På det eksterne Maastro-datasettet oppnådde modellen en gjennomsnittlig Dice-score per pasient på 0,637 både med og uten augmentering. I følge Zijdenbos et al. [70] er 0,70 grensen for en god overlapping. Resultatene virker dermed lovende, men det er fortsatt rom for forbedring. I dette kapittelet diskuteres ytelsen til modellen, samt verdiene til parameterne som inngår i den. Videre diskuteres utfordringer innen modellevaluering, lignende forskning, modellen i en klinisk sammenheng og videre arbeid.

5.2 Effekt av modellparametere på ytelsen

5.2.1 Antall filtre

Resultatene fra de statistiske testene viste en signifikant forskjell mellom antall filtre. Basert på boks-plottet ser det ut som at 32 og 64 filtre i første lag gir omtrent like høy Dice-score, og begge gir høyere Dice-score enn 48 filtre. En modell med 32 filtre er enklere enn en med 64 filtre, og vil dermed være raskere å trene og mindre utsatt for overtilpasning. En enkel modell vil være å foretrekke framfor en mer komplisert modell, og valget falt dermed på 32 filtre.

Til sammenligning hadde U-Net-modellen i Moe et al. [10] og Groendahl et al. [12] 64 filtre i første lag, i likhet med Ronneberger et al. [46] som utviklet det originale U-Nettet. Dense-Net-arkitekturen til Guo et al. [11] derimot, brukte 48 filtre. Flere filtre gir modellen evnen til å lære mer, og vil dermed kunne gi en høyere ytelse. Evnen til å lære mer kan imidlertid også føre til overtilpasning [40]. Basert på resultatene kan det virke som at 2D Dense-Net-

arkitekturen i denne oppgaven klarer å trekke ut den nødvendige informasjonen fra datasettet, slik at det ikke er behov for en mer komplisert modell.

5.2.2 Læringsraten

Basert på de statistiske testene viste det seg at en læringsrate på 10^{-4} og 10^{-5} hadde størst effekt på Dice-scoren til 2D Dense-Net. Ettersom en liten læringsrate kan føre til at modellen konvergerer saktere vil det være hensiktsmessig å bruke 10^{-4} fremfor 10^{-5}. Parameteren hadde høy effekt ifølge effekttesten, som vil si at endring av denne parameteren kan påvirke ytelsen sterkt.

Moe [50] fant at en læringsrate på 10^{-5} førte til en svekket generaliseringsevne for U-Net-modellen, og en større læringsrate enn 10^{-4} førte til at nettverkets treningsevne ble svekket. Dermed ble 10^{-4} også brukt i sammenligningsmodellen i Moe et al. [10].

Testingen av læringsraten skiller seg fra de andre parameterne ved at den ble evaluert underveis i denne oppgaven. Læringsraten 10^{-3} og 10^{-4} ble først testet, før valget av neste parameternivå ble tatt basert på resultatene. Valget for videre testing falt på 10^{-5}, ettersom 10^{-4} gav høyere gjennomsnittlig Dice-score per tverrsnitt enn 10^{-3} gjorde. Det ble dermed antatt at en mindre læringsrate kunne gi en høyere Dice-score. Avgjørelsen ble basert på resultatene Tabell B.1 i Vedlegg B.

5.2.3 Batchnormalisering

Resultatene av de statistiske testene angir at modellen hadde en signifikant høyere ytelse når batchnormalisering ble brukt. Batchnormalisering ble brukt i sammenligningsmodellen i Moe at al. [10] og Groendahl et al. [12]. I likhet med disse studiene brukte Guo et al. [11] også batchnormalisering i sitt 3D Dense-Net. Ioffe og Szegedy [62] utviklet batchnormalisering, og fant at parameteren akselererer treningen av dype nevrale nettverk. Det ble ikke undersøkt om modellen i denne oppgaven konvergerte raskere med batchnormalisering, men det kan være interessant å undersøke i senere studier.

Fire av de 36 modellene som ble kjørt fikk tilnærmet null i Dice-score, som vist i Tabell B.1 i Vedlegg B. Felles for disse fire modellene var at ingen av dem brukte batchnormalisering, og alle hadde en læringsrate på 10^{-3}. En for stor læringsrate kan hindre modellen i å konvergere [39], og som tidligere nevnt kan batchnormalisering gi rom for en høyere læringsrate. I delkapittel 3.5.2 forklares det at prosessen blir sakket ned uten batchnormalisering, og det kreves dermed en lavere læringsrate. Grunnen til at noen modeller fikk så lav Dice-score er ukjent, men det kan tenkes at interaksjonen mellom disse parameternivåene kan ha vært en

faktor. I tillegg kan batchnormalisering eliminere behovet for utelatelse, som nevnt i delkapittel 3.5.2. Ingen av disse interaksjonene ble undersøkt i denne oppgaven, men kan være interessant å utforske ved videre arbeid.

5.2.4 Utelatelsesraten

En utelatelsesrate på 0,5 gav høyest Dice-score for 2D Dense-Nettet, basert på de statistiske testene i denne oppgaven. Likevel har flere studier valgt å ikke bruke utelatelse i sine nettverk. Guo et al. [11] brukte blant annet ikke utelatelse i sitt 3D Dense-Net. Det ble heller ikke brukt i sammenligningsmodellen U-Net i Moe et al. [10] eller Groendahl et al. [12].

Ettersom ReLU har en tendens til å overtilpasse modellen, kan det lønne seg å bruke en regulariseringsmetode sammen med denne aktiveringsfunksjonen [40]. Utelatelse er en regulariseringsmetode, men det er ikke den eneste regulariseringsmetoden som undersøkes i denne oppgaven. Ved bruk av færre filtre eller batchnormalisering, kan det tenkes at viktigheten av utelatelse synker. Sammenhengen mellom disse reguleringsmetodene bør dermed studeres nærmere ved videre arbeid.

Krogstie undersøkte VoxResNet i sin masteroppgave på samme datasett [71], og hennes resultater viste at utelatelse ikke forbedret nettverket. En mulig forklaring for at det likevel hadde en effekt hos Dense-Nettet i denne oppgaven kan være at Dense-Net inneholder flere trenbare parametere enn VoxResNet, og er dermed mer utsatt for overtilpasning [54].

I tillegg til å motvirke overtilpasning, har utelatelse en tendens til å øke treningshastigheten til nettverket [40]. Effekten av dette ble ikke undersøkt i denne oppgaven, men kan være nyttig å undersøke ved videre arbeid. Dense-Net tar relativt lang tid å trene, sammenlignet med U-Net og VoxResNet [64, 71]. Treningshastigheten er derfor et område med rom for forbedring, der utelatelse kan ha en betydning.

I denne oppgaven ble kun to verdier av utelatelse testet. Valget falt da på to helt ulike verdier som var lette å skille fra hverandre. Selv om utelatelsesraten 0,5 viste seg å gi høyest Dice-score for denne modellen, er det uvisst om en annen utelatelsesrate kunne gitt enda høyere ytelse. Som nevnt i delkapittel 3.5.2, er en utelatelsesrate fra 0,2 til 0,5 vanlig. Å teste andre utelatelsesrater kan være et emne for videre arbeid, som muligens kan optimalisere modellen ytterligere.

5.2.5 Augmentering

Modellen med kombinasjonen av de mest optimale parameterne oppnådde en enda høyere Dice-score med augmentering enn uten. Den oppnådde en gjennomsnittlig Dice-score per pasient på 0,742 på testsettet og 0,666 på valideringssettet. Dette var høyere enn modellen uten augmentering, som fikk en Dice-score på 0,734 på testsettet, og 0,635 på valideringssettet. Til sammenligning fikk sammenligningsmodellen U-Net med samme augmentering en Dice-score på 0,751 på testsettet og 0,697 på valideringssettet i Ødegaard sin masteroppgave [64]. Begge modellene oppnådde altså en høyere Dice-score med augmentering enn uten, som tyder på at augmentering var en forbedrende faktor.

Medisinske datasett er allerede variert, ettersom ingen pasienter eller svulsttilfeller er helt like. Til tross for dette, har tidligere studier oppnådd gode resultater ved bruk av augmentering for å variere datasettene ytterligere. I HECKTOR-utfordringen brukte blant annet flere deltakere augmentering i sine segmenteringsmodeller [18]. Yuan [72] brukte for eksempel augmentering på en U-Net-inspirert arkitektur kalt SA-Net, og oppnådde en Dice-score per pasient på 0,732 på testkohorten. Augmentering er et billigere og enklere alternativ til innsamling av mer data for et mer variert datasett. I tillegg vil augmentering kunne forhindre at modellen blir overtilpasset, som kan observeres ved sammenligning av 2D Dense-Net med og uten augmentering, henholdsvis vist i trenings- og valideringskurvene i Figur 4.9 og Figur 4.10.

Augmentering er imidlertid situasjonsbestemt. En augmenteringsmetode som fungerer godt i noen situasjoner, vil kunne forvirre eller forverre modellen i andre situasjoner. Et eksempel er ved gjenkjenning av tall. Roterer man et sekstall 180 grader, vil dette bli et nitall. Augmenteringsmetoden vil da være ugunstig å bruke, ettersom den vil lære modellen at sekstall og nitall er like. Ved tilfellet med en føflekk derimot, vil en rotasjon på 180 grader fortsatt gi en troverdig føflekk-etterligning, og kan dermed forbedre modellen. En kan dermed ikke konkludere med at augmenteringen brukt i denne masteroppgaven vil være forbedrende for alle problem, men at det fungerer godt i dette tilfellet.

5.3 Evaluering av den beste modellen

5.3.1 Ytelse

Som nevnt i delkapittel 4.4 ble den beste modellen definert til å være kombinasjonen av parameterne som hadde størst effekt på ytelsen, der kombinasjonen er gitt i Tabell 4.2. Den

beste modellen fikk en Dice-score per pasient på 0,635 uten augmentering og 0,666 med augmentering på OUS-valideringssettet. På OUS-testsettet fikk den en Dice-score på 0,734 uten augmentering og 0,742 med, og en HD_{95}-score på 17,4 mm uten og 19,0 mm med. Den gjennomsnittlige Dice-scoren per pasient ble brukt istedenfor gjennomsnittlig Dice-score per tverrsnitt, ettersom det gir mer informasjon om hvordan modellen gjør det i en realistisk situasjon, der den totale inntegningsnøyaktigheten på en pasient er viktigere enn hvordan modellen gjør det på kun ett tverrsnitt. I tillegg blir det samme ytelsesmålet brukt i Moe et al. [10], Groendahl et al. [12] og Guo et al. [11], og det blir dermed lettere å sammenligne ytelsen til modellene.

Modellen oppnådde en lavere ytelse ved test på Maastro-datasettet enn den gjorde på testsettet fra OUS. Den fikk en gjennomsnittlig Dice-score per tverrsnitt på 0,541 uten augmentering og 0,550 med augmentering. Den gjennomsnittlige Dice-scoren per pasient ble 0,637 både med og uten augmentering, som tyder på at augmentering ikke hadde effekt for modellen på dette datasettet. Denne ytelsen nådde ikke kravet for en god overlapping, som er definert av Zijdenbos et al. [70]. En lavere Dice-score var imidlertid forventet ettersom Maastro-datasettet er eksternt. Det er dermed ingen mulighet for at dataen kan ha lekket over til treningssettet, slik at modellen kunne trent på akkurat denne typen data. Praksisen innen billedtagning og inntegning i Nederland kan skille seg fra prosedyren i Norge, slik at bildene i Maastro-datasettet kan være noe forskjellige fra OUS-datasettet. Modellen kan dermed få problemer med å gjenkjenne strukturer i bildene, som igjen kan gi lavere ytelse. En mulighet for videre arbeid kan være å overføre et antall bilder fra eksterne datasett som Maastro-datasettet til treningsdatasettet, slik at modellen blir mer generell.

5.3.2 Tidligere arbeid

I senere tid har flere studier utviklet modeller til segmentering av svulster og organer på medisinske bilder. Huang et al. [73] brukte en U-Net-inspirert dyplæringsmodell til å segmentere ut tumorvolum fra hode- og halskreft på PET/CT-bilder med størrelse 512×512. De oppnådde en Dice-score per pasient på 0,74. Lin et al. [17] konstruerte et segmenteringsverktøy ved kunstig intelligens, som tegnet inn svulstvolum hos pasienter med *Nasopharyngeal carcinoma*. Et tredimensjonalt konvolusjonsnettverk, inspirert av arkitekturen til nettverket VoxResNet, ble brukt til å segmentere ut svulstene på MR-bilder med størrelse 96×96×32. Modellen oppnådde en Dice-score med median på 0,79 per pasient. De konkluderte med at verktøyet forbedret svulstinntegningen, og at det kan ha positiv effekt på behandlingsforløpet ved at strålingsterapien kan utføres raskere og nøyaktigere [17]. I

HECKTOR-utfordringen konkurrerte flere dyplæringsmodeller om å oppnå høyest ytelse ved segmentering av hode- og halskreft på et datasett bestående av PET/CT-bilder med størrelse 144×144×144 [18]. Denne utfordringen bidro til mye forskning på området. Iantsen et al. [74] oppnådde den høyeste ytelsen i konkurransen ved bruk av en U-Net-arkitektur med residual-koblinger og en *squeeze and excitation*-normalisering, og oppnådde en Dice-score per pasient på 0,76. Videre i konkurransen ble forskjellige arkitekturer brukt, og modellene oppnådde Dice-scorer per pasient mellom 0,56 til 0,76.

Selv om Lin et al. [17], Huang et al. [73] og bidragene i HECKTOR-utfordringen deler mange likhetstrekk med modellen som ble undersøkt i denne oppgaven, vil det imidlertid ikke være riktig å sammenligne ytelsene direkte. Størrelsen til inputbildene kan blant annet ha en innvirkning på ytelsen til modellen, der et større inputbilde sannsynligvis vil påvirke ytelsen negativt [54]. Lin et al. [17], Huang et al. [73] og modellene i utfordringen segmenterte kun tumorvolum og inkluderte ikke affiserte lymfeknuter, i motsetning til hva som ble gjort i denne oppgaven. Modellene i utfordringen benyttet også en avgrensingsboks som omsluttet dette tumorvolumet, og som kan resultere i høyere ytelse for modellene. Til tross for at resultatene ikke kan sammenlignes direkte, bør det nevnes at ytelsen til 2D Dense-Net kan måle seg med ytelsen til Huang et al. [73] og de høyeste ytelsesscorene i HECTOR-utfordringen [18]. Videre i oppgaven sammenlignes ytelsen til 2D Dense-Net med modeller som har segmentert både tumorvolum og affisert lymfeknutevolum, som sammenligningsmodellen i Moe et al. [10] og Groendahl et al. [12], og 3D Dense-Nettet til Guo et al. [11].

5.3.3 Sammenligning med 3D Dense-Net

Guo et al. [11] sitt 3D Dense-Net fikk en gjennomsnittlig Dice-score per pasient på 0,71 på testsettet. Selv om modellen brukte samme arkitektur som 2D Dense-Net, skiller de seg fra hverandre ved at Guo et al. [11] brukte 3D-konvolusjonslag istedenfor 2D som modellen i denne oppgaven brukte. 2D-konvolusjonene får ikke informasjon fra forrige tverrsnitt, i motsetning til 3D-konvolusjoner som får informasjon fra voxelene i tre dimensjoner. Dette ligner måten radiologer jobber på. De bruker informasjon fra z-aksen, altså forrige og neste bildetverrsnitt, når de tegner inn svulster på et bildetverrsnitt [11]. 3D-konvolusjonslag kan dermed føre til en bedre ytelse enn med 2D-konvolusjonslag, men modellen blir beregningsmessig tyngre [11]. I likhet med denne oppgaven, brukte Guo et al. [11] augmentering på sitt 3D Dense-Net.

En annen forskjell som kan være av betydning for sammenligningen, er at Guo et al. [11] brukte en annen størrelse på inputbildene og et bredere intensitetsvindu enn det som ble brukt i denne oppgaven. Størrelsen på inputbildene var 128×128×48, som nevnt i delkapittel 2.8.3, og vindusintervallet var på [-200, 200] HU. Det ble i denne oppgaven brukt et mindre intervall på [-100, 100] HU, som beskrevet i delkapittel 3.1.2, ettersom kun mykt vev og involverte lymfeknuter som representeres i en liten del av vindusbredden er av interesse. Det blir dermed vanskelig å sammenligne Dice-scorene direkte, gitt at datasettene er forskjellige.

5.3.4 Sammenligning med 2D U-Net

2D Dense-Nettet fikk en gjennomsnittlig Dice-score per pasient uten augmentering (0,73) og med augmentering (0,74) på OUS-testsettet som var noe høyere enn det U-Net-modellen i Moe et al. [10] (0,71) fikk på samme datasett. I denne oppgaven brukes det samme datasettet som ble brukt i Moe et al. [10], der eneste forskjell er at bildene i denne oppgaven ble normalisert. Dice-scorene kan dermed sammenlignes direkte. Begge modellene ligger over grensen for en god overlappsscore på 0,70, definert av Zijdenbos et al. [70]. HD_{95}-scoren til 2D Dense-nettet uten augmentering (17 mm) og med augmentering (19 mm), var begge lavere enn for Moe et al. [10] (24 mm). Disse resultatene tyder på at 2D Dense-Net har potensialet til å gjøre en like god eller bedre automatisk segmentering enn sammenligningsmodellen U-Net.

Sammenligningsmodellen i Groendahl et al. [12] oppnådde en høyere Dice-score per pasient (0,75) enn det 2D Dense-Net i denne oppgaven og U-Nettet i Moe et al. [10] gjorde. Groendahl et al. [12] brukte imidlertid et annerledes datasett, så disse ytelsene kan ikke sammenlignes direkte. Datasettet til Groendahl et al. [12] skiller seg ut ved at de brukte andre verdier i intensitetsvinduet på CT-modaliteten, med et vindussenter på 60 HU og en vindusbredde på 100 HU. Bildene ble også beskjært til et interesseområde med størrelse 176×176 som inneholdt tumorvolumene og de affiserte lymfeknutevolumene, som kan føre til at modellen lettere finner berørt vev. I tillegg brukte Groendahl et al. [12] en læringsrate på 10^{-5} og fem-foldet kryssvalidering, noe som også skiller seg fra Moe et al. [10] og 2D Dense-Nettet i denne oppgaven. K-foldet kryssvalidering er en teknikk som ofte brukes ved mindre datasett [13]. Da deles trenings- og valideringssettet opp i *k* folder, og modellen rullerer treningen på *k*-1 folder og validerer på den siste folden. Når alle foldene er brukt til validering, brukes gjennomsnittet av de *k* valideringsscorene. På denne måten unngår man problemet med at valideringsscoren vil variere mye basert på hvilken del av datasettet modellen validerer på [13].

2D Dense-Net oppnår en ytelse som kan måle seg med U-Net, som betegnes som *state-of-the-art* innen segmenteringsoppgaver [11]. I Guo et al. [11] ble ytelsen til et 3D U-Net og et 3D Dense-Net sammenlignet, og Dense-Nettet oppnådde en høyere gjennomsnittlig Dice-score per pasient (0,71) enn U-Nettet (0,69) på samme datasett. Fordelen med å bruke Dense-Net-strukturen i forhold til U-Net-strukturen er at informasjon lettere propagerer gjennom nettverket ved hjelp av Dense-koblingene, slik at risikoen for forsvinnende gradient og overtilpasning minker [11]. Denne informasjonsflyten vil dermed kunne bidra til en høyere ytelse [11]. Dense-Net har også færre trenbare parametere enn U-Net, som kan redusere prediksjonsvariabiliteten til nettverket [11].

5.3.5 Begrensninger til ytelsen

En utfordring ved å definere at den beste modellen er sammensetningen av de mest ideelle parameterne isolert sett, er at det ikke blir tatt høyde for interaksjonen mellom dem. Noen av parameterne deler samme hensikt, som for eksempel regularisering. Både antall filtre, batchnormalisering og utelatelse kan regularisere modellen hvis et gitt nivå av parameteren brukes. Det regulariserende nivået ble valgt som det beste nivået for hver av parameterne. Dermed består den beste modellen av tre regulariserende parametere. Ved videre arbeid kan det være nyttig å undersøke disse interaksjonene, og finne ut om parameterne påvirker hverandres virkning.

Selv om modellen gir en relativt god gjennomsnittlig ytelse, scorer den fortsatt svært lavt på noen pasienter. Figur 4.11 viser for eksempel at ytelsen for pasient 38, 90 og 177 er betraktelig lavere enn for de andre pasientene. Disse tilfellene bidrar til å trekke gjennomsnittlig Dice-score per pasient på valideringssettet ned. Testsettet inneholder ikke data fra de samme pasientene som valideringssettet, og viser seg å ha en mindre andel lavt-predikerte pasienter, som vist i Figur 4.15. Dette kan bidra til at ytelsen på testsettet er høyere enn ytelsen på valideringssettet. Det motsatte er ofte forventet, slik som observert i Guo et al. [11] der modellen oppnådde en Dice-score på 0,80 på valideringssettet og 0,71 på testsettet. Grunnen til at dette forventes, er fordi data fra valideringssettet har en tendens til å lekke over til treningssettet slik at modellen presterer godt på denne dataen.

Det kan være flere grunner til at modellen scoret så lavt på noen pasienter. En av grunnene kan være fordi anatomien i hode- og halsregionen er svært kompleks, som kan gjøre det vanskelig å skille forskjellige typer vev [11]. Disse pasientene kan ha svulster som ligner annet vev, eller en annerledes anatomi enn majoriteten som modellen trente på. Artefakter i

bildene kan også bidra til lav ytelse, som kan være tilfellet for pasient 93 i Figur 4.17 og Figur 4.18. I tillegg kan atypiske svulster, som for eksempel har et lavt FDG-opptak, bidra til at modellen predikerer feil.

Fra Figur 4.15 er det tydelig at pasient 110 får en Dice-score tilnærmet null. Inntegningene i Figur 4.17 og Figur 4.18 viser at modellen har predikert lyse området som berørt vev, selv om disse områdene er definert som ikke-berørt. Huynh [19] formulerte en hypotese i sin masteroppgave om at U-Net lærer mer fra PET-kanalen enn CT-kanalen, og at lyse områder på PET-kanalen indikerte en høy sannsynlighet for at dette var en svulst. Dermed vil områder med høy SUV, men som ikke er en svulst, kunne føre til en falsk positiv predikering [19]. Eksempelet med pasient 110 belyser også det faktum at modellen ikke bør brukes som en erstatning for radiologer og onkologer, ettersom det er flere aspekter som hensyntas ved inntegning av svulster og affiserte lymfeknuter. Fysiske undersøkelser og bilder fra endoskopi benyttes også [50].

Guo et al. [11] viste at små tumor- og lymfeknutevolum gav en stor varians i Dice-score, og drev den gjennomsnittlige ytelsen ned. En grunn til at 2D Dense-Net scoret lavt på noen pasienter kan dermed være fordi modellen segmenterer ut både tumorvolum og affiserte lymfeknute-volum, der sistnevnte ofte er små. Det kan dermed tenkes at pasientene som får en lav Dice-score har flere små volum av berørt vev, og færre store. Denne trenden vises i Figur 4.12 ved at to av de tre pasientene modellen scoret lavest på i valideringssettet hadde små volum med berørt vev. Den siste lå helt i andre enden av skalaen, og hadde et stort volum. Den sanne inntegningen omfatter for den sistnevnte et større heterogent område som ikke ble predikert som berørt vev av modellen.

I testsettet var trenden liknende. Figur 4.16 viser at de tre pasientene modellen scoret lavest på i testsettet, også hadde små volum med berørt vev. En av årsakene til denne sammenhengen kan blant annet være at noen lymfeknuter er FDG PET-negative. Det vil si at de ikke tar opp FDG og vil dermed ikke lyse opp på bildene. Dette kan observeres hos pasient 16 og 110 i Figur 4.17 og Figur 4.18, og kan føre til at modellen ikke tegner inn disse lymfeknutene [56].

Ytelsen til en modell vil variere noe hver gang den blir kjørt, selv om datasettet og parameterne i modellen er identiske. Dette er fordi de opprinnelige vektene til modellen blir satt til små, tilfeldige tall, som beskrevet i delkapittel 2.8.2. Ettersom tallene er tilfeldige vil de variere litt for hver gang modellen kjøres, som vil utgjøre en endring i ytelsesscoren [75]. I tillegg vil rekkefølgen treningseksemplene presenteres i for modellen variere for hver kjøring

[75]. Ideelt sett burde eksperimentmodellene blitt kjørt flere ganger hver, og et gjennomsnitt av ytelsene blitt brukt videre for hver modell.

5.3.6 Datasettet

Usikkerheter i datasettet kan også påvirke ytelsen til modellen. Som nevnt i delkapittel 3.1, ble dataen samlet inn over en periode på 6 år, og i løpet av denne perioden var det forskjellige radiologer og onkologer som gjorde de manuelle inntegningene. Manuell inntegning er en subjektiv prosess som kan føre til intervariabilitet [76]. Ifølge Gudi et al. [77] har radiologer og onkologer en Dice-enighet på 69 % på PET/CT-bilder. Intervariabiliteten mellom legene er en stor faktor for usikkerhet innen strålingsplanlegging ifølge Weiss og Hess [9], og denne usikkerheten vil overføres til datasettets sanne inntegninger. Datasettet består imidlertid av PET/CT-bilder, som kan bidra til noe nøyaktigere inntegninger enn PET og CT separat. Med PET/CT kan både den anatomiske informasjonen fra CT og den funksjonelle informasjonen fra PET utnyttes, og unionen av disse vil utgjøre inntegningen [76].

5.3.7 Ytelsesmål

I denne oppgaven ble det velkjente og velutprøvde ytelsesmålet Dice-score brukt. Bruken av Dice-score gjør det lettere å sammenlikne med annen forskning, da målet er mye benyttet i svulst-segmenteringsoppgaver [10]. Som nevnt i delkapittel 2.9.2 er ytelsesmålet alene ikke helt ideelt likevel, ettersom det ikke tar hensyn til anatomisk plassering. Dermed ble HD_{95} og MSD kalkulert for modellene kjørt på OUS-testsettet og Maastro-testsettet, for å vurdere ytelsen ytterligere. HD_{95} og MSD brukes dermed som et supplement til Dice-score. De statistiske testene og valg av beste modell ble imidlertid basert på Dice-score alene. Ideelt sett burde dette vært basert på Dice-score og et avstandsmål for å oppnå så korrekte resultater som mulig.

5.4 Klinisk bruk

5.4.1 Tidsbruk

Automatisk svulstinntegning kan bidra positivt i helsesektoren, spesielt med tanke på tidsbruk. Den ferdigtrente modellen i denne oppgaven brukte 53 minutter på predikering av hode- og halskreft hos 114 pasienter i det eksterne Maastro-datasettet ved kjøring i Orion. Det tilsvarer gjennomsnittlig 0,47 minutter per pasient. Manuell segmentering av *Nasopharyngeal carcinoma* på MR-bilder kan ta rundt 30 minutter, ifølge Lin et al. [17]. Ifølge Kosmin et al. [8] tar det gjennomsnittlig 2,7 til 3 timer å tegne inn et fullt sett med tumorvolum og

risikoorganer i en hode- og halskreftspasient manuelt. Dermed har modellen potensialet til å spare legene for betydelig med tid per pasient.

5.4.2 Annet bruk

Resultatene i denne oppgaven har vist at 2D Dense-Net kan lære seg informasjon om anatomi og metabolisme fra PET/CT-bilder, og produsere resultater på samme nivå som sammenligningsmodellen U-Net. Modellen har dermed potensiale til å kunne anvendes som hjelpemiddel for segmentering av hode- og halskreft. Ved videre arbeid kan modellen trenes til bruk på andre segmenteringsoppgaver, som for eksempel segmentering av leverkreft eller lungekreft [11].

5.4.3 Etikk

Lin et al. [17] konstaterte at automatiske segmenteringsverktøy reduserte legenes intravariasjon med 36 % og intervariasjon med 55 %, i tillegg til å redusere inntegningstiden med 39 %. Til tross for dette, er det flere aspekter ved bruk av kunstig intelligens innen helsesektoren som bør overveies før det tas i bruk. EU har laget retningslinjer for etikk innen bruk av kunstig intelligens kalt *Ethics guidelines for thrustworthy AI* [78]. Poenget med disse retningslinjene er å utvikle pålitelige AI-hjelpemidler. Her kreves det at AI-systemer ikke skal undergrave menneskelig autonomi. Det foreslås blant annet å benytte *Human-in-command*-tilnærmingen. Denne tilnærmingen innebærer at en person skal kunne bestemme hvordan AI-teknologien skal brukes i forskjellige situasjoner, og om den skal brukes i det hele tatt, for å utvikle skjønn og sikre muligheten til å overkjøre beslutningen tatt av systemet [78]. I sammenheng med bruk av automatisk segmentering i medisinske bilder, kan kravet oppfylles ved at en lege bruker maskinlæringsmodellen som hjelpemiddel, men står selv ansvarlig for inntegningen som blir gjort.

5.4.4 Sensitiv data

Behandlingen av den sensitive dataen som trengs for å trene en automatisk segmenteringsmodell utgjør en risiko for brudd på Helseregisterloven [79]. Helseregisterloven sikrer at helsedata behandles på en etisk forsvarlig måte, og at den enkeltes personvern blir ivaretatt. Loven krever blant annet at behandling av helseopplysninger gir bedre helse- og omsorgstjenester og er til individets og samfunnets beste [79]. Det samsvarer med formålet til forskningen på automatiske segmenteringsmodeller. Den medisinske dataen kreves imidlertid kun til trening av modellen. Etter at modellen er ferdig trent, er det ikke lenger bruk for dataen, og modellen vil kunne brukes på ny og uavhengig data.

5.5 Videre arbeid

Ved diskusjon av resultatene viser det seg at det er flere aspekter som kan undersøkes nærmere for å optimalisere 2D Dense-Net ytterligere, og oppnå mer pålitelige resultater.

5.5.1 Interaksjon mellom parametere

Som diskutert i delkapittel 5.2, ble ikke interaksjonen mellom parameterne tatt hensyn til i denne oppgaven. Ved videre arbeid kan disse interaksjonene undersøkes, og muligens føre til en annen kombinasjon av parametere for den beste modellen.

N-veis ANOVA skulle i utgangspunktet brukes som statistisk metode for å velge hvilken kombinasjon av parametere som skulle utgjøre den beste modellen. Denne statistiske fremgangsmåten tar hensyn til interaksjonen mellom parameterne, og kunne dermed vært gunstigere for bruk i denne oppgaven. ANOVA krever imidlertid at residualene er normalfordelt, noe de ikke var i dette tilfellet. Til videre arbeid kan flere transformeringer av datasettet prøves ut, for å undersøke om normalfordelte residualer er en mulighet.

5.5.2 Parametere

Et begrenset antall parameternivåer ble valgt ut på forhånd, og testet i denne oppgaven. Ved videre arbeid kan parameterne justeres ytterligere ved testing av flere nivåer, som igjen kan føre til en høyere ytelse for modellen. Det kunne i tillegg vært interessant å undersøke om valg av visse parametere, blant annet utelatelse, kan øke treningshastigheten til modellen.

5.5.3 Automatisk konfigurasjon

Iensee et al. [80] utviklet en segmenteringsmetode kalt nnU-Net som automatisk konfigurerer seg selv. Dette inkluderer preprosessering, nettverksarkitekturen, trening og postprosessering for enhver ny oppgave. Ved videre arbeid kan det være interessant å undersøke om egenskapene til nnU-Net kan implementeres hos 2D Dense-Net. Om disse egenskapene hadde blitt implementert kunne modellen optimalisert parameterkombinasjonen sin selv, og tilpasset seg ethvert nytt datasett. Dette vil være ideelt for brukere som ikke har ekspertisen, ressursene eller tid til å tilpasse modellen til deres problem [80].

5.5.4 Kjøring av modellene

Som nevnt i delkapittel 5.3.5, vil den gjennomsnittlige Dice-scoren fra de 36 eksperimentmodellene som vises i Tabell B.1 i Vedlegg B gi varierende ytelsesscore for hver kjøring. Dermed bør flere kjøringer foretas for hver modell, og unionen av ytelsen til alle

kjøringene brukes videre i statistiske tester. Dette vil kunne gi et mer pålitelig resultat som er mindre basert på tilfeldigheter. K-foldet kryssvalidering kan, som nevnt i delkapittel 5.3.4, være fordelaktig å bruke ved små valideringssett. Det kan dermed være verdt å undersøke ved videre arbeid.

5.5.5 Data

Ved økning av størrelsen på datasettet, vil modellen kunne bli mer generell og mindre overtilpasset. Kjøringen på Maastro-datasettet antydet at modellen var bias ovenfor bildene fra OUS, ettersom ytelsen sank. Som diskutert i delkapittel 5.3.1, er det sannsynlig at dette forekom grunnet små forskjeller mellom bildene fra Nederland og bildene modellen har trent på. Ved å legge til bilder fra ulike klinikker rundt om i verden, kan modellen bli mer generell og prestere bedre på eksterne datasett. *The Cancer Imaging Archives* (TCIA) er et arkiv som inneholder medisinske bilder av kreft, tilgjengelig for offentlig nedlastning [81]. Her ligger det datasett bestående av PET/CT-bilder fra pasienter med hode- og halskreft. Ved videre arbeid kan det være hensiktsmessig å undersøke om dette er noe som kan brukes til å øke størrelsen og variabiliteten til datasettet.

Kapittel 6: Konklusjon

Dyplæringsmodellen 2D Dense-Net ble i denne oppgaven trent, optimalisert og evaluert for bruk som hjelpemiddel til svulstinntegning ved hode- og halskreft. Fire ulike parametere ble justert i optimaliseringsprosessen, der 36 modeller med ulike kombinasjoner av disse parameterne ble trent og validert. Treningen og valideringen ble utført på et datasett fra OUS bestående av PET/CT-bilder fra pasienter med hode- og halskreft. Statistiske tester ble benyttet for å undersøke effekten av de ulike parameterne. Den beste modellen ble definert som kombinasjonen av parameterne som ga høyest Dice-score basert på de statistiske testene.

Den beste modellen hadde batchnormalisering, en læringsrate på 10^{-4}, 32 filtre i første lag og 0,5 i utelatelsesrate. Modellen oppnådde en gjennomsnittlig Dice-score per pasient på 0,73 uten augmentering og 0,74 med augmentering på OUS-testsettet. På Maastro-testsettet var ytelsen noe lavere, med 0,64 både med og uten augmentering. Ytelsen av 2D Dense-Net-modellen var sammenlignbar med ytelsen til en U-Net-modell på OUS-testsettet. Det var også tydelig at augmentering hadde en positiv effekt på ytelsen ved bruk på OUS-datasettet. På Maastro-datasettet var imidlertid effekten av augmentering marginal.

Selv om 2D Dense-Net-modellen viste et godt potensial for bruk til segmentering av tumorer og affiserte lymfeknuter på medisinske bilder, gjorde den fortsatt noen grove feilpredikeringer. I noen av situasjonene der tumorene eller de affiserte lymfeknutene var atypiske, eller bildene generelt skilte seg fra majoriteten av bildene modellen trente på, avvek segmenteringen betydelig fra den sanne inntegningen. Dette belyser viktigheten av at segmenteringsmodellen bør være et støtteverktøy for radiologer og onkologer, ikke en erstatning av dem.

Modellen bør allikevel optimaliseres, slik at støtteverktøyet kan bli så sikkert som mulig. Ved videre arbeid kan det være hensiktsmessig å undersøke om andre nivåer av de testede parameterne kan optimalisere 2D Dense-Net ytterligere. Andre parametere bør også testes, og interaksjonen mellom de forskjellige parameterne. Flere kjøringer for en større mengde data til de statistiske testene, og et større og mer variert datasett til trening og validering, er også temaer som bør undersøkes ved videre arbeid.

Kapittel 7: Referanseliste

[1] NHI. "Kreft." Hentet fra: https://nhi.no/kroppen-var/sykdomsprosesser/kreft/?page=all (lest 26.02.2021).

[2] World Health Organization. "Cancer." Hentet fra: https://www.who.int/news-room/fact-sheets/detail/cancer (lest 20.04.2021).

[3] Helsedirektoratet, *Nasjonalt handlingsprogram med retningslinjer for diagnostikk, behandling og oppfølging av hode-/halskreft*. Oslo: Helsedirektoratet, 2020.

[4] Kreftforeningen. "Kreft i Norge." Hentet fra: https://kreftforeningen.no/om-kreft/kreft-i-norge/ (lest 10.03.2021).

[5] Kreftforeningen. "Hode- og halskreft." Hentet fra: https://kreftforeningen.no/om-kreft/kreftformer/hode-og-halskreft/ (lest 05.02.2021).

[6] Helsedirektoratet. "Hode-halskreft." Hentet fra: https://www.helsedirektoratet.no/pakkeforlop/hode-halskreft/behandling-av-hode-halskreft#hovedgrupper-av-behandlingsforlop (lest 10.03.2021).

[7] Helsenorge. "Hode- og halskreft." Hentet fra: https://www.helsenorge.no/sykdom/kreft/hode-og-halskreft/ (lest 11.03.2021).

[8] M. Kosmin *et al.*, "Rapid advances in auto-segmentation of organs at risk and target volumes in head and neck cancer," *Radiotherapy and Oncology*, Online vol. 135, s. 130-140, mars 2019, doi: 10.1016/j.radonc.2019.03.004.

[9] E. Weiss and C. F. Hess, "The impact of gross tumor volume (GTV) and clinical target volume (CTV) definition on the total accuracy in radiotherapy," *Strahlentherapie und Onkologie*, Online vol. 179, no. 1, s. 21-30, jan. 2003, doi: 10.1007/s00066-003-0976-5.

[10] Y. M. Moe, A. R. Groendahl, O. Tomic, E. Dale, E. Malinen, and C. M. Futsaether, "Deep learning-based auto-delineation of gross tumour volumes and involved nodes in PET/CT images of head and neck cancer patients," *European journal of nuclear medicine and molecular imaging*, Online s. 1-11, feb. 2021, doi: 10.1007/s00259-020-05125-x.

[11] Z. Guo, N. Guo, K. Gong, S. a. Zhong, and Q. Li, "Gross tumor volume segmentation for head and neck cancer radiotherapy using deep dense multi-modality network," *Physics in Medicine & Biology*, Online vol. 64, no. 20, okt. 2019, doi: 10.1088/1361-6560/ab440d.

[12] A. R. Groendahl *et al.*, "A comparison of fully automatic segmentation of tumors and involved nodes in PET/CT of head and neck cancers," *Physics in Medicine & Biology*, Online vol. 66, no. 6, mars 2021, doi: 10.1088/1361-6560/abe553.

[13] F. Chollet, *Deep Learning with Python*, 1. utg. New York: Manning Publications Co., 2018.

[14] D. Ravì *et al.*, "Deep learning for health informatics," *IEEE journal of biomedical and health informatics*, Online vol. 21, no. 1, s. 4-21, jan. 2017, doi: 10.1109/JBHI.2016.2636665.

[15] X. Shen, H. Gao, X. Tao, C. Zhou, and J. Jia, "High-Quality Correspondence and Segmentation Estimation for Dual-Lens Smart-Phone Portraits," 2017. [Online]. Hentet fra: arXiv:1704.02205 [cs.CV].

[16] J. Bresnick. "Arguing the Pros and Cons of Artificial Intelligence in Healthcare." Hentet fra: https://healthitanalytics.com/news/arguing-the-pros-and-cons-of-artificial-intelligence-in-healthcare (lest 26.04.2021).

[17] L. Lin *et al.*, "Deep Learning for Automated Contouring of Primary Tumor Volumes by MRI for Nasopharyngeal Carcinoma," *Radiology*, Online vol. 291, no. 3, s. 677-686, juni 2019, doi: 10.1148/radiol.2019182012.

[18] V. Andrearczyk *et al.*, "Overview of the HECKTOR challenge at MICCAI 2020: Automatic Head and Neck Tumor Segmentation in PET/CT," i *3D Head and Neck Tumor Segmentation in PET/CT Challenge*, Cham, jan. 2020: Springer, s. 1-21, doi: 10.1007/978-3-030-67194-5 1.
[19] B. N. Huynh, "Visualization of deep learning in auto-delineation of cancer tumors.," Masteroppgave, Faculty of Science and Technology, Norwegian University of Life Sciences, Ås, 2020.
[20] Kreftforeningen. "Hva er kreft?" Hentet fra: https://kreftforeningen.no/om-kreft/hva-er-kreft/ (lest 04.02.2021).
[21] Oslo Universitetssykehus. "Hode- og halskreft." Hentet fra: https://oslo-universitetssykehus.no/behandlinger/hode-og-halskreft#les-mer-om-stralebehandling-mot-ore-nese-hals-omradet (lest 04.02.2021).
[22] National Cancer Institute. "Head and Neck Cancers." Hentet fra: https://www.cancer.gov/types/head-and-neck/head-neck-fact-sheet (lest 05.02.2021).
[23] Kreftforeningen. "Cellegift." Hentet fra: https://kreftforeningen.no/om-kreft/behandling/cellegift/ (lest 07.02.2021).
[24] J. K. Shultis and R. E. Faw, *Fundamentals of Nuclear Science and Engineering*, 3. utg. Boca Raton: CRC Pressed, 2016.
[25] S. A. Kane and B. A. Gelman, *Introduction to Physics in Modern Medicine*, 3. utg. Boca Raton: CRC Press, 2020.
[26] J. G. Webster, *Medical Instrumentation Application and Design*, 4. utg. New York: John Wiley & Sons, INC., 2009.
[27] Radiopaedia. "Attenuation coefficient." Hentet fra: https://radiopaedia.org/articles/attenuation-coefficient (lest 04.05.2020).
[28] R. L. Wahl, *Principles and Practice of PET and PET/CT*, 2. utg. Philadelphia: Wolters Kluwer, 2009.
[29] D. J. Bell and M. M. Nadrljanski. "Computed tomography." Hentet fra: https://radiopaedia.org/articles/computed-tomography?lang=us (lest 02.02.2021).
[30] A. Haouimi and A. Murphy. "Image reconstruction (CT)." Hentet fra: https://radiopaedia.org/articles/image-reconstruction-ct?lang=us (lest 03.02.2021).
[31] A. Murphy. "Windowing (CT)." Hentet fra: https://radiopaedia.org/articles/windowing-ct?lang=us (lest 02.02.2021).
[32] DSA Direktoratet for strålevern og atomsikkerhet. "Ord og uttrykk for strålefysikk i stråleterapi." Hentet fra: https://dsa.no/laboratoriene/ord-og-uttrykk-for-stralefysikk-i-straleterapi (lest 04.04.2021).
[33] M. P. Woolley and M. Czarniecki. "PET radiotracers." Hentet fra: https://radiopaedia.org/articles/pet-radiotracers (lest 01.02.2021).
[34] RxList. "Fludeoxyglocose." Hentet fra: https://www.rxlist.com/fludeoxyglucose-drug.htm#description (lest 20.03.2021).
[35] M. Saber and A. Shetty. "Positron emission tomography." Hentet fra: https://radiopaedia.org/articles/positron-emission-tomography?lang=us (lest 01.02.2021).
[36] M. E. Phelps, *PET physics, instrumentation, and scanners*, 1. utg. Los Angeles: Springer, 2006.
[37] P. E. Kinahan and J. W. Fletcher, "PET/CT Standardized Uptake Values (SUVs) in Clinical Practice and Assessing Response to Therapy," Online s. 496-505, des. 2010, doi: 10.1053/j.sult.2010.10.001.
[38] Kreftforeningen. "PET/CT." Hentet fra: https://kreftforeningen.no/om-kreft/undersokelser/pet-ct/ (lest 05.02.2021).

[39] S. Raschka and V. Mirjalili, *Python Machine Learning*, 3. utg. Birmingham: Packt Publishing Ltd., 2019.
[40] V. Zocca, G. Spacagna, D. Slater, and P. Roelants, *Python Deep Learning*, 1. utg. Birmingham: Packt Publishing Ltd., 2017.
[41] C. K. Kausahl, "Deep learning for automatic tumor delineation of anal cancer based on MRI, PET and CT images.," Masteroppgave, Faculty of Science and Technology, Norwegian University of Life Sciences, Ås, 2019.
[42] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation," i *2016 fourth international conference on 3D vision (3DV)*, Standford California, juni 2016: IEEE, s. 565-571, doi: 10.1109/3DV.2016.79.
[43] S. Ruder, "An overview of gradient descent optimization algorithms," 2017. [Online]. Hentet fra: arXiv:1609.04747 [cs.LG].
[44] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," 2015. [Online]. Hentet fra: arXiv:1411.4038 [cs.CV].
[45] H. Noh, S. Hong, and B. Han, "Learning Deconvolution Network for Semantic Segmentation," i *Proceedings of the 2015 IEEE international conference on computer vision (ICCV)*, 2015: IEEE Computer Society, s. 1520-1528, doi: 10.1109/iccv.2015.178.
[46] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," i *Medical Image Computing and Computer-Assisted Intervention*, Munich, Germany, nov. 2015: Springer, s. 234-241, doi: 10.1007/978-3-319-24574-4_28.
[47] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal, "The Importance of Skip Connections in Biomedical Image Segmentation," i *Deep Learning and Data Labeling for Medical Applications*, Cham, 2016: Springer, s. 179-187, doi: 10.1007/978-3-319-46976-8_19.
[48] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," i *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, Hawaii, 2017, doi: 10.1109/CVPR.2017.243.
[49] A. A. Taha and A. Hanbury, "Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool," *BMC Medical Imaging*, Online vol. 15, no. 1, s. 1-28, aug. 2015, doi: 10.1186/s12880-015-0068-x.
[50] Y. M. Moe, "Deep learning for automatic delineation of tumours from PET/CT images," Masteroppgave, Faculty of Science and Technology, Norwegian University og Life Sciences, Ås, 2019.
[51] J. M. Moan, C. D. Amdal, E. Malinen, J. G. Svestad, T. V. Bogsrud, and E. Dale, "The prognostic role of 18F-fluorodeoxyglucose PET in head and neck cancer depends on HPV status," *Radiotherapy and Oncology*, Online vol. 140, s. 54-61, nov. 2019, doi: 10.1016/j.radonc.2019.05.019.
[52] Universitetet i Oslo. "Approvals for medical and health research." Hentet fra: https://www.med.uio.no/english/research/phd/application/how-to-apply/approvals-for-medical-and-health-research.html (lest 24.02.2021).
[53] UICC global cancer control. "What is the TNM cancer staging system?" Hentet fra: https://www.uicc.org/resources/tnm (lest 05.05.2021).
[54] B. N. Huynh, "Personlig kommunikasjon," 2021.
[55] A. Collette, *Python and HDF5: Unlocking Scientific Data*, 1. utg. Sebastopol: O'Reilly Media, Inc., 2013.
[56] A. R. Grøndahl, "Personlig kommunikasjon," 2021.

[57] M. Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," 2015. [Online]. Hentet fra: arXiv:1603.04467 [cs.DC].
[58] CIGENE. "CIGENE computational unit." Hentet fra: https://cigene.no/lab-and-infrastructure/cigene-computational-unit/ (lest 26.02.2021).
[59] NMBU Orion user support. "Partitions." Hentet fra: https://nmbu-orion-support.readthedocs.io/en/latest/partitions.html (lest 11.05.2021).
[60] SchedMD. "Quick Start User Guide." Hentet fra: https://slurm.schedmd.com/quickstart.html (lest 26.02.2021).
[61] NMBU Orion user support. "Connecting to Orion." Hentet fra: https://nmbu-orion-support.readthedocs.io/en/latest/connect.html (lest 26.02.2021).
[62] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," 2015. [Online]. Hentet fra: arXiv:1502.03167 [cs.LG].
[63] W. Burger and M. J. Burge, *Digital Image Processing: An Algorithmic Introduction Using Java*, 2. utg. London: Springer, 2016.
[64] M. Ødegaard, "Effekt av dataaugmentering på dyp læring-basert segmentering av hode- og halskreft i PET/CT-bilder," Masteroppgave, Fakultet for realfag og teknologi, Norges miljø- og biovitenskapelige universitet, Ås, 2021.
[65] D. C. Montgomery, *Design and Analysis of Experiments*, 8. utg. Arizona: John Wiley & Sons, Inc., 2013.
[66] N. M. Razali and Y. B. Wah, "Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests," *Journal of Statistical Modeling and Analytics*, Online vol. 2, no. 1, s. 21-33, 2011.
[67] Alan Grafen and R. Hails, *Modern Statistics for the Life Sciences*, 1. utg. United States: Oxford University Press, 2002.
[68] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, Online vol. 32, no. 200, s. 675-701, des. 1937, doi: 10.2307/2279372.
[69] M. Hollander, D. A. Wolfe, and E. Chicken, *Nonparametric Statistical Methods*, 3. utg. Hoboken: John Wiley & Sons, Inc., 2014.
[70] A. P. Zijdenbos, B. M. Dawant, R. A. Margolin, and A. C. Palmer, "Morphometric analysis of white matter lesions in MR images: method and validation," *IEEE Transactions on Medical Imaging*, Online vol. 13, no. 4, s. 716-724, des. 1994, doi: 10.1109/42.363096.
[71] S. R. Krogstie, "Automatisk segmentering av hode- og halskreft i PET/CT-bilder ved bruk av konvolusjonsnettverk," Masteroppgave, Fakultet for realfag og teknologi, Norges miljø- og biovitenskapelige universitet, Ås, 2021.
[72] Y. Yuan, "Automatic Head and Neck Tumor Segmentation in PET/CT with Scale Attention Network," i *3D Head and Neck Tumor Segmentation in PET/CT Challenge*, Cham, 2020: Springer, s. 44-52, doi: 10.1007/978-3-030-67194-5_5.
[73] B. Huang *et al.*, "Fully Automated Delineation of Gross Tumor Volume for Head and Neck Cancer on PET-CT Using Deep Learning: A Dual-Center Study," *Contrast Media & Molecular Imaging*, vol. 2018, 2018, doi: 10.1155/2018/8923028.
[74] A. Iantsen, D. Visvikis, and M. Hatt, "Squeeze-and-Excitation Normalization for Automated Delineation of Head and Neck Primary Tumors in Combined PET and CT Images," in *3D Head and Neck Tumor Segmentation in PET/CT Challenge*, jan. 2020: Springer, s. 37-43, doi: 10.1007/978-3-030-67194-5_4.
[75] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding Data Augmentation for Classification: When to Warp?," i *2016 international conference on*

digital image computing: techniques and applications (DICTA), Queensland, Australia, des. 2016: IEEE, s. 1-6, doi: 10.1109/DICTA.2016.7797091.
[76] D. Bird *et al.*, "Multimodality imaging with CT, MR and FDG-PET for radiotherapy target volume delineation in oropharyngeal squamous cell carcinoma," *BMC Cancer*, Online vol. 15, no. 1, s. 1-10, nov 2015, doi: 10.1186/s12885-015-1867-8.
[77] S. Gudi *et al.*, "Interobserver Variability in the Delineation of Gross Tumour Volume and Specified Organs-at-risk During IMRT for Head and Neck Cancers and the Impact of FDG-PET/CT on Such Variability at the Primary Site," *Journal of Medical Imaging and Radiation Sciences*, Online vol. 48, no. 2, s. 184-192, jun. 2017, doi: 10.1016/j.jmir.2016.11.003.
[78] High-Level Expert Group on AI (AI HLEG), "Ethics Guidelines for Trustworthy AI," European Commission, Guidelines april 2019. [Online]. Hentet fra: https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai
[79] Regjeringen. "Helseregisterloven." Hentet fra: https://www.regjeringen.no/no/tema/helse-og-omsorg/innsikt/helseregisterloven/id2413825/ (lest 10.04.2021).
[80] F. Isensee, P. F. Jaeger, S. A. Kohl, J. Petersen, and K. H. Maier-Hein, "nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation," *Nature Methods*, Online vol. 18, no. 2, s. 203-211, feb. 2021, doi: 10.1038/s41592-020-01008-z.
[81] The Cancer Imaging Archive. "About The Cancer Imaging Archive." Hentet fra: https://www.cancerimagingarchive.net/about-the-cancer-imaging-archive-tcia/ (lest 20.05.2021).

Vedlegg A

Vedlegg A inneholder en detaljert oversikt over arkitekturen til sammenligningsmodellen U-Net brukt i Moe et al. [10] og 2D Dense-Net brukt i denne masteroppgaven.

Detaljert oversikt over 2D U-Net

Tabell A.1. Oversikt over arkitekturen til sammenligningsmodellen U-Net brukt i Moe et al. [10], hentet fra [64].

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(191, 265, 2)	0	
conv2d (Conv2D)	(191, 265, 64)	1216	input_1
batch_normalization (BatchNormalization)	(191, 265, 64)	256	conv2d
conv2d_1 (Conv2D)	(191, 265, 64)	36928	batch_normalization
batch_normalization_1 (BatchNormalization)	(191, 265, 64)	256	conv2d_1
max_pooling2d (MaxPooling2D)	(95, 132, 64)	0	batch_normalization_1
conv2d_2 (Conv2D)	(95, 132, 128)	73856	max_pooling2d
batch_normalization_2 (BatchNormalization)	(95, 132, 128)	512	conv2d_2
conv2d_3 (Conv2D)	(95, 132, 128)	147584	batch_normalization_2
batch_normalization_3 (BatchNormalization)	(95, 132, 128)	512	conv2d_3
max_pooling2d_1 (MaxPooling2D)	(47, 66, 128)	0	batch_normalization_3
conv2d_4 (Conv2D)	(47, 66, 256)	295168	max_pooling2d_1
batch_normalization_4 (BatchNormalization)	(47, 66, 256)	1024	conv2d_4
conv2d_5 (Conv2D)	(47, 66, 256)	590080	batch_normalization_4
batch_normalization_5 (BatchNormalization)	(47, 66, 256)	1024	conv2d_5
max_pooling2d_2 (MaxPooling2D)	(23, 33, 256)	0	batch_normalization_5
conv2d_6 (Conv2D)	(23, 33, 512)	1180160	max_pooling2d_2
batch_normalization_6 (BatchNormalization)	(23, 33, 512)	2048	conv2d_6
conv2d_7 (Conv2D)	(23, 33, 512)	2359808	batch_normalization_6
batch_normalization_7 (BatchNormalization)	(23, 33, 512)	2048	conv2d_7
max_pooling2d_3 (MaxPooling2D)	(11, 16, 512)	0	batch_normalization_7
conv2d_8 (Conv2D)	(11, 16, 1024)	4719616	max_pooling2d_3
batch_normalization_8 (BatchNormalization)	(11, 16, 1024)	4096	conv2d_8
conv2d_9 (Conv2D)	(11, 16, 1024)	9438208	batch_normalization_8
batch_normalization_9 (BatchNormalization)	(11, 16, 1024)	4096	conv2d_9

Tabell A.2. Fortsettelse av arkitekturen til sammenligningsmodellen U-Net brukt i Moe et al. [10], hentet fra [64].

conv2d_transpose (Conv2DTranspose)	(11, 16, 512)	4719104	batch_normalization_9
tf.image.resize (TFOpLambda)	(23, 33, 512)	0	conv2d_transpose
concatenate (Concatenate)	(23, 33, 1024)	0	batch_normalization_7 tf.image.resize
conv2d_10 (Conv2D)	(23, 33, 512)	4719104	concatenate
batch_normalization_10 (BatchNormalization)	(23, 33, 512)	2048	conv2d_10
conv2d_11 (Conv2D)	(23, 33, 512)	2359808	batch_normalization_10
batch_normalization_11 (BatchNormalization)	(23, 33, 512)	2048	conv2d_11
conv2d_transpose_1 (Conv2DTranspose)	(23, 33, 256)	1179904	batch_normalization_11
tf.image.resize_1 (TFOpLambda)	(47, 66, 256)	0	conv2d_transpose_1
concatenate_1 (Concatenate)	(47, 66, 512)	0	batch_normalization_5 tf.image.resize_1
conv2d_12 (Conv2D)	(47, 66, 256)	1179904	concatenate_1
batch_normalization_12 (BatchNormalization)	(47, 66, 256)	1024	conv2d_12
conv2d_13 (Conv2D)	(47, 66, 256)	590080	batch_normalization_12
batch_normalization_13 (BatchNormalization)	(47, 66, 256)	1024	conv2d_13
conv2d_transpose_2 (Conv2DTranspose)	(47, 66, 128)	295040	batch_normalization_13
tf.image.resize_2 (TFOpLambda)	(95, 132, 128)	0	conv2d_transpose_2
concatenate_2 (Concatenate)	(95, 132, 256)	0	batch_normalization_3 tf.image.resize_2
conv2d_14 (Conv2D)	(95, 132, 128)	295040	concatenate_2
batch_normalization_14 (BatchNormalization)	(95, 132, 128)	512	conv2d_14
conv2d_15 (Conv2D)	(95, 132, 128)	147584	batch_normalization_14
batch_normalization_15 (BatchNormalization)	(95, 132, 128)	512	conv2d_15
conv2d_transpose_3 (Conv2DTranspose)	(95, 132, 64)	73792	batch_normalization_15
tf.image.resize_3 (TFOpLambda)	(191, 265, 64)	0	conv2d_transpose_3
concatenate_3 (Concatenate)	(191, 265, 128)	0	batch_normalization_1 tf.image.resize_3
conv2d_16 (Conv2D)	(191, 265, 64)	73792	concatenate_3
batch_normalization_16 (BatchNormalization)	(191, 265, 64)	256	conv2d_16
conv2d_17 (Conv2D)	(191, 265, 64)	36928	batch_normalization_16
batch_normalization_17 (BatchNormalization)	(191, 265, 64)	256	conv2d_17
conv2d_18 (Conv2D)	(191, 265, 1)	577	batch_normalization_17

Total params: 34,536,833
Trainable params: 34,525,057
Non-trainable params: 11,776

Detaljert oversikt over 2D Dense-Net

Tabell A.3. Oversikt over arkitekturen til 2D Dense-Net, som ble brukt i denne masteroppgaven.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(191, 265, 2)	0	
conv2d (Conv2D)	(191, 265, 32)	608	input_1
batch_normalization (BatchNormalization)	(191, 265, 32)	128	conv2d
dropout (Dropout)	(191, 265, 32)	0	batch_normalization
conv2d_1 (Conv2D)	(191, 265, 32)	9248	dropout
batch_normalization_1 (BatchNormalization)	(191, 265, 32)	128	conv2d_1
concatenate (Concatenate)	(191, 265, 64)	0	dropout batch_normalization_1
conv2d_2 (Conv2D)	(191, 265, 32)	18464	concatenate
batch_normalization_2 (Batchnormalization)	(191, 265, 32)	128	conv2d_2
concatenate_1 (Concatenate)	(191, 265, 96)	0	concatenate batch_normalization_2
conv2d_3 (Conv2D)	(191, 265, 32)	27680	concatenate_1
batch_normalization_3 (BatchNormalization)	(191, 265, 32)	128	conv2d_3
concatenate_2 (Concatenate)	(191, 265, 96)	0	batch_normalization_1 batch_normalization_2 batch_normalization_3
concatenate_3 (Concatenate)	(191, 265, 128)	0	batch_normalization concatenate_2
dropout_1 (Dropout)	(191, 265, 128)	0	concatenate_3
conv2d_5 (Conv2D)	(191, 265, 48)	55344	dropout_1
batch_normalization_5 (BatchNormalization)	(191, 265, 48)	192	conv2d_5
concatenate_4 (Concatenate)	(191, 265, 176)	0	dropout_1 batch_normalization_5
conv2d_6 (Conv2D)	(191, 265, 48)	76080	concatenate_4
batch_normalization_6 (BatchNormalization)	(191, 265, 48)	192	conv2d_6
concatenate_5 (Concatenate)	(191, 265, 224)	0	concatenate_4 batch_normalization_6
conv2d_7 (Conv2D)	(191, 265, 48)	96816	concatenate_5
batch_normalization_7 (BatchNormalization)	(191, 265, 48)	192	conv2d_7
concatenate_6 (Concatenate)	(191, 265, 272)	0	concatenate_5 batch_normalization_7
conv2d_8 (Conv2D)	(191, 265, 48)	117552	concatenate_6
batch_normalization_8 (BatchNormalization)	(191, 265, 48)	192	conv2d_8
conv2d_4 (Conv2D)	(96, 133, 48)	4656	concatenate_2
concatenate_7 (Concatenate)	(191, 265, 192)	0	batch_normalization_5 batch_normalization_6 batch_normalization_7 batch_normalization_8
batch_normalization_4 (BatchNormalization)	(96, 133, 48)	192	conv2d_4
tf.image.resize (TFOpLambda)	(96, 133, 192)	0	concatenate_7
concatenate_8 (Concatenate)	(96, 133, 240)	0	batch_normalization_4 tf.image.resize
dropout_2 (Dropout)	(96, 133, 240)	0	concatenate_8
conv2d_10 (Conv2D)	(96, 133, 64)	138304	dropout_2

Tabell A.4. Fortsettelse av oversikt over arkitekturen til 2D Dense-Net, som ble brukt i denne masteroppgaven.

batch_normalization_10 (BatchNormalization)	(96, 133, 64)	256	conv2d_10
concatenate_9 (Concatenate)	(96, 133, 304)	0	dropout_2 batch_normalization_10
conv2d_11 (Conv2D)	(96, 133, 64)	175168	concatenate_9
batch_normalization_11 (BatchNormalization)	(96, 133, 64)	256	conv2d_11
concatenate_10 (Concatenate)	(96, 133, 368)	0	concatenate_9 batch_normalization_11
conv2d_12 (Conv2D)	(96, 133, 64)	212032	concatenate_10
batch_normalization_12 (BatchNormalization)	(96, 133, 64)	256	conv2d_12
concatenate_11 (Concatenate)	(96, 133, 432)	0	concatenate_10 batch_normalization_12
conv2d_13 (Conv2D)	(96, 133, 64)	248896	concatenate_11
batch_normalization_13 (BatchNormalization)	(96, 133, 64)	256	conv2d_13
concatenate_12 (Concatenate)	(96, 133, 496)	0	concatenate_11 batch_normalization_13
conv2d_14 (Conv2D)	(96, 133, 64)	285760	concatenate_12
conv2d_9 (Conv2D)	(96, 133, 64)	12352	concatenate_7
batch_normalization_14 (BatchNormalization)	(96, 133, 64)	256	conv2d_14
batch_normalization_9 (BatchNormalization)	(96, 133, 64)	256	conv2d_9
concatenate_13 (Concatenate)	(96, 133, 320)	0	batch_normalization_10 batch_normalization_11 batch_normalization_12 batch_normalization_13 batch_normalization_14
concatenate_14 (Concatenate)	(96, 133, 384)	0	batch_normalization_9 concatenate_13
dropout_3 (Dropout)	(96, 133, 384)	0	concatenate_14
conv2d_16 (Conv2D)	(96, 133, 80)	276560	dropout_3
batch_normalization_16 (BatchNormalization)	(96, 133, 80)	320	conv2d_16
concatenate_15 (Concatenate)	(96, 133, 464)	0	dropout_3 batch_normalization_16
conv2d_17 (Conv2D)	(96, 133, 80)	334160	concatenate_15
batch_normalization_17 (BatchNormalization)	(96, 133, 80)	320	conv2d_17
concatenate_16 (Concatenate)	(96, 133, 544)	0	concatenate_15 batch_normalization_17
conv2d_18 (Conv2D)	(96, 133, 80)	391760	concatenate_16
batch_normalization_18 (BatchNormalization)	(96, 133, 80)	320	conv2d_18
concatenate_17 (Concatenate)	(96, 133, 624)	0	concatenate_16 batch_normalization_18
conv2d_19 (Conv2D)	(96, 133, 80)	449360	concatenate_17
batch_normalization_19 (BatchNormalization)	(96, 133, 80)	320	conv2d_19
concatenate_18 (Concatenate)	(96, 133, 704)	0	concatenate_17 batch_normalization_19
conv2d_20 (Conv2D)	(96, 133, 80)	506960	concatenate_18

Tabell A.5. Fortsettelse av oversikt over arkitekturen til 2D Dense-Net, som ble brukt i denne masteroppgaven.

batch_normalization_20 (BatchNormalization)	(96, 133, 80)	320	conv2d_20
concatenate_19 (Concatenate)	(96, 133, 784)	0	concatenate_18 batch_normalization_20
conv2d_21 (Conv2D)	(96, 133, 80)	564560	concatenate_19
batch_normalization_21 (BatchNormalization)	(96, 133, 80)	320	conv2d_21
conv2d_15 (Conv2D)	(48, 67, 80)	25680	concatenate_13
concatenate_20 (Concatenate)	(96, 133, 480)	0	batch_normalization_16 batch_normalization_17 batch_normalization_18 batch_normalization_19 batch_normalization_20 batch_normalization_21
batch_normalization_15 (BatchNormalization)	(48, 67, 80)	320	conv2d_15
tf.image.resize_1 (TFOpLambda)	(48, 67, 480)	0	concatenate_20
concatenate_21 (Concatenate)	(48, 67, 560)	0	batch_normalization_15 tf.image.resize_1
dropout_4 (Dropout)	(48, 67, 560)	0	concatenate_21
conv2d_23 (Conv2D)	(48, 67, 96)	483936	dropout_4
batch_normalization_23 (BatchNormalization)	(48, 67, 96)	384	conv2d_23
concatenate_22 (Concatenate)	(48, 67, 656)	0	dropout_4 batch_normalization_23
conv2d_24 (Conv2D)	(48, 67, 96)	566880	concatenate_22
batch_normalization_24 (BatchNormalization)	(48, 67, 96)	384	conv2d_24
concatenate_23 (Concatenate)	(48, 67, 752)	0	concatenate_22 batch_normalization_24
conv2d_25 (Conv2D)	(48, 67, 96)	649824	concatenate_23
batch_normalization_25 (BatchNormalization)	(48, 67, 96)	384	conv2d_25
concatenate_24 (Concatenate)	(48, 67, 848)	0	concatenate_23 batch_normalization_25
conv2d_26 (Conv2D)	(48, 67, 96)	732768	concatenate_24
batch_normalization_26 (BatchNormalization)	(48, 67, 96)	384	conv2d_26
concatenate_25 (Concatenate)	(48, 67, 944)	0	concatenate_24 batch_normalization_26
conv2d_27 (Conv2D)	(48, 67, 96)	815712	concatenate_25
batch_normalization_27	(48, 67, 96)	384	conv2d_27
concatenate_26 (Concatenate)	(48, 67, 1040)	0	concatenate_25 batch_normalization_27
conv2d_28 (Conv2D)	(48, 67, 96)	898656	concatenate_26
batch_normalization_28 (BatchNormalization)	(48, 67, 96)	384	conv2d_28
concatenate_27 (Concatenate)	(48, 67, 1136)	0	concatenate_26 batch_normalization_28
conv2d_29 (Conv2D)	(48, 67, 96)	981600	concatenate_27
batch_normalization_29 (BatchNormalization)	(48, 67, 96)	384	conv2d_29
concatenate_28 (Concatenate)	(48, 67, 672)	0	batch_normalization_23 batch_normalization_24 batch_normalization_25 batch_normalization_26 batch_normalization_27 batch_normalization_28 batch_normalization_29
conv2d_transpose (Conv2DTranspose)	(96, 134, 80)	53840	concatenate_28

Tabell A.6. Fortsettelse av oversikt over arkitekturen til 2D Dense-Net, som ble brukt i denne masteroppgaven.

batch_normalization_30 (BatchNormalization)	(96, 134, 80)	320	conv2d_transpose
tf.image.resize_2 (TFOpLambda)	(48, 67, 480)	0	concatenate_20
tf.image.resize_3 (TFOpLambda)	(48, 67, 80)	0	batch_normalization_30
concatenate_29 (Concatenate)	(48, 67, 640)	0	batch_normalization_15 tf.image.resize_2 tf.image.resize_3
dropout_5 (Dropout)	(48, 67, 640)	0	concatenate_29
conv2d_30 (Conv2D)	(48, 67, 80)	460880	dropout_5
batch_normalization_31 (BatchNormalization)	(48, 67, 80)	320	conv2d_30
concatenate_30 (Concatenate)	(48, 67, 720)	0	dropout_5 batch_normalization_31
conv2d_31 (Conv2D)	(48, 67, 80)	518480	concatenate_30
batch_normalization_32 (BatchNormalization)	(48, 67, 80)	320	conv2d_31
concatenate_31 (Concatenate)	(48, 67, 800)	0	concatenate_30 batch_normalization_32
conv2d_32 (Conv2D)	(48, 67, 80)	576080	concatenate_31
batch_normalization_33	(48, 67, 80)	320	conv2d_32
concatenate_32 (Concatenate)	(48, 67, 880)	0	concatenate_31 batch_normalization_33
conv2d_33 (Conv2D)	(48, 67, 80)	633680	concatenate_32
batch_normalization_34 (BatchNormalization)	(48, 67, 80)	320	conv2d_33
concatenate_33 (Concatenate)	(48, 67, 960)	0	concatenate_32 batch_normalization_34
conv2d_34 (Conv2D)	(48, 67, 80)	691280	concatenate_33
batch_normalization_35 (BatchNormalization)	(48, 67, 80)	320	conv2d_34
concatenate_34 (Concatenate)	(48, 67, 1040)	0	concatenate_33 batch_normalization_35
conv2d_35 (Conv2D)	(48, 67, 80)	748880	concatenate_34
batch_normalization_36	(48, 67, 80)	320	conv2d_35
concatenate_35 (Concatenate)	(48, 67, 480)	0	batch_normalization_31 batch_normalization_32 batch_normalization_33 batch_normalization_34 batch_normalization_35 batch_normalization_36
conv2d_transpose_1 (Conv2DTranspose)	(96, 134, 64)	30784	concatenate_35
batch_normalization_37 (BatchNormalization)	(96, 134, 64)	256	conv2d_transpose_1
tf.image.resize_4 (TFOpLambda)	(96, 133, 64)	0	batch_normalization_37
concatenate_36 (Concatenate)	(96, 133, 448)	0	batch_normalization_9 concatenate_13 tf.image.resize_4
dropout_6 (Dropout)	(96, 133, 448)	0	concatenate_36
conv2d_36 (Conv2D)	(96, 133, 64)	258112	dropout_6
batch_normalization_38 (BatchNormalization)	(96, 133, 64)	256	conv2d_36
concatenate_37 (Concatenate)	(96, 133, 512)	0	dropout_6 batch_normalization_38
conv2d_37 (Conv2D)	(96, 133, 64)	294976	concatenate_37
batch_normalization_39 (BatchNormalization)	(96, 133, 64)	256	conv2d_37

Tabell A.7. Fortsettelse av oversikt over arkitekturen til 2D Dense-Net, som ble brukt i denne masteroppgaven.

concatenate_38 (Concatenate)	(96, 133, 576)	0	concatenate_37 batch_normalization_39
conv2d_38 (Conv2D)	(96, 133, 64)	331840	concatenate_38
batch_normalization_40 (BatchNormalization)	(96, 133, 64)	256	conv2d_38
concatenate_39 (Concatenate)	(96, 133, 640)	0	concatenate_38 batch_normalization_40
conv2d_39 (Conv2D)	(96, 133, 64)	368704	concatenate_39
batch_normalization_41 (BatchNormalization)	(96, 133, 64)	256	conv2d_39
concatenate_40 (Concatenate)	(96, 133, 704)	0	concatenate_39 batch_normalization_41
conv2d_40 (Conv2D)	(96, 133, 64)	405568	concatenate_40
batch_normalization_42 (BatchNormalization)	(96, 133, 64)	256	conv2d_40
concatenate_41 (Concatenate)	(96, 133, 320)	0	batch_normalization_38 batch_normalization_39 batch_normalization_40 batch_normalization_41 batch_normalization_42
conv2d_transpose_2 (Conv2DTranspose)	(192, 266, 48)	15408	concatenate_41
batch_normalization_43 (BatchNormalization)	(192, 266, 48)	192	conv2d_transpose_2
tf.image.resize_5 (TFOpLambda)	(96, 133, 192)	0	concatenate_7
tf.image.resize_6 (TFOpLambda)	(96, 133, 48)	0	batch_normalization_43
concatenate_42 (Concatenate)	(96, 133, 288)	0	batch_normalization_4 tf.image.resize_5 tf.image.resize_6
dropout_7 (Dropout)	(96, 133, 288)	0	concatenate_42
conv2d_41 (Conv2D)	(96, 133, 48)	124464	dropout_7
batch_normalization_44 (BatchNormalization)	(96, 133, 48)	192	conv2d_41
concatenate_43 (Concatenate)	(96, 133, 336)	0	dropout_7 batch_normalization_44
conv2d_42 (Conv2D)	(96, 133, 48)	145200	concatenate_43
batch_normalization_45 (BatchNormalization)	(96, 133, 48)	192	conv2d_42
concatenate_44 (Concatenate)	(96, 133, 384)	0	concatenate_43 batch_normalization_45
conv2d_43 (Conv2D)	(96, 133, 48)	165936	concatenate_44
batch_normalization_46	(96, 133, 48)	192	conv2d_43
concatenate_45 (Concatenate)	(96, 133, 432)	0	concatenate_44 batch_normalization_46
conv2d_44 (Conv2D)	(96, 133, 48)	186672	concatenate_45
batch_normalization_47 (BatchNormalization)	(96, 133, 48)	192	conv2d_44
concatenate_46 (Concatenate)	(96, 133, 192)	0	batch_normalization_44 batch_normalization_45 batch_normalization_46 batch_normalization_47
conv2d_transpose_3 (Conv2DTranspose)	(192, 266, 32)	6176	concatenate_46
batch_normalization_48 (BatchNormalization)	(192, 266, 32)	128	conv2d_transpose_3
tf.image.resize_7 (TFOpLambda)	(191, 265, 32)	0	batch_normalization_48
concatenate_47 (Concatenate)	(191, 265, 160)	0	batch_normalization concatenate_2 tf.image.resize_7

Tabell A.8. Fortsettelse av oversikt over arkitekturen til 2D Dense-Net, som ble brukt i denne masteroppgaven.

dropout_8 (Dropout)	(191, 265, 160)	0	concatenate_47
conv2d_45 (Conv2D)	(191, 265, 32)	46112	dropout_8
batch_normalization_49 (BatchNormalization)	(191, 265, 32)	128	conv2d_45
concatenate_48 (Concatenate)	(191, 265, 192)	0	dropout_8 batch_normalization_49
conv2d_46 (Conv2D)	(191, 265, 32)	55328	concatenate_48
batch_normalization_50 (BatchNormalization)	(191, 265, 32)	128	conv2d_46
concatenate_49 (Concatenate)	(191, 265, 224)	0	concatenate_48 batch_normalization_50
conv2d_47 (Conv2D)	(191, 265, 32)	64544	concatenate_49
batch_normalization_51 (BatchNormalization)	(191, 265, 32)	128	conv2d_47
concatenate_50 (Concatenate)	(191, 265, 96)	0	batch_normalization_49 batch_normalization_50 batch_normalization_51
conv2d_48 (Conv2D)	(191, 265, 1)	865	concatenate_50

Total params: 15,354,369
Trainable params: 15,347,777
Non-trainable params: 6,592

Vedlegg B

Vedlegg B inneholder oversikt over de 36 eksperimentmodellene som ble kjørt og den gjennomsnittlige Dice-scoren per tverrsnitt hvert eksperiment oppnådde. Vedlagt ligger også et eksempel på en konfigurasjonsfil som ble bruk til kjøringen av eksperimentene. Resten av konfigurasjonsfilene kan finnes på GitHub-siden: https://github.com/malenegjeng/cnn-template.

Eksperimentmodellene

Tabell B.1. Oversikt over kombinasjonene av parameterne i modellene som ble trent og kjørt på valideringssettet, med gjennomsnittlig Dice-score per tverrsnitt.

Eksperiment	Batchnormalisering	Utelatelsesrate	Læringsrate	Antall filter	Dice-score
1	Ja	0	1,00E-03	48 + 16	0,615
2	Nei	0	1,00E-03	48 + 16	4,481e-11
3	Ja	0,5	1,00E-03	48 + 16	0,595
4	Nei	0,5	1,00E-03	48 + 16	4,481e-11
5	Ja	0	1,00E-04	48 + 16	0,602
6	Nei	0	1,00E-04	48 + 16	0,620
7	Ja	0,5	1,00E-04	48 + 16	0,591
8	Nei	0,5	1,00E-04	48 + 16	0,620
9	Ja	0	1,00E-02 eller 1,00E-05	48 + 16	0,580
10	Nei	0	1,00E-02 eller 1,00E-05	48 + 16	0,596
11	Ja	0,5	1,00E-02 eller 1,00E-05	48 + 16	0,570
12	Nei	0,5	1,00E-02 eller 1,00E-05	48 + 16	0,612
13	Ja	0	1,00E-03	32 + 16	0,610
14	Nei	0	1,00E-03	32 + 16	4,481e-11
15	Ja	0,5	1,00E-03	32 + 16	0,605
16	Nei	0,5	1,00E-03	32 + 16	0,597
17	Ja	0	1,00E-04	32 + 16	0,594
18	Nei	0	1,00E-04	32 + 16	0,621
19	Ja	0,5	1,00E-04	32 + 16	0,609
20	Nei	0,5	1,00E-04	32 + 16	0,600
21	Ja	0	1,00E-02 eller 1,00E-05	32 + 16	0,568
22	Nei	0	1,00E-02 eller 1,00E-05	32 + 16	0,594
23	Ja	0,5	1,00E-02 eller 1,00E-05	32 + 16	0,578
24	Nei	0,5	1,00E-02 eller 1,00E-05	32 + 16	0,614
25	Ja	0	1,00E-03	64 + 16	0,615
26	Nei	0	1,00E-03	64 + 16	4,481e-11
27	Ja	0,5	1,00E-03	64 + 16	0,598
28	Nei	0,5	1,00E-03	64 + 16	0,582
29	Ja	0	1,00E-04	64 + 16	0,611
30	Nei	0	1,00E-04	64 + 16	0,625
31	Ja	0,5	1,00E-04	64 + 16	0,621
32	Nei	0,5	1,00E-04	64 + 16	0,610
33	Ja	0	1,00E-02 eller 1,00E-05	64 + 16	0,570
34	Nei	0	1,00E-02 eller 1,00E-05	64 + 16	0,605
35	Ja	0,5	1,00E-02 eller 1,00E-05	64 + 16	0,577
36	Nei	0,5	1,00E-02 eller 1,00E-05	64 + 16	0,605

Eksempel på JSON-konfigurasjonsfil

Under vises et eksempel på en konfigurasjonsfil som ble brukt til å kjøre et eksperiment i Orion. Eksperimentet hadde batchnormalisering, ingen utelatelse, 10^{-4} som læringsrate og 48 filtre i første lag.

```
{
    "dataset_params": {
        "class_name": "H5Reader",
        "config": {
            "filename": "/home/work/maleneg/hn_delin/full_dataset_singleclass.h5",
            "batch_size": 16,
            "x_name": "input",
            "y_name": "target",
            "batch_cache": 10,
            "shuffle": true,
            "fold_prefix": "",
            "train_folds": [
                "train"
            ],
            "val_folds": [
                "val"
            ],
            "test_folds": [
                "test"
            ],
            "preprocessors": [
                {
                    "class_name": "HounsfieldWindowingPreprocessor",
                    "config": {
                        "window_center": 70,
                        "window_width": 200,
                        "channel": 0
                    }
                },
                {
                    "class_name": "ImageNormalizerPreprocessor",
                    "config": {
                        "vmin": [
                            -100,
                            0
                        ],
                        ],
                        "vmax": [
                            100,
                            25
                        ]
                    }
                }
            ]
        }
    },
    "train_params": {
        "epochs": 200,
        "callbacks": [
            {
                "class_name": "EarlyStopping",
                "config": {
                    "monitor": "val_loss",
                    "patience": 30
                }
            }
        ]
    },
    "input_params": {
        "shape": [
            191,
            265,
            2
        ]
    },
    "model_params": {
        "loss": {
            "class_name": "BinaryFbetaLoss"
        },
```

```
        "optimizer": {
            "class_name": "adam",
            "config": {
                "learning_rate": 0.0001
            }
        },
        "metrics": [
            {
                "class_name": "BinaryFbeta"
            },
            {
                "class_name": "Dice"
            }
        ]
    },
    "architecture": {
        "type": "DenseNet",
        "layers": [
            {
                "name": "down_conv1",
                "class_name": "Conv2D",
                "config": {
                    "filters": 48,
                    "kernel_size": 3,
                    "activation": "relu",
                    "kernel_initializer": "he_normal",
                    "padding": "same", "strides": 1
                },
                "normalizer": {
                    "class_name": "BatchNormalization"
                }
            },
            {
```

```
                "dense_block": 3,
                "name": "dense_block1",
                "class_name": "Conv2D",
                "config": {
                    "filters": 48,
                    "kernel_size": 3,
                    "activation": "relu",
                    "kernel_initializer": "he_normal",
                    "padding": "same"
                },
                "normalizer": {
                    "class_name": "BatchNormalization"
                }
            },
            {
                "name": "down_conv2",
                "class_name": "Conv2D",
                "config": {
                    "filters": 64,
                    "kernel_size": 1,
                    "activation": "relu",
                    "kernel_initializer": "he_normal",
                    "padding": "same",
                    "strides": 2
                },
                "normalizer": {
                    "class_name": "BatchNormalization"
                },

                "inputs": [
                    "down_conv1",
                    "dense_block1"
                ]
```

```
        },
        {
            "dense_block": 4,
            "name": "dense_block2",
            "class_name": "Conv2D",
            "config": {
                "filters": 64,
                "kernel_size": 3,
                "activation": "relu",
                "kernel_initializer": "he_normal",
                "padding": "same"
            },
            "normalizer": {
                "class_name": "BatchNormalization"
            }
        },
        {
            "name": "down_conv3",
            "class_name": "Conv2D",
            "config": {
                "filters": 80,
                "kernel_size": 1,
                "activation": "relu",
                "kernel_initializer": "he_normal",
                "padding": "same",
                "strides": 2
            },
            "normalizer": {
                "class_name": "BatchNormalization"
            },
            "inputs": [
                "down_conv2",
                "dense_block2"
            ]
        },
        {
            "dense_block": 5,
            "name": "dense_block3",
            "class_name": "Conv2D",
            "config": {
                "filters": 80,
                "kernel_size": 3,
                "activation": "relu",
                "kernel_initializer": "he_normal",
                "padding": "same"
            },
            "normalizer": {
                "class_name": "BatchNormalization"
            }
        },
        {
            "name": "down_conv4",
            "class_name": "Conv2D",
            "config": {
                "filters": 96,
                "kernel_size": 1,
                "activation": "relu",
                "kernel_initializer": "he_normal",
                "padding": "same",
                "strides": 2
            },
            "normalizer": {
                "class_name": "BatchNormalization"
            },
            "inputs": [
                "down_conv3",
                "dense_block3"
                ]
        },
```

```
        {
            "dense_block": 6,
            "name": "dense_block4",
            "class_name": "Conv2D",
            "config": {
                "filters": 96,
                "kernel_size": 3,
                "activation": "relu",
                "kernel_initializer": "he_normal",
                "padding": "same"
            },
            "normalizer": {
                "class_name": "BatchNormalization"
            }
        },
        {
            "name": "down_conv5",
            "class_name": "Conv2D",
            "config": {
                "filters": 112,
                "kernel_size": 1,
                "activation": "relu",
                "kernel_initializer": "he_normal",
                "padding": "same",
                "strides": 2
            },
            "normalizer": {
                "class_name": "BatchNormalization"
            },
            "inputs": [
                "down_conv4",
                "dense_block4"
                ]
        },
        {
```

```
            "dense_block": 7,
            "name": "dense_block5",
            "class_name": "Conv2D",
            "config": {
                "filters": 112,
                "kernel_size": 3,
                "activation": "relu",
                "kernel_initializer": "he_normal",
                "padding": "same"
            },
            "normalizer": {
                "class_name": "BatchNormalization"
            }
        },
        {
            "name": "up_conv4",
            "class_name": "Conv2DTranspose",
            "config": {
                "filters": 96,
                "kernel_size": 1,
                "activation": "relu",
                "kernel_initializer": "he_normal",
                "padding": "same",
                "strides": 2
            },
            "normalizer": {
                "class_name": "BatchNormalization"
            }
        },
        {
            "dense_block": 6,
            "class_name": "Conv2D",
            "config": {
                "filters": 96,
                "kernel_size": 3,
                "activation": "relu",
```

```
                "kernel_initializer": "he_normal",
                "padding": "same"
            },
            "normalizer": {
                "class_name": "BatchNormalization"
            },
            "inputs": [
                "down_conv4",
                "dense_block4",
                "up_conv4"
                ]
        },
        {
            "name": "up_conv3",
            "class_name": "Conv2DTranspose",
            "config": {
                "filters": 80,
                "kernel_size": 1,
                "activation": "relu",
                "kernel_initializer": "he_normal",
                "padding": "same", "strides": 2
            },
            "normalizer": {
                "class_name": "BatchNormalization"
            }
        },
        {
            "dense_block": 5,
            "class_name": "Conv2D",
            "config": {
                "filters": 80,
                "kernel_size": 3,
                "activation": "relu",
                "kernel_initializer": "he_normal",
                "padding": "same"
            },
```

```
            "normalizer": {
                "class_name": "BatchNormalization"
            },
            "inputs": [
                "down_conv3",
                "dense_block3",
                "up_conv3"
                ]
        },
        {
            "name": "up_conv2",
            "class_name": "Conv2DTranspose",
            "config": {
                "filters": 64,
                "kernel_size": 1,
                "activation": "relu",
                "kernel_initializer": "he_normal",
                "padding": "same",
                "strides": 2
            },
            "normalizer": {
                "class_name": "BatchNormalization"
            }
        },
        {
            "dense_block": 4,
            "class_name": "Conv2D",
            "config": {
                "filters": 64,
                "kernel_size": 3,
                "activation": "relu",
                "kernel_initializer": "he_normal",
                "padding": "same"
            },
            "normalizer": {
                "class_name": "BatchNormalization"
```

```
                },
                "inputs": [
                    "down_conv2",
                    "dense_block2",
                    "up_conv2"
                    ]
            },
            {
                "name": "up_conv1",
                "class_name": "Conv2DTranspose",
                "config": {
                    "filters": 48,
                    "kernel_size": 1,
                    "activation": "relu",
                    "kernel_initializer": "he_normal",
                    "padding": "same",
                    "strides": 2
                },
                "normalizer": {
                    "class_name": "BatchNormalization"
                }
            },
            {
                "dense_block": 3,
                "class_name": "Conv2D",
                "config": {
                    "filters": 48,
                    "kernel_size": 3,
                    "activation": "relu",
                    "kernel_initializer": "he_normal",
                    "padding": "same"
                },
                "normalizer": {
                    "class_name": "BatchNormalization"
                },
                "inputs": [
                    "down_conv1",
                    "dense_block1",
                    "up_conv1"
                    ]
            },
            {
                "class_name": "Conv2D",
                "config": {
                    "filters": 1,
                    "kernel_size": 3,
                    "activation": "sigmoid",
                    "kernel_initializer": "he_normal",
                    "padding": "same"
                }
            }
        ]
    }
}
```

Vedlegg C

Vedlegg C inneholder en oversikt over statistiske metoder som ble brukt i denne masteroppgaven. Skript fra RStudio for bruk av de statistiske testene, og tilhørende plott er vedlagt.

Undersøkelse av krav om normalfordeling samt transformasjoner

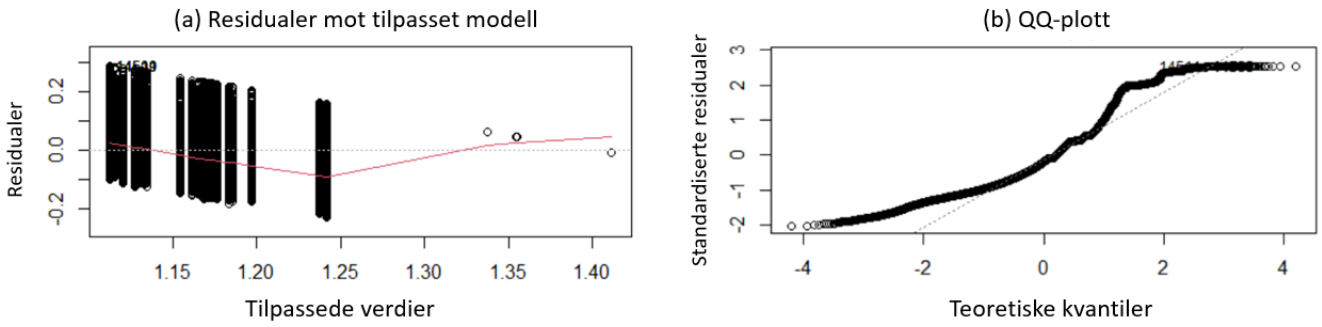
```r
#Undersøker kravet om normalfordelte residualer

#Laster inn resultatdata
data <- Dice_per_slice_totalt_med_eksperimentnummer

#Lager uavhengige faktorvariabler
data$n_filters <- factor(data$n_filters,
                         levels = c(32,48,64),
                         labels = c("F_32", "F_48", "F_64"))
data$BatchNorm <- factor(data$BatchNorm,
                         levels = c("Yes","No"),
                         labels = c("Yes", "No"))
data$Dropout_rate <- factor(data$Dropout_rate,
                            levels = c("0","0,5"),
                            labels = c("0", "0,5"))
data$Learning_rate <- factor(data$Learning_rate,
                             levels = c("1,00E-03","1,00E-04","1,00E-05"),
                             labels = c("3","4","5"))
#ANOVA med interaksjoner
AOV.1 <- aov(f1_score ~n_filters*BatchNorm + n_filters*Dropout_rate + n_filters*Learning_rate + BatchNorm*Dropout_rate +
               BatchNorm*Learning_rate + Dropout_rate*Learning_rate, data = data)
summary(AOV.1)


# Plotter diagnoseplott
plot(AOV.1, 1) #Residuals vs. fitted
plot(AOV.1, 2) #QQ-plott

```

```r
#Bruker Anderson Darling for å sjekke om residualene er normalfordelt
library(nortest)
ad.test(AOV.1$residuals)

#Residualene er ikke normalfordelt -> utfører transformasjoner

#Logaritmisk transformasjon
data$f1_scoreTransformed_log = log10(max(data$f1_score+1) - data$f1_score)

#Kvadratrot-transformasjon
data$f1_scoreTransformed_sqrt = sqrt(max(data$f1_score+1) - data$f1_score)

#Invers-transformasjon
data$f1_scoreTransformed_invers = 1/(max(data$f1_score+1) - data$f1_score)

#Lager en ANOVA-modell for den transformerte dataen ved å sette inn for "f1_scoreTransformed" i koden under:
#1. f1_scoreTransformed_log
#2. f1_scoreTransformed_sqrt
#3. f1_scoreTransformed_invers
AOV.2 <- aov(f1_scoreTransformed ~n_filters*BatchNorm + n_filters*Dropout_rate + n_filters*Learning_rate
             + BatchNorm*Dropout_rate + BatchNorm*Learning_rate + Dropout_rate*Learning_rate, data = data)
summary(AOV.2)

#Plotter diagnoseplott for de transformerte modellene
plot(AOV.2, 1)#Residuals vs. fitted
plot(AOV.2, 2) #QQ-plott

```

```r
#Bruker Anderson Darling for å sjekke om residualene er normalfordelt
ad.test(AOV.2$residuals)

#Boxcox-trasformasjon
library(MASS)

y <- data$f1_score + 0.5
m <- aov(y ~ n_filters*BatchNorm + n_filters*Dropout_rate + n_filters*Learning_rate + BatchNorm*Dropout_rate +
           BatchNorm*Learning_rate + Dropout_rate*Learning_rate, data = data)
b <- boxcox(m, lambda = seq(1,2, by = 0.01))

#Lamda (1.7) velges basert på boxcox-plottet
m_transformert <- aov((((y ^ 1.7) - 1) / 1.7) ~ data$BatchNorm + data$Dropout_rate + data$Learning_rate + data$n_filters)

#Plotter diagnoseplott av den transformerte dataen
#Fitted vs residuals
plot(fitted(m_transformert), resid(m_transformert), col = "dodgerblue",
     pch = 20, cex = 1.5, xlab = "Fitted", ylab = "Residuals")
abline(h = 0, lty = 2, col = "darkorange", lwd = 2)
#QQ-plot
plot(m_transformert, 2)

#Bruker Anderson Darling for å sjekke om residualene er normalfordelt
ad.test(m_transformert$residuals)

```

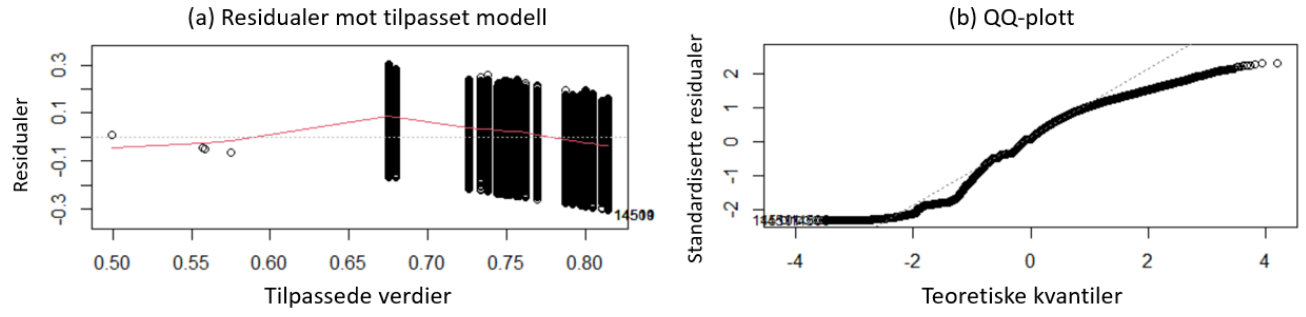
Diagnoseplott

Kvadratrot-transformering



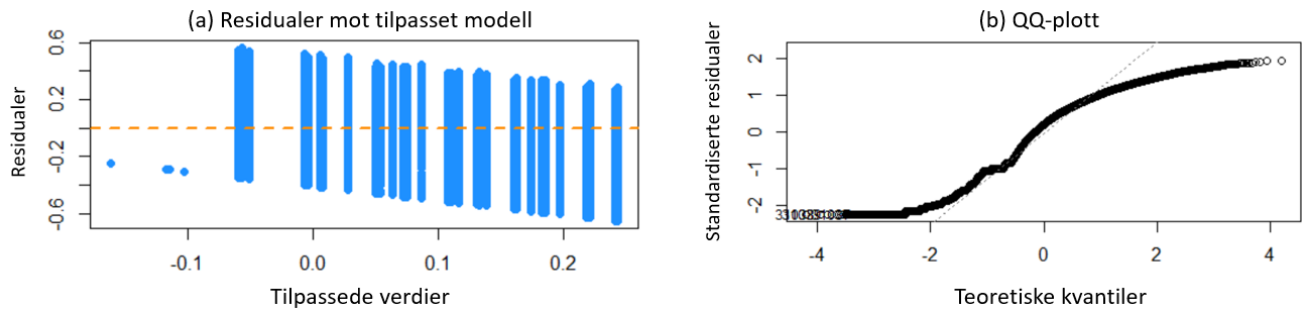
Figur C.1. Her vises residualer mot tilpasset modell (a) og QQ-plott (b) for datasettet med Dice-score per tverrsnitt for alle modellene etter kvadratrot-transformeringen.

Invers-transformering



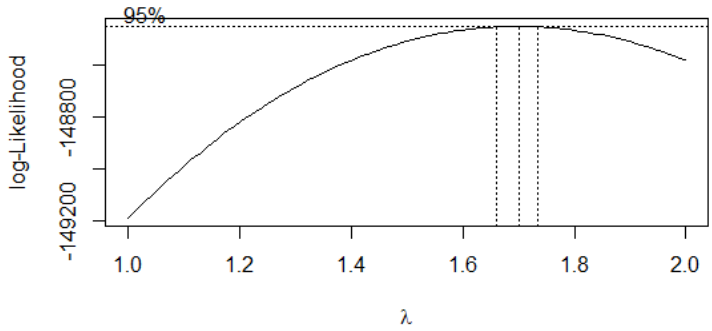
Figur C.2. Her vises residualer mot tilpasset modell (a) og QQ-plott (b) for datasettet med Dice-score per tverrsnitt for alle modellene etter invers-transformeringen.

Boxcox-transformering



Figur C.3. Her vises residualer mot tilpasset modell (a) og QQ-plott (b) for datasettet med Dice-score per tverrsnitt for alle modellene etter boxcox-transformeringen.

Under vises Log-likelihood-plottet som ble brukt for å velge lambda-verdi for Boxcox-transformeringen. Lambda ble satt til x-verdien 1,7, som var det høyeste punktet på grafen.



Figur C.4. Her vises et Log-likelihood-plott som brukes for å velge lambda-verdi til boxcox-transformeringen. Verdien bør velges basert på toppunktet på kurven.

Friedmantest og Wilcoxon signed rank-test

```

#Laster inn data for hver parameter
data_nfilters <- mean_nfilters
data_learningrate <- mean_learningrate
data_batchnorm <- mean_batchnorm
data_dropout <- mean_dropout

#Laster inn pakker
library(PMCMRplus)
library(ggpubr)
library(rstatix)

# Antall filtre ----------------------------------------------------------------

y_nfilter <- data_nfilters$f1_score
block_nfilter <- 1:1033
blocks_nfilter <- rep(block_nfilter,times=3)
groups_nfilter <- factor(c(rep("32", times=1033), rep("48", times=1033),
                           rep("64", times=1033)))

# Friedmantest
friedman.t.chidist <- friedmanTest(y_nfilter, groups_nfilter, blocks_nfilter)
friedman.t.chidist[["p.value"]]

#Post-hoc med Nemenyi
nemenyi.all.comp <- frdAllPairsNemenyiTest(y_nfilter, groups_nfilter,
                                           blocks_nfilter,
                                           alternative = "two.sided")
nemenyi.all.comp

```

```

#Boksplott
ggboxplot(data_nfilters, x = "n_filters", y = "f1_score", add = "jitter")

#Effektstørrelse
data_nfilters %>% friedman_effsize(f1_score ~ n_filters |Teller)

# Læringsraten -----------------------------------------------------------------

y_learningrate <- data_learningrate$f1_score
block_learningrate <- 1:1033
blocks_learningrate <- rep(block_learningrate,times=3)
groups_learningrate <- factor(c(rep("1,00E-03", times=1033),
                                rep("1,00E-04", times=1033),
                                rep("1,00E-05", times=1033)))

# Friedmantest
friedman.t.chidist <- friedmanTest(y_learningrate, groups_learningrate,
                                   blocks_learningrate)
friedman.t.chidist[["p.value"]]

#Post-hoc med Nemenyi
nemenyi.all.comp <- frdAllPairsNemenyiTest(y_learningrate, groups_learningrate,
                                           blocks_learningrate,
                                           alternative = "two.sided")
nemenyi.all.comp

#Boksplott
ggboxplot(data_learningrate, x = "Learning_rate", y = "f1_score",
          add = "jitter")

```

```r
#Effektstørrelse
data_learningrate %>% friedman_effsize(f1_score ~ Learning_rate |Teller)

# Batchnormaliserig ------------------------------------------------------------

#Visualsering med boksplott
bxp <- ggpaired(data_batchnorm, x = "BatchNorm", y = "f1_score",
                order = c("Yes", "No"),
                ylab = "f1_score", xlab = "BatchNorm")
bxp

#Wilcoxon signed rank test
stat.test <- data_batchnorm  %>%
  wilcox_test(f1_score ~ BatchNorm, paired = TRUE) %>%
  add_significance()
stat.test

#Effektstørrelse
data_batchnorm  %>%
  wilcox_effsize(f1_score ~ BatchNorm, paired = TRUE)

# Utelatelsesrate --------------------------------------------------------------

#Visualisering med boksplott
bxp <- ggpaired(data_dropout, x = "Dropout_rate", y = "f1_score",
                order = c("0", "0,5"),
                ylab = "f1_score", xlab = "Dropout_rate")
bxp

```

```r
#Wilcoxon signed rank test
stat.test <- data_dropout  %>%
  wilcox_test(f1_score ~ Dropout_rate, paired = TRUE) %>%
  add_significance()
stat.test

#Effektstørrelse
data_dropout  %>%
  wilcox_effsize(f1_score ~ Dropout_rate, paired = TRUE)

```

Vedlegg D

Vedlegg D inneholder oversikter over gjennomsnittlig Dice-score, HD_{95} og MSD per pasient fra den beste modellen med og uten augmentering, som ble kjørt på OUS-testsettet og Maastro-testsettet.

Tabell D.1. Resultatene fra den beste modellen med og uten augmentering, kjørt på OUS-testsettet.

	Uten augmentering			Med augmentering		
Pasientnummer	Dice-score	HD_{95} [mm]	MSD [mm]	Dice-score	HD_{95} [mm]	MSD [mm]
5	0,729	32,3	2,24	0,789	38,3	1,00
8	0,772	19,6	1,00	0,755	23,3	1,00
13	0,777	15,0	0,000	0,775	22,9	1,00
16	0,412	20,7	4,36	0,400	22,4	4,90
18	0,825	4,00	0,000	0,817	3,00	0,000
21	0,626	21,2	1,00	0,596	37,0	1,00
36	0,800	2,24	0,000	0,788	19,0	0,000
44	0,797	3,61	0,000	0,790	3,61	0,000
52	0,858	2,24	0,000	0,824	5,39	1,00
55	0,763	13,9	1,00	0,793	4,24	0,000
60	0,785	18,6	1,00	0,760	21,6	1,00
61	0,744	3,00	0,000	0,735	5,00	0,000
67	0,721	49,5	1,00	0,739	35,9	1,00
73	0,715	3,16	1,00	0,681	15,0	1,41
74	0,768	11,6	1,00	0,815	4,58	0,000
77	0,869	3,00	0,000	0,851	3,16	1,00
82	0,732	31,4	1,41	0,635	13,9	2,00
91	0,840	12,7	0,000	0,840	6,08	0,000
93	0,305	25,1	3,74	0,443	23,5	2,24
99	0,887	2,24	0,000	0,892	3,61	0,000
110	0,0	43,9	24,0	0,0	42,8	26,2
116	0,694	21,5	1,00	0,688	57,0	1,41
120	0,848	40,9	0,000	0,832	37,7	1,00
130	0,724	3,61	0,000	0,705	4,00	0,000
140	0,749	39,4	1,00	0,779	18,3	1,00
148	0,750	18,4	1,00	0,821	17,5	0,000
153	0,718	4,36	0,000	0,697	4,24	0,000
154	0,767	3,00	0,000	0,757	3,61	0,000
162	0,808	12,2	1,00	0,813	40,3	1,00
164	0,742	7,62	1,00	0,749	47,6	1,41
169	0,708	53,8	2,00	0,784	48,9	1,00
184	0,880	10,9	0,000	0,882	35,6	0,000
191	0,821	4,00	0,000	0,793	5,83	1,00
194	0,660	17,9	1,00	0,756	11,4	1,00
209	0,826	4,12	1,00	0,826	10,0	1,00
217	0,819	4,12	0,000	0,820	4,00	0,000
223	0,638	30,9	0,000	0,740	5,45	0,000

233	0,889	9,22	0,000	0,875	13,9	0,000
242	0,775	44,7	0,000	0,815	3,74	0,000
249	0,805	28,0	1,00	0,822	34,8	1,00

Tabell D.2. Resultatene fra den beste modellen med og uten augmentering, kjørt på Maastro-testsettet.

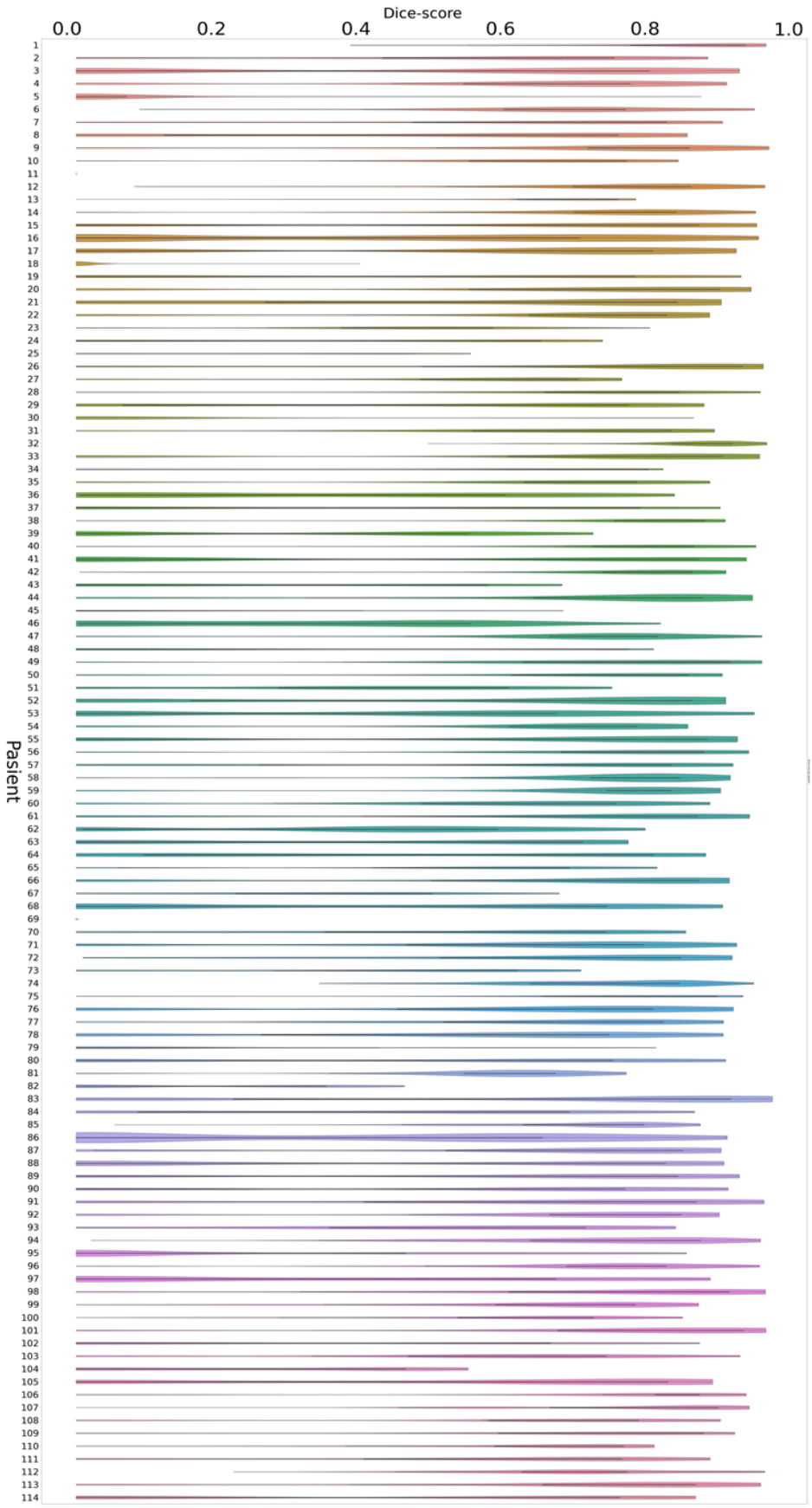
	Uten augmentering			Med augmentering		
Pasientnummer	Dice-score	HD_{95} [mm]	MSD [mm]	Dice-score	HD_{95} [mm]	MSD [mm]
1	0,873	3,32	0,000	0,836	15,9	1,00
2	0,603	24,6	1,41	0,618	42,2	2,00
3	0,767	17,0	0,000	0,710	31,8	1,00
4	0,722	12,1	0,000	0,652	18,4	1,00
5	0,071	52,6	4,58	0,001	57,5	26,6
6	0,699	38,0	2,00	0,759	19,6	1,00
7	0,732	3,32	0,000	0,659	70,7	1,00
8	0,577	37,3	2,83	0,684	26,7	1,00
9	0,807	4,24	1,00	0,785	16,6	1,00
10	0,673	20,2	1,00	0,714	4,69	1,00
11	0,000	14,0	11,0	0,000	24,0	16,1
12	0,783	18,1	1,00	0,791	15,7	1,00
13	0,685	8,72	1,00	0,686	3,00	1,00
14	0,772	5,00	0,000	0,745	5,00	1,00
15	0,796	2,24	0,000	0,811	2,00	0,000
16	0,532	13,2	2,83	0,560	17,7	3,00
17	0,721	9,38	1,00	0,765	6,32	1,00
18	0,006	66,2	58,8	0,007	59,5	40,3
19	0,653	24,7	0,000	0,586	41,3	1,00
20	0,782	24,9	1,00	0,773	32,5	1,41
21	0,717	7,35	1,00	0,705	6,08	1,00
22	0,737	45,1	1,00	0,709	45,9	1,00
23	0,494	4,12	1,00	0,468	9,00	1,00
24	0,540	9,00	1,00	0,486	44,91	2,24
25	0,360	25,2	1,00	0,315	8,35	0,000
26	0,863	5,00	1,00	0,862	9,90	1,00
27	0,607	7,00	1,00	0,642	7,81	1,00
28	0,732	31,2	1,00	0,818	20,6	0,000
29	0,665	14,0	1,00	0,701	15,0	1,00
30	0,170	65,5	8,12	0,127	30,0	19,0
31	0,721	15,8	1,00	0,782	33,5	1,00
32	0,878	16,4	0,000	0,882	20,0	0,000
33	0,840	14,0	0,000	0,857	3,00	0,000
34	0,708	13,4	1,00	0,720	26,2	0,000
35	0,735	3,16	0,000	0,760	3,16	1,00
36	0,479	43,4	2,24	0,255	43,7	1,00
37	0,538	4,5	0,000	0,474	86,7	4,12

38	0,806	4,24	1,00	0,803	18,1	1,00
39	0,409	23,8	2,45	0,533	5,00	1,00
40	0,802	2,83	0,000	0,804	2,83	0,000
41	0,552	6,26	0,000	0,594	3,61	0,000
42	0,765	7,07	1,00	0,727	15,0	1,00
43	0,380	37,3	1,73	0,333	32,6	1,00
44	0,781	42,0	1,00	0,747	18,5	1,00
45	0,292	1,73	0,000	0,276	35,5	0,000
46	0,366	13,2	5,10	0,390	20,0	5,74
47	0,731	8,94	1,41	0,752	8,60	1,00
48	0,513	28,7	1,00	0,359	27,6	2,00
49	0,821	18,4	1,00	0,832	7,81	0,000
50	0,751	37,9	1,00	0,669	40,3	1,00
51	0,492	50,4	3,00	0,528	53,2	2,45
52	0,803	6,40	1,00	0,810	5,10	1,00
53	0,611	17,1	1,00	0,648	40,0	1,00
54	0,713	13,3	1,00	0,695	18,2	1,00
55	0,811	23,6	1,00	0,835	17,1	1,00
56	0,798	26,2	1,00	0,829	12,8	0,000
57	0,724	5,83	0,000	0,742	15,8	0,000
58	0,775	4,24	1,00	0,746	5,00	1,00
59	0,798	4,12	0,000	0,806	3,74	0,000
60	0,687	6,71	1,00	0,740	5,00	0,000
61	0,761	24,2	1,00	0,729	26,6	1,00
62	0,479	15,1	1,41	0,537	14,0	1,00
63	0,523	36,9	5,00	0,369	37,8	6,78
64	0,520	18,4	2,00	0,630	17,3	1,00
65	0,587	22,2	1,00	0,650	2,83	0,000
66	0,825	4,90	1,00	0,826	5,00	1,00
67	0,369	13,3	3,00	0,403	18,5	4,90
68	0,533	53,1	2,24	0,548	18,0	1,00
69	0,000	56,1	25,2	0,000	61,2	27,7
70	0,551	20,2	3,61	0,573	22,0	2,24
71	0,752	4,47	0,000	0,739	6,40	1,00
72	0,762	23,0	1,00	0,763	32,9	1,00
73	0,497	4,12	1,00	0,318	25,8	1,41
74	0,745	16,2	1,00	0,767	26,2	1,00
75	0,796	2,00	0,000	0,781	2,24	0,000
76	0,729	27,5	1,00	0,743	7,35	1,00
77	0,716	16,2	1,00	0,728	13,2	1,00
78	0,668	5,39	1,00	0,646	4,12	1,00
79	0,255	15,9	0,000	0,334	5,74	0,000
80	0,676	39,9	1,00	0,750	38,9	1,00
81	0,604	34,3	1,73	0,580	34,5	2,00
82	0,182	31,8	19,0	0,185	15,8	1,00
83	0,832	47,5	1,00	0,855	45,0	1,00
84	0,561	14,0	1,00	0,655	5,20	0,000

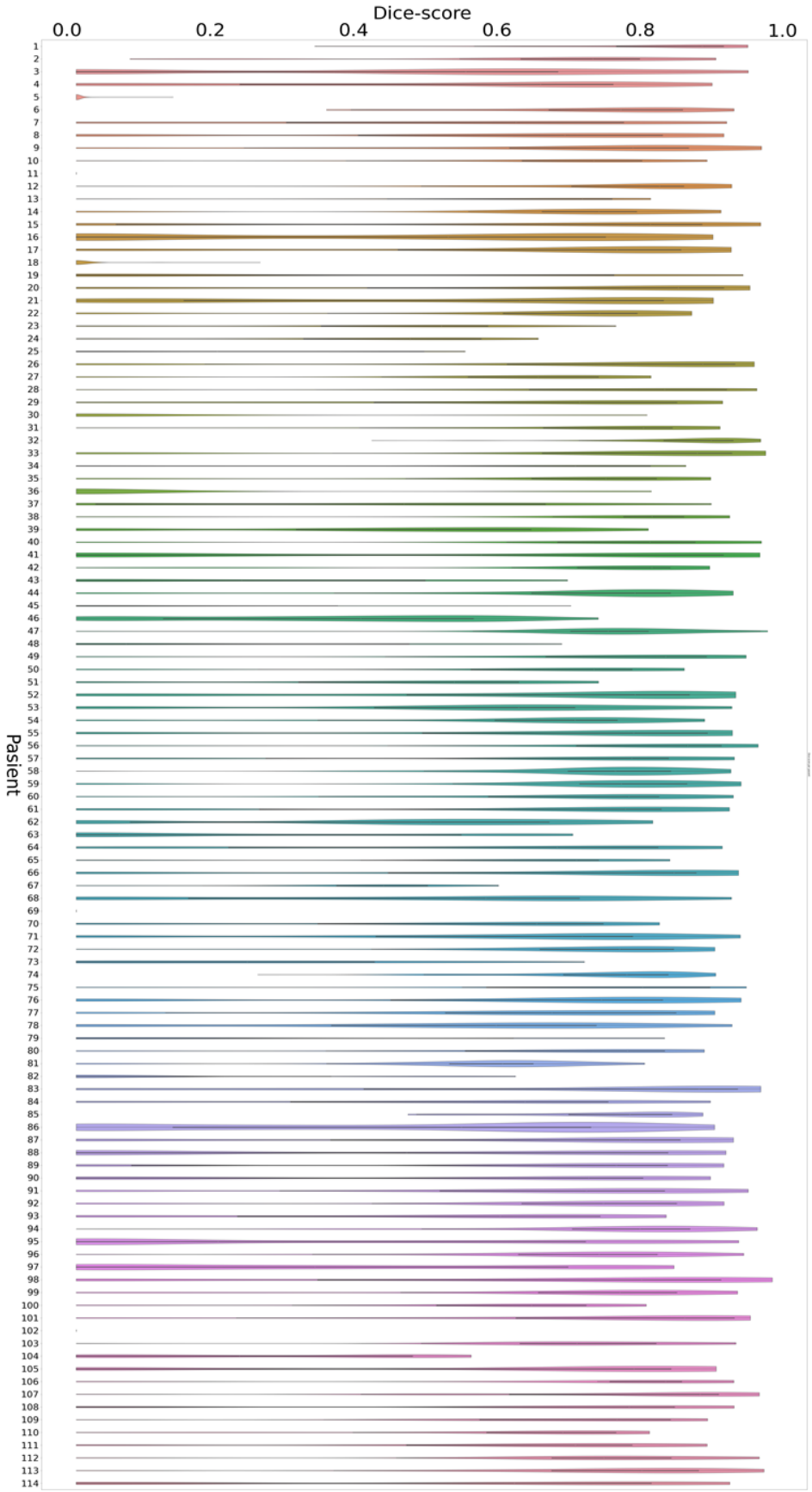
85	0,730	10,1	1,41	0,771	5,74	1,00
86	0,490	12,7	2,24	0,580	9,90	1,41
87	0,731	97,8	1,41	0,780	91,6	1,00
88	0,700	43,4	1,00	0,701	31,6	0,000
89	0,779	15,0	1,00	0,789	4,47	0,000
90	0,684	4,24	1,00	0,687	16,3	1,00
91	0,767	4,00	0,000	0,724	21,8	1,00
92	0,768	7,00	0,000	0,766	7,87	0,000
93	0,645	26,3	1,00	0,657	26,6	1,00
94	0,727	20,0	1,00	0,747	15,8	1,00
95	0,387	19,0	1,41	0,464	19,0	1,00
96	0,747	55,0	1,00	0,717	57,5	1,41
97	0,582	9,27	1,00	0,538	8,77	1,00
98	0,819	4,12	0,000	0,799	14,2	1,00
99	0,703	6,16	1,00	0,760	5,39	1,00
100	0,637	52,5	1,41	0,623	5,39	1,00
101	0,841	4,12	0,000	0,818	10,5	1,00
102	0,383	3,61	1,00	0,000	100	98,6
103	0,660	25,6	1,00	0,703	50,6	0,000
104	0,362	9,00	5,10	0,356	9,00	5,00
105	0,754	6,00	1,00	0,793	17,7	1,00
106	0,820	2,83	0,000	0,793	3,16	0,000
107	0,854	10,3	1,00	0,864	12,8	0,000
108	0,715	12,4	0,000	0,764	3,16	0,000
109	0,779	3,61	0,000	0,769	9,70	0,000
110	0,706	5,10	1,00	0,700	5,66	1,00
111	0,677	4,58	1,00	0,696	4,47	1,00
112	0,686	26,2	1,00	0,730	40,1	1,00
113	0,789	16,0	1,00	0,808	14,8	1,00
114	0,618	5,48	0,000	0,663	37,1	0,000

Fiolinplott

Figur D.1 og Figur D.2 viser henholdsvis Dice-score per tverrsnitt for hver pasient i Maastro-testsettet for modellen uten og med augmentering.



Figur D.1. Fiolinplottet viser Dice-score per tverrsnitt for alle pasientene i Maastro-datasettet, for modellen uten augmentering. Bredden representerer antallet tverrsnitt med den gitte Dice-scoreverdien, og lengden representerer variansen til Dice-score per tverrsnitt for pasienten.



Figur D.2. Fiolinplottet viser Dice-score per tverrsnitt for alle pasientene i Maastro-datasettet, for modellen med augmentering. Bredden representerer antallet tverrsnitt med den gitte Dice-scoreverdien, og lengden representerer variansen til Dice-score per tverrsnitt for pasienten.

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

Postboks 5003
NO-1432 Ås
Norway