Norges miljø- og biovitenskapelige universitet

Masteroppgave 2021 30 stp
Fakultet for realfag og teknologi

Effekt av dataaugmentering på dyp læring-basert segmentering av hode- og halskreft i PET/CT-bilder

Effect of data augmentation on deep learning-based segmentation of head and neck cancers in PET/CT images

Maria Ødegaard
Miljøfysikk og fornybar energi

Forord

Denne masteroppgaven markerer avslutningen på en femårig mastergrad i Miljøfysikk og fornybar energi ved Norges miljø- og biovitenskapelige universitetet (NMBU) og er skrevet ved fakultet for realfag og teknologi våren 2021.

Jeg vil gjerne starte med å rette en stor takk til min hovedveileder Professor Cecilia Marie Futsæther for grundige tilbakemeldinger, oppløftende ord og god oppfølging gjennom hele denne prosessen. En stor takk må også rettes til Ph.d-stipendiat Bao Ngoc Huynh som har laget rammeverket *deoxys* benyttet i denne masteroppgaven. Uten hennes veiledning, råd og hjelp underveis, hadde denne masteroppgaven vært vanskelig å fullføre. Jeg vil også gjerne få takket Ph.d-stipendiat Aurora Rosvoll Grøndahl for gode råd og svar på spørsmål gjennom denne perioden. Takk til Førsteamanuensis Oliver Tomic og gruppen til Professor Kathrine Røe Redalen ved NTNU for interessante diskusjoner og veiledning på møter. Professor Eirik Malinen ved universitetet i Oslo fortjener også en takk for å ha anskaffet datasettet med PET/CT-bilder benyttet i masteroppgaven.

En spesiell takk må rettes til mine medstudenter Malene Elise Gjengedal og Sofie Roko Krogstie som jeg har arbeidet tett med gjennom denne perioden. Deres råd, hjelp og samarbeid har vært uvurderlig. En stor takk må også rettes til min familie, kjæreste og mine venner som har støttet meg under arbeidet med denne masteroppgaven og hjulpet meg med rettskriving. En ekstra takk går til mine samboere i kollektivet som jeg har fått æren av å bo med i fem fine år. Deres støtte, kjærlighet og positivitet har gjort denne studietiden uforglemmelig.

Helt til slutt vil jeg gjerne takke alle mine medstudenter for fem fantastiske år på NMBU.

Ås, 01.06.2021

_____

Maria Ødegaard

Sammendrag

Formål

Inntegning av tumorvolum i hode- og halsområdet er en tidkrevende utfordring, utsatt for intra- og intervariabiliteter, samt feilinntegninger, som kan gi store konsekvenser for behandlingsutfallet til den enkelte pasient. Ved bruk av dyp læring kan den manuelle inntegningsprosessen automatiseres og dermed brukes som et hjelpemiddel innenfor medisinsk behandling. Målet med denne oppgaven var å forbedre segmenteringsytelsen til et konvolusjonsnettverk med 2D U-Net-arkitektur ved å benytte dataaugmenteringsteknikker på inputbilder. Mangel på tilstrekkelig treningsdata er en av de vanligste utfordringene innenfor maskinlæring og kan føre til overtilpassede modeller. Dette gjelder spesielt innen medisin, der det både er dyrt og vanskelig å anskaffe bilder nettverket kan trene på. Ved å endre inputbildene på en rekke tilfeldige måter, blir modellen introdusert for bilder med større variasjoner, som kan gjøre modellen mer robust og generell, og dermed øke segmenteringsytelsen til 2D U-Net-arkitekturen.

Metode

Rammeverket *deoxys*, utviklet ved fakultet for realfag og teknologi, NMBU, ble benyttet for å lage 65 eksperimentmodeller med 2D U-Net-arkitektur som ble trent på treningsdatasett påført ulike kombinasjoner av augmenteringsteknikker. Datasettet bestod av PET/CT-bilder av hode- og halskreftpasienter fra Oslo universitetssykehus (OUS) og ble delt i treningssett (142 pasienter), valideringssett (40 pasienter) og testsett (15 pasienter). Datasettet ble delt slik at settene inneholdt et representativt utvalg av pasienter med forskjellig tumorstadium, der hvert tverrsnitt inneholdt sann inntegning av tumorvolum og påvirkede lymfeknuter utført av spesialister. Overlappsmålet Dice-score ble benyttet for å evaluere modellenes ytelse på valideringssettet.

48 eksperimentmodeller brukte de affine transformasjonsteknikkene rotasjon, zoom, forskyvning og flipp, kalt eksperimentplan 1. 16 eksperimenter brukte punkt- og filteroperasjonene lysstyrke, kontrast, støy og uskarphet, kalt eksperimentplan 2. Videre ble et kombinasjonseksperiment utført, som bestod av teknikkene som oppnådde høyest ytelse per tverrsnitt på valideringssettet i eksperimentplan 1 og eksperimentplan 2. De statistiske testene Friedmantest etterfulgt av Nemenyi post-hoc test og Wilcoxon signed rank-test ble benyttet i sammenheng med Dice-score for å finne eksperimentmodellen med kombinasjonen av augmenteringsteknikker som oppnådde høyest segmenteringsytelse i forhold til ikke-

augmenterte modeller. Denne modellen ble videre testet på OUS-testsettet og et eksternt testsett innhentet fra Maastro Clinic i Nederland for å evaluere modellens ytelse på usett data.

Resultat og diskusjon

Affine transformasjoner hadde signifikant innvirkning på segmenteringsytelsen, i motsetning til punkt- og filteroperasjonene som i noen tilfeller hadde negativ effekt på modellens ytelse. Augmenteringsteknikken med den mest markante innvirkningen var den affine transformasjonen flipp. Videre tydet de statistiske testene på at den sterkeste augmenteringen oppnådde de høyeste Dice-scorene. Modellen med tilfeldig rotasjon mellom -90° og 90°, zoomfaktor mellom 0,5 og 1,5, et antall pikselforskyvning mellom -10 og 10 i *x*- og *y*-retning og flipping om *x*-aksen, oppnådde høyest ytelse på valideringssettet av alle eksperimentmodellene og ble derfor videre evaluert på testsettene. Denne modellen oppnådde den gjennomsnittlige Dice-scoren 0,75 ± 0,16 per pasient på OUS-testsettet og 0,65 ± 0,19 på det eksterne Maastro-settet.

Observasjoner fra visualiserte inntegninger viste at modellen hadde vanskeligheter for å tegne inn små tumorvolum og områder der den sanne inntegningen inneholdt piksler med lavt FDG-opptak eller atypiske Hounsfield-verdier. I tillegg inkluderte modellen falske positive inntegninger i typiske lyse områder som ikke var tegnet inn av spesialistene.

Konklusjon

Augmentering økte modellytelsen og gav inntegninger med høy overlapp med sanne inntegninger, i forhold til ikke-augmenterte modeller. Spesielt kraftige affine transformasjoner økte segmenteringsytelsen til modellen. Til tross for lovende resultater utførte modellen feilaktige inntegninger som kan få store konsekvenser for pasientens behandlingsutfall. Den augmenterte modellen må derfor videre forbedres for å kunne brukes som et assisterende hjelpemiddel i klinisk sammenheng. Videre arbeid bør undersøke post-prosessering, innhenting av treningsdata fra forskjellige institusjoner, samt undersøke andre augmenteringsteknikker for å kunne ytterligere forbedre modellens segmenteringsytelse.

Abstract

Purpose

Tumour delineation in the head and neck region is a time-consuming challenge, prone to intra- and interobserver variability as well as erroneous delineations which have large consequences for the treatment outcome for each individual patient. By using deep learning, the manual delineation can be automated and thereby be used as a tool in medical treatment. The goal of this thesis was to improve the segmentation performance of a convolutional neural network with 2D U-Net architecture by using data augmentation techniques on the input data. The lack of sufficient training data is one of the most common challenges within the field of machine learning, which can cause overfitted models, especially in the medical field, where it is both expensive and demanding to procure images for network training. By distorting the input images in various ways, the model will be introduced to images with larger variations. This can make the model more robust and general, thereby increasing the delineation performance of the 2D U-Net.

Method

The framework *deoxys,* developed at the Faculty of Science and Technology, NMBU, was used to generate 65 experiment models with the 2D U-Net architecture, which were trained on a dataset containing images augmented with different combinations of augmentation techniques. The dataset contained PET/CT images of head and neck cancer patients from the Oslo University Hospital (OUS) and was split into a training set (142 patients), a validation set (40 patients) and a test set (15 patients). The dataset was stratified by tumour stage where each image slice contained a true segmentation of the tumour volume and affected lymph nodes, provided by specialists. The performance metric Dice-score was used to evaluate the model's performance on the validation set.

48 experiment models used the affine transformations rotation, zoom, shift and flip, called experimental plan 1. 16 experiments used the points and filter operations brightness, contrast, noise, and blur, called experimental plan 2. Furthermore, a combination experiment was conducted that used the techniques within each experimental plan which achieved the highest performance per slice on the validation set. The statistical test Friedman test followed by a Nemenyi post-hoc test, and Wilcoxon signed-rank test was used in combination with the Dice-score to find the experiment model with the augmentation techniques that obtained the highest segmentation performance relative to models without augmentation. This model was

further tested on the OUS test set and on an external test set retrieved from Maastro Clinic in the Netherlands for evaluating the model performance on unseen data.

Results and discussion

Affine transformations were shown to have a significant effect on the segmentation performance, unlike the points and filter operations which in some cases had a negative influence on the model performance. The augmentation technique which had the clearest effect was the affine transformation flip. Furthermore, the statistical test showed that the most substantial augmentation achieved the highest Dice-scores. The model, consisting of random rotation between -90° and 90°, a zoom factor between 0.5 and 1.5, a shift between -10 and 10 pixels along each axis and vertical flip, obtained the highest performance among all experiment models, and was therefore further evaluated on the test sets. This model achieved a mean Dice-score 0.75 ± 0.16 per patient on the OUS test set and 0.65 ± 0.19 on the external Maastro set.

Observations from visualised delineations show that the model struggled to delineate small tumour volumes or regions where the true segmentations contained pixels with a low FDG-uptake or atypical Hounsfield-values. In addition, the model included false positive delineations in bright areas not delineated by specialists.

Conclusion

Augmentation was found to increase model performance, providing auto-delineations with higher overlap with the ground truth relative to models without augmentation. Particularly affine transformations with a substantial augmentation increased the segmentation performance of the 2D U-Net model. Despite promising segmentation results, the augmented models included erroneous delineations, which have large consequences for the patient treatment outcome. The models must therefore be further improved if they are to be used as an assistance tool in clinical use. For future research and development, post-processing should be explored, more image data from different institutions should be added to the training set and other augmentation techniques should be investigated to further increase the model's segmentation performance.

Innholdsfortegnelse

Liste over forkortelser

Adam:	Adaptive moment estimation
AI:	Artificial Intelligence
ANOVA:	Analysis of Variance
CIGENE:	Center of Interactive Genetics
CNN:	Convolutional Neural Network / Konvolusjonsnettverk
CPU:	Central Processing Unit
CT:	Computertomografi
DBMS:	Database Management System
FCN:	Fully Convolutional Network / Fullt konvolusjonsnettverk
FDG:	Fluorodeoksyglukose
FN:	False Negative / Falsk negativ
FP:	False Positive / Falsk positiv
FPR:	False Positive Rate / Falsk positiv rate
GANs:	Generative Adversarial Networks
GPU:	Graphics Processing Unit
GTV:	Gross Tumor Volume
GTV-N:	Gross Tumor Volume Node
GTV-T:	Gross Tumor Volume Tumor
HD_{95}:	95. persentil Hausdorff avstand
HDF5:	Hierarchical Data Format version 5
HECKTOR:	HEad and neCK TumOR
HU:	Hounsfield Unit
JSON:	JavaScript Object Notation
KI:	Kunstig intelligens
LOR:	Line of Response
MICCAI:	Medical Image Computing and Computer Assisted Intervention
MR:	Magnetisk resonans
MSD:	Median Surface Distance / Median overflateavstand

MSE:	Mean Squared Error / Midlere kvadratisk feil
OUS:	Oslo universitetssykehus
PET:	Positronemisjonstomografi
PRE:	Precision / Presisjon
QQ-plott:	Quantile-Quantile plot
RAM:	Random Access Memory
ReLU:	Rectified Linear Unit
RGB:	Rød, Grønn og Blå
SGD:	Stocastic gradient descent / Stokastisk gradient-nedstigning
SSH:	Secure Shell
SUV:	Standardized Uptake Value
Tanh:	Tangens hyperbolikus
TN:	True Negative / Sann negativ
TNM:	Tumor, node, metastase
TP:	True Positive / Sann positiv
TPR:	True Positive Rate / Sann positiv rate
WL:	Window Level / vinduslevel
WW:	Window Width / vindusbredde

Kapittel 1: Introduksjon

1.1 Motivasjon

1.1.1 Hode- og halskreft

Kreft er en fellesbetegnelse på en dødelig sykdom som i 2020 tok livet av nærmere 10 millioner mennesker på verdensbasis [1]. Det finnes over 200 ulike kreftformer som alle omhandler ukontrollert celledeling grunnet mutasjoner i cellens arvestoff [2]. Den ukontrollerte celledelingen fører til akkumulering av kreftceller som danner kreftsvulster, tumorer, i regionen hvor mutasjonen oppstod. Løsrivning av kreftceller som fraktes via blodårer og lymfebaner kan videre resultere i spredning, metastase, til andre deler av kroppen [2]. I 2019 ble nærmere 35 000 [3] mennesker diagnostiert med kreft i Norge, hvor 648 av tilfellene var av typen kreft i hode- og halsregionen [4].

Hode- og halskreft utgjør ondartede tumorer lokalisert i områdene nese og bihuler, leppe, munnhule, svelg, strupehode eller spyttkjertler [5]. De vanligste behandlingsmetodene for hode- og halskreft er kirurgi, cellegift og radioterapi i kombinasjon eller alene [5]. Valg av behandlingsmetode avhenger av nøyaktig diagnose som ofte stilles ved bruk av de medisinske avbildningsteknikkene PET og CT [2]. En PET-skanning kan avsløre områder med forhøyet metabolsk aktivitet som ofte kjennetegner kreftsvulster, mens en CT-skanning kan avsløre posisjonen til den forhøyede aktiviteten i kroppen [6]. I dag kombineres nesten alle PET- og CT-skannere, som gjør at en fullkroppsundersøkelse av både anatomien og den forhøyede metabolske aktiviteten raskt og enkelt kan utføres [7]. Størrelsen eller beliggenheten til tumorvolumet i hode- og halsregionen gjør i mange tilfeller operasjon vanskelig [4]. Radioterapi er derfor mye brukt, og er en av de mest effektive behandlingsformene som gir bedre funksjonelle utfall sammenlignet med andre metoder [8].

Radioterapi, også kalt strålebehandling, er en effektiv behandlingsmetode som bruker høyenergetisk stråling for å drepe kreftceller og dermed hindre videre ukontrollert celledeling [9]. Radioterapi virker på området som bestråles, og effekten øker med økt dose [9]. Målet er å gi tilstrekkelig dose til kreftsvulstvolumet og samtidig begrense dosen til nærliggende vev og organer for å forhindre skade [10, 11]. Den komplekse anatomien i hode- og halsområdet, hvor tumorvolumet ofte befinner seg nærme andre organer, gjør strålebehandling utfordrende [12]. For å unngå skadelig strålebehandling på friskt vev er det derfor viktig med presis og nøyaktig kreftsvulstinntegning.

1.1.2 Utfordringer

Nåværende gullstandard av kreftsvulstinntegning innenfor klinisk praksis er manuell inntegning utført av medisinske spesialister [11]. Til tross for at en PET/CT-skanning kan forsterke tumor-visualisering, er inntegning av tumorvolum i hode- og halsområdet en tidkrevende utfordring, utsatt for feilinntegninger, spesielt når det gjelder store og irregulære tumorer som er omgitt av kompleks og kritisk anatomi [8, 13]. Unøyaktighet eller feilinntegninger kan føre til skader på friskt vev og i noen tilfeller underdosering av tumorvolum som kan gjøre at kreften ikke forsvinner og pasienten ikke kan erklæres frisk [8].

Selv med eksisterende globale veiledningsguider for inntegning av hode- og halskrefttumorer, er det mye usikkerhet tilknyttet den individuelle variabiliteten av den unike tumor og anatomi for hver enkelt pasient [13]. Inntegning av tilsynelatende friskt vev uten diskrete tumorer, men som likevel utgjør en risiko for å være eller bli påvirkede områder, er i tillegg en utfordrende prosess. Det er ikke uvanlig med varierende inntegninger av slike områder blant de forskjellige CT-tverrsnittene til en og samme pasient hvor anatomien er tilnærmet identisk. Inntegningen kan variere blant spesialistene, intervariabiliteter, men og mellom pasienter som er behandlet av samme spesialist, intravariabiliteter [13]. Intervariabiliteter av inntegning på tvers av spesialister, samt intravariabiliteter hos spesialistene, er en faktor som i tidligere studier har vist seg å gi større unøyaktigheter i inntegning av tumorvolum enn unøyaktigheten tilknyttet pasientposisjon og organbevegelse [14].

En automatisering av kreftsvulstinntegningen vil derfor kunne være en stor fordel for både spesialistene og pasientene. Ved bruk av dyp læring kan den manuelle inntegningsprosessen automatiseres, og dermed brukes som et hjelpemiddel innenfor medisinsk behandling. Automatiseringen kan bidra til å redusere tid og usikkerhet tilknyttet den manuelle inntegningen, og dermed bidra til en forbedret behandlingsplan for den enkelte pasient [15].

1.2 Bruk av dyp læring

Dyp læring er en sentral del innenfor maskinlæring som ved bruk av selvlærende algoritmer henter ut nyttig informasjon fra presentert data og basert på dette trekker selvstendige slutninger [16]. Ved å gi datamaskinen evnen til å lære, kan oppgaver som vanligvis utføres av mennesker automatiseres og mulig forbedres [17]. Bruk av maskinlæring i helsesektoren har vist lovende resultater, hvor dyplæringsmodeller, sammen med mennesker eller alene, har utkonkurrert spesialister i både tumor klassifisering, identifisering, prognose estimering og evaluering av behandlingsforløp [18]. I flere studier har konvolusjonsnettverk med ulike

arkitekturer blitt tatt i bruk for å automatisere den tidkrevende og utfordrende inntegningsprosessen.

Studien til Lin et al. [15] undersøkte bruk av konvolusjonsnettverk med VoxResNet-arkitektur for automatisk inntegning av krefttumorvolum i neseområdet («nasopharyngeal cancer») på MR-bilder. Videre funn tyder på at den automatiske inntegningsmodellen benyttet som hjelpemiddel i klinisk praksis reduserte intraobservasjoner med 36 %, interobservasjoner med 55 % og inntegningstid med 39 % [15]. Studien til Guo et al. [8] oppnådde høy overlapp mellom predikert og sann inntegning ved bruk av konvolusjonsnettverk med Dense-Net-arkitektur for segmentering av både krefttumorvolum og store påvirkede lymfeknuter i PET, CT og PET/CT-bilder. I 2020 ble konkurransen HECKTOR (HEad and neCK TumOR) arrangert for automatisk segmentering av hode- og halskrefttumorvolum i FDG-PET/CT-bilder på den internasjonale konferansen *Medical Image Computing and Computer Assisted Intervention* (MICCAI) [19]. Ved bruk av blant annet ulike varianter av U-Net-arkitekturen oppnådde flere deltakere høy overlapp mellom predikert inntegning utført av modellen og sann inntegning utført av spesialister. I studien til Isensee et al. [20] ble en U-Net modell kalt nnU-Net utviklet, som automatisk konfigurerer preprosessering, arkitektur, trening og post-prosessering for hvilken som helst oppgave og datasett i det biomedisinske feltet.

Den populære U-Net-arkitekturen ble også benyttet i studiene til Moe et al. [21], Groendahl et al. [11] og masteroppgaven til Huynh [22]. Studiene undersøkte 2D U-Net-modellens segmenteringsytelse av kreftsvulster og påvirkede lymfeknuter i PET/CT-bilder av hode- og halskreftpasienter fra Oslo universitetssykehus, hvor de to førstnevnte studiene oppnådde segmenteringsoverlappsmål per pasient på henholdsvis 0,71 og 0,75. Til tross for lovende resultater, kan mangel på tilstrekkelig treningsdata være en begrensing for å oppnå høyest mulig segmenteringsytelse. Ved bruk av visualiseringsteknikker fant Huynh [22] ut at noen av egenskapene hentet ut av ulike filtre i samme lag hadde mange likheter. Dette kan komme av mangel på et stort og variert treningsdatasett som gjør at modellen ikke klarer å utnytte filtrene på best mulig måte [22].

Et dypt nevralt nettverk trenger en tilstrekkelig mengde treningsdata for å kunne klassifisere usett data riktig [17]. Komplekse modeller som trener på små datasett risikerer å bli overtilpasset, ved å lære for mange detaljer og støy fra treningsdataene, som dermed gjør modellen lite generell [17]. Et av de vanligste problemene innenfor maskinlæring er mangel på tilstrekkelig treningsdata av god kvalitet [23]. Store datasett er ofte en utfordring, spesielt innenfor medisin, hvor det både kan være dyrt og vanskelig å produsere nye

treningseksempler [24]. Personvern tilknyttet pasienter og deres tilhørende data gjør det i tillegg utfordrende å anskaffe bilder nettverket kan trene på [25]. En kjent metode som håndterer mangel på treningsdata, er dataaugmentering [23].

1.3 Dataaugmentering

Dataaugmentering er en samling av teknikker som skaper nye treningseksempler. Ved å endre allerede eksisterende treningsdata på en rekke tilfeldige måter kan nye troverdig treningsvarianter med nyttig informasjon skapes, som øker variasjonen og størrelsen på datasettet [17, 24]. Bruk av augmenteringsteknikker på inputdata har i flere tidligere studier vist å kunne forbedre ytelsen til dyplæringsmodeller [26-29]. Teknikkene brukes blant annet for å forhindre overtilpasning og for å håndtere ubalanserte klasser som ofte er en utfordring i medisinske datasett [30]. Ved å trene en dyplæringsmodell på augmenterte bilder, blir modellen eksponert for nye aspekter og varianter av inputdata, som gjør at modellen kan generalisere bedre, bli mer robust og øke ytelsen [17, 24].

1.4 Mål med oppgaven

Målet med denne masteroppgaven var å forbedre segmenteringen av tumorvolum og påvirkede lymfeknuter til 2D U-Net-modellen undersøkt i studiene til Moe et al. [21], Groendahl et al. [11] og Huynh [22] ved bruk av dataaugmenteringsteknikker på treningsdata. Datasettet benyttet er hentet fra Oslo universitetssykehus, Radiumhospitalet, og består av PET/CT-bilder til hode- og halskreftpasienter som hadde planlagt gjennomført radioterapibehandling i perioden fra januar 2007 til desember 2013. Totalt 65 eksperimentmodeller med et konvolusjonsnettverk med U-Net-arkitektur ble trent på treningsdatasett påført ulike kombinasjoner av tradisjonelle augmenteringsteknikknivåer. Formålet var å finne kombinasjonen av teknikker som optimaliserte 2D U-Net-modellens inntegning. Eksperimentmodellen som oppnådde høyest segmenteringsytelse på valideringssettet ble testet på testsett fra Oslo universitetssykehus og på testsett fra Maastro Clinic i Nederland for å evaluere modellens ytelse på usett data. Modellen ble videre sammenlignet med 2D U-Net-modellen undersøkt i studiene til Moe et al. [21], Groendahl et al. [11] og Huynh [22], hvor dataaugmenteringsteknikker ikke ble benyttet.

1.5 Organisering

Denne masteroppgaven starter med å gi en oversikt over det teoretiske grunnlaget bak avbildningsteknikkene CT og PET i Kapittel 2, og det teoretiske grunnlaget bak maskinlæring i Kapittel 3. Videre beskriver Kapittel 4 metodikken benyttet i oppgaven. Resultater blir

presentert i Kapittel 5, etterfulgt av en diskusjon i Kapittel 6. Oppgaven avslutter med konklusjon i Kapittel 7, referanser i Kapittel 8 og vedleggene Vedlegg A – Vedlegg E.

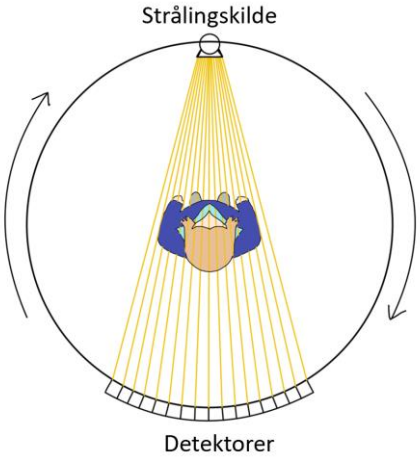
Kapittel 2: Medisinsk avbildning

2.1 Prinsipper innen medisinsk avbildning

Medisinsk avbildning er et samlebegrep som omhandler ulike teknikker brukt for å avbilde kroppens indre vev og organer. Disse teknikkene er både nyttige og nødvendige for å kunne gi rett diagnose og behandling til kreftpasienter [31]. Bruken av medisinske avbildningsteknikker startet allerede tidlig på 1900-tallet, kort tid etter at Wilhelm Röntgen oppdaget røntgenstrålingens evne til å gjengi skyggegrafer av kroppens indre [31]. Videre har teknikkene utviklet seg gjennom tidene. En revolusjon innen bildeteknologien kom da datamaskinen ble tatt i bruk for å lagre, prosessere og visualisere medisinske bilder [32]. Computertomografi (CT) var den første teknikken hvor matematiske operasjoner ble utført på innsamlet data for å produsere og manipulere bilder ved bruk av en datamaskin [32]. En annen nyttig avbildning- og diagnostiseringsteknikk som har utviklet seg gjennom tidene er positronemisjonstomografi (PET). PET-teknikken inkluderer blant annet bruken av radioisotoper og stråling for avbildning av kroppens metabolske aktivitet, som kan brukes for å detektere kreftregioner [33].

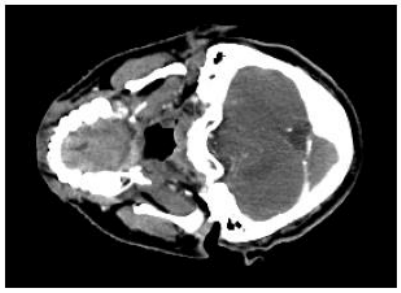
2.2 Computertomografi

Bildeteknikken computertomografi bruker røntgenstråling for å rekonstruere 2D-tverrsnitt av kroppens indre [31, 34]. I en standard CT-skanner plasseres pasienten med det ønskede undersøkelsesområdet mellom en strålingskilde og en ring av detektorer [32]. Strålingskilden emitterer energirike røntgenfotoner som passerer gjennom undersøkelsesområdet, vist i Figur 2.1 [34].



Figur 2.1: Illustrasjon av konseptet med en CT-skanner, hvor det ønskede undersøkelsesområdet plasseres mellom en strålingskilde og en ring av detektorer, som roterer rundt pasienten. Illustrasjonen er laget med inspirasjon fra figur i boken Introduction to Physics in Modern medicine [35].

Detektorringen måler antall fotoner transmittert gjennom pasienten, som indikerer absorpsjonen i de ulike delene av kroppen. CT-skanneren beveger seg sammen med detektoren og foretar rundt 160 målinger av røntgenstrålingen absorbert i punkter jevnt fordelt langs det skannede tverrsnittet. Slike målinger kalles projeksjoner [32]. Et bilde dannes basert på absorpsjonen i de ulike delene av kroppen. Områder med høy absorpsjon fremkommer hvite, og områder med lav absorpsjon fremkommer mørke på det rekonstruerte bildet [35]. Figur 2.2 viser et eksempel på et CT-bilde av et menneskehode.

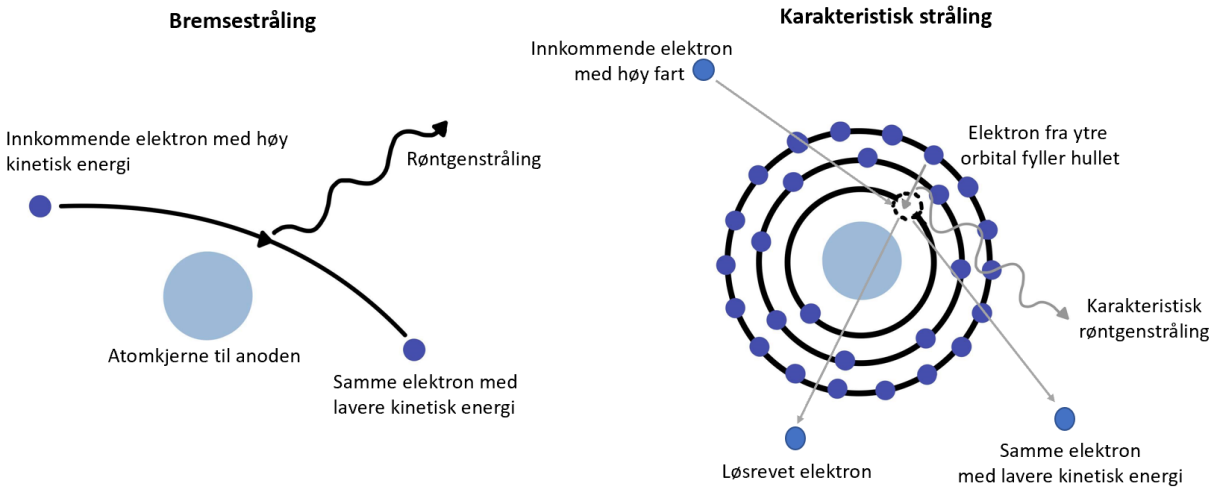


Figur 2.2: Figuren viser et CT-bilde av et menneskehode. Områder med høy absorpsjon fremkommer hvite, mens områder med lav absorpsjon fremkommer mørke.

Strålingskilden og detektorringen beveger seg en liten grad og gjentar prosessen helt til CT-systemet har skannet en vinkel på 180 grader. Bestråles det ønskede området med røntgenstråling fra forskjellige vinkler, klarer CT-maskinen å gjengi anatomisk informasjon ved digital rekonstruering basert på de innsamlede detektormålingene [32].

2.2.1 Røntgenstråling

Røntgenstråling er elektromagnetisk stråling med bølgelengde kortere enn 10 nm [35]. I CT-maskinen produseres røntgenstråling når akselererende elektroner emittert fra en katode kolliderer med atomene i en anode. Den dominerende produserte røntgenstrålingseffekten kalles bremsestråling [35]. Her bremses elektronene opp og endrer sin retning nær anoden grunnet tiltrekningskrefter, som resulterer i emittert røntgenstråling. I tillegg produseres det røntgenstråling ved karakteristisk stråling, hvor energirike elektroner løsriver et K-skall-elektron fra anoden. Det etterlatte hullet i orbitalen blir raskt fylt av et nærliggende elektron og det sendes ut elektromagnetisk stråling [36]. Figur 2.3 illustrerer eksempler på bremsestråling og karakteristisk stråling. Røntgenstråling emittert gjennom undersøkelsesområdet absorberes i varierende grad av de ulike vevstypene som kan beskrives ved attenuering [7, 35].



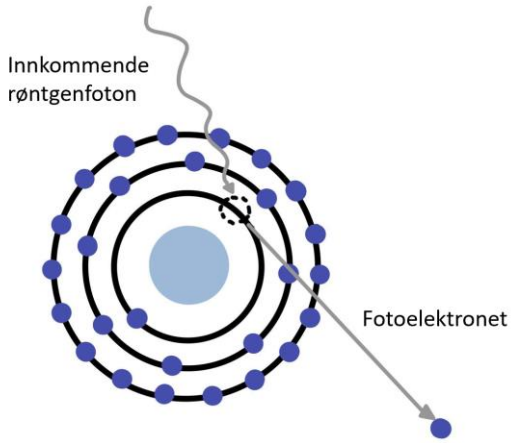
Figur 2.3: Figuren illustrerer fenomenet bremsestråling til venstre og karakteristisk stråling til høyre som begge gir opphav til røntgenstråling. Figuren er laget etter inspirasjon fra illustrasjon i boken Introduction to Physics in Modern Medicine [35].

2.2.2 Attenuering

Attenuering betegner hvor lett en stråle kan passere gjennom et materiale og gir derfor et mål på hvor mye strålen svekkes av det utsatte området [37]. Ved å måle attenueringskoeffisienten til vev som har blitt utsatt for røntgenstråling, kan tettheten til vevet beregnes og videre brukes av CT-maskinen for å rekonstruere et 2D-tversnitt [34]. Attenueringskoeffisienten vil variere fra ulike vev som gjør det mulig å danne et bilde av kroppens indre [7]. Variasjonene avhenger i hovedsak av fotoelektrisk absorpsjon og Compton-spredning [35, 38].

2.2.2.1 Fotoelektrisk absorpsjon

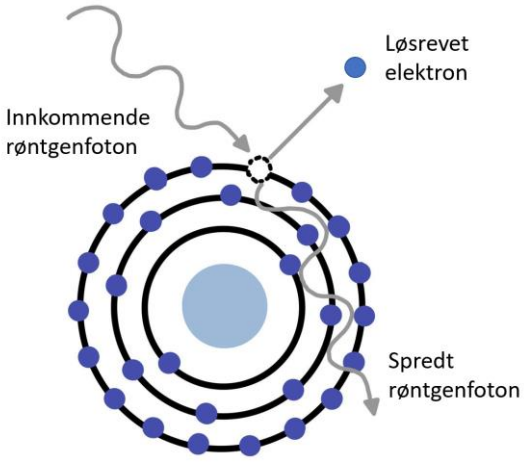
Ved fotoelektrisk absorpsjon vekselvirker et røntgenfoton med et elektron i en av de innerste orbitalene til et atom [35]. All energi overføres fra fotonet til elektronet i sammenstøtet. Dersom fotonenergien er høyere eller lik elektronets bindingsenergi vil elektronet løsrives fra orbitalen. Dette elektronet, kalt fotoelektronet, beveger seg en kort avstand i vevet før det fanges av et nærliggende atom, samtidig som den tomme elektronplassen i orbitalen raskt fylles [35]. Figur 2.4 illustrerer fotoelektrisk absorpsjon. Fotoelektrisk absorpsjon avhenger av vevets tetthet og atomtall, Z, og dominerer når røntgenstråleenergien er under 25 keV [35, 38].



Figur 2.4: Figuren illustrerer fotoelektrisk effekt. Et innkommende røntgenfoton overfører all energi til et elektron i det innerste skallet som løsrives (fotoelektronet). Figuren er tegnet etter inspirasjon fra illustrasjon i boken Introduction to Physics in Modern Medicine [35].

2.2.2.2 Compton-spredning

Compton-spredning har størst innvirkning på attenueringen når røntgenfotonenergien overstiger 25 keV og avhenger bare av vevets tetthet [35, 38]. Ved Compton-spredning løsriver det energirike røntgenfotonet et elektron fra orbitalen, som vist i Figur 2.5. Røntgenfotonet mister noe energi i vekselvirkningen med elektronet og blir avbøyd. Energien i vekselvirkningen er bevart og fordeles mellom det avbøyde røntgenfotonet og det frie elektronet [35].



Figur 2.5: Figuren illustrerer Compton-spredning der et innkommende røntgenfoton løsriver et elektron fra orbitalen. Røntgenfotonet mister noe energi til elektronet og blir spredt. Figuren er tegnet etter inspirasjon fra illustrasjon i boken Introduction to Physics in Modern Medicine [35].

Både Compton-spredning og fotoelektrisk absorpsjon bidrar til å svekke den transmitterte røntgenstråleintensiteten gjennom ulike typer vev [35]. Sammenhengen mellom attenueringskoeffisienten og røntgenstråleintensiteten kan derfor beskrives med ligning 2.1.

$$I_t = I_0 e^{(-\mu \Delta x)} \tag{2.1}$$

Her er I_t et mål på intensiteten til den transmitterte røntgenstrålen gjennom et materiale med tykkelse Δx, I_0 et mål på intensiteten til innkommende røntgenstråling og μ betegner attenueringskoeffisienten til det spesifikke området [7].

I rekonstruering av et bilde i CT-teknikken bestemmes attenueringen av røntgenstrålingen gjennom materialet [7]. Et CT-bilde kan enkelt forklares som en matrise bestående av piksler. Hver piksel i det rekonstruerte CT-bildet er et mål på den gjennomsnittlige attenueringen i et tredimensjonalt pikselelement. Dette tredimensjonale pikselelementet kalles en voxel og representerer et bestrålt volum av vevstypen under undesøkelse [7, 34].

2.2.3 CT-tall

For å tydeliggjøre vevstypens representasjon i de innsamlede attenueringsdataene brukes Hounsfield-skalaen, der hver voxel normaliseres etter voxler som inneholder vann [7, 34]. Denne nye størrelsen kalles CT-tall og definerer den relative forskjellen av strålingsabsorbsjon mellom vev og vann [39]. CT-tallet, med den dimensjonsløse Hounsfield-enheten HU, kan beskrives med ligning 2.2.

$$CT\text{-}tall = k\left[\frac{\mu_m - \mu_w}{\mu_w}\right] \tag{2.2}$$

Her representerer *k* en skaleringsfaktor ganget med brøken hvor μ_w er attenueringskoeffisienten til vann og μ_m er attenueringskoeffisienten til vevet under undersøkelse [7]. Gitt en skaleringsfaktor k lik 1000, vil CT-tallet til vann være 0 HU, CT-tallet til luft vil være -1000 HU, CT-tallet til fett vil være -60 HU til +70 HU og CT-tallet til kompakt ben ligge på +1000 HU [7, 34]. CT-tallet blir videre representert på en gråskala, hvor de ulike vevstypene kan diskrimineres i det rekonstruerte CT-bildet [7].

2.2.4 Rekonstruering

De innsamlede skalerte attenueringsdataene brukes videre til å rekonstruere et digitalt bilde av tverrsnittet ved hjelp av komplekse matematiske algoritmer [34]. Det finnes i hovedsak to kategorier av matematiske algoritmer som brukes av CT-systemer: itererende- og analytiske metoder [32]. I de itererende metodene starter CT-systemet med en vilkårlig gjetning av pikslenes attenueringsverdi. Gjetningen blir sammenlignet med de innsamlede dataene, og verdiene justeres. Prosessen gjentas til systemets iterative justeringer og de målte projeksjonene stemmer overens. I de analytiske metodene rekonstrueres tverrsnittet direkte fra de innsamlende dataene ved teknikken *back projection* [32]. Metoden danner bilder ved å projisere tilbake de innsamlede attenueringsdataene i den gitte vinkelen projeksjonen ble tatt.

Dette foretas i alle vinkler som resulterer i et uklart rekonstruert bilde [32, 34]. Ved å legge på et filter på innsamlingsdataene før rekonstrueringsteknikken *back projection,* forhindrer CT-maskinen den uklare effekten. Denne teknikken kalles *filtered back projection* og er en vanlig metode brukt av CT-systemer [32, 34]. Videre matematiske operasjoner og justeringer kan foretas for å forsterke detaljer og forskjeller i CT-bildet [32].

2.2.5 CT-Windowing

Ved å velge skalaen av CT-tall som bildet skal inneholde, kan ulike strukturer bli fremhevet og CT-bildet kan manipuleres [40]. Prosessen kalles ofte *Windowing* eller intensitetsvindusinnstilling, hvor vindusbredden («window width», WW) definerer CT-tallskalaen brukt i bildet. En stor vindusbredde vil resultere i bilder med en stor overgang fra mørke til lyse piksler. Stor vindusbredde kan være gunstig å bruke i undersøkelsesområder hvor attenueringsverdiene har store variasjoner, som for eksempel lunger der blodårer og luft møtes. Smal vindusbredde brukes i områder hvor det er gunstig å skille mellom attenueringsverdier med små variasjoner, som for eksempel myke vev [40]. Vindussenteret, også kalt vinduslevel («window level», WL), definerer midtpunktet på vindusbreddeskalaen [40]. Ulike verdier av WL og WW vil ha innvirkning på bildets representasjon og kan i mange tilfeller fremheve ønskelige kontraster. Noen ganger kreves det derimot ekstra tiltak for å få tilstrekkelig fremtoninger.

2.2.6 CT-kontrastmiddel

I tilfeller hvor radiologene ønsker å undersøke områder med lignende attenueringsverdi kan pasienten injiseres med et kontrastmiddel. Områder som for eksempel inneholder blodårer og andre myke vev kommer da tydeligere frem på CT-bildet [41]. Grunnstoffene barium og jod brukes ofte som kontrastmidler grunnet deres høye atomtall sammenlignet med annet vev i kroppen, henholdsvis med atomtall Z lik 56 og Z lik 53 [35, 38]. Ettersom fotoelektrisk absorpsjon både avhenger av vevets tetthet og vevets atomtall, vil kontrasten i bildet økes [38].

2.3 Positronemisjonstomografi

Positronemisjonstomografi er en bildeteknikk som i radiologi og onkologi kan brukes for å detektere og gradere svulster i kroppen [7, 33]. Metoden er en ikke-invasiv bildeteknikk som kvantitativt måler mengden radioaktivitet i levende vev [42]. Pasienten som skal undersøkes injiseres med radioaktivt stoff som inneholder positronemitterende radioaktive isotoper [42]. PET-teknikken danner bilder basert på det faktum at visse radioisotoper avgir positroner som

annihilerer med elektroner. Denne annihileringsprosessen danner videre to fotoner på 511 keV som sendes ut i motsatt retning og blir detektert i ulike detektorer [7, 42]. Radioaktivitetens posisjon blir beregnet basert på hvor fotonene detekteres, og et bilde kan dannes.

2.3.1 Isotoper

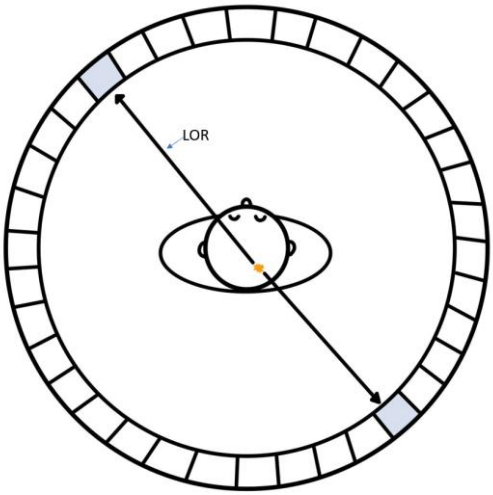
En atomkjerne er sammensatt av protoner og nøytroner med lignende masse og ulik ladning [7, 35]. Atomer med likt atomtall, men forskjellig nøytronantall kalles isotoper. Visse kombinasjoner av nøytroner og protoner kan føre til ustabile nuklider, også kalt radionuklider [7]. Ustabile-nuklider vil desintegrere, som resulterer i utsendelse av strålingsenergi [7, 35].

Radionuklider med overflod av protoner vil kunne emittere positroner ved desintegrering [7]. Etter at positronet har avsatt mesteparten av energien interagerer positronet med dens antipartikkel, elektronet. Antipartikkelen har lik masse som positronet, men ulik ladning, som gjør at de tiltrekkes [35]. Når positronet og elektronet interagerer oppstår den fysiske prosessen annihilering. Her konverteres massen til positron-elektronparet om til to gamma-fotoner på 511 keV som frigjøres i motsatt retning [7]. Forekommer slik desintegrering av en radionuklide inne i kroppens vev, kan de energirike gamma-fotonene unnslippe kroppen og detekteres av PET-skanneren [35].

2.3.2 PET-skanner

I en standard PET-skanner plasseres pasientens ønskede undersøkelsesområde inn i en ring av detektorer. Hvis de frigjorte gamma-fotonene innenfor detektorplanet oppdages samtidig i to forskjellige detektorer kan man anta at annihileringen skjedde langs en rett linje, kalt *Line of Response* (LOR), vist i Figur 2.6 [7]. Under en PET-undersøkelse registreres det hver gang det oppstår slike samtidige fotontreff. For å være sikker på at de to detekterte fotonene kommer fra samme annihilering, registreres de bare hvis de treffer innen et tidsvindu på noen få nanosekunder [31].

Data fra en PET-undersøkelse er enkelt forklart en liste med antall treff langs hver enkel LOR [7]. Fordelingen av de registrerte fotontreffene langs en retning representerer distribusjonen av radioaktiviteten innenfor det gitte undersøkelsesområdet. Ved å måle distribusjonen fra alle mulige vinkler, samler PET-skanneren inn informasjon som videre kan brukes for å rekonstruere et tverrsnittbilde av kroppens indre [7].



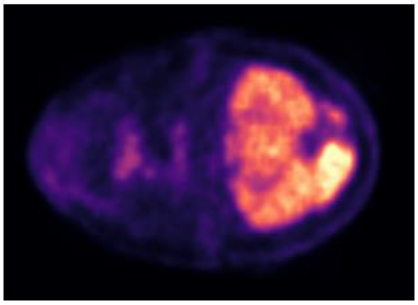
Figur 2.6: Figuren illustrerer en PET-skanning hvor pasientens ønskede undersøkelsesområde plasseres inn i en ring av detektorer. To frigjorte gamma-fotonene oppdages samtidig i to forskjellige detektorer, markert med blått i figuren, og det trekkes en linje, LOR, som man kan anta annihileringen skjedde langs. Figuren er tegnet etter inspirasjon fra illustrasjon i boken Introduction to Physics in Modern Medicine [35].

2.3.3 FDG

FDG, F-18 fluorodeoksyglukose (2-deoxy-2-[18F]-fluoro-D-glucose), er en av de vanligste radioaktive stoffene brukt for PET-avbildningsteknikken [33]. Radionukliden F-18, kombinert med glukose, danner FDG som intravenøst injiseres i kroppen. Den lave dosen radioaktivitet injisert, utgjør lav til ingen risiko for pasienten [33]. FDG fordeles videre i vev og akkumuleres i celler med høy metabolisme. Tumorceller er eksempler på områder som har en forhøyet metabolisme og vil derfor ta opp mye av det radioaktive stoffet [33, 42]. SUV-verdier, *Standardized Uptake Value*, brukes som mål på det relative FDG-opptaket i kroppen og oppgis ofte i g/mL (tetthet). SUV-indeksen normaliserer opptaket i en piksel til det gjennomsnittlige opptaket i resten av kroppen, som vist i ligning 2.3 [7].

$$SUV = \frac{(konsentrasjon\ av\ radioaktiv\ substans)}{(aktivitet\ til\ injisert\ dose)/(kroppsmasse)} \tag{2.3}$$

Områder med høy konsentrasjon av FDG forekommer mørke på svart-hvitt-PET-bilder som kan gi en indikasjon på tumorers lokasjon [33]. Ved å måle den radioaktive distribusjonen i kroppen får radiologene informasjon om kroppens anatomiske struktur og fysiologiske funksjoner [33]. Figur 2.7 viser et eksempel på et PET-bilde av hode til en pasient med hode- og halskreft. Den forhøyede aktiviteten i tumorcellene fremkommer lysende i PET-bildet.



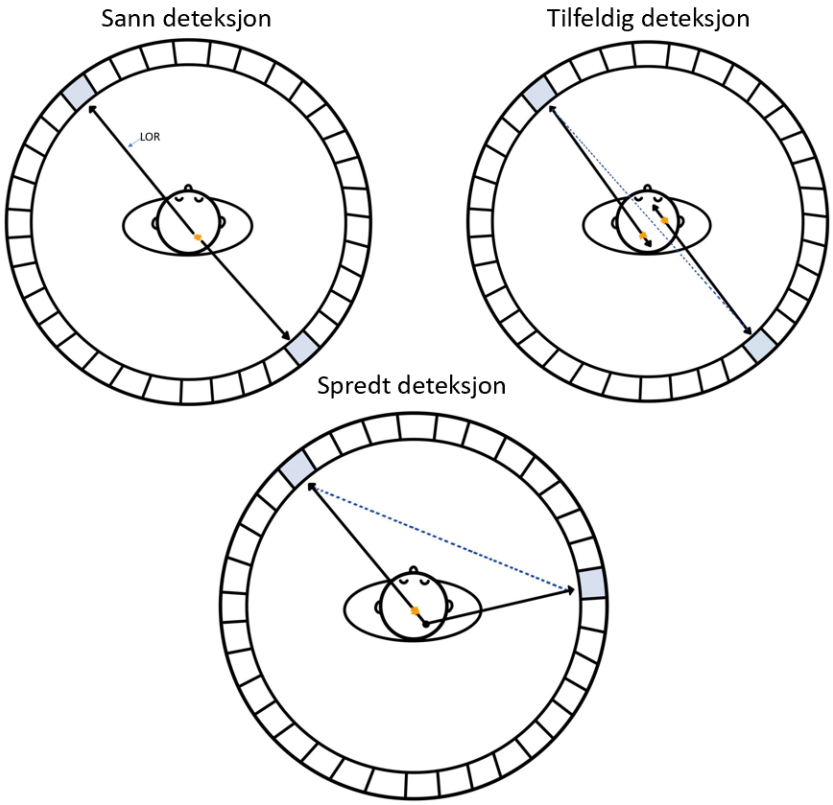
Figur 2.7: Figuren viser et eksempel på et PET-bilde av hode til en pasient med hode- og halskreft. Kreftsvulsten og hjernen fremkommer lysende i bildet.

2.3.4 Bilderekonstruering

De innsamlede dataene fra PET-skanneren benyttes videre til å rekonstruere tverrsnittbilder av radioaktiviteten distribuert langs det skannede området [43]. Ved å bruke de matematiske algoritmene til computertomografi, kan et bilde av kroppens indre gjenskapes på en ikke-invasiv måte. Som tidligere nevnt, er det i hovedsak to kategorier av metoder som benyttes til bilderekonstruering: analytiske og iterative. De analytiske metodene bruker algoritmene kalt *filtered back projection* for å gjenskape todimensjonale tverrsnittbilder. De iterative metodene gjenskaper bilder ved hjelp av en serie med iterasjoner som prøver å finne en passende representasjon av målingene. På denne måten rekonstrueres de innsamlede dataene til bilder som reflekterer distribusjonen av positron-emitterende atomer i området under undersøkelse [43].

2.3.5 Feilaktige deteksjoner

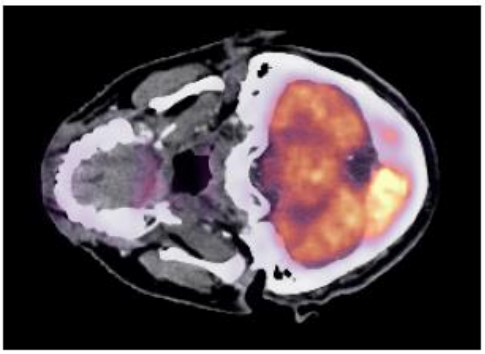
Det finnes forskjellige typer deteksjoner av fotoner [35], som vist i Figur 2.8. Sann deteksjon representerer detektering av to fotoner fra samme annihilering. Tilfeldig deteksjon forekommer når to fotoner fra forskjellige annihileringer detekteres. Spredt deteksjon representerer fotoner som har blitt attenuert i vevet og dermed endret retning grunnet Compton-spredning. Både tilfeldig og spredt deteksjon gir feilaktige LOR og det oppstår støy i bildet, samt unøyaktige representasjoner av radioaktivitet i undersøkelsesområdet [7]. CT-bilder kan brukes til å korrigere for attenueringen som oppstår, og det er derfor vanlig å kombinere en PET-skanner med en CT-maskin [7, 35].



Figur 2.8: Figuren illustrerer eksempler på sann deteksjon, tilfeldig deteksjon og spredt deteksjon som kan oppstå under en PET-skanning. Figuren er illustrert med inspirasjon fra figur i boken Introduction to Physics in Modern Medicine [35].

2.4 PET/CT

I dag kombineres nesten alle PET-skannere som lages med en CT-maskin, som gjør at en fullkropps undersøkelse kan utføres raskt og enkelt, der både anatomien og undersøkelsesområdets aktivitet blir avbildet [7]. Hos kreftpasienter kan CT-maskinen brukes til å tegne inn mistenkelige områder som videre blir undersøkt for forhøyet metabolsk aktivitet av PET-skanneren [35]. Figur 2.9 viser et eksempel på et PET/CT-bilde av en pasient med hode- og halskreft.



Figur 2.9: Figuren viser et PET/CT-bilde av en pasient diagnostisert med hode- og halskreft. Krefttumoren og hjernen fremkommer lysende i bildet.

Radiologer og onkologer bruker PET/CT-avbildninger for diagnostisering, overvåkning og planlegging av behandlingsforløpet [35]. Ved å kombinere PET-skanner med en CT-maskin kan CT-maskinen korrigere for attenuering i PET-bilder. Dette sparer mye tid og reduserer støy i bildet, som er en stor fordel [7]. En annen fordel med PET/CT-kombinasjonen er reduksjonen av falske negative og falske positive krefttumordeteksjoner, som kan forbedre diagnostiseringen hos kreftpasienter [35]. Ved å kombinere modalitetene kan CT brukes for å detektere tumorer som ellers ikke ville vært synlig i en PET-skanning grunnet lav metabolsk aktivitet, falsk negativ deteksjon. PET kan brukes for å skille ondartede og godartede tumorer som begge har blitt detektert av CT-skanneren, falsk positiv deteksjon [35].

Kapittel 3: Maskinlæring

3.1 Kunstig intelligens og maskinlæring

Kunstig intelligens (KI) er et felt innenfor datavitenskap som handler om å automatisere intellektuelle menneskelige oppgaver ved å gi datamaskinen evnen til å lære fra et sett med regler og innsamlet data [17]. Tanken om kunstig intelligens startet allerede på 1950-tallet og har siden den tid utviklet seg fort. På slutten av 1900-tallet oppstod underkategorien maskinlæring som har vist seg å bli en av de mest suksessfulle og populære grenene innenfor KI. Maskinlæring omhandler bruken av selvlærende algoritmer som ut ifra et datasett henter ut informasjon og basert på dette trekker selvstendige slutninger [17]. Underkategorien brukes i økende grad i det dagligdagse liv og har vist lovende fremgang, blant annet innenfor medisinsk diagnostisering [16].

3.2 Maskinlæring

I maskinlæring trenes en maskinlæringsmodell til å finne komplekse mønstre og sammenhenger fra datasett og foretar videre prediksjoner på ny presentert data [44, 45]. Sammenhengen funnet i datasettet beskrives av ligninger med tilhørende koeffisienter, også kalt vekter, som justeres etter hvert som modellen trenes [44]. Det at maskinlæringsmodeller trenes, skiller maskinlæring fra andre grener innenfor KI, hvor modellene blir forhåndsprogrammert med regler som brukes for å hente ut informasjon fra datasettet [17]. Maskinlæring kan grovt deles inn i tre underkategorier: veiledet læring, ikke-veiledet læring og forsterket læring, hvor veiledet- og ikke-veiledet læring er hovedkategorier [16].

3.2.1 Veiledet læring

Veiledet læring omhandler modeller hvor både inputdata og korresponderende outputdata er kjent [17]. En modell blir i denne kategorien trent opp til å finne en god representasjon av sammenhengen mellom inputverdier som gir den tilhørende outputverdien. Klassifisering er en vanlig oppgave innenfor veiledet læring der målet er å kategorisere et nytt treningseksempel basert på tidligere observasjoner [16]. Modellen trenes på et treningsdatasett med kjente outputverdier gjentatte ganger helt til modellen har klart å finne en funksjon, f, som klassifiserer hvert treningseksempel korrekt. Modellen kan da kjøres på et nytt datasett og predikerer klasser uten menneskelig interaksjon [45]. En annen vanlig oppgave innenfor veiledet læring er regresjonsanalyse. I regresjonsanalyse prøver modellen å finne sammenhengen mellom et sett med forklarende variabler, inputdata, som gir kontinuerlige responsverdier, outputdata [16].

3.2.2 Ikke-veiledet læring

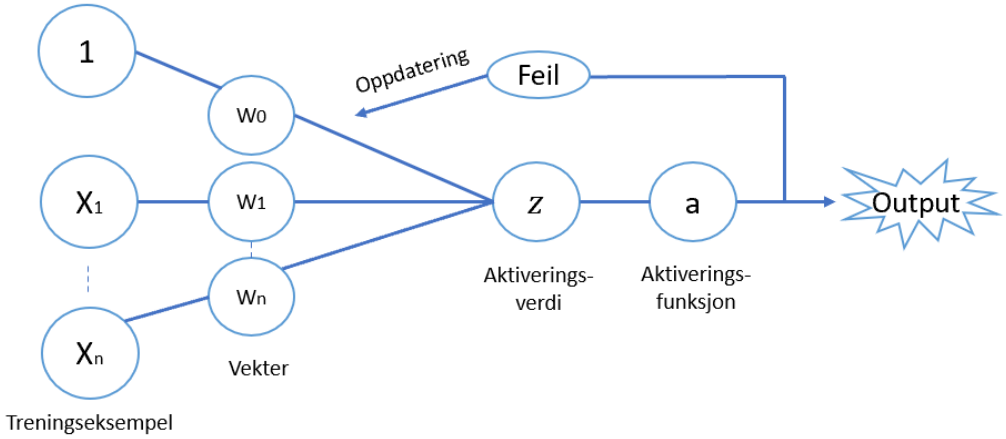
Ikke-veiledet læring håndterer ukjente datasett som mangler en tilhørende outputverdi. Modellen som trenes på datasettet har bare tilgang til inputdataen og må lete etter skjulte mønstre og sammenhenger uten veiledning. En teknikk mye brukt innenfor ikke-veiledet læring er gruppering, som organiserer datasettet i ulike grupper for å hente ut informasjon [16].

3.2.3 Forsterket læring

I forsterket læring får modellen en belønning eller straff ut ifra adferden. Målet er å utvikle et system som forbedrer sin ytelse basert på hvordan modellen handler i miljøet [16]. Forsterket læring er fremdeles på forskerstadiet og praktiseres ikke i like stor grad som veiledet og ikke-veiledet læring [17].

3.2.4 Nevroner

Noen av algoritmene brukt i maskinlæring og kunstig intelligens er inspirert av hvordan nevroner i hjernen fungerer [16]. I biologisk sammenheng er nevroner nerveceller i hjernen som bidrar til prosessering og transportering av kjemiske og elektriske signaler. Nevronene kan beskrives som enkle, logiske porter som mottar inputsignaler fra andre nærliggende celler. Hvis inputsignalene overstiger en viss grense, vil et outputsignal bli avfyrt og sendt videre til andre nærliggende nevroner. Kort tid etter nevroners funksjon ble beskrevet, ble den første maskinlæringsalgoritmen som baserer seg på nevronets funksjonalitet, Perceptron, laget [16]. Figur 3.1 viser en skjematisk fremstilling av læringsprosessen til Perceptron som mottar input fra et treningseksempel X og kombinerer dette med vekter W for beregning av en aktiveringsverdi z. Aktiveringsverdien sendes til aktiveringsfunksjonen a, som avgjør om et outputsignal blir avfyrt. Outputsignalet brukes til å beregne den predikerte feilen og til å oppdatere vektene. Maskinlæringsmodeller kan bestå av slike nevroninspirerte algoritmer, også kalt noder, som satt sammen i flere lag kalles nevrale nettverk. Slike nevrale nettverk brukes i dyp læring som er en sentral del innenfor maskinlæring [17].



Figur 3.1: Skjematisk fremstilling av maskinlæringsalgoritmen Perceptron. Input fra treningseksempel, X, kombineres med vekter, W, og det beregnes en aktiveringsverdi, z. Aktiveringsverdien går inn i aktiveringsfunksjonen, a, som gir ut en outputverdi. Outputverdien brukes til å beregne predikert feil og til å oppdatere vektene. Figuren er tegnet med inspirasjon fra illustrasjon i boken Python Machine Learning [16].

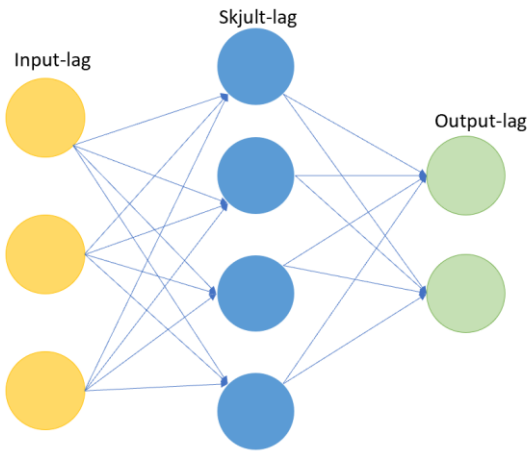
3.3 Dyp læring

Dyp læring kan sees på som et matematisk rammeverk som lærer å representere data gjennom flere påfølgende lag. Antall lag avgjør hvor dyp modellen er [17]. Jo flere lag, jo flere egenskaper klarer nettverket å lære fra treningseksemplene [45]. Hvert lag i et dypt nevralt nettverk består av en eller flere noder koblet sammen med andre lag. Nodene fungerer som en bryter på lik linje som nevroner i hjernen og utfører en transformasjon på inputdataen når de blir aktivert [44].

Dype nevrale nettverk har flere fordeler sammenlignet med konvensjonelle maskinlæringsalgoritmer [45]. Nettverket finner automatisk de beste egenskapene som representerer datasettet. Dermed reduseres mye tid brukt av mennesker på manuell utvelgelse av egenskapene, som de konvensjonelle maskinlæringsalgoritmene skal lære fra. Dype nevrale nettverk er heller ikke sensitive til støy og kan derfor gjenkjenne lærte objekter og strukturer som har blitt forvrengt, eller inneholder mangelfull informasjon [45].

3.3.1 Lag i et nevralt nettverk

Figur 3.2 viser strukturen til et dypt nevralt nettverk som består av et input-lag, et skjult lag og et output-lag.



Figur 3.2: Figuren illustrerer strukturen til et nevralt nettverk bestående av input-lag, et skjult lag og et output-lag koblet sammen med vekter. Figuren er laget etter inspirasjon fra boken Deep Learning with Python [17].

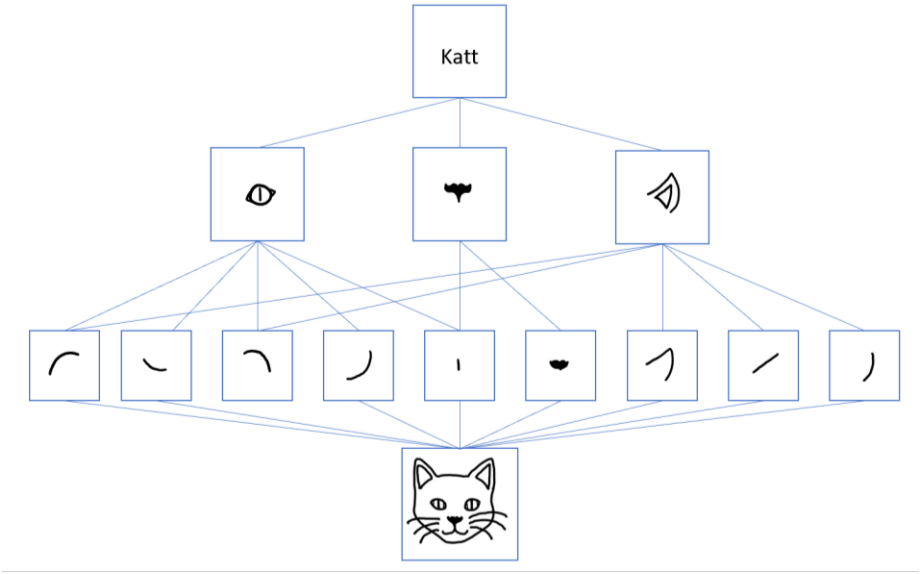
Det som foregår i de ulike lagene, kan enkelt forklares ved hjelp av en klassifiseringsoppgave som handler om å skille datapunkter fra hverandre [45]. Oppgaven kan for eksempel være å lære modellen å skille bilder med hunder og katter, som brukes som eksempel videre.

3.3.1.1 Inputlaget

Inputlaget representerer datasettet hvor outputverdien holdes konstant. Dette skiller inputnodene fra noder i de andre lagene hvor dataen blir transformert [45]. Inputlaget tar inn et treningseksempel fra datasettet som skal undersøkes, for eksempel et bilde av en katt. Antall noder i inputlaget er likt antall bildeelementer (piksler) i bildet [45].

3.3.1.2 Skjulte lag

Videre sendes treningseksempelet gjennom nodene i de skjulte lagene i nettverket. Nodene transformerer treningseksempelet til nye representasjoner som gradvis skiller seg fra originaldataene for å hente ut nyttig informasjon [17]. De skjulte lagene i visse dype nevrale nettverk har en hierarkisk oppbygning, der de lærer mer og mer komplekse egenskaper hos treningseksemplene [16, 45]. Nettverket starter med å lære store strukturer, for eksempel linjer og kanter, som er likt for både hunder og katter. I neste lag lærer nettverket å sette sammen disse linjene og kantene og lærer hva som skiller kategoriene fra hverandre. De påfølgende lagene lærer mer komplekse egenskaper, som for eksempel posisjonen til høyre øye eller venstre pote, spesifikt for hunder og katter [45]. Figur 3.3 viser et eksempel på hierarkisk uthenting av informasjon fra et treningseksempel av en katt hvor kanter og linjer settes sammen til øye, nese og øre, som til slutt resulterer i prediksjonen katt.



Figur 3.3: Illustrasjon av hierarkisk uthenting av informasjon fra et treningseksempel, katt. Kanter og linjer settes sammen til lokale objekter, som til slutt resulterer i prediksjonen katt øverst i figuren. Figuren er laget etter inspirasjon fra boken Deep Learning with Python [17].

3.3.1.3 Outputlaget

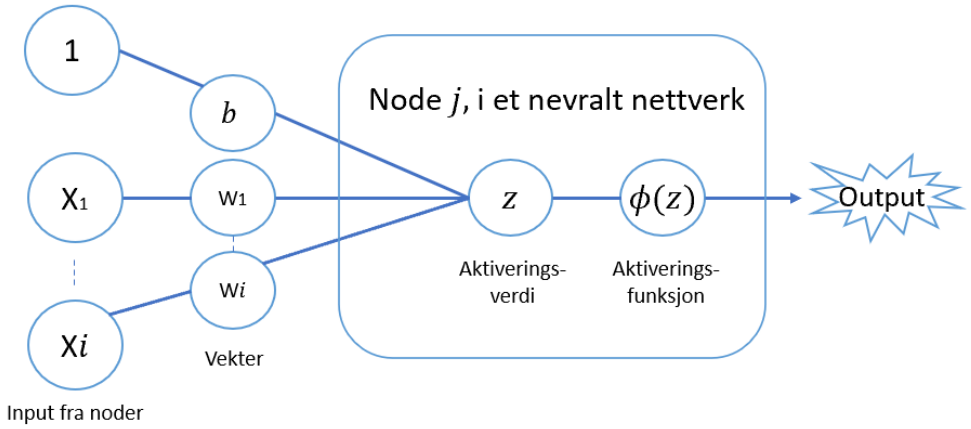
I outputlaget blir de nye representasjonene av bildet samlet. Outputlaget kan bestå av en eller flere noder ut ifra oppgaven det nevrale nettverket skal løse [45]. En binær klassifiseringsoppgave har en eller to outputnoder, mens multiklassifiseringsoppgaver har like mange outputnoder som antall klasser. Outputverdien kan ses på som en sannsynlighet for at treningseksempelet representerer en gitt klasse. Outputnoden med den høyeste verdien vil derfor være prediksjonen til nettverket, i dette tilfellet hund eller katt [45].

3.3.2 Vekter

Datatransformasjonene utført i de skjulte lagene blir justert etter hvert som nettverket blir eksponert for nye treningseksempler. Transformasjonene vektlegges med et sett med vekter som beskriver hvor viktig transformasjonene er for klassifiseringen [17]. Vektene kan sammenlignes med synapseovergangen mellom to nevroner i hjernen [45]. Synapseovergangen markerer kontaktstedet mellom to nerveceller hvor signaler overføres. Jo viktigere signalet er, jo sterkere er synapseovergangen. På lik linje, vil overganger mellom to noder i et nevralt nettverk ha høye vekter hvis overgangen er viktig [45]. Oppgaven til et nevralt nettverk kan derfor defineres som å finne et sett med vekter til alle lag, slik at treningseksempler blir riktig klassifisert [17].

3.3.3 Aktiveringsfunksjon

En node brukt i et nevralt nettverk kan skjematisk fremstilles som vist i Figur 3.4.



Figur 3.4: Illustrasjon av en node (indikert med boks) i et nevralt nettverk. Noden mottar input fra andre noder i nettverket, som summeres med vekter i en aktiveringsverdi z. Aktiveringsverdien sendes videre gjennom en aktiveringsfunksjon og det genereres et outputsignal. Figuren er tegnet med inspirasjon fra illustrasjon i boken Python Deep Learning [45].

Inputdata fra foregående noder kombineres med tilhørende vekter og summeres. Den vektede summen z, beskriver aktiveringsverdien til noden og er gitt ved ligning 3.1 [45].

$$z = \Sigma_i w_i x_i \ + \ b \tag{3.1}$$

z representerer den vektede summasjonen mellom inputdataen x_i fra node i i det foregående laget, vektlagt med w_i som representerer styrken mellom node i og den nåværende noden j. b representerer her en bias som gjør at hyperplanet ikke nødvendigvis går igjennom origo. Hyperplanet er en geometrisk fremstilling og defineres av vektene som prøver å separere datapunktene på best mulig måte [45]. Den vektede summen inngår videre i en aktiveringsfunksjon, $\phi(z)$, som avgjør outputverdien til noden. En node i et nevralt nettverk kan ses på som en selvstendig klassifiseringsmodell som aktiveres når aktiveringsverdien er større eller lik en gitt grense, eller hvis inputdataen befinner seg på den ene siden av hyperplanet, geometrisk fremstilt [45].

Aktiveringsfunksjoner er lineære og ikke-lineære kontinuerlige funksjoner som avgjør outputen til en node [45]. Det finnes mange forskjellige aktiveringsfunksjoner som velges ut ifra oppgaven maskinlæringsmodellen skal løse. Aktiveringsfunksjonen kan variere fra lag til lag i et nevralt nettverk, men er lik for alle noder i et og samme lag [45]. Nevrale nettverk som inneholder lineære aktiveringsfunksjoner, kan bare lære lineære transformasjoner og representasjoner av inputdataene [17]. Dette gir en begrensning på informasjonen nettverket klarer å hente ut fra datasett, og vil i noen tilfeller føre til at det nevrale nettverket ikke finner

en god nok representasjon av dataene. Det er derfor vanlig å velge differensierbare ikke-lineære aktiveringsfunksjoner i nevrale nettverk. Ved å introdusere ikke-lineære funksjoner, øker mengden representasjoner og transformasjoner tilgjengelig for å hente ut informasjon [17, 45]. Noen av de vanligste aktiveringsfunksjonene brukt i nevrale nettverk er blant annet sigmoid, tanh og ReLU [45].

3.3.3.1 Sigmoid aktiveringsfunksjon

Ligning 3.2 viser sigmoid funksjonen, også kalt logistisk funksjon, som vanligvis brukes som outputnode-aktiveringsfunksjon i binære klassifiseringsoppgaver [45].

$$\phi(z) = \frac{1}{1 + e^{(-z)}} \tag{3.2}$$

Noder som inneholder sigmoid funksjonen transformerer aktiveringsverdien til en verdi mellom 0 og 1 og kan tolkes som sannsynligheten for at noden aktiveres [45]. Hvis nodene har negative inputverdier, kan sigmoid funksjonen skape problemer i de skjulte lagene [16]. Aktiveringsfunksjonen fører da til outputverdier nær null, som gjør at det nevrale nettverket lærer sakte og potensielt ikke finner en god nok representasjon av datasettet. Det er derfor vanlig å bruke tanh eller ReLU som aktiveringsfunksjon i de skjulte lagene [16].

3.3.3.2 Tanh aktiveringsfunksjon

Aktiveringsfunksjonen tanh, også kalt tangens hyperbolikus, vist i ligning 3.3, ligner sigmoid funksjonens transformasjon av data, og kan tolkes som en reskalert versjon av sigmoid funksjonen [16].

$$\phi(z) = \frac{e^{(z)} - e^{(-z)}}{e^{(z)} + e^{(-z)}} \tag{3.3}$$

Tanh transformerer aktiveringsverdien til en verdi mellom -1 og 1 [16, 45].

3.3.3.3 ReLU aktiveringsfunksjon

ReLU, også kalt *Rectified Linear Unit*, er en populær aktiveringsfunksjon og kan matematisk fremstilles som vist i ligning 3.4 [16, 17].

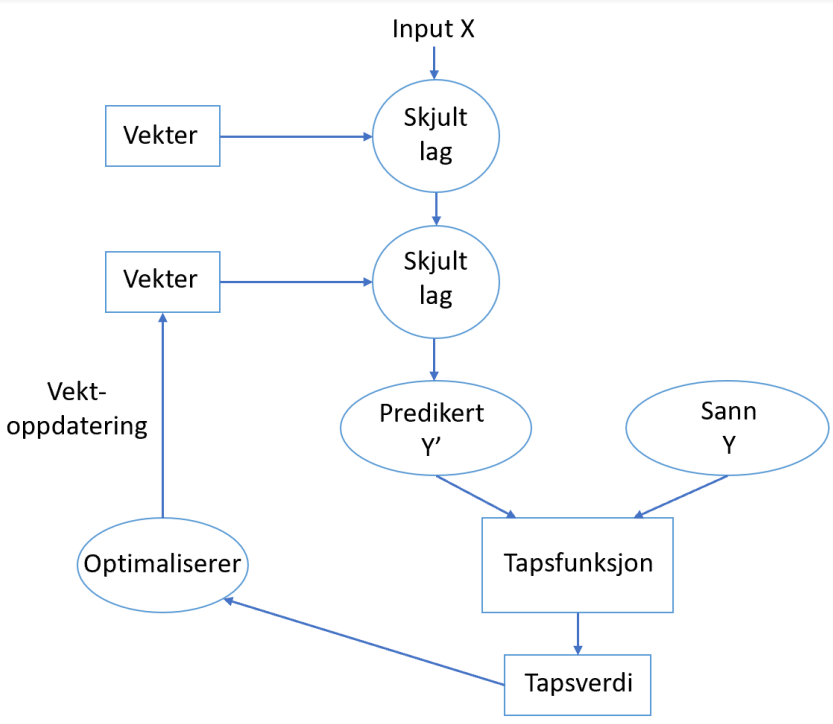
$$\phi(z) = max(0, z) \tag{3.4}$$

Aktiveringsfunksjonen transformerer alle aktiveringsverdier til en verdi mellom 0 og uendelig, og egner seg bra til å lære komplekse funksjoner i nevrale nettverk [16].

Den deriverte av aktiveringsfunksjonen brukes i sammenheng med oppdateringen av nettverkets vekter. For visse aktiveringsfunksjoner går oppdateringen av vektene mot null etter hvert som antall lag i nettverket øker. Dette fører til ineffektiv oppdatering av vektene i de tidlige lagene. Ettersom den deriverte til aktiveringsfunksjonen ReLU alltid er 1 for positive tall, hindres dette problemet. ReLU brukes derfor mye i de skjulte lagene [16].

3.3.4 Optimalisering av et nevralt nettverk

Et nevralt nettverk har som oppgave å finne et sett med funksjoner som viser sammenhengen mellom inputverdier og deres tilhørende outputverdi [45]. Funksjonene avhenger av vektene til hvert lag, og oppgaven til et nevralt nettverk kan, som tidligere nevnt, forenkles til å finne de mest optimale vektene [17]. Figur 3.5 viser et eksempel på læringsfasen til et nevralt nettverk. Når et nevralt nettverk trener på et datasett får nettverket som input et treningseksempel, indikert med X i figuren, og gir som output en prediksjon, Y'. Denne prediksjonen blir videre sammenlignet med den korrekte tilhørende outputverdien, sann Y, og det kalkuleres en tapsverdi ved hjelp av en tapsfunksjon, som definerer forskjellen mellom verdiene. Målet til det nevrale nettverket er å minimere denne tapsverdien slik at prediksjonen blir så korrekt som mulig [16]. Den beregnede tapsverdien brukes av en optimaliserer, en optimaliseringsteknikk som oppdaterer vektene til nettverket. Læringsprosessen gjentas til nettverket har funnet vektene som minimerer tapsfunksjonen [16, 17].



Figur 3.5: Figuren viser læringsprosessen til et nevralt nettverk som får et treningseksempel X som input. Inputen transformeres gjennom de skjulte lagene og det predikeres en outputverdi Y'. Den predikerte outputverdien og den sanne

outputverdien Y går inn i en tapsfunksjon hvor det beregnes en tapsverdi, som videre brukes av en optimaliserer for å oppdatere nettverkets vekter. Prosessen gjentas til vektene som minimerer tapsfunksjonen er funnet. Figuren er illustrert med inspirasjon fra illustrasjon i boken Deep Learning with Python [17].

3.3.5 Tapsfunksjon

Tapsfunksjon, også kalt kostfunksjon, kontrollerer kvaliteten på nettverkets outputverdier. Tapsfunksjonen gir en vurdering av sammenhengen mellom nettverkets predikerte outputverdi og den sanne outputverdien fra treningsdatasettet, som beregnes ut ifra et avstandsmål [17]. I starten av treningsprosessen tildeles lagene tilfeldige vekter, og det utføres derfor tilfeldige transformasjoner på treningsdataen. Outputverdien til nettverket vil av den grunn ofte skille seg i stor grad fra den korrekte verdien [17]. Etter hvert som nettverket blir eksponert for nye treningseksempler blir vektene justert og tapsfunksjonen minker. Det finnes forskjellige tapsfunksjoner som velges ut ifra oppgaven det nevrale nettverket står ovenfor. Midlere kvadratisk feil, («Mean Squared Error», MSE) er en vanlig tapsfunksjon brukt i regresjonsoppgaver, mens ulike kryssentropi tapsfunksjoner («cross-entropy loss») ofte brukes i klassifiseringsoppgaver [16]. En tapsfunksjon som spesialiserer seg på segmenteringsoppgaver er DiceLoss definert av Milleatri et al. [46]. Utrykket for DiceLoss-funksjonen er gitt i ligning 3.5.

$$DiceLoss = 1 - \frac{2\Sigma_i p_i g_i}{\Sigma_i (p_i)^2 + \Sigma_i (g_i)^2} \tag{3.5}$$

I sammenheng med kreftsvulstinntegning i PET/CT-bilder betegner p_i den predikerte inntegningen for et bilde, g_i betegner den sanne inntegningen for et bilde, mens i representerer en piksel [46].

En tapsfunksjon kan matematisk fremstilles som et sett med punkter på en linje hvor bunnpunktet representerer den laveste tapsverdien [45]. Ved å velge et punkt på denne linjen og følge kurven hvor den deriverte til linjen er minst, kan modellen finne et minimum som vil kunne gi de optimale vektene. Teknikkene brukt for å finne bunnpunktet på best mulig måte kalles optimaliseringsteknikker [45].

3.3.6 Optimaliseringsteknikker

Optimaliseringsteknikker bruker den beregnede tapsfunksjonsverdien for å oppdatere nettverkets vekter [17]. En vanlig metode for å minimere tapsfunksjonen er å oppdatere vektene med en gitt steglengde i motsatt retning av gradienten til tapsfunksjonskurven [16, 17]. En gradient er en generalisering av derivasjonen til funksjoner med multidimensjonale inputverdier som, på lik linje med den deriverte, beskriver stigningen til funksjonen. Ved å bevege seg i motsatt retning av stigningen vil metoden dermed kunne finne et globalt

minimum. Denne optimaliseringsalgoritmen kalles gradient-nedstigning («gradient descent») og er mye brukt i maskinlæringsmodeller [16, 17]. Ligning 3.6 og ligning 3.7 beskriver hvordan vektene kan oppdateres.

$$\boldsymbol{w} = \boldsymbol{w} + \Delta\boldsymbol{w} \tag{3.6}$$

$$\Delta\boldsymbol{w} = -\eta\Delta J(\boldsymbol{w}) \tag{3.7}$$

$\boldsymbol{w}$ representerer her summen av vektene som skal endres med oppdateringen $\Delta\boldsymbol{w}$. $\Delta\boldsymbol{w}$ er gitt ved gradienten ∇J til tapsfunksjonen J, beregnet med hensyn på vektene ganget med steglengden, også kalt læringsraten, η. Minustegnet indikerer at oppdateringen beveger seg i motsatt retning av gradienten for å minimere tapsfunksjonen [16].

Det finnes mange varianter av optimaliseringsalgoritmer. Stokastisk gradient-nedstigning («Stocastic gradient descent» SGD) og *Adaptive moment estimation* (Adam) [47], er de vanligste brukte optimaliseringsalgoritmene [16]. SGD-metoden oppdaterer vektene i motsatt retning av gradienten til en tilfeldig samling med treningseksempler etter hver gjennomkjøring [17]. Adam er en effekt optimaliseringsteknikk som krever lite minne og kan brukes på store datasett. Den gradientbaserte optimaliseringsteknikken Adam beregner tilpassende læringsrater η (steglengde) til forskjellige parametere ut ifra estimater av gjennomsnittet og variansen til tidligere gradienter [16, 47]. Ved å velge passende tapsfunksjon og passende optimaliseringsalgoritme, vil det nevrale nettverket raskt finne vektene som på best mulig måte løser oppgaven nettverket står ovenfor [16, 17].

3.3.7 Tilbakepropagering

For et nevralt nettverk bestående av bare ett lag kan vektene oppdateres ved bruk av differansen mellom outputlagets prediksjon og den faktiske outputverdien, som beskrevet tidligere. Dype nettverk som består av flere lag, må derimot bruke en litt annen metode for å oppdatere vektene gjennom nettverket [45]. Ettersom den sanne outputverdien til en node i det skjulte laget ikke er kjent slik som i outputlaget, må tapsverdien (feilen) som brukes for å oppdatere de skjulte lagenes vekter, estimeres. Feilen estimeres ved hjelp av den beregnede kjente feilen fra outputlaget og føres bakover gjennom nettverket. Denne metoden kalles tilbakepropagering. Tilbakepropagering er en komplisert algoritme som baserer seg på kjerneregelen for å oppdatere vektene i lagene etter hvert som den beveger seg fra outputlaget til inputlaget [45]. På denne måten endres vektene i et dypt nevralt nettverk mot det gunstige.

3.4 Trenings-, valideringss- og testsett

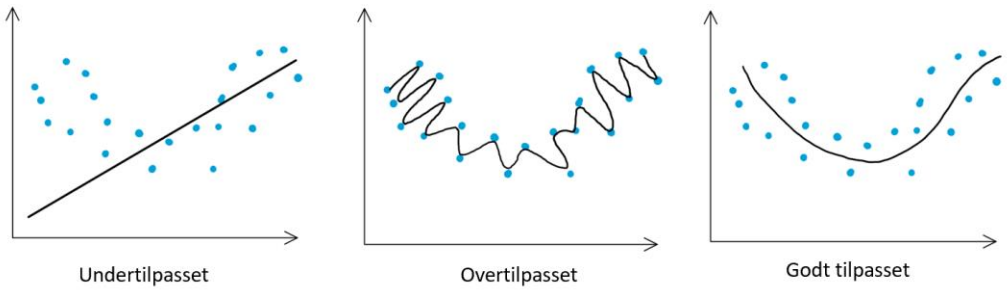
Når en maskinlæringsmodell og nevrale nettverk trenes på et datasett, får modellene treningseksempler som input. For å kunne evaluere om modellen har lært fra treningseksemplene kreves det i tillegg et valideringssett og et testsett [17].

Etter hvert som modellen trener på et treningssett vil prestasjonen til modellen forbedres på treningseksemplene. Trenes modellen i det uendelige, vil modellen klare å klassifisere hvert treningseksempel i treningssettet korrekt. Introduseres modellen derimot for ny data, vil prestasjonen kunne minke. Modellen kan ha lært alle detaljene i treningsdatasettet og klarer dermed ikke generalisere til ny usett data. Ved å bruke et valideringssett kan modellen evalueres etter hver gjennomkjøring og generaliseringsevnen til modellen kan overvåkes. Etter at modellen er ferdig trent, testes modellen på usett data i testsettet, som vurderer modellens ytelse [17].

For å korrekt kunne evaluere om modellen har lært det den skal lære fra treningsdatasettet må settene inneholde en tilnærmet lik andel av de ulike klassene. Inneholder treningssettet treningseksempler for klasse 1, men ikke klasse 2, vil modellen predikere dårlig på klasse 2 i testsettet. Når datasettet deles opp i de ulike settene er det derfor viktig at mangfoldet av klassene bevares i hvert sett. Denne prosessen kalles stratifisering [16].

3.5 Overtilpasning

Fenomenet overtilpasning brukes om modeller som klassifiserer alle treningseksemplene korrekt, men generaliserer dårlig til ny usett data [16, 17, 45]. Modellen har da lært mønstre og strukturer i datasettet som kan kategoriseres som støy uten betydningsfull informasjon [45]. I maskinlæring er det en fin balanse mellom optimalisering og generalisering [17]. En maskinlæringsmodell har som mål å lære tilstrekkelig informasjon fra et datasett, optimalisering, for så å predikere korrekt på et nytt datasett, generalisering. Lærer modellen for mye fra treningssettet, vil modellen kunne gi feilaktige prediksjoner på usett data, og er da overtilpasset. Lærer modellen for lite fra datasettet vil maskinlæringsmodellen også kunne gi feilaktige prediksjoner på usett data, og modellen er da undertilpasset [17]. Figur 3.6 viser eksempler på undertilpassing, overtilpassing og god tilpassing av datapunkter.



Figur 3.6: Figuren illustrerer eksempler på en modell som er undertilpasset (venstre), overtilpasset (midten) og godt tilpasset (høyre) datapunkter i en gitt oppgave.

Komplekse modeller som trener på et lite datasett er utsatt for overtilpasning. Modellen har da for mange parametere som fører til høy varians [16]. Det finnes flere metoder og teknikker som kan brukes for å forhindre overtilpasning.

3.5.1 Regularisering

Regularisering av en modells vekter er teknikker som endrer fordelingen av vektene mot det generelle. Regulariseringsmetodene legger på et ekstra ledd, en kostnad, til tapsfunksjonen som reduserer ekstreme vektverdier [16, 17].

3.5.2 Utelatelse

Utelatelse («dropout») er en vanlig og effektiv regulariseringsmetode mye brukt i nevrale nettverk. Metoden baserer seg på å sette et bestemt antall tilfeldige noder i et lag lik null [17]. Ved å «slå av» visse noder, tvinges nettverket til å lære en mer generell og robust representasjon av dataene [16, 17].

3.5.3 Dataaugmentering

Den beste metoden for å redusere overtilpasning er å øke variasjonen og størrelsen på datasettet som modellen trener på [17]. Dataaugmentering er en samling av teknikker, der noen baserer seg på å endre de allerede eksisterende treningseksemplene til nye varianter. På denne måten utvides variasjonen og mengden treningseksempler i datasettet, som kan øke modellens generaliseringsevne [17]. Delkapittel 3.8 beskriver videre ulike dataaugmenteringsteknikker.

3.5.4 Redusere kapasiteten til et nettverk

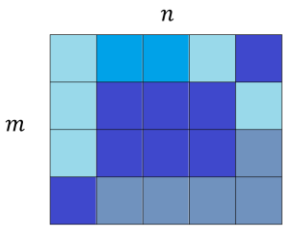
Modellen som trenes kan i tillegg reduseres i antall trenbare parametere, som er den enkleste metoden for å redusere overtilpasning [17]. Trenbare parametere bestemmes av antall lag og antall noder i lagene, som definerer modellens kapasitet. Jo flere trenbare parametere, jo mer kan modellen lære fra datasettet [17]. En form for dype nevrale nettverk, kalt

konvolusjonsnettverk, har et redusert antall trenbare parametere, ettersom nodene i nettverket kobles lokalt sammen med andre noder som representerer nabopiksler i inputbilder fra datasettet [45]. På denne måten reduseres risikoen for at nettverket blir overtilpasset.

3.6 Konvolusjonsnettverk

Konvolusjonsnettverk («Convolutional Neural Networks», CNNs) er en samling dype nevrale nettverk som egner seg spesielt godt til bildeklassifiseringsoppgaver [16]. CNN-modellene er inspirert av synssenteret i den menneskelige hjerne som kan gjenkjenne objekter og strukturer [16, 45]. Synssenteret er hierarkisk oppbygd i prosessering og forståelse av informasjon. Todimensjonale vektorer bestående av ulike fargeintensiteter fanges opp som input til netthinnen i øyet. Synssignalene sendes videre gjennom den optiske nerven til thalamus i hjernen, som mottar sensorisk informasjon fra sansene. Signalet sendes så gjennom ulike områder i hjernen som ekstraherer og prosesserer informasjonen. Et område er for eksempel ansvarlig for fargetolkning, et annet for form- og ansiktsgjenkjenning. De enkle synssignalrepresentasjonene hentet ut på ulike nivåer, settes til slutt sammen til en tolkning av signalet [45]. På lik linje som synssenteret i hjernen, klarer CNN-modeller å lære seg de viktigste egenskapene for klassifisering av data ved hierarkisk uthenting og prosessering av informasjon [16]. CNN-modeller kalles derfor ofte «egenskapsuthentere» («feature extractors») [16].

Et bilde med høyde m og bredde n består av $m \times n$ firkantede bildeelementer, som vist i Figur 3.7. Disse elementene kalles piksler og representerer en intensitetsverdi. Intensitetsverdien angir intensiteten og fargen i det gitte området [48].



Figur 3.7: Figuren illustrerer et bilde med høyde m og bredde n som består av m x n bildeelementer, piksler, som representerer fargeintensiteten i det gitte området.

CNN-modeller utnytter det faktum at nærliggende piksler i et bilde ofte inneholder mer relevant informasjon enn piksler som er plassert langt fra hverandre [16, 45]. Nettverket kan på denne måten lære lokale mønstre fra et område med piksler i inputbildet. Et lokalt mønster trukket ut fra et område i inputbildet kan gjenkjennes i en annen region og CNN-modeller kan av den grunn bruke de samme vektene. Som tidligere nevnt har nettverket derfor færre

trenbare parametere, som gjør CNN-modeller effektive og robuste mot overtilpasning [16, 17, 45].

3.6.1 Lag i konvolusjonsnettverk

Et konvolusjonsnettverk består i hovedsak av tre typer lag: konvolusjonslag («convolutional layer»), undersamplingslag («subsampling layer») og fullt koblet-lag («fully connected layer») [16]. Konvolusjonslag foretar transformasjoner og operasjoner på inputdata i nettverket, etterfulgt av undersamplingslag, også kalt samlelag, som endrer outputstørrelsen fra inputstørrelsen. Nettverket avsluttes med et eller flere fullt koblet-lag, hvor alle nodene i laget er koblet sammen med alle outputnodene [16].

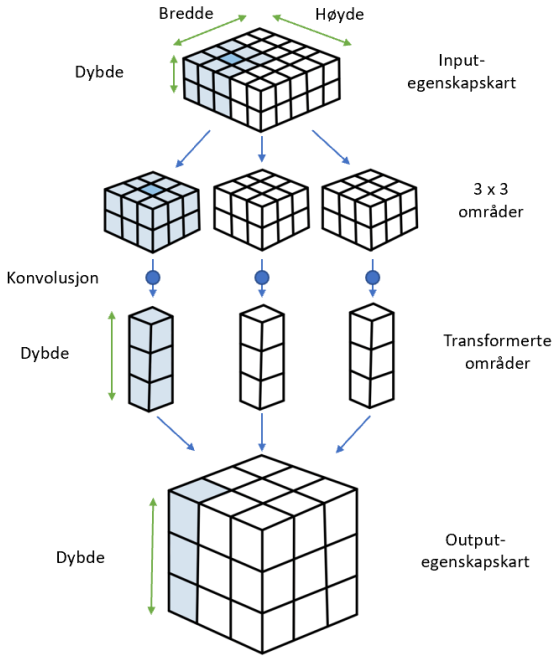
3.6.1.1 Konvolusjonslag

En konvolusjon er en operasjon som transformerer et område med piksler om til en ny representasjon [16]. Konvolusjonslag består av filtre som opererer over tredimensjonal inputdata med høyde, bredde og dybde [17]. Høyden og bredden indikerer antall piksler i inputbildet, mens dybden representerer antall kanaler. Et svart-hvitt-bilde består av én kanal som representerer gråtoneskalaen, mens et RGB-bilde består av tre kanaler, en for hver av fargene rød, grønn og blå. Transformasjoner foretas på alle kanalene i et inputbilde og resulterer i nye representasjoner, hvor antall kanaler, dybden, avhenger av antall filtre i laget [17]. Disse nye representasjonene av inputbildet kalles egenskapskart («feature maps») [16]. Et inputbilde med flere kanaler kan sees på som en stabel med matriser, der det utføres en konvolusjonsoperasjon på hver av matrisene som legges sammen til et resultat [16]. Et konvolusjonslag kan bestå av flere filtre som lærer spesifikke egenskaper av dataene. Jo flere filtre i et lag, jo mer komplekse og avanserte egenskaper kan nettverket lære seg fra treningseksemplene [17].

Filter

Et filter består av et område, eller vindu, med en gitt mengde verdier som ofte har størrelse 3×3 eller 5×5 [16]. Filtrene brukes til å endre inputdata for å fremheve egenskaper som kan være nyttig for korrekt klassifisering [45]. En konvolusjonsoperasjon beveger et filter en gitt avstand, kalt steglengde («stride»), over pikslene i inputbildet. Når filteret dekker et område, foretas det en transformasjon via en vektmatrise i filteret, kalt en kernel [17]. Hver verdi i vektmatrisen multipliseres med overlappende pikselverdi i inputbildet som summeres til en sluttverdi [45]. Denne sluttverdien kan sammenlignes med aktiveringsverdien i nevrale nettverk [45]. På denne måten glir filteret over hele inputbildet og ekstraherer overlappende

3D-områder som blir transformert til en endimensjonal vektor [17]. Dimensjonen til den endimensjonale vektoren representerer dybde og avhenger av antall filtre i foregående lag eller antall kanaler i inputlaget. Vektorene settes til slutt sammen til et 3D-outputegenskapskart. Verdiene i outputegenskapskartet inneholder informasjon om korresponderende posisjon i inputbildet, der venstre hjørne i outputegenskapskartet inneholder informasjon om venstre hjørne i inputegenskapskartet [17]. Figur 3.8 viser hvordan en konvolusjonsoperasjon transformerer et inputegenskapskart om til et outputegenskapskart.



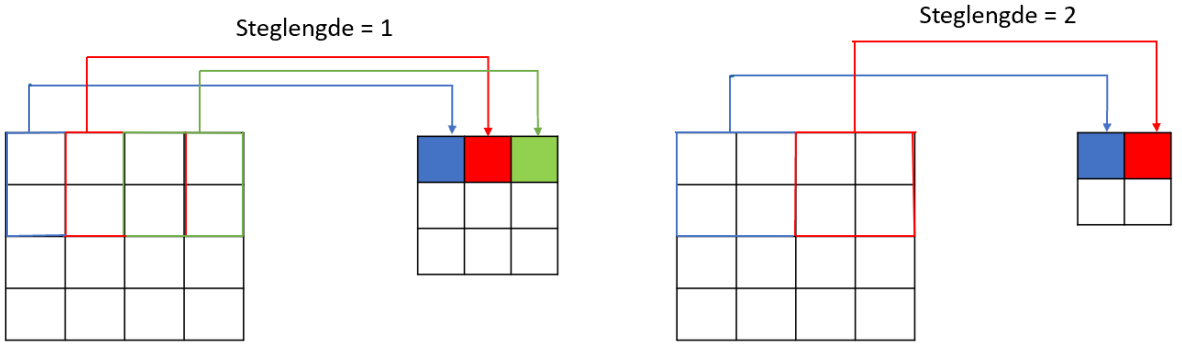
Figur 3.8: Figuren illustrerer hvordan en konvolusjon fungerer. Et filter beveger seg over 3×3 områder i et inputegenskapskart og transformerer områdene om til endimensjonale vektorer. Disse vektorene settes videre sammen til et outputegenskapskart. Figuren er tegnet med inspirasjon fra illustrasjon i boken Deep Learning with Python [17].

Steglengde og polstring

Outputbildets størrelse fra et konvolusjonslag er ofte forskjellig fra størrelsen på lagets inputbilde [17]. Måten et filter beveger seg på over inputdata, størrelsen på filteret og filterets oppførsel langs kantene, har innvirkning på outputegenskapskartet produsert [45].

Steglengde

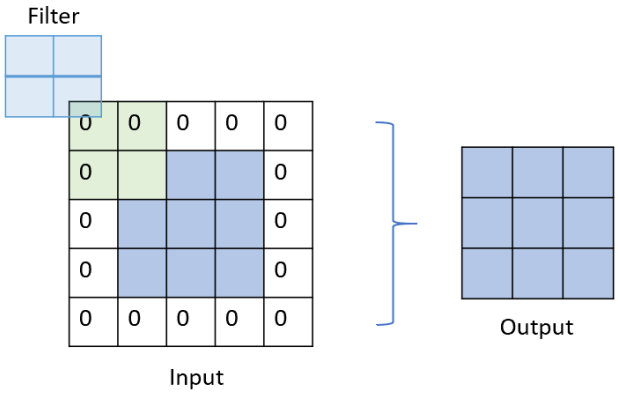
Steglengde avgjør hvor mange piksler filteret forflyttes med av gangen [45]. Steglengde større enn 1 fører ofte til et outputegenskapskart med redusert høyde og bredde. Figur 3.9 viser forskjellen på outputegenskapskart produsert med steglengde lik 1 og steglengde lik 2.



Figur 3.9: Figuren illustrerer forskjellen på et outputegenskapskart laget med steglengde lik 1 og steglengde lik 2. De omrissede fargene representerer et filter som beveger seg over inputbildet og gir en output representert med tilhørende farge i outputegenskapskartet.

Polstring

Inputbilder påført filtre med visse størrelser kan resultere i forminskede outputbilder [17]. For å få samme outputstørrelse som inputstørrelse kan teknikken polstring («padding») benyttes. Polstring handler om å legge til rader og kolonner bestående av nullverdier rundt inputbildet. På denne måten kan filteret påført, bevege seg over alle posisjoner som vil resultere i et like stort outputbilde som inputbilde, gitt at steglengden er lik 1. Ved å justere antall tillagte rader og kolonner, kan outputbildets størrelse manipuleres til å bli større, mindre, eller lik inputbildets dimensjoner [17]. Figur 3.10 viser et inputbilde med polstring slik at outputstørrelsen blir lik inputstørrelsen.



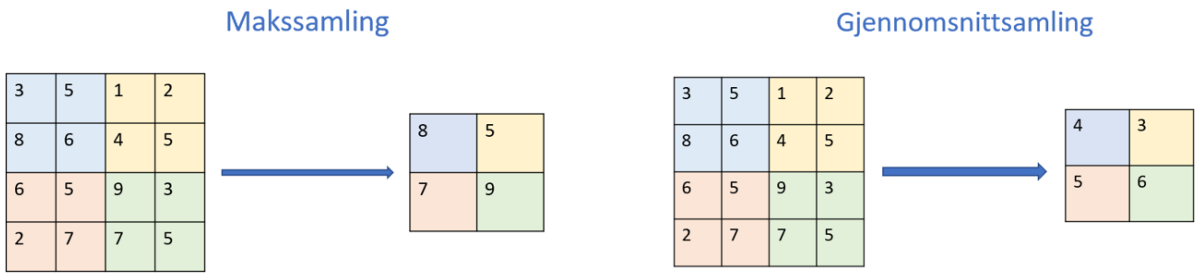
Figur 3.10: Figuren illustrerer polstring (markert med 0) påført et inputbilde som gir lik outputstørrelse når et filter beveges over inputbildet med steglengde lik 1.

3.6.2 Samlelag

Et konvolusjonslag i et konvolusjonsnettverk etterfølges ofte av et undersamplingslag i form av samleoperasjoner [16, 45]. Samlelag har som hensikt å redusere inputbildets størrelse ved å velge ut visse egenskaper. Dette resulterer i et mer effektivt og robust nettverk som er mindre

utsatt for overtilpasning [16]. Makssamling og gjennomsnittligsamling er de vanligste eksemplene på samleoperasjoner [16, 45]. Makssamling ekstraherer områder i inputegenskapskartet og gir som output den høyeste verdien i området. Gjennomsnittligsamling gir som output gjennomsnittet til det lokale området i inputbildet [17]. Små forandringer i det lokale området vil ikke påvirke resultatet fra en makssamlingsoperasjon i motsetning til resultatet fra en gjennomsnittsamlingsoperasjon, noe som gjør makssamlingslag mer robust mot støy i inputdata [16].

Samleoperasjonene fører ikke til nye trenbare parametere ettersom de bare ekstraherer verdier fra inputbildet, som øker beregningshastigheten til nettverk [45]. Figur 3.11 viser eksempel på makssamling og gjennomsnittssamling påført et bilde.



Figur 3.11: Figuren illustrerer et eksempel på makssamling (venstre) og et eksempel på gjennomsnittssamling (høyre) påført et bilde. Makssamling gir som output den høyeste verdien, mens gjennomsnittsamling gir som output gjennomsnittet til det lokale området i inputbildet. Figuren er illustrert med inspirasjon fra illustrasjon i boken Python Machine Learning [16].

3.6.3 Fullt koblet-lag

Et konvolusjonsnettverk avsluttes ofte med et fullt koblet-lag der er alle nodene i foregående lag er koblet sammen med alle nodene i nåværende lag. Ved å avslutte med et fullt koblet-lag bestående av en eller flere noder kan nettverket gi ut outputverdier som indikerer nettverkets prediksjon [16].

3.7 Semantisk segmentering

CNN-modeller er kjent for å finne gode representasjoner av inputdata, og brukes derfor mye til klassifiseringsoppgaver av bilder som helhet, men også til lokal strukturklassifisering [49]. I noen tilfeller er både klassen og posisjonen til en gitt piksel ønskelig å detektere. Dette gjelder spesielt ved bruk av medisinsk avbildning til kreftsvulstinntegning, hvor målet er å klassifisere og lokalisere kreftpiksler [50]. Semantisk segmentering handler om å tilordne hver piksel i et bilde en klasse, som beskriver objektet eller området pikselen befinner seg i [49]. Objekter i inputdata kan på denne måten segmenteres fra hverandre.

Ulike CNN-modeller har vist seg å gi gode resultater innenfor semantisk segmentering [49, 50]. Fullt konvolusjonsnettverk («Fully Convolutional Network», FCN) er et eksempel på et nettverk mye brukt til semantisk segmentering, der fullt koblet-lag byttes ut med konvolusjonslag [49]. Selv om FCN-modeller både lokaliserer og detekterer objekter i inputdata, gir de som output en grov fremstilling av objektstruktur med få detaljer [51]. Dette kommer i hovedsak av samlelagene i arkitekturen, som velger ut de viktigste egenskapene fra foregående lag og av den grunn fører til tapt posisjonsinformasjon [51]. Ved å legge til skip-koblinger («skip-connections») som kombinerer informasjon fra tidligere lag med nåværende lag, kan denne tapte informasjon gjenopprettes [52]. Et populært nevralt nettverk som baserer seg på fulle konvolusjonsnettverk, er U-Net. Ronneberger et al. [50] presenterte en variant av arkitekturen som har vist lovende resultater innenfor forskjellige biomedisinske segmenteringsoppgaver.

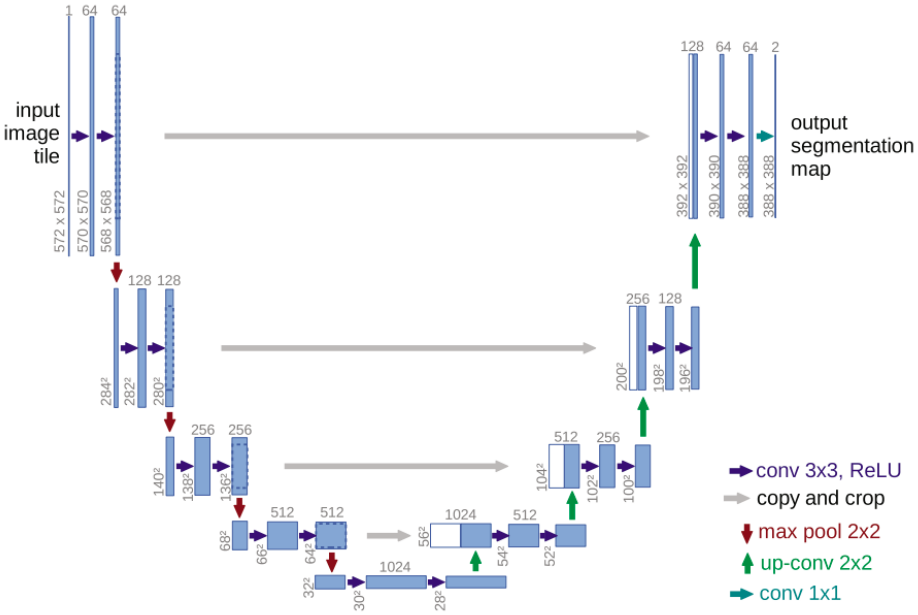
3.7.1 U-Net

U-Net-arkitekturen har form som en U og består i hovedsak av to symmetriske veier, sammentrekningsvei («contracting path») og ekspanderendevei («expansive path»). Sammentrekningsveien, også kalt nedsamplingsveien, inneholder blant annet makssamlingslag som reduserer den romlige dimensjonen til inputbildene [50]. Den ekspanderende veien, også kalt oppsamplingsvei, inneholder blant annet oppsamplingsoperatorer som øker oppløsningen til dataene i nettverket [50]. Oppsamplingsoperatorer kan være transponerte konvolusjonslag som ligner konvolusjonslag, men bruker transformasjoner for å øke den romlige dimensjonen i inputdata samtidig som konvolusjonens tilkoblingsmønster («connectivity pattern») beholdes [53]. Ved hjelp av skip-koblinger kombineres informasjon med høy oppløsning fra sammentrekningsveien med den ekspanderende veien, noe som hindrer strukturinformasjon i å gå tapt [50].

U-Net-arkitekturen er en form for koder-dekoder nettverk som trekker ut viktige egenskaper i koder-delen og gjenoppretter tapt informasjon om struktur og posisjon til objekter i dekoder-delen [51]. Nettverket kan på denne måten lære presis segmentering som er viktig i biomedisinsk analyse [50].

Figur 3.12 viser U-Net-arkitekturen presentert av Ronneberger et al. [50]. Sammentrekningsveien er avbildet til venstre i figuren. I denne delen av nettverket påføres to 3×3-konvolusjoner etterfulgt av aktiveringsfunksjonen ReLU flere ganger, indikert med blå piler. Videre påføres 2×2 makssamlingsoperasjoner med steglengde 2 (røde piler), som

reduserer den romlige dimensjonen til inputdataene og samtidig dobler antall egenskapskanaler for hvert nedsamplingssteg. De to veiene er koblet sammen med en flaskehals i bunnen av nettverket, som består av to konvolusjonslag. Den ekspanderende veien er avbildet til høyre i figuren. I denne veien foregår en oppsampling av egenskapskartet etterfulgt av 2×2-konvolusjoner som halverer antall egenskapskanaler, indikert med grønne piler i figuren. Veien består videre av skip-koblinger (grå piler) fra sammentrekningsveien og to 3×3-konvolusjoner som etterfølges av aktiveringsfunksjonen ReLU. Det siste steget i den ekspanderende veien inneholder en 1×1-konvolusjon for klassifisering [50].



Figur 3.12: Illustrasjon av U-Net-arkitekturen utviklet av Ronneberger et al. [50], gjengitt med tillatelse av forfatter Olaf Ronneberger. Boksene representerer egenskapskart som blir gitt som input og output i de ulike lagene, hvor antall kanaler er indikert på toppen av hver boks og størrelsen er indikert med tall på siden. Pilene representerer de ulike operasjonene som foretas, hvor fargen og tilhørende operasjon er gitt til høyre i figuren. De hvite boksene representerer kopierte egenskapskart fra sammentrekningsveien (til venstre) som blir sammenkoblet (vha. skip-koblinger, grå pil) med output i den ekspanderende veien (til høyre). Den siste gråblå pilen representerer 1×1-konvolusjon som klassifiserer pikslene i bildet.

3.8 Dataaugmentering

En utfordring blant CNN-modeller er mangel på tilstrekkelig treningsdata som kan føre til at modellen generaliserer dårlig til nye datasett [23]. Jo flere treningseksempler en maskinlæringsmodell kan trene på, jo mer nøyaktig kan modellen bli [27]. Som nevnt i delkapittel 3.5, kan komplekse modeller som trener på små datasett risikere å bli overtilpasset ved å lære for mange detaljer og støy fra treningsdataen, som kan føre til feilaktige klassifiseringer på usett data [17]. Store datasett er ofte en utfordring, spesielt innenfor feltet biomedisinske avbildning, hvor det både er dyrt og utfordrende å generere nye treningseksempler [24]. Personvern tilknyttet pasienter og deres tilhørende data gjør det i

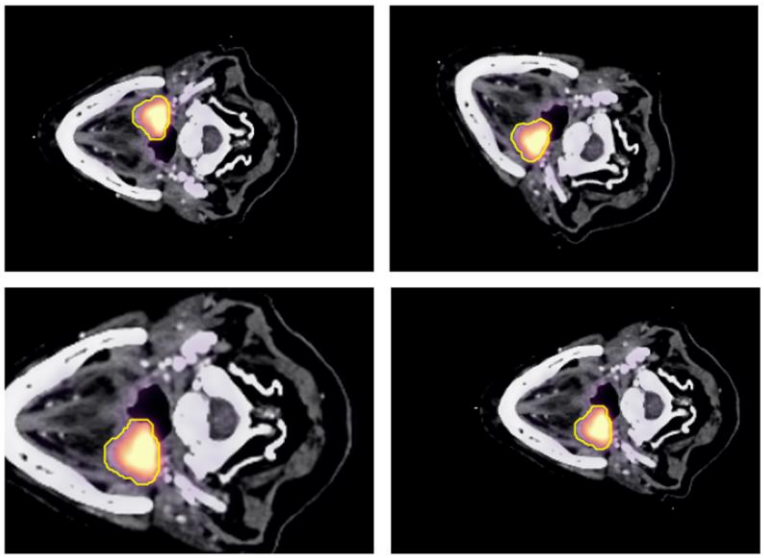
tillegg vanskelig å anskaffe bilder nettverket kan trene på [27]. En kjent metode som øker mengden treningsdata er dataaugmentering.

Som beskrevet i delkapittel 3.5.3 er dataaugmentering, også kalt bildeaugmentering, teknikker som genererer nye treningseksempler. Ved å påføre ulike transformasjoner på allerede eksisterende treningsdata, samtidig som forholdet til det sanne målet holdes konstant, skapes nye varianter av treningseksempler med nyttig informasjon, som øker variasjonen og dermed størrelsen på datasettet [17, 24, 29]. Bruken av dataaugmentering har i flere tidligere studier vist lovende resultater for forbedring av et nettverks ytelse [23, 26, 27]. Teknikkene brukes blant annet for å forhindre overtilpasning og for å håndtere ubalanserte klasser som ofte er en utfordring i medisinske datasett [30]. En dyplæringsmodell som trener på augmentert inputdata med nye varianter av treningseksempler vil kunne ignorere små forandringer i datasettet som gjør at modellen kan generalisere bedre, bli mer robust og vil kunne øke ytelsen [17, 24, 27].

Dataaugmenteringsteknikker dekker en rekke teknikker som benyttes til ulike oppgaver [27]. De vanligste augmenteringsmetodene faller under kategorien datavridning («data warping») hvor mengden data økes ved å endre den allerede eksisterende datamengden [27]. Populære og effektive augmenteringsmetoder innenfor denne kategorien er tradisjonelle affine transformasjonsmetoder og punkt- og filteroperasjoner [23, 27].

3.8.1 Affine transformasjoner

Affine transformasjoner er en klasse av geometriske operasjoner som transformerer inputdata ved å endre på geometrien i bildet [48]. Transformasjonene bevarer parallelle- og rette linjer i originalbildet, samt avstandsforholdet mellom to punkter. Ved å modifisere koordinatene til inputbildets pikselverdier genereres det et nytt bilde, hvor pikslene har en ny posisjon. På denne måten holdes intensitetsverdiene til pikslene konstant, mens posisjonen endres [48]. De manglende pikslene i bildet etter affine transformasjoner er påført, blir fylt med den konstante verdien 0 [54]. Figur 3.13 viser eksempler på et PET/CT-bilde som har blitt påført de affine transformasjonene flipp, rotasjon, zoom og forskyvning.

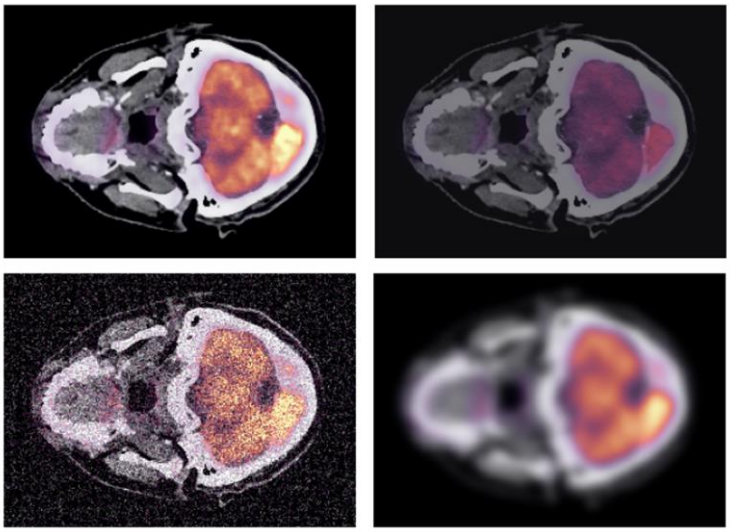


Figur 3.13: Figuren illustrerer fire PET/CT-bilder med kreftsvulstinntegning (gult omriss), som har blitt påført affine transformasjoner. Øverst til venstre er bildet speilet om x-aksen. Øverst til høyre er bildet rotert en gitt vinkel. Nederst til venstre er bildet zoomet inn, mens bildet nederst til høyre er forskjøvet.

3.8.2 Punkt- og filteroperasjoner

Punkt- og filteroperasjoner er en samling av operasjoner som generer nye treningseksempler ved å utføre transformasjoner og modifikasjoner på pikselverdiene i inputdata uten å endre størrelse, geometri eller den lokale strukturen [48]. Eksempler på slike metoder er blant annet kontrastendring, justering av lysstyrke, tillagt støy og endring av uskarphet. De tre førstnevnte teknikkene er punktoperasjoner, mens den sistnevnte teknikken uskarphet, er en filteroperasjon [48].

I punktoperasjoner avhenger hver piksel i det transformerte bildet kun av pikselen i samme posisjon fra originalbildet [48]. En punktoperasjon transformerer pikselintensitetsverdiene i et inputbilde til nye representasjoner ved hjelp av en funksjon f. Ligningen $b = f(a)$ er et eksempel på en transformasjon hvor pikselintensitetene, a, tilordnes nye verdier, b, ved hjelp av funksjonen f. Filteroperasjoner skiller seg fra punktoperasjoner ved at de bruker mer enn én piksel fra originalbildet for å danne hver nye pikselverdi i det transformerte bildet [48]. Figur 3.14 viser eksempler på punkt- og filteroperasjoner påført et PET/CT-bilde.



Figur 3.14: Figuren viser punkt- og filteroperasjoner påført et PET/CT-bilde. Originalbildet uten augmenteringsteknikker er avbildet øverst til venstre. Øverst til høyre er lysstyrken i bildet justert ned (punktoperasjon). Nede til venstre er det lagt på støy (punktoperasjon). Nede til høyre er skarpheten i bildet justert ned (filteroperasjon).

3.8.3 Elastisk deformasjon

Elastisk deformasjon er en annen augmenteringsmetode som ved å endre formen, volum eller lengden, deformerer objektet i bildet. I et kontinuerlig legeme vil en deformasjon forekomme hvis ytre krefter påføres. Gjenopprettes formen etter påførte krefter, er legemet elastisk [55]. I studien til Ronneberger et al. [50] ble teknikken elastisk deformasjon benyttet for å øke mengden treningseksempler blant bilder tatt gjennom mikroskop. Ifølge Ronneberger et al. [50] er deformasjoner den vanligste årsaken til variasjon mellom ulike vev i kroppen, og elastisk deformasjon er derfor en effektiv teknikk som skaper realistiske treningseksempler. Elastiske deformasjoner påført bilder av hode- og halsregion kan imidlertid generere anatomisk ukorrekte treningseksempler, ettersom regionen ikke er spesielt elastisk. Augmenteringsteknikken har også vist å være tidkrevende, samt vanskelig å implementere [56]. Elastisk deformasjon benyttes derfor ikke som augmenteringsteknikk i denne masteroppgaven.

3.8.4 Generative Adversarial Networks

Syntetisk dataaugmentering er en nyere form for dataaugmentering hvor en genererende modell lærer datadistribusjonen i et treningssett og basert på dette fremstiller syntetiske nye treningseksempler [28]. *Generative Adversarial Networks* (GANs) er en slik modell som genererer nye treningseksempler uten veiledning, ved bruk av min-maks strategien [23, 27, 28]. Strategien går ut på å bruke to nettverk, genererende nettverk og diskriminerende nettverk, hvor det genererende nettverket har som oppgave å lage troverdige bilder, mens det diskriminerende nettverket har som oppgave å skille de genererte treningseksemplene fra de

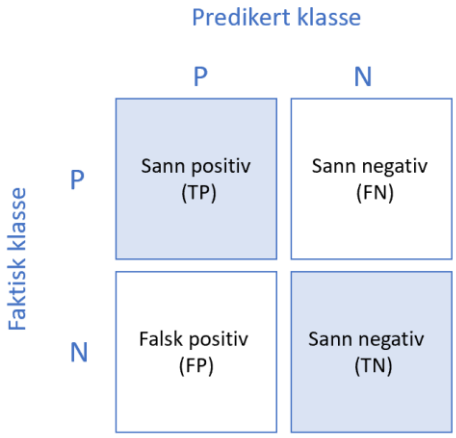
originale (sanne) bildene i datasettet [23, 27]. En kostverdi brukes for å angi graden av diskriminering mellom de genererte og sanne bildene. Det genererende nettverket trenes for å lure det diskriminerende nettverket, og dermed minimere kostverdien, mens det diskriminerende nettverket trenes for å skille bildene, og dermed maksimere kostverdien [23]. Selv om GANs, har vist å gi gode resultater er det mye usikkerhet tilknyttet evaluering av modellene [23]. Teknikken er i tillegg tidkrevende, vanskelig å implementere og kan i enkelte tilfeller generere treningseksempler som er for like og dermed gjør at modellens generaliseringsevne ikke øker [56]. GANs som augmenteringsteknikk benyttes derfor ikke i denne masteroppgaven.

3.9 Ytelsesmål

For å kontrollere om en maskinlæringsmodell oppnår det den skal lære må modellen overvåkes med et mål som beskriver ytelsen. Valg av målemetode avhenger av oppgaven maskinlæringsmodellen står ovenfor. En av de vanligste målene brukt i balanserte klassifiseringsoppgaver er nøyaktighet («accuracy») [17], som gir generell informasjon om hvor mange treningseksempler som er klassifisert rett [16]. Er klassene derimot ubalanserte hjelper det ikke å måle antall korrekte klassifiseringer, ettersom målemetoden nøyaktighet vil få en høy score dersom modellen klassifiserer majoritetsklassen korrekt, men minoritetsklassen feil. I slike klassifiseringsoppgaver er det mer vanlig å bruke målemetoder som tar med informasjon angående andelen av klasser representert i treningseksemplene [17]. Flere av disse ytelsesmålene baserer seg på en tabell, kalt forvirringsmatrise, som visualiserer prestasjonen til modellen [16].

3.9.1 Forvirringsmatrise

Figur 3.15 viser forvirringsmatrisen, som består av fire ruter der antall sanne positive («true positive», TP), sanne negative («true negative», TN), falske positive («false positive», FP) og antall falske negative («false negative», FN) prediksjoner er oppgitt [16].



Figur 3.15: Figuren illustrerer en forvirringsmatrise som brukes for å visualisere prestasjonen til en modell. Radene representerer faktisk klasse (sann klasse) og kolonnene representerer predikert klasse av modellen som klassifiserer treningseksempler. Figuren er illustrert med inspirasjon fra boken Python Machine Learning [16].

I sammenheng med kreftsvulstinntegning representerer TP antall kreftpiksler som modellen har klassifisert korrekt, TN antall ikke-kreftpiksler modellen har klassifisert korrekt, FP antall piksler modellen har feilaktig klassifisert som kreft og FN antall piksler som modellen feilaktig har klassifisert som piksler uten kreft. Informasjonen forvirringsmatrisen gir kan brukes for å beregne ulike målemetoder som måler ytelsen til modellen [16].

3.9.2 Overlappsmål

Målene falsk positiv rate («false Positive Rate», FPR), og sann positiv rate, («True Positive Rate», TPR), vist i henholdsvis ligning 3.8 og 3.9, tar høyde for andelen eksempler som er representert i treningsdatasettet og egner seg derfor godt til ubalanserte klassifiseringsoppgaver [16].

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} \tag{3.8}$$

$$TPR = \frac{TP}{P} = \frac{TP}{FN + TP} \tag{3.9}$$

FPR gir i sammenheng med kreftsvulstinntegning et mål på andelen treningseksempler som ble feilaktig klassifisert som kreft i forhold til totalen av piksler uten kreft [16]. TPR, også kalt sensitivitet, gir et mål på andelen korrekte klassifiserte kreftpiksler i forhold til den totale andelen av kreftpiksler.

Målemetoden presisjon («precision», PRE) vist i ligning 3.10 gir et mål på andelen piksler korrekt klassifisert som kreft i forhold til totalen av piksler predikert som kreft.

$$PRE = \frac{TP}{TP\ +\ FP} \tag{3.10}$$

En maskinlæringsmodell som bruker sensitivitet som målemetode jobber mot å detektere så mange kreftsvulster som mulig og minimerer dermed sjansen for å klassifisere en pasient som falsk negativ, men øker sjansen for å klassifisere en pasient som falsk positiv. Brukes målemetoden presisjon kan maskinlæringsmodellen risikere å klassifisere en pasient som falsk negativ [16]. En vanlig målemetode som muliggjør vekting av presisjon og sensitivitet er F_β gitt i ligning 3.11 [57].

$$F_\beta = \frac{(\beta^2 + 1) * PRE * TPR}{\beta^2 * PRE + TPR} \tag{3.11}$$

For å balansere ut optimaliseringen av presisjon og sensitivitet kan β setter lik 1 som gir ytelsesmålet f1-score, også kalt Dice-score, vist i ligning 3.12 [16].

$$Dice\text{-}score = 2 * \frac{PRE\ *\ TPR}{PRE\ +\ TPR} = \frac{2TP}{2TP\ +\ FN\ +\ FN} \tag{3.12}$$

Dice-koeffisienten (Dice-score) kalles i mange tilfeller overlappsindeks som gir et mål på grad av overlapp mellom sann inntegning og predikert inntegning. Overlappsmålet Dice-score er ifølge Guo et al. [8] gullstandarden innenfor segmenteringsoppgaver, men kan i noen tilfeller gi misvisende resultater. En modell som segmenterer tumorvolum og påvirkede lymfeknuter i medisinske avbildninger vil oppnå en høy Dice-score hvis modellen korrekt klassifiserer tumorvolumet, men feilaktig klassifiserer små påvirkede lymfeknuter, ettersom ytelsesmålet baserer seg på størrelsen til det ønskede volumet og ikke tar hensyn til avstanden mellom inntegningene [8]. Det kan derfor være gunstig å bruke et avstandsbasert mål i sammenheng med overlappsmålet for å korrekt evaluere modellens ytelse.

3.9.3 Avstandsmål

Et avstandsbasert mål som ofte brukes i sammenheng med segmenteringsoppgaver er Hausdorff avstand («Hausdorff distance»), gitt i ligning 3.13. Hausdorff avstand gir et mål på den maksimale avstanden mellom overflate voxler til den predikerte inntegningen, P, og den sanne inntegningen G [57].

$$HD(G,P) = max\big(h(G,P), h(P,G)\big), \qquad h(G,P)\ = \max_{y \in G}\min_{\rho \in P}\|g - \rho\| \tag{3.13}$$

Her indikerer $\|g - \rho\|$ den euklidiske avstanden mellom overflatepunkt g og p [11]. Ettersom Hausdorff avstandsmålet er sensitiv for ekstreme observasjoner, utliggere («outliers»), kan 95.

persentil Hausdorff avstand (HD_{95}) benyttes, som ekskluderer slike utliggere [8, 11, 58]. HD_{95} gir et mål på den mest alvorlige inntegningsfeilen utført av modellen, ettersom avstandsmålet måler den lengste avstanden mellom den sanne og den predikerte inntegningen [21].

Et annet avstandsbasert mål er median overflateavstand («Median Surface Distance», MSD). MSD gir et mål på median-avstand mellom den predikerte inntegningen og den sanne inntegningen og gir dermed et mål på den totale inntegningsfeilen [21]. En lav MSD og en lav HD_{95} indikerer som oftest en god segmentering. MSD og HD_{95} gir relevant komplementær informasjon angående modellens prediksjoner og er derfor gunstig å benytte for å evaluere modellens totale segmenteringsytelse.

Kapittel 4: Metode

4.1 Datasettet

Datasettet som ble benyttet i denne masteroppgaven består av tredimensjonale PET/CT-bilder fra pasienter diagnostisert med hode- og halskreft av typen plateepitelkreft i munnhulen («oral cavity»), munnsvelg («oropharynx»), strupesvelg («hypopharynx») og strupehodet («larynx») [21]. Dataene er hentet fra Oslo universitetssykehus (OUS), Radiumhospitalet, hvor 400 hode- og halskreftpasienter gjennomførte planlagt radioterapibehandling i perioden fra januar 2007 til desember 2013 [59]. Ut ifra ulike kriterier oppnådde 197 pasienter kravene for videre analyse, og utgjør datasettet dyplæringsmodellene i denne oppgaven har trent på [11].

De innsamlede FDG-PET/CT-dataene består av tverrsnitt tatt med PET/CT-skanneren Siemens Biograph 16, fra hodeskallen og ned til midbrystparti [11, 59]. Videre ble kontrastforsterket CT optimalisert for nakkeregionen brukt for blant annet attenueringskorreksjon og bildetolkning [59]. Tumorer (GTV-T) og påvirkede lymfeknuter (GTV-N) ble basert på FDG-opptaket, manuelt tegnet inn av en erfaren spesialist i nukleærmedisin. Inntegningene ble videre prosessert av en til to onkologer som baserte seg på klinisk informasjon og kontrastforsterket CT. Til slutt ble arbeidet godkjent av en av flere senior onkologer [21, 59]. I bildene som inneholdt flere inntegninger på tvers av spesialistene, ble unionen av segmenteringen valgt som sann inntegning [11, 21]. Datasettet inneholder to klasser, friskt vev og ikke-friskt vev, hvor pikslene som inneholder inntegnede tumorer eller påvirkede lymfeknuter tilhører klassen ikke-friskt vev [11, 60].

4.1.1 Trenings-, validerings- og testsett

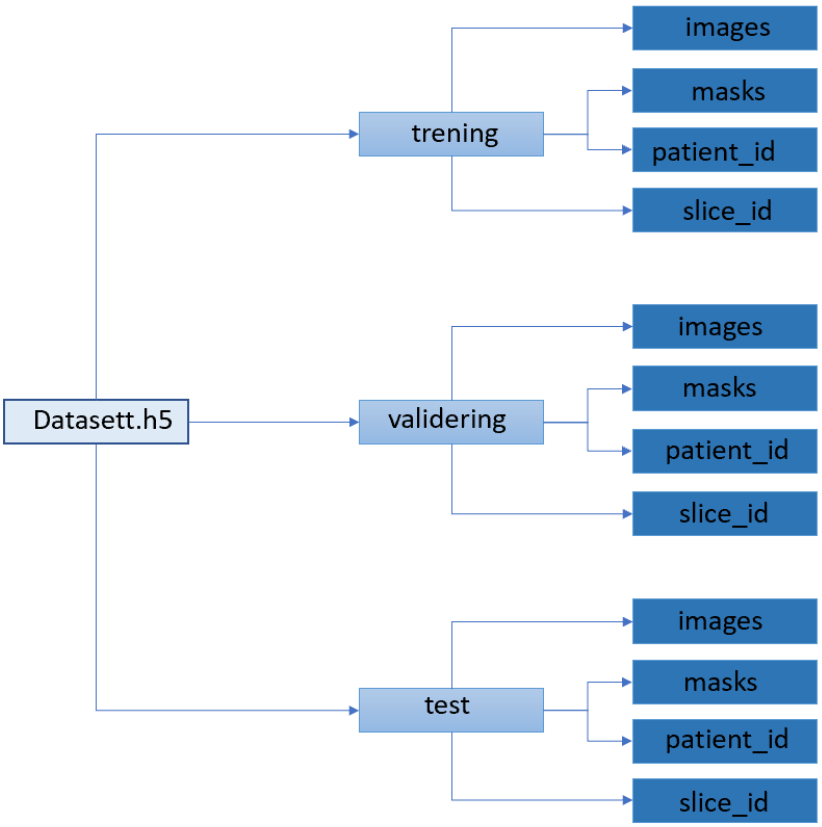
Før datasettet kunne brukes for å trene modellene ble det delt opp i et treningssett, valideringssett og testsett med tilhørende sanne inntegninger. De medisinske 3D-avbildningene til hver pasient er i datasettet delt opp i 2D-tverrsnitt som inneholder to kanaler, PET og kontrastforsterket CT, der dimensjonene til inputbildene er 191×256×2 [22]. Datasettet tilsvarer settet brukt i [21] hvor datasettet ble delt etter tumorstadium definert i TNM-systemet [61]. Delingen sørget for at hvert av de tre settene inneholdt et representativt utvalg av pasienter med forskjellig tumorstadium [21].

De 197 pasientene i datasettet ble delt slik at det var 142 pasienter i treningssettet, 15 pasienter i valideringssettet og 40 pasienter i testsettet [21]. Flere detaljer angående datasettet er gitt i [21, 60].

4.1.2 HDF5-format

Datasettet ble lagret i HDF5-format, *Hierarchical Data Format version 5*, som er en standard lagringsmetode for store datasett og muliggjør for hierarkisk organisering av data [62]. HDF5-formatet består av tre hovedelementer: datasett, grupper og attributter. Datasett kan være array-lignende deler av data lagret på disk, grupper er hierarkiske beholdere som inneholder datasett eller andre grupper, mens attributter inneholder tilleggsinformasjon om gruppene og datasettet [62].

Datasettet i denne oppgaven ble organisert i tre grupper: trening, validering og test. Disse gruppene inneholder alle fire datasett hver: *images*, *masks*, *patient_id* og *slice_id* [60]. Strukturen til datasettet lagret i HDF5-format er illustrert med Figur 4.1.



Figur 4.1: Illustrasjon av strukturen til datasettet lagret i HDF5-format. Filen består av de tre gruppene trening, validering og test, som videre inneholder fire datasett hver: images, masks, patient_id og slice_id.

Datasettet *images* består av 2D-bildetverrsnitt fra PET/CT-avbildningen av hode- og halskreftpasientene. Datasettet *masks* består av kreftsvulstinntegningene utført av spesialister. Disse brukes som sann inntegning under trening av dyplæringsmodellene. De to siste datasettene, *patient_id* og *slice_id* innad i de tre gruppene, inneholder henholdsvis

informasjon om identifikasjon til pasientene og identifikasjon til de ulike tverrsnittene. Tabell 4.1 gir en videre beskrivelse av HDF5-filen.

Tabell 4.1: Tabellen gir en beskrivelse av strukturen til datasettfilen, datasett.h5, som inneholder datasettene med tilhørende størrelse, datatype og innhold, listet i tabellen.

Datasett	Størrelse	Datatype	Innhold
images	[n_images, x, y, c]	float32	Inputbildene
masks	[n_images, x, y, m]	float32	Segmenteringsbildene
patient_id	[n_images]	unit16	Pasient-ID-nummer
slice_id	[n_images]	unit16	Tverrsnitt-ID til pasientene

I Tabell 4.1 representerer *n_images* antall bilder i datasettet, *x* antall piksler langs *x*-aksen i bildet, *y* antall piksler langs *y*-aksen i bildet, *c* antall kanaler og *m* antall inntegninger. Ettersom det er en binær segmenteringsoppgave er antall inntegninger, *m*, lik 1 [60].

4.2 Maastro-datasett

Dyplæringsmodellen som oppnådde høyest segmenteringsytelse i denne masteroppgaven ble videre testet på Maastro-datasettet. Maastro-datasettet består av PET/CT-bilder til 114 pasienter behandlet ved Maastro Clinic i Nederland. Pasientene har samme typer diagnoser som i OUS-treningsdatasettet og har mottatt tilsvarende type behandling. Tumorvolum (GTV-T) og påvirkede lymfeknuter (GTV-N) er tegnet inn i PET/CT-bildene slik som i treningseksemplene [63].

4.3 Rammeverk og programvare

4.3.1 Deoxys

Rammeverket *deoxys* utviklet av Bao Ngoc Huynh [22] ble i denne masteroppgaven brukt for å definere og trene dyplæringsmodeller. *Deoxys* er et Keras[1]-basert rammeverk som muliggjør bruk av dyp læring for tumorsegmentering i medisinske bilder. Ifølge Huynh [22] kan *deoxys*-brukere både lage og trene konvolusjonsnettverk på bildedatasett, samt visualisere ytelsen og prediksjonen til modellen [22]. Ved å bruke JSON (JavaScript Object Notation)-konfigurasjonsfiler, kan brukere av rammeverket definere ulike CNN-arkitekturer. Brukere kan blant annet velge komponenter som tapsfunksjon, ulike lag i nettverket, aktiveringsfunksjon og ytelsesmål for det definerte nettverket. Den tilhørende koden og flere

[1] https://keras.io

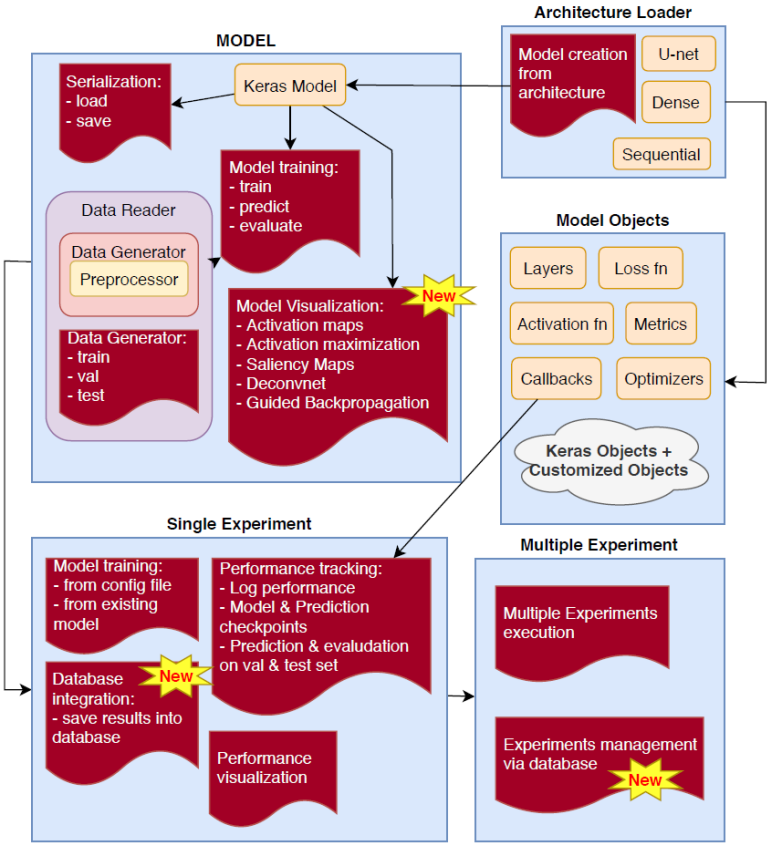
detaljer for rammeverket *deoxys* kan finnes på kodeplattformen GitHub: GitHub - huynhngoc/deoxys.

4.3.1.1 Keras og TensorFlow

Keras er et brukervennlig bibliotek og dyplæringsrammeverk for Python som *deoxys* er basert på. I Keras kan nesten hvilken som helst dyplæringsmodell defineres og trenes [17]. Keras er bygget på toppen av andre plattformer, blant annet TensorFlow [64] utviklet av Google, som muliggjør for multidimensjonale array-operasjoner brukt i dyp læring [17]. TensorFlow gjør i tillegg at Keras kan kjøre på både GPU og CPU [17]. For å kunne ta i bruk rammeverket *deoxys* må brukeren ha i minimum versjonene Python 3.7[2] og Keras 2.3.0 [22].

4.3.1.2 Komponenter

Rammeverket *deoxys* består i hovedsak av fire komponenter: dataleser, Keras modell *wrapper*, arkitektur innlastningsmoduler og eksperiment moduler [22]. Figur 4.2 viser en oversikt av strukturen til rammeverket *deoxys* med komponentene og tilgjengelige funksjoner.



Figur 4.2: Oversikt over strukturen til rammeverket deoxys utviklet av Bao Ngoc Huynh [22], med komponentene og tilgjengelige funksjoner. Gjengitt med tillatelse av Bao Ngoc Huynh.

[2] http://python.org

4.3.1.3 Dataleser

Dataleseren har i prinsippet tre funksjoner. Først leses bildedata inn fra HDF5-filen, videre splittes de innleste dataene i trenings- validerings- og testsett. Til slutt gir dataleseren bestemte små deler («batches») av dataene til modellen som skal trenes, valideres og testes [22]. Medisinske datasett, som *deoxys* er beregnet på å håndtere, er ofte store i størrelse og passer derfor ikke alltid inn i det tilgjengelige minnet. Komponenten splitter dataene i mindre mengder for å redusere sjansen for at minnet går fullt [22]. Dataleseren er i Figur 4.2 vist i boksen *Model* med navn *Data Reader*.

4.3.1.4 Keras modell *wrapper*

Komponenten Keras modell *wrapper* muliggjør bruken av Keras-metoder for trening og testing av dyplæringsmodeller. I tillegg inneholder komponenten teknikker som gjør det mulig å lagre og laste modeller til og fra disken. I Figur 4.2 er komponenten vist med navn *Keras Model* i boksen kalt *Model*.

4.3.1.5 Arkitektur innlastningsmoduler

I disse modulene settes dyplæringsmodellen sammen ut ifra informasjonen gitt i JSON-konfigurasjonsfiler [22]. Elementer som inngår er blant annet tapsfunksjon, ytelsesmål og aktiveringsfunksjon, eller egendefinerte elementer fra brukeren av rammeverket [22]. Arkitektur innlastningsmoduler inneholder også forhåndsdefinerte arkitekturer til dyplæringsmodellene. Eksempler på disse arkitekturene er blant annet sekvens arkitektur, som er den enkleste formen for CNNs og består av stabler med lag, eller U-Net-arkitektur, en litt mer avansert arkitektur introdusert i delkapittelet 3.7.1 [22]. Arkitektur innlastningsmoduler representerer boksen *Architecture Loader* og boksen *Model Objects* vist i Figur 4.2.

4.3.1.6 Eksperiment moduler

Eksperiment modulene brukes for å trene ulike dyplæringsmodeller med forskjellig arkitekturer og konfigurerbare variabler i nettverket (hyperparametere) [22]. Prestasjonen til modellen blir loggført etter hvert som modellen trenes, som muliggjør visualisering av modellens ytelse og prediksjoner av kreftsvulstinntegning på de medisinske dataene. Ved å loggføre etter hver gjennomkjøring, kan den beste modellen velges ut til videre evaluering på testsett [22]. I Figur 4.2 representerer eksperiment moduler boksene med navn *Single Experiment* og *Multiple Experiments*.

4.3.1.7 Database system (DBMS)

Rammeverket *deoxys* inneholder i tillegg et databasesystem («Database management system») som brukes for å håndtere eksperimenter [22]. Eksperimentene kan inneholde brukerdefinerte navn og beskrivelser, samt konfigurasjonsinformasjon om nettverkets arkitektur og parametere. Databasesystemet samler og organiserer data, som blant annet lagrede prediksjoner, ytelseslogg og lagrede modeller. På denne måten kan brukere analysere flere eksperimenter på en enkel og rask måte [22].

4.3.2 Regneklyngen Orion

Modellene brukt i denne masteroppgaven for å utføre kreftsvulstinntegninger er konvolusjonsnettverk. Konvolusjonsnettverk som utfører konvolusjonsoperasjoner på medisinsk inputdata krever mye minne og stor beregningskraft [17]. Eksperimentene har derfor blitt utført ved å bruke den eksterne enheten *Orion Compute Cluster*[3], som er en tungregningsklynge med åpen kilde infrastruktur [65]. Norges miljø- og biovitenskapelige universitet er vert for regneklyngen og den driftes av forskningsgruppen CIGENE, *Center of Interactive Genetics* [66].

Orion er ifølge CIGENE en SLURM-basert LINUX klynge med 580 CPU-er, 7 TB RAM og 550 TB lagringskapasitet [66]. Regneklyngen har i tillegg 4 GPU-servere med tre grafikkort hver, hvor hvert grafikkort har GPU-minne på 256 GB [67]. Den tilgjengelige store lagringsplassen og den høye prosesseringsevnen gjør at regneklyngen kan hjelpe brukere til å kjøre både småskala- og storskala eksperimenter [65].

For å koble til den eksterne Orion klyngen kreves det programvare som gir sikker tilkobling, en SSH-klient (Secure-Shell) [68]. Programvaren MobaXterm[4] ble i denne masteroppgaven benyttet for å koble til regneklyngen. MobaXterm inneholder funksjoner for å overføre og redigere filer og sende eksperimenter til regneklyngen Orion.

4.3.3 Bruk av *deoxys* og Orion

Konfigurasjonsfilene til eksperimentene kjørt i denne masteroppgaven er laget på plattformen Spyder[5] med programmeringsspråket Python versjon 3.7.9. Her ble rammeverket *deoxys* importert og brukt for å definere dyplæringsmodellene som ble trent. Ved bruk av *deoxys* ble konfigureringsfiler i JSON-filformat laget, hvor nødvendig preprosessering, arkitektur og

[3] NMBU Orion Compute Cluster — NMBU Orion user support documentation (nmbu-orion-support.readthedocs.io)
[4] https://mobaxterm.mobatek.net/
[5] Home — Spyder IDE (spyder-ide.org)

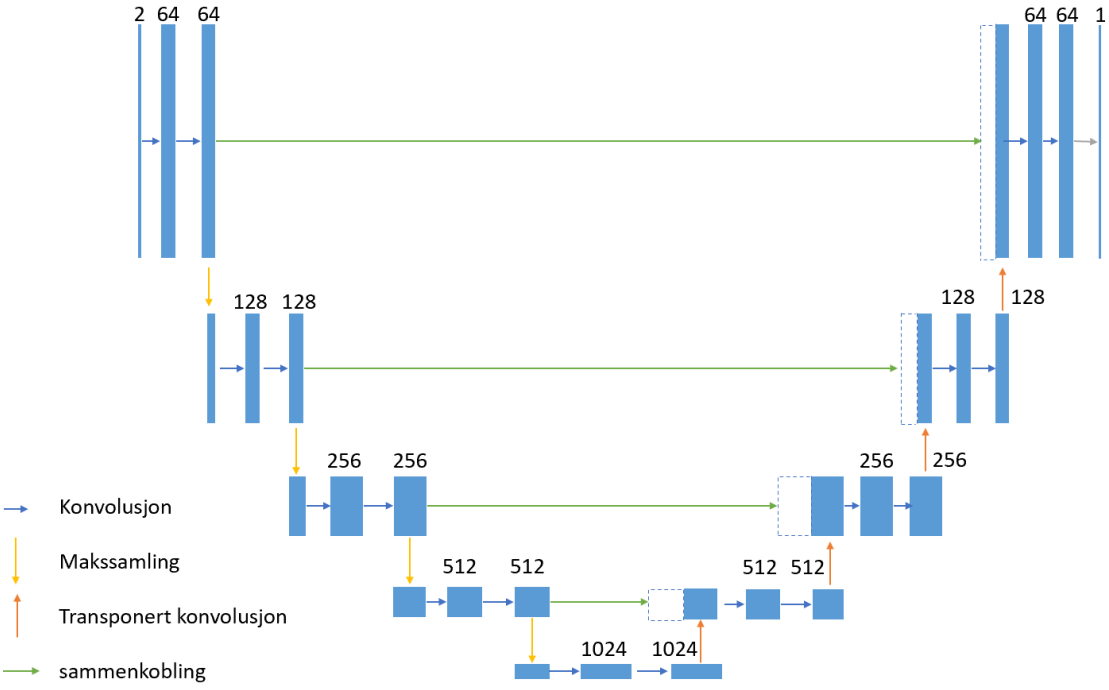
ulike parametere til dyplæringsmodellene ble definert. JSON-filene ble videre lastet opp til kodeplattformen GitHub[6], for så å bli lastet ned til programvaren MobaXterm. Her ble modellene sendt til regneklyngen Orion og kjørt på tilgjengelige regnenoder.

4.4 Modellene

Totalt 65 eksperimentmodeller med et konvolusjonsnettverk med 2D U-Net-arkitektur, ble i denne masteroppgaven trent på treningsdatasett påført ulike kombinasjoner av augmenteringsteknikknivåer. Eksperimentmodellen som oppnådde høyest ytelsesmål ble videre evaluert på testsett, og deretter sammenlignet med 2D U-Net-modellen, baselinemodellen, undersøkt i studiene til Moe et al. [21], Groendahl et al. [11] og Huynh [22], hvor augmenteringsteknikker ikke ble benyttet. Målet var å finne en kombinasjon av augmenteringsteknikker som kunne forbedre segmenteringsytelsen i forhold til baselinemodellen. Augmenteringsteknikkene benyttet i de ulike modellene blir introdusert i delkapittel 4.6.

4.4.1 U-Net-arkitekturen

Den U-formede arkitekturen til konvolusjonsnettverket brukt i denne masteroppgaven består av veiene sammentrekningsvei og ekspanderendevei, som beskrevet i delkapittel 3.7.1. Figur 4.3 viser U-Net-arkitekturen benyttet i baselinemodellen og eksperimentmodellene.



Figur 4.3: Illustrasjon av U-Net-arkitekturen benyttet i eksperimentmodellene og i baselinemodellen. De blå boksene representerer egenskapskart som blir gitt som input og output i de ulike lagene. Antall kanaler er indikert på toppen av hver

[6] https://github.com

boks. Pilene representerer de ulike operasjonene som foretas på egenskapskartene, der fargen og tilhørende operasjon er gitt til venstre i figuren. De stiplede boksene representerer kopierte egenskapskart fra sammentrekningsveien (til venstre) som blir sammenkoblet (vha. skip-koblinger, grønn pil) med output i den ekspanderende veien (til høyre). Den siste gråblå pilen representerer 1×1-konvolusjon som klassifiserer pikslene i bildet.

4.4.1.1 Strukturen til U-Net-modellene

U-Net-arkitekturen til baselinemodellen og eksperimentmodellene skiller seg fra den opprinnelige U-Net-arkitekturen presentert av Ronneberger et al. [50], beskrevet i delkapittel 3.7.1. I baselinemodellen og eksperimentmodellene ble batchnormalisering lagt til etter hver ReLU, noe det opprinnelige nettverket ikke benyttet [21]. Batchnormalisering brukes for å normalisere data når gjennomsnittet og variansen varierer under trening av det nevrale nettverket [17]. Den opprinnelige arkitekturen brukte oppsamplingslag med 2×2-konvolusjoner, i motsetning til 3×3 som ble benyttet i baselinemodellen og eksperimentmodellenes arkitektur [21]. I den opprinnelige U-Net-arkitekturen benyttes det heller ikke polstring, som fører til at kantpiksler forsvinner i konvolusjonslagene. Inputbildene til makssamlingslagene ble derfor beskjært før de ble koblet sammen med tilhørende output i den ekspanderende veien [50].

Vedlagt i Vedlegg A ligger Tabell A.1, Tabell A.2, Tabell A.3 og Tabell A.4 som gir en oversikt over strukturen i U-Net-arkitekturen benyttet i eksperimentmodellene og baselinemodellen. Tabell A.1 gir en oversikt over sammentrekningsveien, også kalt nedsamplingsveien, som består av inputlag, konvolusjonslag, batchnormaliseringslag og makssamlingslag, der størrelsen til inputdataen halveres [22]. Videre i nettverket består flaskehalsen av to konvolusjonslag og to batchnormaliseringslag som er vist i Tabell A.2. Den andre veien i U-Net-arkitekturen, ekspanderende vei, også kalt oppsamplingsveien, består av konvolusjonslag, transponerte konvolusjonslag, sammenkoblinger og batchnormaliseringslag vist i Tabell A.3 og Tabell A.4. De transponerte konvolusjonslagene dobler størrelsen til inputbildene [22].

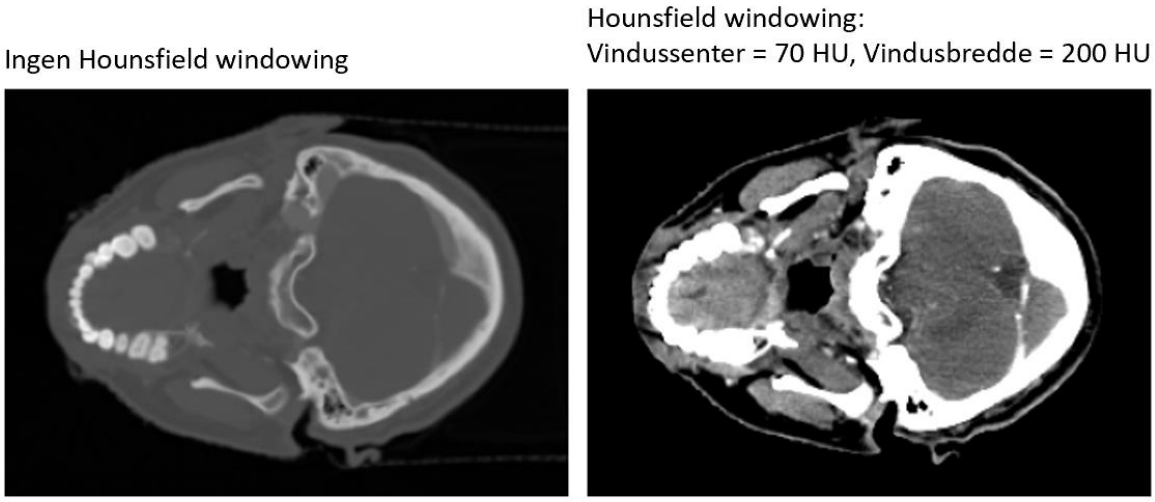
4.4.1.2 Parametere

Aktiveringsfunksjonene ReLU og Sigmoid, beskrevet i delkapittel 3.3.3, ble brukt i nettverket til eksperimentmodellene og baselinemodellen. ReLU er brukt i nettverkenes konvolusjonslag med unntak av det siste laget, hvor aktiveringsfunksjonen Sigmoid er brukt for å avgjøre om pikslene er av type friskt-vev eller ikke. Tapsfunksjonen som ble brukt i baselinemodellen og eksperimentmodellene kalles DiceLoss, beskrevet i delkapittel 3.3.5. DiceLoss ble definert av Milleatri et al. [46] og er spesialisert for segmenteringsoppgaver.

Videre ble Adam (introdusert i delkapittel 3.3.6) brukt som optimaliseringsteknikk med en læringsrate på 10^{-4} i eksperimentmodellene og i baselinemodellen benyttet i studien til Moe et al. [21] og Huynh [22]. I studien til Groendahl et al. [11] ble en læringsrate på 10^{-5} benyttet. De nevrale nettverkene i eksperimentmodellene fikk som input en batch av 16 bilder fra datasettet under treningsprosessen, før vektene ble oppdatert ved hjelp av algoritmen tilbakepropagering beskrevet i delkapittel 3.3.7. Eksperimentmodellene som trente på hode- og halskreftdatasettet hadde en maksimalgrense på 200 epoker, antall ganger nettverket kjører igjennom hele treningssettet. Det er i konfigurasjonsfilen lagt inn en funksjon kalt tidlig stopping («early stopping»), som stopper modellene når ytelsesmål-scoren på valideringssettet ikke forbedres med en gitt rate i løpet av 30 epoker. På denne måten reduseres overflødig tid brukt på å trene, samtidig som beregningskraft begrenses. Ytelsesmålet brukt for å evaluere modellenes prestasjon på valideringssettet var Dice-score. I tillegg ble de avstandsbaserte ytelsesmålene HD_{95} og MSD benyttet for å gi komplementær informasjon angående kvaliteten til modellens prediksjoner på testsettene. Detaljer og ligninger for ytelsesmålene er gitt i delkapittel 3.9. Modaliteten PET/CT ble benyttet i baselinemodellen og eksperimentmodellene siden denne modaliteten viste seg å oppnå høyest segmenteringsytelse i studien utført av Moe et al. [21].

4.5 Preprosessering

Datasettet benyttet i denne masteroppgaven er det samme preprosesserte hode- og halskreftdatasettet brukt i [21]. For å redusere de ubalanserte dataene ble bildene beskjært, som førte til at hvert tverrsnitt i datasettet inneholdt mellom 0,02 % og 32 % tumor- eller påvirkede lymfeknutepiksler [60]. Bilder i kanalen CT i datasettet, ble preprosessert med følgende CT-vindusinnstilinger i Hounsfield-enheter (HU): vindussenter på 70 HU med vindusbredde lik 200 HU. Vinduparameterne ble ifølge Moe [60] valgt etter konsultasjon med en erfaren radiolog. Vindussenteret på 70 HU representerer median intensiteten i tumorvolumet i CT-bildene, mens vindusbredden på 200 HU omfatter det meste av det myke vevet [21]. Figur 4.4 viser hvordan CT-vindusinnstilling påvirker et CT-bilde.



Figur 4.4: Eksempel på et CT-bilde før (venstre) og etter (høyre) preprosessering med CT-vindusinnstillingen («Hounsfield windowing») vindussenter lik 70 HU og vindusbredde lik 200 HU.

Bildene brukt i denne oppgaven ble normalisert og består av intensitetsverdier mellom 0 og 1. Normalisering av data er viktig for modell generalisering. Modalitetene PET og CT har i utgangspunktet ulikt spenn på pikselverdiene som kan forvirre dyplæringsmodellen. Ved hjelp av normalisering får pikselverdiene i de ulike modalitetene lik fordeling, som gjør at treningsprosessen kan konvergere raskere [54]. Normalisering ble ikke benyttet på datasettet i studien til Moe et al. [21].

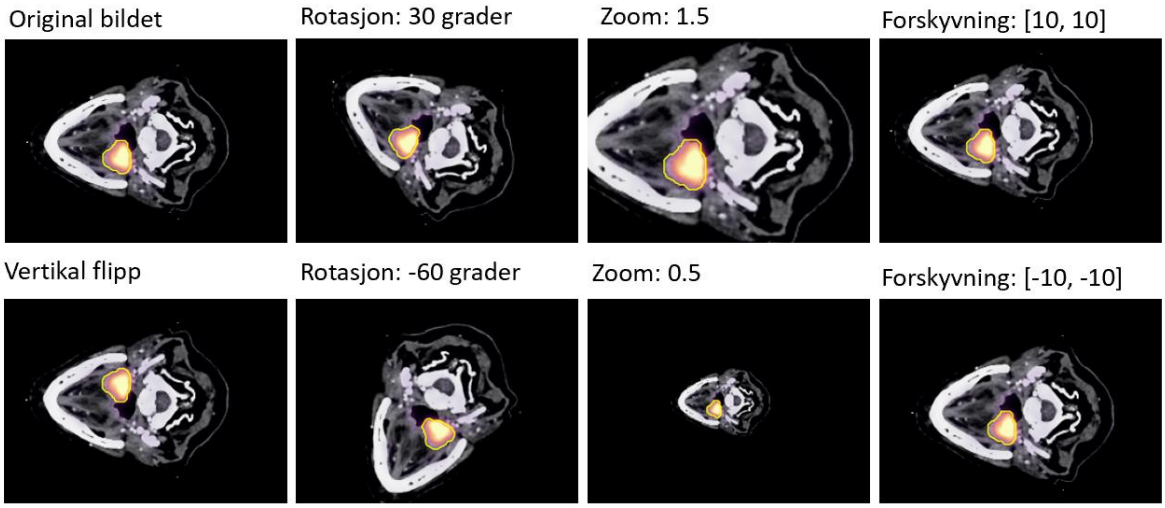
4.6 Dataaugmentering

I denne masteroppgaven har en rekke augmenteringsteknikker blitt benyttet på hode- og halskreft treningsdatasettet og de tilhørende kreftsvulstinntegningene, som en del av preprosesserings-prosessen. Formålet var å utvide treningsdatasettet med nye variasjoner, ved å foreta endringer på det allerede eksisterende treningsdatasettet [24]. En dyplæringsmodell som trener på et større datasett med nye varianter, kan bli mer generell og dermed potensielt forbedre modellens ytelse [17, 24, 27]. Ulike kombinasjoner av geometriske operasjoner kalt affine transformasjoner, og ulike kombinasjoner av punkt- og filteroperasjoner, har i denne masteroppgaven blitt undersøkt. Disse teknikkene og tilhørende nivåer var innebygget i rammeverket *deoxys*, der flere detaljer kan finnes på [69]. Figur E.1 i Vedlegg E viser eksempler på et PET/CT-bilde påført kombinasjoner av de ulike teknikkene.

4.6.1 Affine transformasjoner

Affine transformasjoner (beskrevet i delkapittel 3.8.1) har i tidligere studier vist seg å være effektive og populære dataaugmenteringsmetoder [23, 27]. Teknikkene rotasjon, zoom, forskyvning og flipp er de affine dataaugmenteringsmetodene som har blitt benyttet i denne

masteroppgaven. Figur 4.5 viser eksempler på et PET/CT-bilde i datasettet som har blitt påført de forskjellige affine augmenteringsteknikkene med ulike nivåer.



Figur 4.5: Eksempler på et PET/CT-bilde som har blitt påført ulike affine augmenteringsteknikker, der den gule inntegningen representer den sanne inntegningen. Bildene indikerer hvilket augmenteringsteknikknivå som har blitt påført. Øverst til venstre er originalbildet avbildet.

4.6.1.1 Rotasjon

Dataaugmenteringsteknikken rotasjon («rotation») skaper nye treningseksempler ved å rotere inputbilder et antall grader. Hvilke inputbilder som påføres transformasjonen rotasjon er tilfeldig. Sannsynligheten for at et treningseksempel roteres er i denne masteroppgaven 20 % [69]. Kjører nettverket gjennom treningsdatasettet fem ganger skal hvert treningseksempel i teorien ha blitt rotert minst en gang. Antall grader som inputbildet roteres med velges tilfeldig av modellen fra et bestemt intervall. I konfigurasjonsfilen til dyplæringsmodellen spesifiseres det hvilke antall grader bildene maksimalt skal roteres med. Er antall grader lik 60, vil et inputbilde rotere med en vinkel mellom -60°og 60°. Ettersom rotasjonsvinkelen velges tilfeldig innenfor det gitte intervallet er det liten sannsynlighet for at det samme inputbilde roteres med likt antall grader. Nettverket vil derfor kunne skape nye varianter av treningsdata etter hver gjennomkjøring. Jo flere ganger nettverket trener, jo flere treningseksempler genereres [54]. Rotasjonsvinklene som har blitt testet i denne masteroppgaven er 0°, 30°, 60°, 90°.

4.6.1.2 Zoom

Zoom («zoom») er en vanlig dataaugmenteringsteknikk som endrer inputdata ved å foreta forstørringer og forminskninger på treningseksempler. Det er i denne masteroppgaven 20 % sannsynligheten for å påføre treningseksempler augmenteringsteknikken zoom [69]. I

konfigurasjonsfilen defineres det et intervall som angir zoomområdet modellen tilfeldig velger en zooomfaktor fra. Et tall under 1 indikerer forminskning, utzooming, mens et tall over 1 indikerer en forstørring, innzooming. En zoomfaktor lik 1 indikerer ingen zoomeffekt. Er det gitte intervallet for eksempel lik [0,8, 1,2] kan et inputbilde tilfeldig bli forstørret eller forminsket, men det resulterende bildet kan ikke være mindre enn 0,8 eller større enn 1,2 av original bildet [54]. I denne masteroppgaven har zoomfaktoren 1 (ingen zoom) og zoomområdene [0,5, 1,5] og [0,8, 1,2] blitt undersøkt.

4.6.1.3 Forskyvning

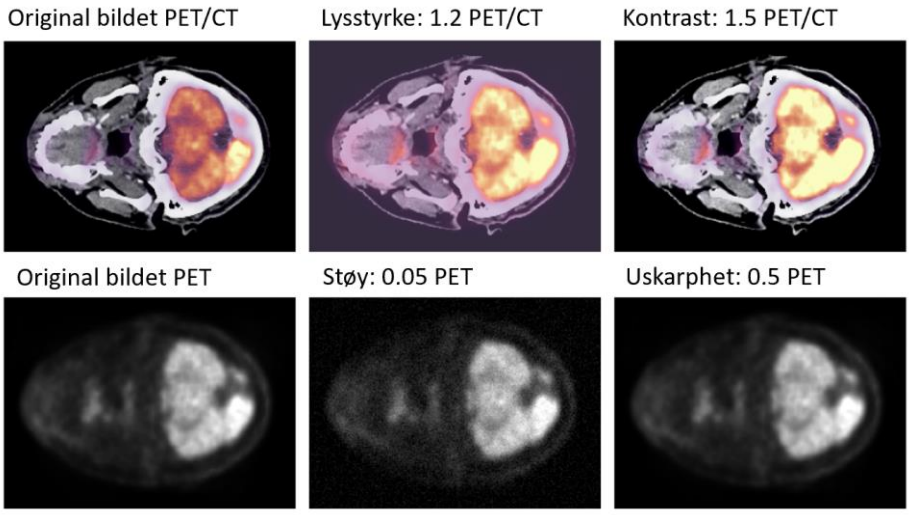
Dataaugmenteringsteknikken forskyvning («shift») skaper nye treningseksempler ved å forskyve originalbildet enten opp, ned, til venstre eller til høyre [26]. Ved å trene modeller på inputdata som har blitt påført teknikken forskyvning kan potensielle biaser for posisjon av kreftsvulster eller påvirkede lymfeknuter unngås [30]. I konfigurasjonsfilen til dyplæringsmodellen angis det tall som indikerer et intervall i hver akse som modellen tilfeldig velger en forskyvningsfaktor fra. Bildet kan forskyves både langs *x*-aksen og langs *y*-aksen. Spesifiseres tallene [10, 20] i konfigurasjonsfilen indikerer dette at dyplæringsmodellen kan forskyve treningseksempelet antall piksler langs *x*-aksen med en tilfeldig faktor mellom [-20, 20] og langs *y*-aksen med en tilfeldig faktor valgt fra intervallet [-10, 10]. Et negativt tall for *y*-aksen indikerer forskyvning av bildet nedover, mens et positivt tall indikerer forskyvning av bildet oppover. Et negativt tall for *x*-aksen indikerer forskyvning til høyre, mens et positivt tall indikerer forskyvning til venstre [54]. Sannsynligheten for at et inputbilde blir påført augmenteringsteknikken forskyvning er 10 % [69]. Denne masteroppgaven har testet forskyvning i intervallet [-10, 10] i hver akse.

4.6.1.4 Flipp

Metoden flipp («flip») endrer inputdata og skaper dermed nye treningseksempler ved å flippe inputdata vertikalt, speiling om *x*-aksen, eller horisontalt, speiling om *y*-aksen. I denne masteroppgaven testes nivåene vertikal flipp og ingen flipp. Ifølge Hussain et al. [26] avdekker en vertikal flipp unike egenskaper i medisinsk data, som kan forbedre dyplæringsmodellens ytelse. Ved å flippe et inputbilde om *x*-aksen genereres det nye troverdige treningseksempler som kan forbedre prestasjonen til modellen på usett data. Sannsynligheten for at et bilde påføres dataaugmenteringsteknikken flipp er i denne masteroppgaven 50 % [69].

4.6.2 Punkt- og filteroperasjoner

Andre tradisjonelle dataaugmenteringsteknikker som har vist lovende resultater i tidligere studier er blant annet punkt- og filteroperasjoner (beskrevet i delkapittel 3.8.2) som foretar fargemodifiseringer på inputdata [23, 27]. Fargemodifiseringsmetodene er både raske, reproduserbare og enkle å forstå, som gjør de til populære dataaugmenteringsteknikker [23]. I denne masteroppgaven har nye treningseksempler blitt generert ved å endre lysstyrke, endre kontrast, legge til støy og justere uskarphet i inputbilder. I masteroppgaven til Huynh [22] ble det formulert en hypotese om at U-Net-modellen lærte at lyse områder i PET-kanalen indikerte høy sannsynligheten for kreft, og derfor tegnet inn slike områder. Ettersom dyplæringsmodellen lærte mest fra PET-kanalen, ble noen strukturer hos pasienter med høy SUV klassifisert feil. Støy og uskarphet ble derfor påført PET-kanalen for å potensielt øke CT-kanalen sin innvirkning på modellens avgjørelse. Teknikkene lysstyrke og kontrast ble påført PET/CT-modaliteten. Figur 4.6 viser eksempler på et bilde fra datasettet med modalitetene PET/CT og PET som har blitt påført forskjellige punkt- og filteroperasjoner.



Figur 4.6: Eksempler på et PET/CT-bilde og et PET-bilde som har blitt påført ulike punkt- og filteroperasjoner. Bildene indikerer hvilke teknikk og nivå som har blitt påført.

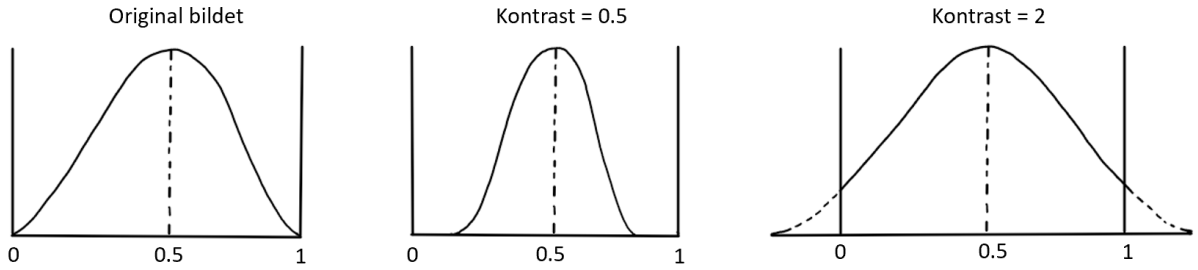
4.6.2.1 Lysstyrke

Justering av lysstyrke («brightness») er en enkel form for fargemodifikasjon under kategorien punktoperasjoner. Endring av lysstyrke handler om å øke eller minke intensiteten til pikslene i et inputbilde [48]. Distribusjonen av intensitetsverdier, og andelen av disse representert i et bilde, kan fremstilles ved bruk av et histogram. Økes lysstyrken med en gitt verdi flyttes histogrammet mot høyre. Minker lysstyrken med en gitt verdi flyttes histogrammet mot venstre [48].

I denne masteroppgaven endres lysstyrken i inputbildet ved å legge til et tilfeldig tall fra et beregnet lysstyrkeintervall. Dette intervallet beregnes basert på inputbildets maksimale pikselintensitetsverdi i hver kanal og et angitt lysstyrkespenn i konfigurasjonsfilen. Har inputbildet pikselverdier i området [0, 1] og det angitte lysstyrkespennet er [0,8, 1,2] blir lysstyrkeintervallet beregnet til [-0,2, 0,2] ettersom $1 * (0{,}8 - 1) = -0{,}2$ og $1 * (1{,}2 - 1) = 0{,}2$. Dette impliseres at alle pikselverdier i inputbildet som påføres augmenteringsteknikken vil bli addert med en tilfeldig valgt verdi innenfor det gitte området -0,2 til 0,2 [54]. Spennet for lysstyrkeendringen som i denne masteroppgaven har blitt undersøkt er [0,8, 1,2] og 1 (ingen lysstyrkeendring). Sannsynligheten for at et inputbilde påføres lysstyrkeaugmentering er 10 % [69].

4.6.2.2 Kontrast

En annen vanlig punktoperasjon er endring av kontrast («contrast») i inputdata [48]. Generelt brukes begrepet kontrast om forskjellen mellom den største og den minste intensitetsverdien til pikslene i bildet, og beskriver dermed intervallet av intensitetsverdier. En økning i kontrasten vil gi en økning av bredden i bildets histogram. På lik linje vil en minking i kontrasten resultere i et smalere histogram [48]. Ved å endre kontrasten i inputdata og dermed intervallet av intensitetsverdier, kan det genereres nye variasjoner av treningseksempler. I konfigurasjonsfilen defineres intervallet konfigurasjonsfilen tilfeldig veldig en kontrastfaktor fra. Forandringen av pikslenes intensitetsverdi er basert på inputbildets histogram. Hvis M er gjennomsnittsintensitetsverdien i inputbildet og F er den tilfeldige valgte kontrastfaktoren, vil det augmenterte bildet være $\big((I - M) * F\big) + M$, hvor I er intensitetsverdien til hver piksel. Etter denne endringen blir pikselverdier lavere enn 0 omgjort til 0 og pikselverdier høyere enn 1 omgjort til 1 [54]. Figur 4.7 viser eksempel på histogrammet til et bilde med intensitetsverdier mellom 0 og 1 som endres ved kontrastendring lik 0,5 og 2. Sannsynligheten for å endre kontrasten i et treningsbilde er 10 % [69]. Intervallet som i denne oppgaven har blitt testet er [0,7, 1,3] og 1 (ingen zoom).



Figur 4.7: Figuren viser eksempel på et histogram til et bilde med intensitetsverdier (x-aksen) mellom 0 og 1 som endres ved kontrastendring. Høyden på kurven representerer antall piksler med den gitte intensitetsverdien. Verdier som er lavere enn 0 og høyere enn 1 blir satt til henholdsvis 0 og 1 ved kontrastendring lik 2.

4.6.2.3 Støy

Teknikken støy («noise») er under kategorien punktoperasjoner [48]. Denne teknikken legger til tilfeldig Gaussisk støy fra en Gaussisk fordeling (normalfordeling) med en definert varians σ^2, til hver piksel i et inputbilde. I konfigureringsfilen defineres det hvilken kanal og hvilket intervall modellen tilfeldig velger σ^2 fra. I denne masteroppgaven har augmenteringsteknikken blitt testet på PET-kanalen med et intervall på [0, 0,05]. Den forventede prosentandelen inputbilder som påføres augmenteringsteknikken støy i hver batch er 10 % [69].

4.6.2.4 Uskarphet

Augmenteringsteknikken uskarphet («blur») er en filteroperasjon [48]. Bruk av filter til å justere skarpheten i inputdata er en vanlig teknikk innenfor bildeprosessering [30]. Filtermatrisen med en gitt størrelse $n \times n$ beveges over pikslene til treningseksempelet og gir et uskarpt bilde. På denne måten eksponeres modellen for uskarpe treningseksempler som kunne blitt generert grunnet bevegelse under en PET/CT-skanning [30]. Et Gaussisk filter ble i denne masteroppgaven benyttet på inputbildene. Det lineære Gaussiske filteret bruker den Gaussiske funksjonen, vist i ligning 4.1, for å beregne transformasjonen som påføres inputbildets piksler [48].

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.1}$$

Her representerer x pikselposisjonen i *x*-aksen, y pikselposisjonen i *y*-aksen, og σ standardavviket til fordelingen [48]. Sannsynligheten for at et treningseksempel påføres augmenteringsteknikken uskarphet er 10 % [69]. I konfigureringsfilen defineres det både hvilken kanal som skal påføres teknikken og hvilket σ-intervall modellen tilfeldig skal velge en faktor fra. Filteroperasjonen har i denne masteroppgaven blitt testet på kanalen PET med σ-intervallet satt til [0,5, 1,5] og 0 (ingen uskarphet påført).

4.6.3 Sannsynligheten for å bevare originalbildet

Et treningseksempel blir påført augmenteringsteknikkene definert i konfigurasjonsfilen med en sannsynlighet som kan beregnes. Består konfigurasjonsfilen til modellen av augmenteringsteknikkene zoom, rotasjon, forskyvning og flipp, der sannsynligheten for påført transformasjon henholdsvis er 0,2, 0,2, 0,1 og 0,5, vil sannsynligheten for å bevare originalbildet være gitt som vist i ligning 4.2 og ligning 4.3:

$$P(orginalbildet) = (1 - 0{,}2) * (1 - 0{,}2) * (1 - 0{,}1) * (1 - 0{,}5) \tag{4.2}$$

$$P(originalbildet) = 0{,}288 \quad (4.3)$$

Det er altså tilnærmet 30 % sannsynlighet for at treningseksempelet gitt som input til nettverket ikke transformeres. Antall nye treningseksempler som genereres og legges til treningsdatasettet avhenger av antall ganger modellen trener. Jo flere treningseksempler som skapes, jo flere aspekter og variasjoner av datasettet blir modellen eksponert for, som kan bidra til en mer robust modell med høy generaliseringsevne [17, 27].

4.7 Eksperimentplan

De 65 utførte eksperimentene med ulike kombinasjoner av augmenteringsteknikknivåer ble delt opp i tre eksperimentplaner, basert på kategori av dataaugmenteringsteknikker. Den første eksperimentplanen, kalt eksperimentplan 1, bestod av 48 ulike kombinasjoner av nivåene til de fire affine transformasjonene rotasjon, forskyvning, zoom og flipp, definert i 48 konfigurasjonsfiler. Tabell 4.2 viser nivåene som ble testet for hver teknikk.

Tabell 4.2: Tabellen viser en oversikt over de ulike teknikkene og nivåene definert i eksperimentplan og konfigurasjonsfil for de affine transformasjonene, som utgjør eksperimentplan 1.

Augmenteringsteknikk	Nivå i eksperimentplan	Nivå i konfigureringsfil
Rotasjon	0, 30, 60, 90	0, 30, 60, 90
Zoom	1, [0,5, 1,5], [0,8, 1,2]	1, [0,5, 1,5], [0,8, 1,2]
Forskyvning	null, [10, 10]	null, [-10, 10]
Flipp	no, yes	null, 0

Den andre eksperimentplanen, kalt eksperimentplan 2, bestod av 16 eksperimenter med ulike nivåer av de fire punkt- og filteroperasjonsteknikkene lysstyrke, kontrast, støy og uskarphet. Detaljer angående augmenteringsteknikkene er beskrevet i delkapittel 3.8. Tabell 4.3 viser hvilke nivåer som ble testet for hver teknikk.

Tabell 4.3: Tabellen viser en oversikt over de ulike teknikkene og nivåene definert i eksperimentplan og konfigurasjonsfil for punkt- og filteroperasjonene som utgjør eksperimentplan 2.

Augmenteringsteknikk	Nivå i eksperimentplan	Nivå i konfigurasjonsfil
Lysstyrke	no, yes	1, [0,8, 1,2]
Kontrast	no, yes	1, [0,7, 1,3]
Støy	no, yes	0, 0,05
Uskarphet	no, yes	0, [0,5, 1,5]

Videre ble et kombinasjonseksperiment utført, som bestod av teknikknivåene som oppnådde høyest ytelse per tverrsnitt på valideringssettet i eksperimentplan 1 og eksperimentplan 2, og

utgjør eksperimentplan 3. Eksperimentplaner og eksempel på konfigurasjonsfil ligger vedlagt i Vedlegg B. Alle konfigurasjonsfilene kan finnes på: https://github.com/mariaodegaard/cnn-template.

Et ferdig trent eksperiment brukte ca. 25–30 GB minne i regneklyngen Orion. Ettersom Orion tilbydde maksimallagringsminne på 300 GB, ble bare 4–5 eksperimentmodeller kjørt daglig i eksperimentkjøringsfasen for å unngå «tom for minne-problemet». For å frigjøre tilgjengelig plass ble eksperimentmodellens vekter, prediksjoner og bilder med predikerte inntegninger lagret for den epoken som oppnådde høyest Dice-score på valideringssettet.

4.8 Evaluering av modellene

For å finne kombinasjonen av nivåene til de ulike augmenteringsteknikkene innad i eksperimentplan 1 og eksperimentplan 2 som oppnådde høyest segmenteringsytelse ble både den gjennomsnittlige Dice-scoren per tverrsnitt på valideringssettet og statistiske tester benyttet. De statistiske testene ble utført på eksperimentplan 1 og eksperimentplan 2, hver for seg, i plattformen RStudios[7]. RStudios-scriptene benyttet for de statistiske testene ligger vedlagt i Vedlegg C. Kombinasjonen av de beste nivåene innad i eksperimentplan 1 og eksperimentplan 2 ble satt sammen og kjørt i eksperimentplan 3. Modellen som oppnådde høyest segmenteringsytelse på valideringssettet av de tre eksperimentplanene ble videre kjørt på testsettene.

4.8.1 Organisering av resultatdataene

Før de statistiske testene kunne gjennomføres ble resultatdataene fra eksperimentmodellene post-prosessert i plattformen Spyder. Ettersom valideringssettet bestod av 15 pasienter med totalt 1033 tverrsnitt, ble 15 Dice-scorer per pasient og 1033 Dice-scorer per tverrsnitt beregnet for hver eksperimentmodell. Dice-score per tverrsnitt til modellene i eksperimentplan 1 og eksperimentplan 2 ble videre organisert i to separate datasett.

4.8.2 Statistiske tester

N-veis ANOVA [70] som ser på hver parameter (augmenteringsteknikk) og interaksjonene mellom de ulike parameterne, ble benyttet for å evaluere parameternivåenes innvirkning på segmenteringsytelsen Dice-score. I ANOVA ble Dice-score per tverrsnitt brukt som responsvariabel og augmenteringsteknikker brukt som forklaringsvariabler. Siden en

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

betingelse for å kunne bruke ANOVA er normalfordelte residualer, ble fordelingen til residualene undersøkt ved bruk av diagnoseplott og Anderson-Darling test [71].

Hypotesene for Anderson-Darling testen var:

H0: Residualene er normalfordelte

H1: Residualene er ikke normalfordelte

Dersom ikke betingelsen for normalitet ble oppfylt, ble ulike transformasjoner som logaritmisk-transformasjon, kvadratrot-transformasjon, invers-transformasjon og boxcox-transformasjon utført på datasettene for å undersøke om de transformerte dataene oppfylte normalitetskravet [72].

Dersom normalitetskravet ikke kunne oppnås, ble de ikke-parametriske testene Friedmantest [73] etterfulgt av tosidig Nemenyi post-hoc test [74] og tosidig Wilcoxon signed rank-test [74] benyttet for å evaluere resultatdataene. Friedmantest og Wilcoxon signed rank-test undersøker innvirkningen av de ulike nivåene innad i hver augmenteringsteknikk på modellenes segmenteringsytelse. Den statistiske testen Nemenyi post-hoc ble benyttet etter Friedmantesten for å finne nivåene innad i parameterne som eventuelt ga signifikant forskjell. Wilcoxon signed rank-test ble brukt på parametere med to nivåer, mens Friedmantesten etterfulgt av post-hoc test ble brukt på parametere med tre eller flere nivåer. Hypotesene til begge testene var:

H0: Ingen forskjell mellom nivåene på segmenteringsytelse

H1: Forskjell mellom nivåene på segmenteringsytelse

En forutsetning for å kunne utføre Friedmantesten er et datasett uten replikasjoner («unreplicated complete block design») [73]. For å oppnå dette, må resultatdataene organiseres ved å legge til en kolonne «Teller» som indekserer fra 1-1033 per eksperiment i hvert datasett. Videre må gjennomsnittet av Dice-scoren til tverrsnitt med samme teller-nummer og samme nivå innad i parameteren bli beregnet. Dette krever at det må lages fire datasett for eksperimentplan 1 og fire datasett for eksperimentplan 2, et for hver parameter, bestående av 1033 rader ganger antall nivåer innad i parameteren.

Kapittel 5: Resultater

5.1 Eksperimentene

Som beskrevet i delkapittel 4.7 ble totalt 65 eksperimentmodeller kjørt i regneklyngen Orion på hode- og halskreftdatasettet (OUS) med ulike kombinasjoner av dataaugmenteringsteknikker fordelt i tre ekseprimentplaner. Målet var å finne modellen bestående av augmenteringsteknikkene som oppnådde høyest ytelse på segmenteringsoppgaven, heretter kalt den beste modellen, som videre ble kjørt på OUS-testsettet og det eksterne Maastro-testsettet.

5.2 Modellytelse for de innledende eksperimentene

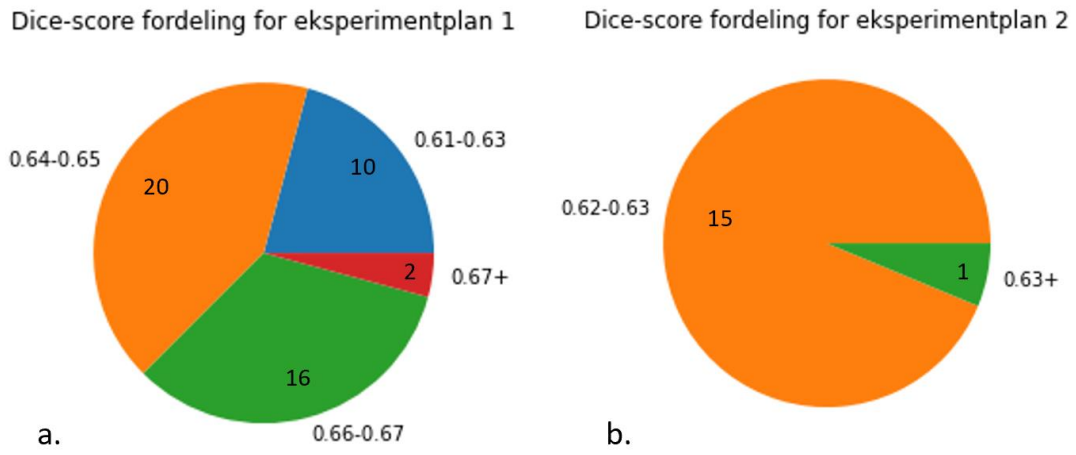
Basert på valideringssettet som bestod av 15 pasienter med ulikt antall tverrsnitt per pasient, ble en gjennomsnittlig Dice-score per tverrsnitt beregnet for hver eksperimentmodell. Fordelingen av Dice-scorene oppnådd innad i eksperimentplanene er visualisert i Figur 5.1 i sektordiagrammer. En oversikt over augmenteringsnivåene og den oppnådde ytelsen for hver eksperimentmodell er gitt i Tabell B.1 og Tabell B.2 vedlagt i Vedlegg B, for henholdsvis eksperimentplan 1 og 2.

Fra sektordiagrammet i Figur 5.1 (diagram 5.1a.) og fra Tabell B.1, oppnådde 10 eksperimentmodeller en Dice-score innenfor intervallet 0,61–0,63 i eksperimentplan 1. Felles for disse eksperimentene var at augmenteringsteknikken flipp ikke var brukt. Teknikkene rotasjon og zoom var heller ikke benyttet i de fleste av disse eksperimentmodellene. Forskyvning ble benyttet i 5 av 10 eksperimentmodeller, men så ikke ut til å gi en tydelig økning i ytelsen. Videre viser sektordiagrammet at flertallet av eksperimentmodellene oppnådde en gjennomsnittlig Dice-score i intervallet 0,64–0,65. I dette intervallet observeres det ingen tydelige trender av augmenteringsteknikknivåer i Tabell B.1. 16 eksperimentmodeller oppnådde en gjennomsnittlig Dice-score innenfor intervallet 0,66–0,67. I disse eksperimentene ble vertikal flipp om *x*-aksen, rotasjon og zoom brukt. 7 av 16 eksperimentmodeller brukte augmenteringsteknikken forskyvning. To eksperimentmodeller oppnådde Dice-score over 0,67. Disse brukte augmenteringsteknikkene tilfeldig rotasjon mellom -90° og 90°, flipping om *x*-aksen og forskyvning.

Basert på høyest oppnådd Dice-score i Tabell B.1 ble eksperimentmodell 46 utnevnt til beste modell for eksperimentplan 1. Denne modellen brukte augmenteringsteknikkene tilfeldig

rotasjon mellom -90° og 90°, zoomfaktor mellom 0,5 og 1,5, et antall pikselforskyvning mellom -10 og 10 i *x*- og *y*-retning og flipping om *x*-aksen.

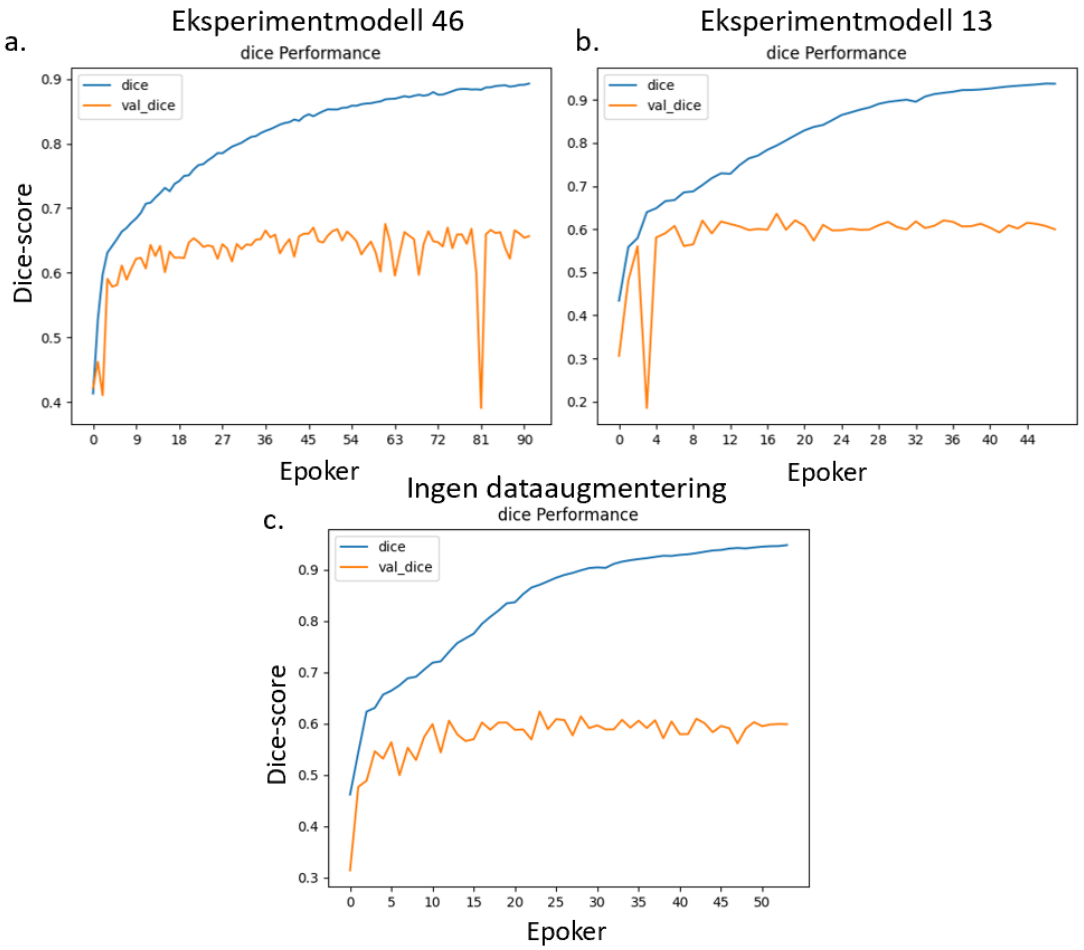
Sektordiagrammet for eksperimentplan 2 (diagram i Figur 5.1b.) viser at de fleste eksperimentmodellene oppnådde en gjennomsnittlig Dice-score per tverrsnitt på 0,62. Kun eksperimentmodell 13 som brukte augmenteringsteknikkene uskarphet og støy, oppnådde en Dice-score over 0,63 (Tabell B.2) og ble derfor utnevnt til beste modell for eksperimentplan 2.



Figur 5.1: Figuren viser sektordiagram av gjennomsnittlig Dice-score per tverrsnitt fra OUS-valideringssettet for eksperimentplan 1 (diagram a.) og 2 (diagram b.). Størrelsen på sektoren og det oppgitte tallet representerer andelen av eksperimentmodeller som oppnådde en Dice-score innenfor de ulike intervallene.

5.2.1 Trenings- og valideringskurve

Figur 5.2 viser trenings- og valideringskurven til eksperimentmodell 46 (graf 5.2a., plan 1) og eksperimentmodell 13 (graf 5.2b., plan 2). Trenings- og valideringskurven til eksperimentmodell 1 i plan 2 (graf 5.2c.) uten noen form for augmentering, er også vist.



Figur 5.2: Figuren viser utviklingen av Dice-score for treningssettet (blå) og valideringssettet (oransje) etter hver epoke, for eksperimentmodellene 46 (a.), 13 (b.) og for eksperimentmodell 1 (c.) i eksperimentplan 2 uten augmentering.

Dersom avstanden mellom treningskurven og valideringskurven er stor, kan dette indikere at dyplæringsmodellen er overtilpasset treningsdatasettet. Figur 5.2 viser at avstanden mellom treningskurven og valideringskurven er større uten augmentering (graf 5.2c.) enn med. Dette tyder på at augmenteringsteknikkene hjelper mot overtilpasning. Likevel er avstanden mellom treningskurven og valideringskurven fremdeles relativt stor i modellene med augmentering. Eksperimentmodellene 46 og 13 er godt tilpasset treningsdatasettet, men generaliserer ikke like godt til valideringssettet, som kan tyde på at modellene fremdeles er overtilpasset.

Valideringskurven til eksperimentmodell 46 og 13 (Figur 5.2a. og b.) har tydelige nedoverpigger, spesielt ved epoker 81 og 3 for henholdsvis modell 46 og 13. Dette kan tyde på at modellene er sensitive for visse vektoppdateringer.

Eksperimentmodell 46 bruker flere epoker enn de andre modellene og har dermed trent på et større treningsdatasett ettersom det for hver epoke legges til nye augmenterte treningseksempler. Mekanismen tidlig stopping, beskrevet i delkapittel 4.4.1.2, har ikke stoppet eksperimentmodell 46 før epoke 92 som kan det tyde på at ytelsen har økt ved å

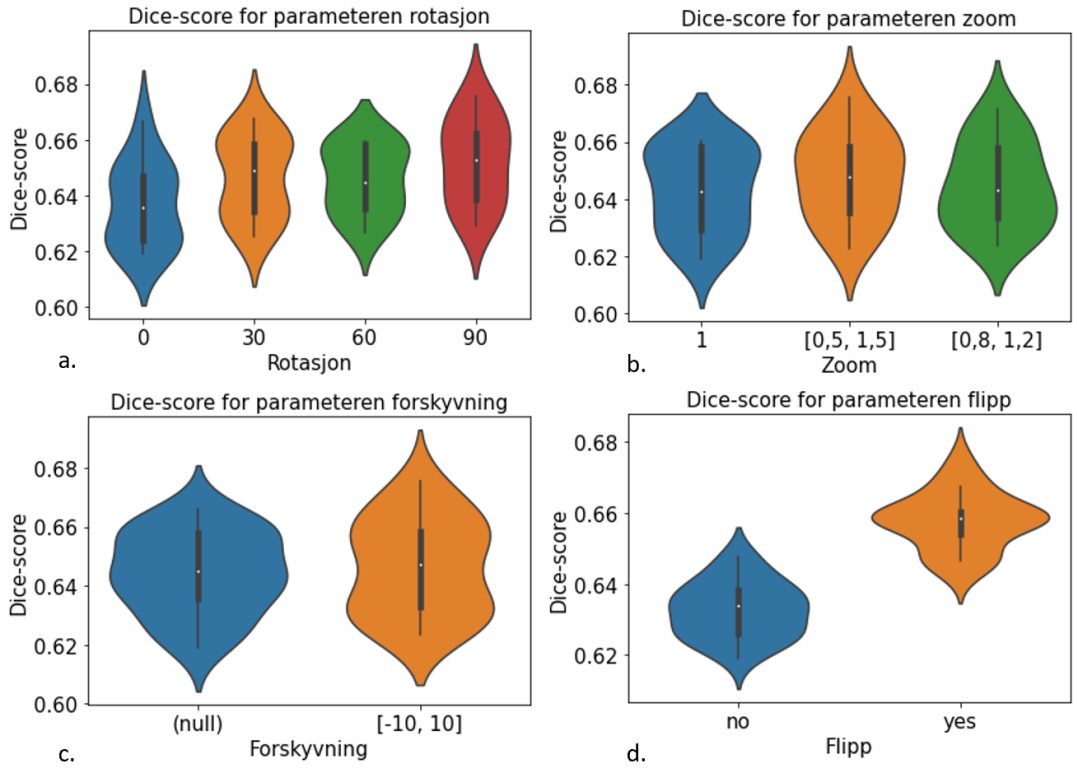
inkludere augmenterte treningseksempler frem til epoke 62. Etter denne epoken forbedres ikke ytelsen med en gitt rate i løpet av 30 epoker, selv om nye augmenterte treningseksempler legges til. Eksperimentmodell 13 ble stoppet etter epoke 48, som kan tyde på at de tillagte augmenterte treningseksemplene ikke har hatt effekt på modellytelsen etter epoke 18.

5.2.2 Visualisering av modellytelse

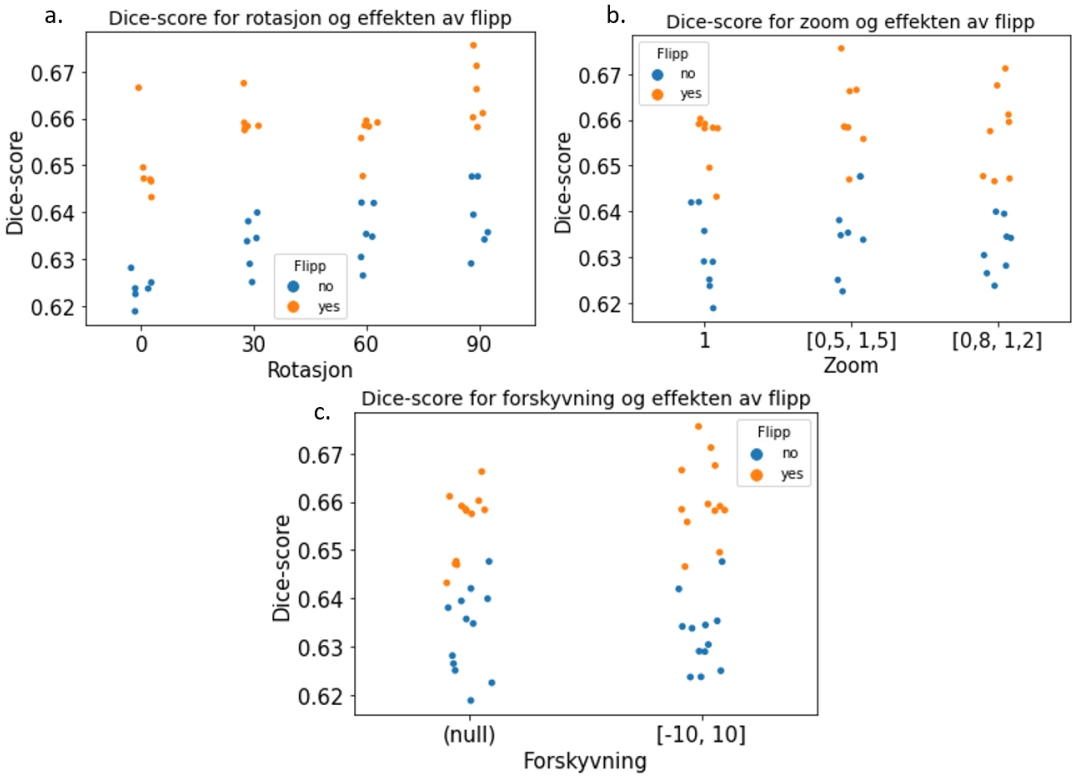
Ved bruk av fiolinplott ble augmenteringnivåenes innvirkning på segmenteringsytelsen visualisert. Et fiolinplott per parameter ble laget for affine augmenteringer (eksperimentplan 1), vist i Figur 5.3, og punkt- og filteraugmenteringer (eksperimentplan 2), vist i Figur 5.5.

Fra Figur 5.3 kan det se ut til at affine augmenteringer har positiv innvirkning på Dice-score for alle teknikkene. For parameteren rotasjon (fiolinplott i Figur 5.3a.) øker Dice-scoren i takt med økt rotasjonsnivå, med unntak av overgangen mellom rotasjonsnivå 30 og 60 grader, hvor ytelsen og medianen blir noe lavere. Fiolinplott i Figur 5.3b. viser at modeller med augmenteringsteknikken zoom oppnår høyere Dice-scorer, og at zoomnivået med videst intervall har den største innvirkningen. Likevel observeres det fra formen til fiolinplottene en større fordeling av eksperimenter som oppnådde en relativt høy Dice-score med nivå 1 (ingen zoom), sammenlignet med de andre nivåene. Augmenteringsteknikken forskyvning (Figur 5.3c.) gav noe høyere Dice-score enn modeller uten forskyvning. Augmenteringsteknikken flipp (Figur 5.3d.) førte til den mest markante økningen i Dice-score.

For eksperimentplan 1 ble det i tillegg laget jitter-plott for parameterne rotasjon, zoom og forskyvning, hvor effekten av parameteren flipp er fremhevet, vist i Figur 5.4. Disse jitter-plottene viser de samme trendene som observert i fiolinplottene, og at nedgangen mellom rotasjonsnivå 30 og 60 grader i jitter-plott i Figur 5.4a. skyldes få datapunkter. Videre viser jitter-plottene tydelig at bruk av augmenteringstenikken flipp resulterte i de høyeste Dice-scorene.

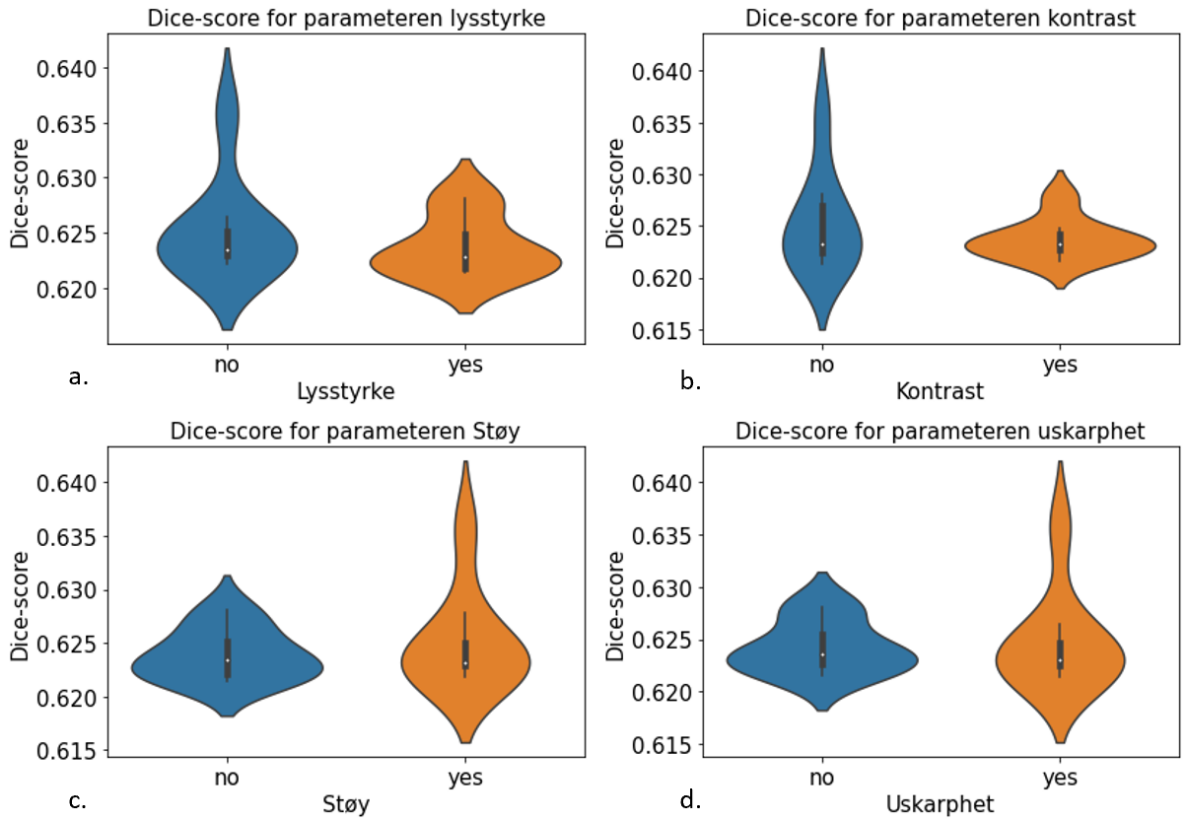


Figur 5.3: Figuren viser fiolinplott for augmenteringsteknikkene rotasjon, zoom, forskyvning og flipp i eksperimentplan 1. Plottene visualiserer fordelingen av gjennomsnittlig Dice-score per tverrsnitt på valideringssettet for augmenteringsnivåene. Fiolinplottets form indikerer fordelingen av oppnådd Dice-score. Tykkelsen representerer antall modeller, mens lengden representerer Dice-score-variansen. Den svarte boksen indikerer verdier mellom første og tredje kvartil, den hvite prikken indikerer Dice-score-medianen, mens verdier utenfor den vertikale linjen representerer Dice-score-utliggere.



Figur 5.4: Figuren viser jitter-plott for parameterne rotasjon, zoom og forskyvning hvor modeller med (oransje punkter) og modeller uten (blå punkter) flipp-augmentering er fremhevet. Punktene viser fordelingen av den gjennomsnittlige Dice-scoren per tverrsnitt på valideringssettet oppnådd med de ulike augmenteringstenikknivåene.

Modeller med punkt- og filteraugmenteringene resulterte i tilnærmet lik median-Dice-score som modeller uten disse augmenteringene. Punktaugmenteringene lysstyrke (fiolinplott i Figur 5.5a.) og kontrast (fiolinplott i Figur 5.5b.) ser ut til å redusere ytelsen for noen modeller. Variansen i modellytelse ved bruk av lysstyrke- og kontrastaugmentering ble dermed mindre. Augmentering med støy (fiolinplott i Figur 5.5c.) og uskarphet (fiolinplott i Figur 5.5d.) ser ut til å ha den største andelen modeller med høy Dice-score. Variansen i modellytelse ved bruk av støy og uskarphet ble dermed større.



Figur 5.5: Figuren viser fiolinplott for augmenteringsteknikkene lysstyrke, kontrast, støy og uskarphet i eksperimentplan 2. Plottene visualiserer fordelingen av gjennomsnittlig Dice-score per tverrsnitt på valideringssettet for augmenteringsnivåene. Fiolinplottets form indikerer fordelingen av oppnådd Dice-score. Tykkelsen representerer antall modeller, mens lengden representerer Dice-score-variansen. Den svarte boksen indikerer verdier mellom første og tredje kvartil, den hvite prikken indikerer Dice-score-medianen, mens verdier utenfor den vertikale linjen representerer Dice-score-utliggere.

5.3 Statistiske tester

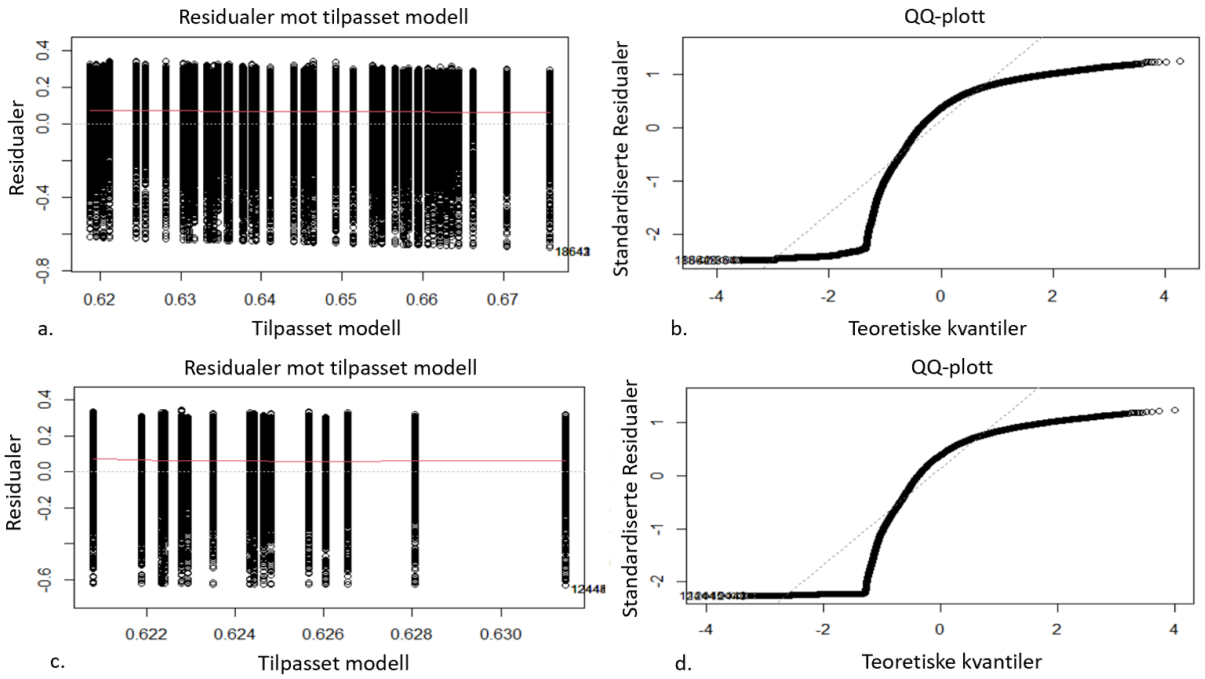
For å underbygge valg av beste modell i eksperimentplan 1 og 2, ble statistiske tester benyttet. Testene ble utført på datasett bestående av Dice-score per tverrsnitt for alle tverrsnitt for hver modell, som beskrevet i delkapittel 4.8. Disse datasettene, heretter kalt resultatdataene, inneholder dermed flere datapunkter enn datasettene benyttet i Figur 5.1 til Figur 5.5, hvor modellenes gjennomsnittlig Dice-score per tverrsnitt ble benyttet. Disse statistiske analysene kan dermed gi mer informasjon angående modellenes ytelse.

5.3.1 N-veis ANOVA

En betingelse for å kunne utføre en ANOVA er normalfordelte residualer, som nevnt i delkapittel 4.8.2. For å undersøke om residualene til resultatdataene oppfylte kravet om normalfordeling, ble diagnoseplott og Anderson-Darling test benyttet.

Diagnoseplottene for resultatdataene til eksperimentplan 1 og 2 er vist i Figur 5.6. Diagnoseplottet residualer mot tilpasset modell er vist i Figur 5.6a. og Figur 5.6c. Er kravet om normalitet oppfylt, skal den røde linjen i disse plottene være rett og nær den stiplede linjen hvor *y* er lik null. Spredningen av residualene skal i tillegg være lik over og under *y*-aksen. Til tross for at de røde linjene i disse plottene er tilnærmet rette, er ikke fordelingen av residualpunker over og under *y*-aksen like.

I QQ-plottet vist i Figur 5.6b. og Figur 5.6d. skal de svarte residualpunktene følge den stiplede linjen dersom residualene oppfyller kravet om normalitet. Dette var ikke tilfellet, og diagnoseplottene tyder dermed på at betingelsen for normalfordeling ikke var oppfylt. Dette bekreftes videre med den lave p-verdien fra Anderson-Darling testen gitt i Tabell 5.1 og Tabell 5.2, som indikerer at H0 kan forkastes på 0,05 signifikansnivå.

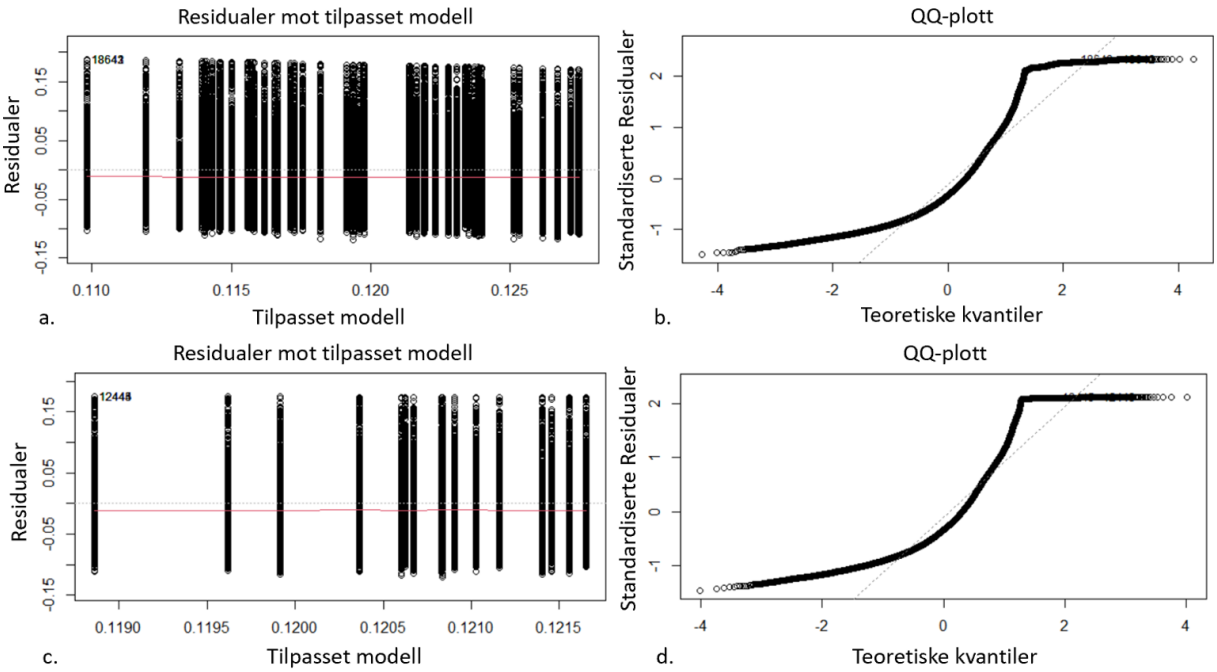


Figur 5.6: Figuren viser diagnoseplottene residualer mot tilpasset modell og QQ-plott for resultatdataene til eksperimentplan 1 (a. og b.) og 2 (c. og d.).

Ettersom residualene ikke var normalfordelt, ble resultatdataene transformert ved hjelp av logaritmisk-transformasjon, kvadratrot-transformasjon, invers-transformasjon og boxcox-transformasjon, for å videre undersøke om normalitetskravet kunne oppfylles. Figur 5.7 viser

diagnoseplott for resultatdataene etter logaritmisk transformasjon. Diagnoseplott for de andre transformasjonsteknikkene er vedlagt i Vedlegg C.

Fra Figur 5.7 observeres de samme trendene som i Figur 5.6, som indikerer at kravet om normalfordeling ikke var oppfylt for de logaritmisk transformerte resultatdataene. Dette gjelder også for diagnoseplottene vedlagt i Vedlegg C.



Figur 5.7: Figuren viser diagnoseplottene residualer mot tilpasset modell og QQ-plott for logaritmisk transformert resultatdata til eksperimentplan 1 (a. og b.) og 2 (c. og d.).

P-verdien fra Anderson-Darling testen bekrefter videre at residualene ikke var normalfordelte, vist i Tabell 5.1 og Tabell 5.2. Dermed ble en N-veis ANOVA ikke benyttet for å evaluere resultatdataene.

Tabell 5.1: Tabellen gir en oversikt over p-verdi fra Anderson-Darling testen utført på både transformert og ikke transformert resultatdata til eksperimentplan 1.

Transformasjon	p-verdi
Ingen transformasjon	p-verdi < 0,001
Logaritmisk	p-verdi < 0,001
Kvadratrot	p-verdi < 0,001
Invers	p-verdi < 0,001
Boxcox	p-verdi < 0,001

Tabell 5.2: Tabellen gir en oversikt over p-verdi fra Anderson-Darling testen utført på både transformert og ikke transformert resultatdata til eksperimentplan 2.

Transformasjon	p-verdi
Ingen transformasjon	p-verdi < 0,001
Logaritmisk	p-verdi < 0,001
Kvadratrot	p-verdi < 0,001
Invers	p-verdi < 0,001
Boxcox	p-verdi < 0,001

5.3.2 Friedmantest og Wilcoxon signed rank-test

En Friedmantest etterfulgt av Nemenyi post-hoc test og Wilcoxon signed rank-test ble dermed benyttet for å evaluere augmenteringsnivåenes innvirkning på modellenes segmenteringsytelse. Testene ble utført på datasett med Dice-score per tverrsnitt per modell på valideringssettet uten replikasjoner, for hver parameter. Friedmantest etterfulgt av post-hoc test ble utført på augmenteringsteknikker med tre eller flere nivåer, mens Wilcoxon signed rank-test ble benyttet på augmenteringsteknikker med to nivåer, som vist i Tabell C.1 og Tabell C.2 i Vedlegg C.

5.3.2.1 Eksperimentplan 1

Figur 5.8 viser boksplott som visualiserer Dice-score per tverrsnitt per modell for hver affine augmenteringsteknikk. Nivåene på disse teknikkene ser ut til å gi tilnærmet lik Dice-score. Tabell 5.3 gir en oversikt over de statistiske testene utført på resultatdataene til parameterne i eksperimentplan 1 (rotasjon, zoom, forskyvning og flipp), hvor oppnådd p-verdi og effektstørrelse til de ulike parameterne er vist. Alle augmenteringsteknikkene oppnådde en lavere p-verdi enn signifikansnivået 0,05. H0 kan dermed forkastes, som indikerer at det var signifikante forskjeller på segmenteringsytelsen mellom minst et av nivåene innad i hver teknikk. For parameterne rotasjon, zoom og forskyvning ble effektstørrelsen beregnet til liten. Valg av augmenteringsnivå for disse teknikkene vil dermed trolig ikke føre til en drastisk forskjell på Dice-scoren. For teknikken flipp var effekten beregnet til moderat.

Tabell 5.3: Tabellen gir en oversikt over de statistiske testene utført på resultatdataene til augmenteringsteknikker i eksperimentplan 1 (rotasjon, zoom, forskyvning og flipp), hvor p-verdien og effektstørrelsen er vist.

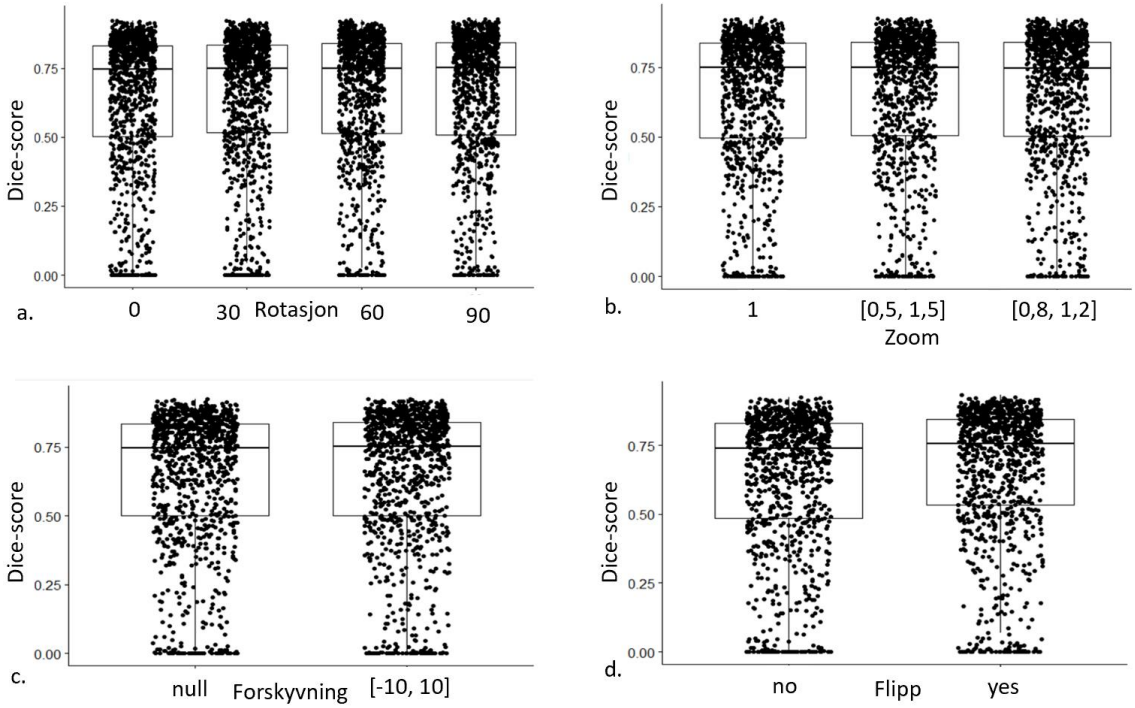
Parameter	Test	p-verdi	Effektstørrelse	Effektindikasjon
Rotasjon	Friedman + post-hoc	p-verdi < 0,001	0,055	Liten
Zoom	Friedman + post-hoc	p-verdi < 0,001	0,012	Liten
Forskyvning	Wilcoxon signed rank-test	p-verdi < 0,001	0,127	Liten
Flipp	Wilcoxon signed rank-test	p-verdi < 0,001	0,473	Moderat

Etter Friedmantest, ble Nemenyi post-hoc testen brukt til å se hvilke nivåer av augmenteringsteknikkene rotasjon og zoom som ga signifikante forskjeller mellom segmenteringsytelsene. For teknikken rotasjon var det signifikant forskjell mellom alle nivåer ettersom oppnådd p-verdi var mindre enn 0,001, med unntak av rotasjonsnivå 30 grader mot rotasjonsnivå 60 grader som ga en p-verdi lik 0,73. For zoom augmentering ga Nemenyi post-hoc testen p-verdi lik 0,001 mellom nivåene [0,5, 1,5] mot [0,8, 1,2] og p-verdi < 0,001 mellom nivå [0,5, 1,5] mot nivå 1, ingen zoom. Dette tyder på at det var signifikant forskjell mellom de oppgitte zoomnivåene. Zoomnivå [0,8, 1,2] mot ingen zoom fikk imidlertid en p-verdi lik 0,555, som indikerer at det ikke var signifikant forskjell på segmenteringsytelsen mellom disse nivåene for signifikansnivå 0,05.

De statistiske testene tyder på at det kan være fordelaktig å benytte augmenteringsteknikkene tilfeldig rotasjon mellom -90° og 90°, zoomfaktor mellom 0,5 og 1,5, et antall pikselforskyvning mellom -10 og 10 i *x*- og *y*-retning og flipping om *x*-aksen for segmenteringene i denne oppgaven, som vist i Tabell 5.4.

Tabell 5.4: Tabellen viser en oversikt over nivåene til de affine augmenteringene i eksperimentplan 1 som oppnådde høyest segmenteringsresultat basert på statistiske tester.

Augmenteringsteknikk	Nivå
Rotasjon	90
Zoom	[0,5, 1,5]
Forskyvning	[-10, 10]
Flipp	yes



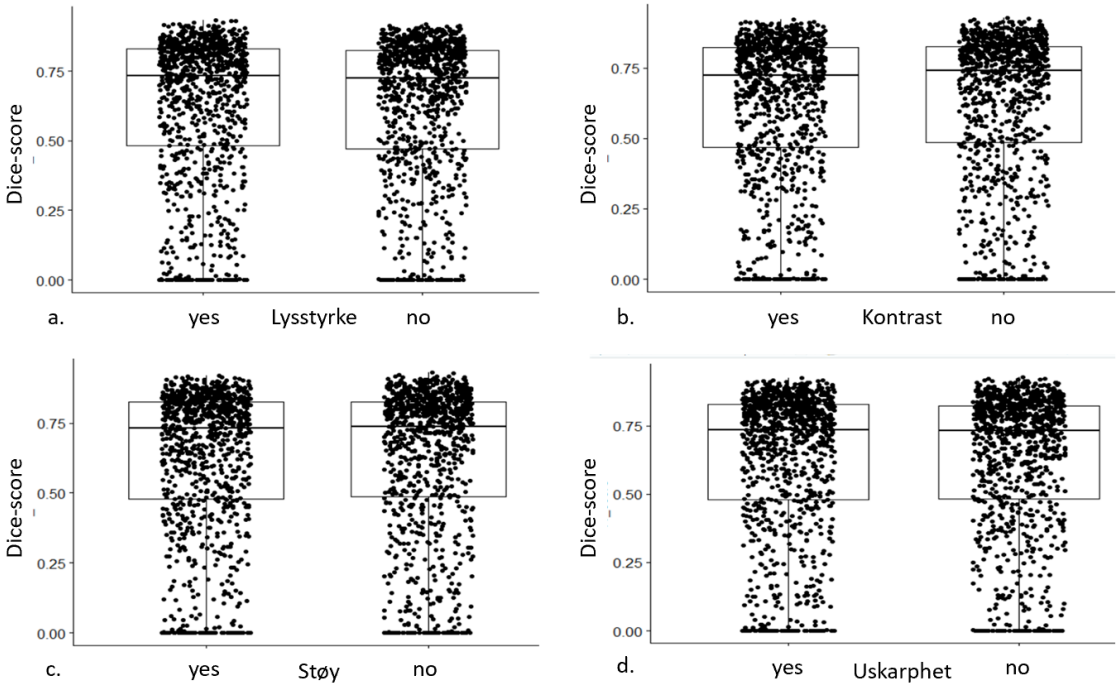
Figur 5.8: Figuren viser boksplott som visualiserer Dice-score per tverrsnitt per modell for hver affine augmenteringsteknikk (eksperimentplan 1). De svarte datapunktene i plottene representerer de observerte Dice-scorene for de ulike augmenteringsteknikknivåene. Boksen i plottene indikerer observasjonene mellom nedre kvartil, 25 %, og øvre kvartil, 75 %. Streken i boksen indikerer median-Dice-score per tverrsnitt oppnådd for hvert parameternivå.

5.3.2.2 Eksperimentplan 2

Tabell 5.5 gir en oversikt over de statistiske testene utført på resultatdataene til parameterne i eksperimentplan 2 (lysstyrke, kontrast, støy og uskarphet) hvor p-verdien og effektstørrelsen for de ulike parameterne er oppgitt. For alle punkt- og filteraugmenteringene oppnås en p-verdi høyere enn signifikansnivå 0,05 og 0,01. Dette indikerer at H0 ikke kan forkastes og at det derfor ikke kan konstateres at disse teknikkene ga signifikant forskjell i ytelse. Det er heller ikke mulig å se tydelige forskjeller mellom boksplottene i Figur 5.9 som visualiserer Dice-score per tverrsnitt per modell for punkt- og filteraugmenteringsteknikkene.

Tabell 5.5: Tabellen gir en oversikt over de statistiske testene utført på resultatdataene til augmenteringsteknikker i eksperimentplan 1 (lysstyrke, kontrast, støy og uskarphet), hvor p-verdien og effektstørrelsen er vist.

Parameter	Test	p-verdi	Effektstørrelse	Effektindikasjon
Lysstyrke	Wilcoxon signed rank-test	0,187	0,041	Liten
Kontrast	Wilcoxon signed rank-test	0,256	0,034	Liten
Støy	Wilcoxon signed rank-test	0,312	0,037	Liten
Uskarphet	Wilcoxon signed rank-test	0,170	0,042	Liten



Figur 5.9: Figuren viser boksplott som visualiserer Dice-score per tverrsnitt per modell for hver punkt- og filteraugmenteringsteknikk (eksperimentplan 2). De svarte datapunktene i plottene representerer de observerte Dice-scorene for de ulike augmenteringsteknikknivåene. Boksen i plottene indikerer observasjonene mellom nedre kvartil, 25 %, og øvre kvartil, 75 %. Streken i boksen indikerer median-Dice-score per tverrsnitt oppnådd for hvert parameternivå.

5.4 Eksperimentplan 3

Eksperimentmodell 46 med affine transformasjoner, spesifisert i Tabell 5.6, ble utnevnt til beste modell i eksperimentplan 1. Augmenteringene i denne modellen sammenfaller med nivåene valgt basert på de statistiske testene, vist i Tabell 5.4.

Tabell 5.6: Tabellen gir en oversikt over augmenteringsnivåene i eksperimentplan 1 som oppnådde høyest segmenteringsytelse basert på statistiske tester og gjennomsnittlig Dice-score per tverrsnitt på valideringssettet.

Eksperiment	Rotasjon	Zoom	Forskyvning	Flipp	Dice-score
46	90	[0,5, 1,5]	[-10, 10]	yes	0,676

Ettersom statistiske tester ikke fant signifikante forskjeller i punkt- og filteraugmenteringene i eksperimentplan 2, ble valget av beste nivå for disse augmenteringene basert på de innledende forsøkene, hvor eksperimentmodell 13 oppnådde høyest gjennomsnittlig Dice-score per tverrsnitt. Nivåene for punkt- og filteraugmenteringsteknikkene som eksperimentmodell 13 bestod av er vist i Tabell 5.7.

Tabell 5.7: Tabellen gir en oversikt over augmenteringsnivåene i eksperimentplan 2 som oppnådde høyest segmenteringsytelse basert på statistiske tester og gjennomsnittlig Dice-score per tverrsnitt på valideringssettet.

Eksperiment	Lysstyrke	Kontrast	Støy	Uskarphet	Dice-score
13	no	no	yes	yes	0,636

Fra eksperimentplan 1 og 2 ble kombinasjonen av augmenteringsnivåene som oppnådde høyest segmenteringsytelse, heretter kalt kombinasjonsmodellen, satt sammen til eksperimentplan 3. Nivåene til kombinasjonsmodellen og den oppnådde gjennomsnittlige Dice-scoren per tverrsnitt på valideringssettet er vist i Tabell 5.8. Denne ga noe lavere ytelse enn modell 46 som kun benyttet affine augmenteringsteknikker. Dette kan tyde på at augmenteringsteknikkene støy og uskarphet kan ha negativ innvirkning på segmenteringsresultatene.

Tabell 5.8: Tabellen gir en oversikt over augmenteringsnivåene til eksperimentplan 3 og den oppnådde gjennomsnittlige Dice-scoren per tverrsnitt på valideringssettet.

Rotasjon	Zoom	Forskyvning	Flipp	Lysstyrke	Kontrast	Støy	Uskarphet	Dice-score
90	[0,5, 1,5]	[-10, 10]	yes	no	no	yes	yes	0,666

5.5 Augmenteringsmodellen

Ettersom eksperimentmodell 46 med kun affine transformasjoner oppnådde høyest Dice-score, ble modellen utnevnt til beste modell blant de tre eksperimentplanene og videre kjørt på testsettene til OUS-datasettet og det eksterne Maastro-datasettet. Den beste modellen refereres heretter til som augmenteringsmodellen.

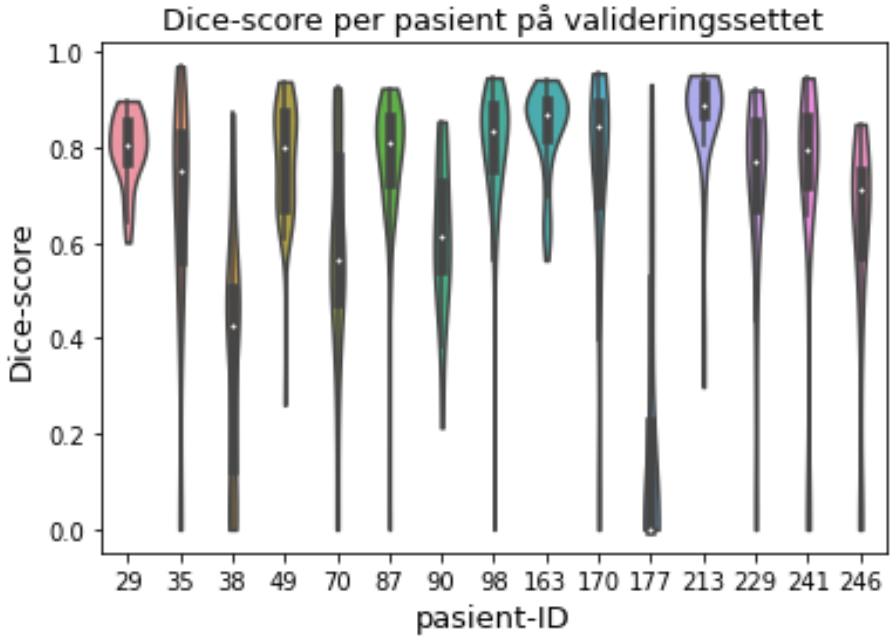
5.5.1 Ytelse oppnådd på valideringssettet

Augmenteringsmodellen oppnådde gjennomsnittlig Dice-score per tverrsnitt på 0,676, som vist i Tabell 5.6. Beregnede Dice-score per pasient for valideringssettet er gitt i Tabell 5.9, der gjennomsnittet er beregnet til 0,697 med standardavvik lik 0,211.

Tabell 5.9: Tabellen viser Dice-score per pasient til pasientene i valideringssettet oppnådd med den beste modellen, augmenteringsmodellen.

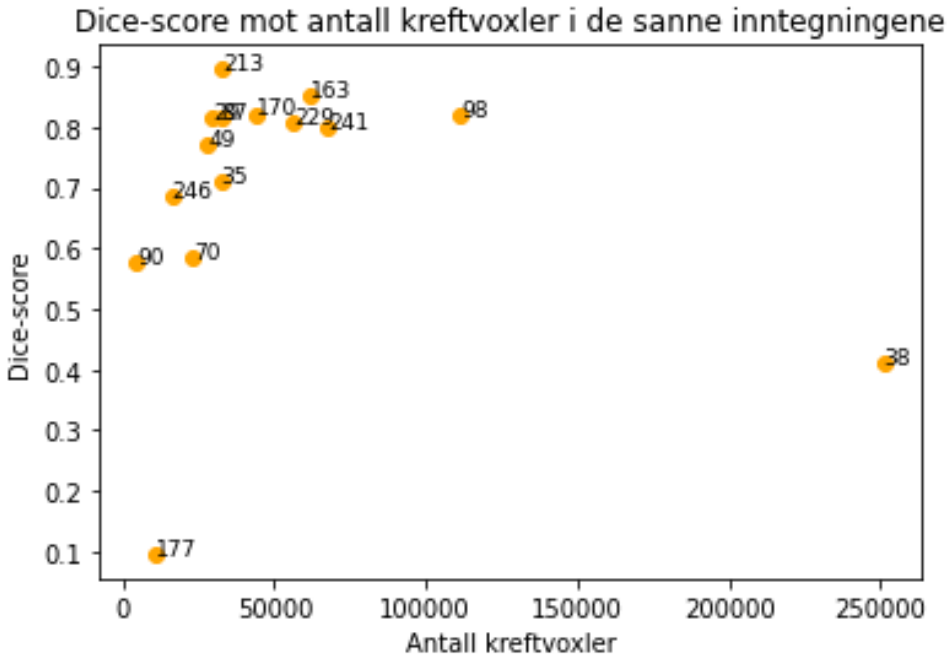
Pasient-ID	Dice-score
29	0,815
35	0,712
38	0,411
49	0,771
70	0,583
87	0,814
90	0,575
98	0,819
163	0,852
170	0,821
177	0,096
213	0,896
229	0,807
241	0,798
246	0,685

Fordelingen av Dice-score oppnådd på pasientenes tverrsnitt i valideringssettet ble visualisert ved hjelp av fiolinplott, vist i Figur 5.10. Augmenteringsmodellen oppnådde høy segmenteringsytelse for de fleste pasientene i valideringssettet, med unntak av pasient 38 og pasient 177, som oppnådde betraktelig lavere Dice-score. For pasient 177 er medianen, indikert med den hvite prikken, tilnærmet lik null. For pasient 29 og pasient 163 oppnådde augmenteringsmodellen høye Dice-scorer med lav varians. De resterende pasientene har alle lange haler som representerer Dice-score-utliggere, hvor den predikerte og den sanne inntegningen i noen tverrsnitt ikke overlapper og derfor har oppnådd ytelsen 0,000.



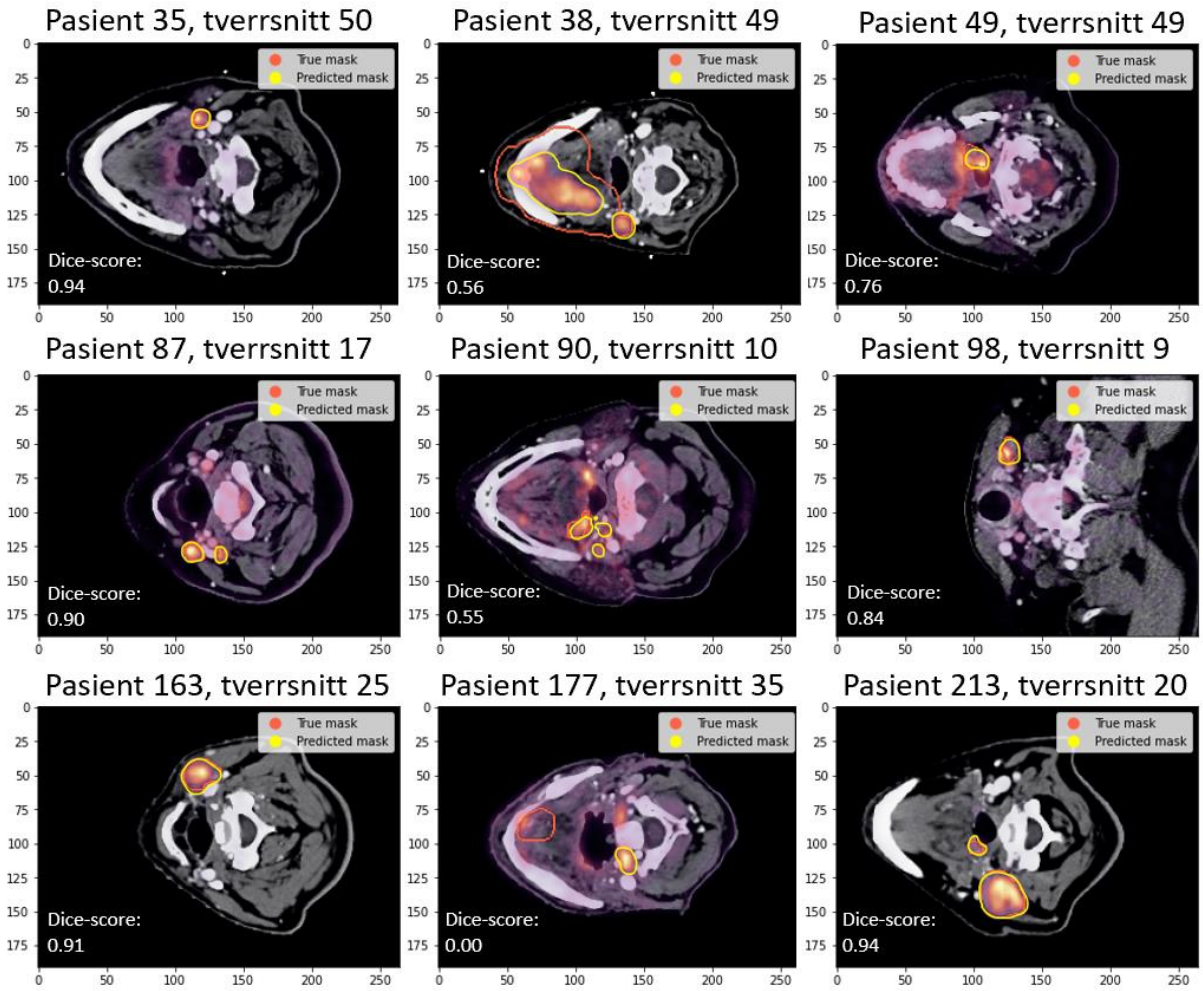
Figur 5.10: Figuren visualiserer fordelingen av Dice-score beregnet for tverrsnittene til hver pasient oppnådd med augmenteringsmodellen. Fiolinplottets form indikerer fordelingen av oppnådd Dice-score, der tykkelsen representerer antall tverrsnitt med den gitte Dice-scoren, mens lengden representerer variansen av Dice-score for hver pasient. Den svarte boksen indikerer verdier som ligger mellom første og tredje kvartil, mens den hvite prikken indikerer medianen til Dice-score for hver pasient.

Dice-score per pasient er i Figur 5.11 vist mot antall voxler av klassen ikke-friskt vev (kreftvoxler) i den sanne inntegningen til valideringssettet, for å undersøke om størrelsen av inntegningsvolum har påvirkende effekt på ytelsesmålet. Fra figuren observeres det at augmenteringsmodellen oppnådde lav segmenteringsytelse for pasient 177, som har blant de minste kreftsvulstene, og for pasient 38, som har et betydeligere høyere antall kreftvoxler i de sanne inntegningene sammenlignet med andre pasienter. Med unntak av pasient 38, ser det ut til at augmenteringsmodellen oppnådde høyere ytelse for pasienter med et større tumorvolum.



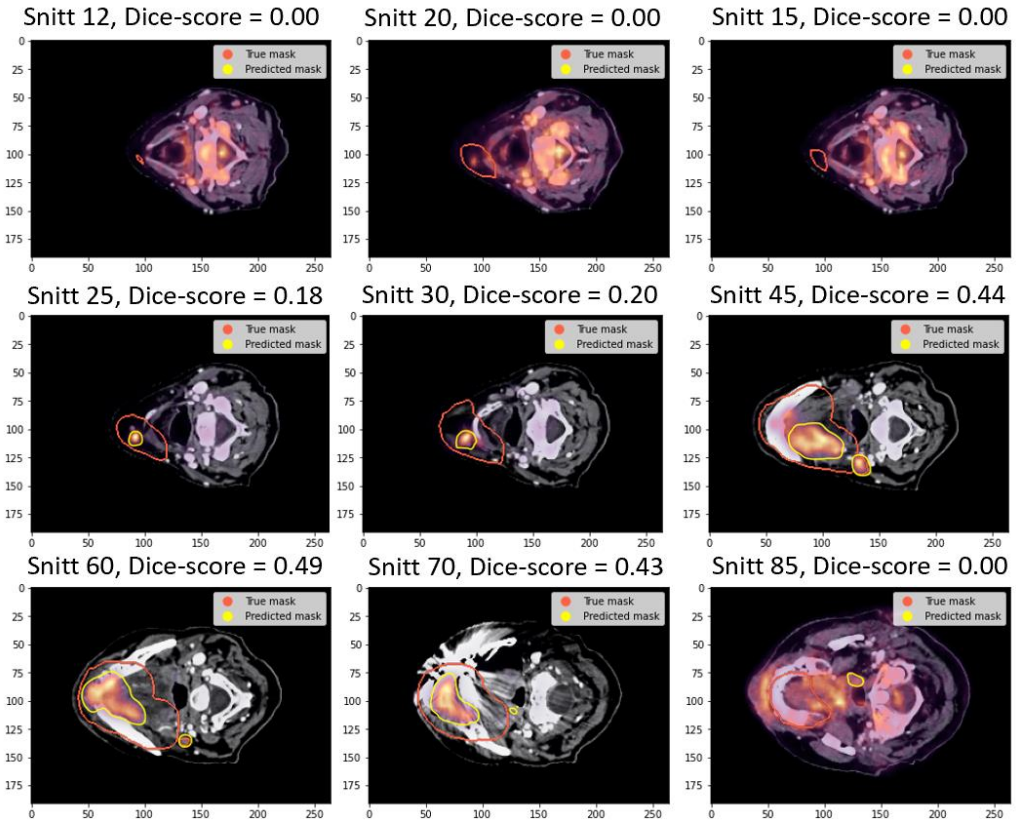
Figur 5.11: Spredningsplottet viser oppnådd Dice-score per pasient (oransje datapunkter med pasient-ID) mot antall voxler av klassen ikke-friskt vev (kreftvoxler) i den sanne inntegningen til valideringssettet. En voxel har dimensjon 1×1×1 mm^3.

Eksempler på kreftsvulstinntegninger utført av augmenteringsmodellen på tverrsnitt til pasienter i valideringssettet er vist i Figur 5.12. PET/CT-bildene representerer ulike tverrsnitt av pasientene tatt fra hodeskallen ned til midbrystparti. Den røde inntegningen representerer den sanne inntegningen utført av spesialister, mens den gule inntegningen representerer augmenteringsmodellens predikerte inntegning. Bildene viser at augmenteringsmodellen utførte lovende inntegning på de fleste pasienttverrsnittene avbildet, med unntak for pasient 38 og pasient 177, hvor den predikerte inntegningen ikke overlapper like godt med den sanne inntegningen i tverrsnittene.

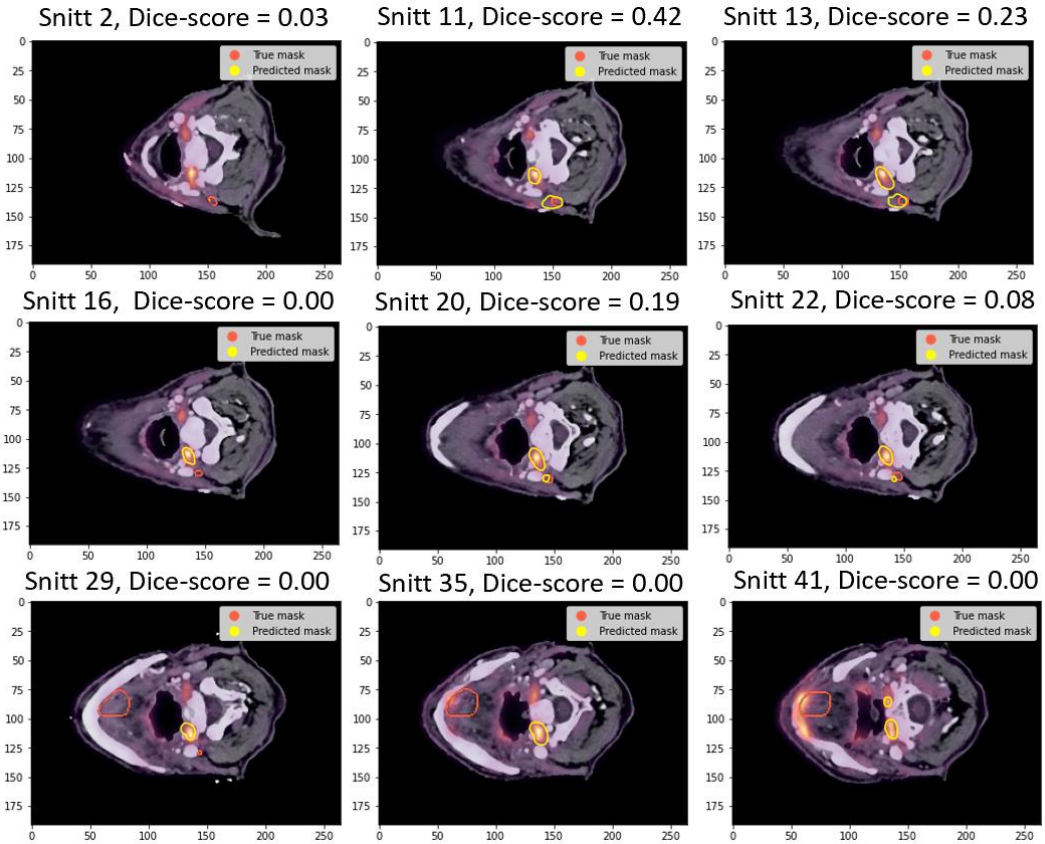


Figur 5.12: Figuren viser eksempler på segmenteringsresultater fra OUS-valideringssettet hvor den gule inntegningen er utført av augmenteringsmodellen, den beste modellen, og den røde inntegningen er utført av spesialister.

Flere eksempler av inntegninger utført av augmenteringsmodellen på tverrsnitt for pasient 38 og pasient 177 er vist i henholdsvis Figur 5.13 og Figur 5.14. Figurene viser en mengde tverrsnitt for pasient 38 og pasient 177, der segmenteringsytelsen er lik 0,000. Undersøkes disse tverrsnittene er det flere inputbilder hvor augmenteringsmodellen ikke har utført kreftsvulstinntegning til tross for at inputbildene inneholder sann inntegning utført av spesialister. Felles for disse snittene er områder med lavt FDG-opptak eller atypiske Hounsfield-verdier. Figur 5.13 og Figur 5.14 viser videre at modellen har tegnet inn typiske lyse områder som ikke har blitt tegnet inn av spesialistene. Noen tverrsnitt i Figur 5.13 viser også at modellen bare har tegnet inn områder i den sanne inntegningen hvor FDG-opptaket er høyt og ikke de resterende voxlene av den sanne inntegningen.



Figur 5.13: Figuren viser eksempler på segmenteringsresultater til pasient 38 fra valideringssettet hvor den gule inntegningen er utført av augmenteringsmodellen, den beste modellen, og den røde inntegningen er utført av spesialister.



Figur 5.14: Figuren viser eksempler på segmenteringsresultater til pasient 177 fra valideringssettet hvor den gule inntegningen er utført av augmenteringsmodellen, den beste modellen, og den røde inntegningen er utført av spesialister.

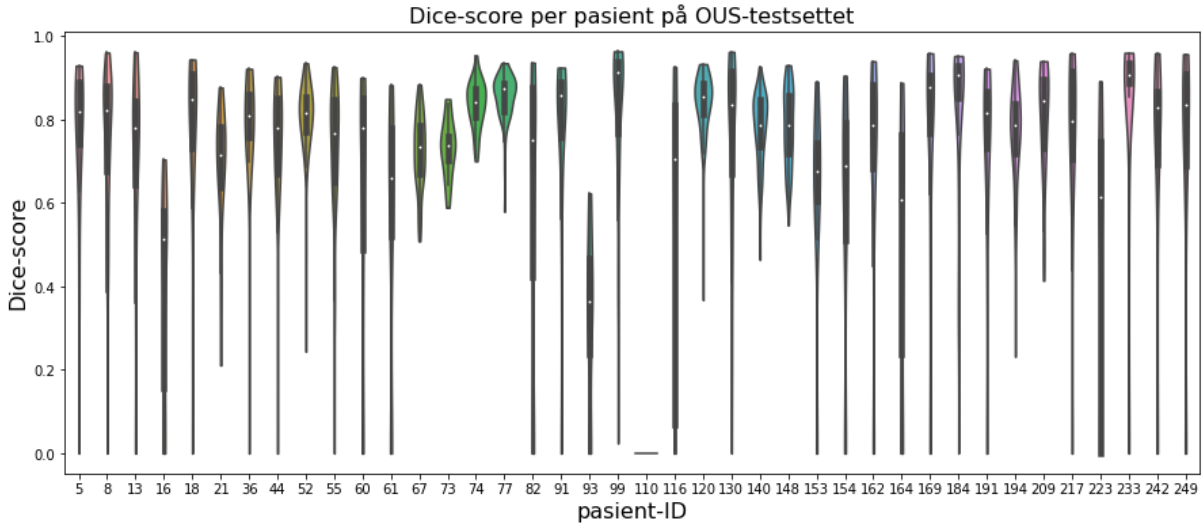
5.5.2 Ytelse oppnådd på OUS-testsettet

Augmenteringsmodellen ble videre kjørt på OUS-testsettet for å evaluere ytelsen til modellen på usett data. Modellen oppnådde en gjennomsnittlig Dice-score per tverrsnitt på 0,709 ± 0,257 (standardavvik). Beregnet ytelse per pasient for Dice-score, HD_{95} og MSD på testsettet til augmenteringsmodellen er gitt i Tabell D.1 vedlagt i Vedlegg D, hvor gjennomsnittsverdien og standardavviket til de ulike ytelsesmålene er gitt i Tabell 5.10. Fra Tabell D.2 observeres det varierende segmenteringsytelser, hvor modellen oppnår høy ytelse for de fleste pasienter. En bemerkelsesverdig pasient er pasient 110, hvor modellen oppnådde Dice-scoren på 0,000, HD_{95} på 42,6 mm og MSD på 25,0 mm, som indikerer at modellen har tegnet inn et område med relativt stor avstand fra den sanne inntegningen.

Tabell 5.10: Tabellen gir en oversikt over beregnet gjennomsnitt og standardavvik per pasient for ytelsesmålene Dice-score, HD_{95} og MSD for OUS-testsettet.

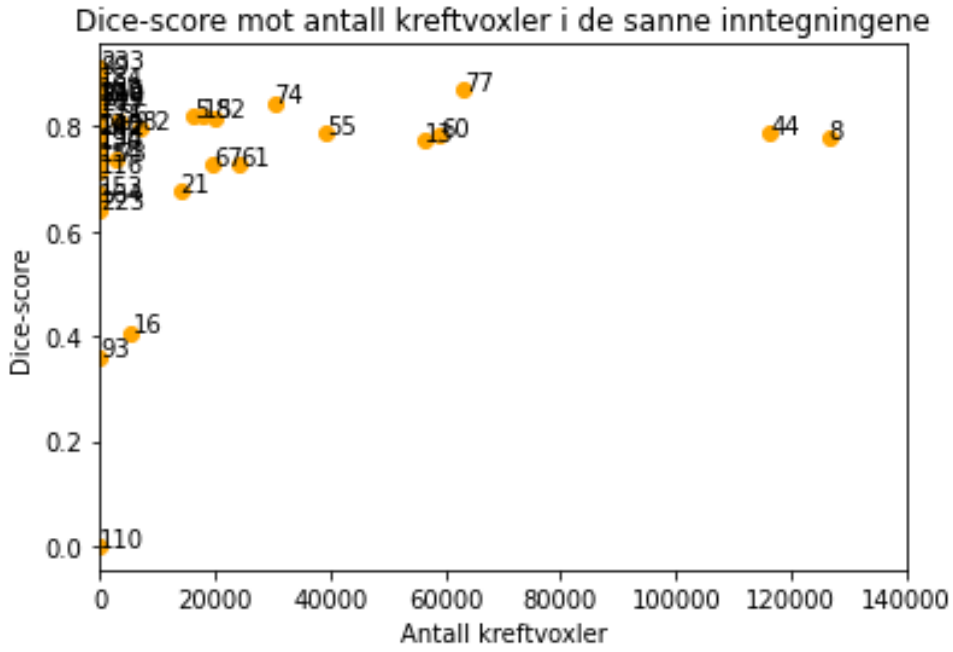
	Dice-score	HD_{95} [mm]	MSD [mm]
Gjennomsnitt ± standardavvik	0,751 ± 0,164	11,4 ± 11,9	1,09 ± 3,95

Visualisering av oppnådd Dice-score til tverrsnittene for hver pasient på OUS-testsettet er vist i Figur 5.15. Formen til fiolinplottene til pasientene 67, 73, 74 og 148 tyder på at modellen har oppnådd Dice-score med lav varians. De fleste andre pasienter har en lang hale som indikerer Dice-score-utliggere for noen av tverrsnittene. Augmenteringsmodellen oppnådde bemerkelsesverdig lav ytelse for pasient 16 og pasient 93 med Dice-score på henholdsvis 0,407 og 0,361, i tillegg til pasient 110 hvor oppnådd Dice-score var beregnet til null.



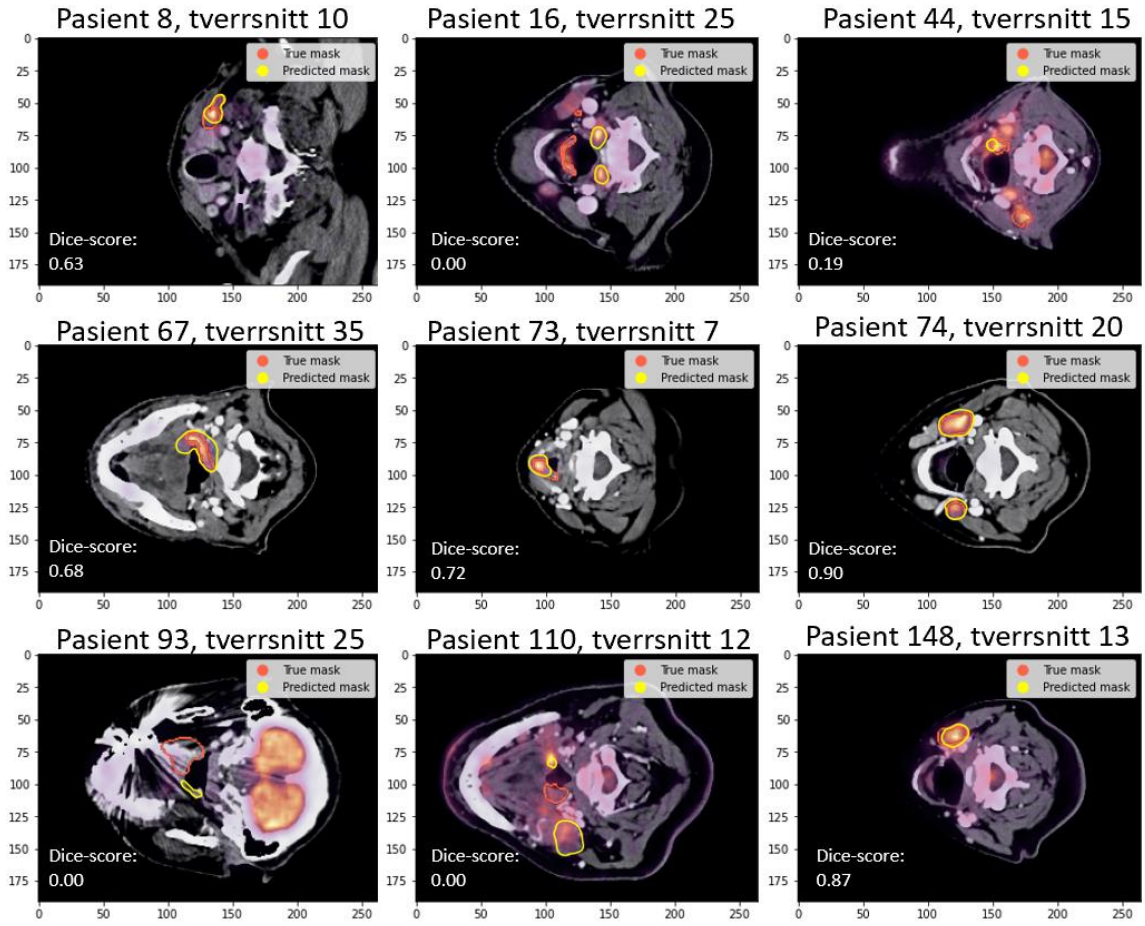
Figur 5.15: Figuren visualiserer fordelingen av Dice-score beregnet for tverrsnittene til hver pasient oppnådd med augmenteringsmodellen på OUS-testsettet. Fiolinplottets form indikerer fordelingen av oppnådde Dice-score, der tykkelsen representerer antall tverrsnitt med den gitte Dice-scoren, mens lengden representerer variansen av Dice-score for hver pasient. Den svarte boksen indikerer verdier som ligger mellom første og tredje kvartil, mens den hvite prikken indikerer medianen til Dice-score for hver pasient. Verdier utenfor den vertikale linjen representerer Dice-score-utliggere.

Oppnådd Dice-score i forhold til antall kreftvoxler (klassen ikke-friskt vev) i de sanne inntegningene i testsettet er visualisert i Figur 5.16. Fra figuren observeres det at augmenteringsmodellen oppnådde varierende ytelse for pasientene med et lavt antall kreftvoxler i de sanne inntegningene. Det observeres videre at modellen oppnådde lav ytelse for pasient 93 og 16 og lavest ytelse for pasient 110, som alle har et lavt antall kreftvoxler. Spredningsplottet tyder i tillegg på at augmenteringsmodellen oppnådde høy segmenteringsytelse for pasienter med et større tumorvolum, som for pasient 44 og pasient 8.



Figur 5.16: Spredningsplottet viser oppnådd Dice-score per pasient (oransje datapunkter med pasient-ID) mot antall voxler av klassen ikke-friskt vev (kreftvoxler) i den sanne inntegningen til OUS-testsettet. En voxel har dimensjon 1×1×1 mm^3.

Figur 5.17 viser eksempler på PET/CT-tverrsnitt til et utvalg av pasienter i testsettet, hvor den gule inntegningen representerer augmenteringsmodellens predikerte inntegning og den røde inntegningen representerer den sanne inntegningen utført av spesialister. Fra figuren observeres det at modellen utfører inntegning med høy overlapp for pasient 8, 73, 74 og 148. Disse pasientene har i tillegg oppnådd relativt lav HD_{95} og MSD, som vist i Tabell D.1 i Vedlegg D. For pasient 16 har modellen tegnet inn lyse områder (falske positive) som ikke har blitt klassifisert som klassen ikke-friskt vev av spesialistene. I noen av de avbildede tverrsnittene overlapper ikke den predikerte inntegningen med den sanne inntegningen. Dette gjelder for pasientene 16, 93 og 110 som i Figur 5.15 og Figur 5.16 var pasientene som oppnådde lav Dice-score og relativt høy HD_{95} og MSD, vist i Tabell D.1.



Figur 5.17: Figuren viser eksempler på segmenteringsresultater fra OUS-testsettet hvor den gule inntegningen er utført av augmenteringsmodellen, den beste modellen, og den røde inntegningen er utført av spesialister.

5.5.3 Ytelse oppnådd på Maastro-testsettet

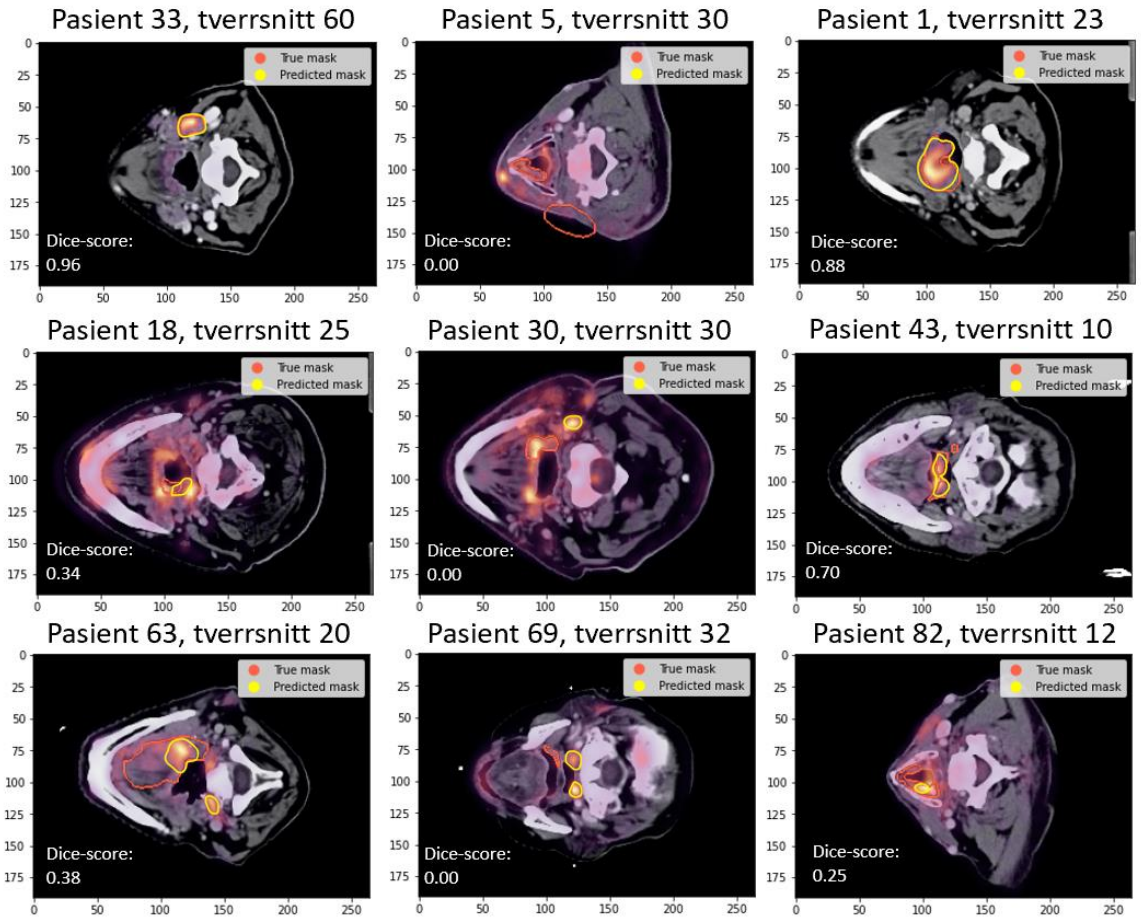
Augmenteringsmodellen som ble kjørt på Maastro-testsettet oppnådde den gjennomsnittlige Dice-scoren per tverrsnitt på 0,577 ± 0,310. Beregnet ytelse per pasient for Dice-score, HD_{95} og MSD er gitt i Tabell D.2 og Figur D.1 vedlagt i Vedlegg D, hvor beregnet gjennomsnitt og standardavvik til de ulike ytelsesmålene er gitt i Tabell 5.11. Fra Figur D.1 er variansen av Dice-score til de fleste pasientene stor. Augmenteringsmodellen gjør det dårligere på noen pasienter sammenlignet med andre. Dette gjelder spesielt for noen pasienter som er avbildet i Figur 5.18.

Tabell 5.11: Tabellen gir en oversikt over beregnet gjennomsnitt og standardavvik per pasient for ytelsesmålene Dice-score, HD_{95} og MSD for Maastro-testsettet.

	Dice-score	HD_{95} [mm]	MSD [mm]
Gjennomsnitt ± standardavvik	0,646 ± 0,185	14,4 ± 13,5	1,55 ± 3,36

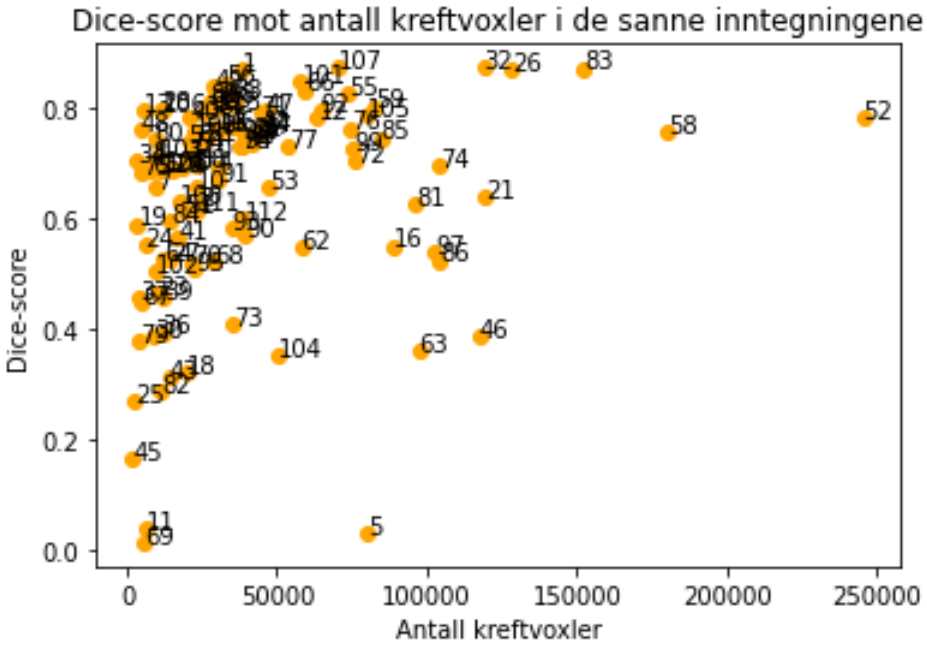
Figur 5.18 viser eksempler på tverrsnitt til pasienter i Maastro-settet som oppnådde lav ytelse, middels ytelse og høy ytelse. Et bemerkelsesverdig tverrsnitt er tverrsnitt 30 til pasient 5 som

ikke inneholder predikert inntegning og hvor den sanne inntegningen er inntegnet delvis utenfor den synlige avbildningen. Videre har modellen tegnet inn karakteristiske lyse områder for pasient 30 og pasient 63 som ikke har blitt tegnet inn av spesialistene, falske positive inntegninger. Felles for falske negative strukturer er områder med atypisk FDG-opptak eller atypisk Hounsfield-verdier.



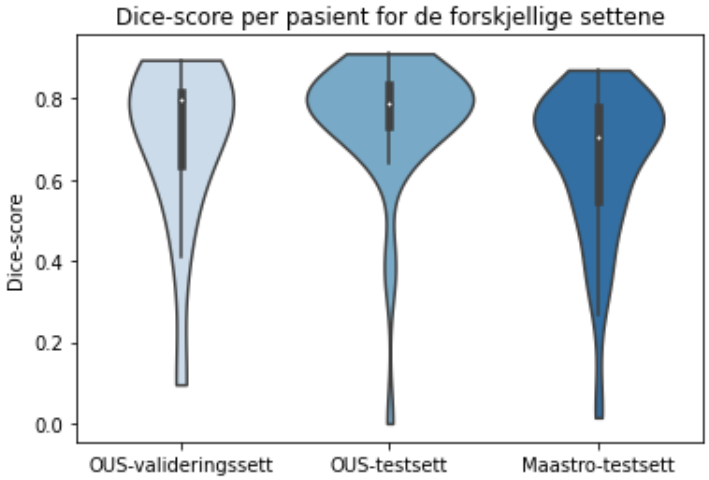
Figur 5.18: Figuren viser PET/CT-bilder av pasienter diagnostisert med hode- og halskreft fra Maastro-testsettet med predikerte inntegninger utført av augmenteringsmodellen (gul farge) og sanne inntegninger utført av spesialister (rød farge).

Figur 5.19 visualiserer oppnådd Dice-score i forhold til antall kreftvoxler (klassen ikke-friskt vev) i de sanne inntegningene i Maastro-testsettet. Fra figuren observeres det at augmenteringsmodellen oppnådde varierende ytelse for pasientene med et lavt antall kreftvoxler i de sanne inntegningene, og at modellen har lettere for å tegne inn krefttumorer og påvirkede lymfeknuter hos pasienter med et større antall kreftvoxler. Det observeres videre at modellen oppnådde lav ytelse for pasient 69 og 11, som har et relativt lavt antall kreftvoxler. Augmenteringsmodellen oppnådde i tillegg lav ytelse for pasient 5 som har et mye større tumorvolum sammenlignet med volumstørrelsen til andre pasienter som modellen oppnådde høyere ytelse for. I Tabell D.2 vedlagt i Vedlegg D er avstandsscoren HD_{95} og MSD for de nevnte pasientene høy, som indikerer lav segmenteringsytelse.



Figur 5.19: Figuren viser et spredningsplott som visualiserer oppnådd gjennomsnittlig Dice-score per pasient (oransje datapunkter med pasient-ID) mot antall voxler av klassen ikke-friskt vev (kreftvoxler) i den sanne inntegningen til Maastro-testsettet. En voxel har dimensjon 1×1×1 mm^3.

Figur 5.20 visualiserer oppnådd Dice-score per pasient på OUS-valideringssettet (15 pasienter), OUS-testsettet (40 pasienter) og det eksterne Maastro-testsettet (114 pasienter). Figuren viser at augmenteringsmodellen oppnådde høyest Dice-score på OUS-testsettet. Det observeres videre at OUS-valideringssettet har den høyeste medianen.



Figur 5.20: Fiolinplottene visualiserer fordelingen av Dice-score per pasient augmenteringsmodellen oppnådde på de forskjellige datasettene. Fiolinplottets form indikerer fordelingen av oppnådd Dice-score, der tykkelsen representerer antall tverrsnitt med den gitte Dice-scoren, mens lengden representerer variansen av Dice-score for hver pasient. Den svarte boksen indikerer verdier som ligger mellom første og tredje kvartil, mens den hvite prikken indikerer medianen til Dice-score for hver pasient. Verdier utenfor den vertikale linjen representerer Dice-score-utliggere.

Kapittel 6: Diskusjon

6.1 Målet med masteroppgaven

2D U-Net-modellen benyttet for segmentering av hode- og halskreftsvulster og påvirkede lymfeknuter i studiene til Moe et al. [21], Groendahl et al. [11] og masteroppgaven til Huynh [22] har videre blitt utforsket i denne oppgaven. Ved bruk av dataaugmenteringsteknikker på inputbildene i treningsdatasettet har denne masteroppgaven satt søkelys på å forbedre 2D U-Net-modellens segmenteringsytelse. Totalt 65 eksperimenter med ulike nivåer av augmenteringsteknikker har blitt utført. Modellene oppnådde lovende resultater, der den gjennomsnittlige Dice-scoren per tverrsnitt på valideringssettet var i intervallet 0,62 til 0,68. Som vist i Kapittel 5 ble den beste modellen, basert på statistiske tester og gjennomsnittlig Dice-score per tverrsnitt, videre kjørt på OUS-testsettet og det eksterne Maastro-testsettet der modellen oppnådde en gjennomsnittlig Dice-score per pasient på henholdsvis 0,75 og 0,65. Ifølge Zijdenbos et al. [75] er en Dice-score over 0,70 regnet som en god segmentering. Den beste av eksperimentmodellene oppnådde dermed en god ytelse på OUS-testsettet.

6.2 Effekt av augmenteringsteknikker

I denne masteroppgaven har ulike nivåer av tradisjonelle augmenteringsteknikker innenfor kategoriene affine transformasjoner og punkt- og filteroperasjoner blitt testet ut. Som beskrevet i Kapittel 5 ble augmenteringsteknikkenes innvirkning på segmenteringsresultatet undersøkt ved bruk av visualiseringsplott (Figur 5.3, Figur 5.4, Figur 5.5, Figur 5.8 og Figur 5.9) og statistiske tester. Fra de statistiske testene, samt visualiseringsplottene, er det tydelig at valg av augmenteringsnivå påvirker segmenteringsytelsen. Dette gjelder spesielt for de affine transformasjonene i eksperimentplan 1, mens punkt- og filteroperasjonene ikke hadde signifikant effekt. Kombinasjonen av augmenteringsteknikker som oppnådde høyest segmenteringsytelse var eksperimentmodell 46 i eksperimentplan 1, som bestod av de affine transformasjonsnivåene tilfeldig rotasjon mellom -90° og 90°, zoomfaktor mellom 0,5 og 1,5, et antall pikselforskyvning mellom -10 og 10 i x- og y-retning og flipping om x-aksen. Eksperimentmodell 46 refereres videre til som augmenteringsmodellen.

6.2.1 Affine transformasjoner

Plassering av pasienten i PET/CT-skanneren er en faktor som kan påvirke PET/CT-bilder. Det samme gjelder hvis bildene er tatt med forskjellige PET/CT-skannere, utført av ulike spesialister eller tatt på andre institusjoner [35, 56]. En dyplæringsmodell som trener på bilder med samme karakteristiske trekk, for eksempel lik orientering av pasientene, vil ikke være

robust ovenfor små forandringer av pasientplasseringen, noe som kan føre til feilklassifiseringer. Ved bruk av affine transformasjoner kan pasientens plassering i inputbildet endres, samtidig som objektets struktur og form bevares. Dyplæringsmodellen som trener på de augmenterte inputbildene kan på denne måten lære ulike orienteringer av pasientene.

Augmenteringsteknikken rotasjon kan skape troverdige treningseksempler ved å endre vinkel i inputbildet, som vil kunne bidra til å forhindre mulige biaser av kreftsvulstlokasjonen. Resultater fra de innledende forsøkene og de statistiske testene viste at ytelsen økte med økt rotasjonsnivå, med unntak av overgangen mellom rotasjonsnivå 30 og rotasjonsnivå 60. Nedgangen kan imidlertid skyldes tilfeldigheter som kan komme av startvektene som nettverket begynner å trene med, eller rekkefølgen på inputbildene nettverket presenteres for [25]. Rotasjonsnivået som oppnådde høyest Dice-score var rotasjonsnivå 90 grader, tilfeldig rotasjon mellom -90° og +90°. Ettersom økt rotasjonsvinkel så ut til å gi økt Dice-score, kan det virke som at mer ekstreme augmenteringsnivåer fører til forbedret ytelse. En rotasjonsvinkel større enn 90 grader vil imidlertid ikke gi mening å teste, ettersom pasienten ikke kan rotere hodet til en slik vinkel og det vil derfor ikke gi representative treningseksempler for datasettet.

Fra de statistiske testene oppnådde zoomnivået med videst intervall høyest Dice-score. Funn for parameteren tyder derfor på at ekstreme augmenteringsverdier kan føre til økt ytelse, som også ble observert for augmenteringsteknikken rotasjon. Det er derimot ikke gitt at ytelsen vil øke i takt med økt augmenteringsverdi. Forstørres inputbildet med en faktor som gjør at viktig informasjon i bildet går tapt, kan augmenteringsteknikken potensielt føre til feilaktige klassifiseringer. Det samme gjelder hvis inputbildet forminskes med en faktor som gjør at kreftsvulstpikslene blir vanskelig å identifisere.

Teknikken forskyvning, som endrer plasseringen av bildet, kan på lik linje som teknikkene rotasjon og zoom brukes for å forhindre eventuelle biaser av kreftsvulster i inputdata, ettersom modellen får mulighet til å lære romlige invariante representasjoner [56]. Fra den statistiske testen ble det vist at forskyvning hadde signifikant effekt på segmenteringsytelsen, men at effektstørrelsen var liten. Dette kan stemme med observasjoner i fiolinplott Figur 5.3c., som viste at en stor andel eksperimentmodeller uten påført forskyvning også oppnådde høy ytelse. Tidligere studier tyder i tillegg på at visse konvolusjonsnettverk er forskyvnings-invariante, som blant annet kan komme av makssamlingslagene eller det stadige økte

mottakelige feltet («receptive field») til noder i de påfølgende lagene [56, 76]. Ettersom U-Net-arkitekturen inneholder makssamlingslag, kan dette derfor være en grunn for den lave innvirkningen augmenteringsteknikken hadde på segmenteringsytelsen. Likevel tyder studien til Kauderer-Abrams [76], som undersøkte forskyvnings-invarians til konvolusjonsnettverk med ulik arkitektur, at arkitekturen alene ikke fører til forskyvnings-invarians. Studien viste videre at augmenteringsteknikken forskyvning, benyttet på treningsdatasettet, førte til at nettverket ble invariant ovenfor bilder som var forskjøvet [76]. Funn tyder dermed på at teknikken kan ha positiv effekt. Dette bekreftes videre i denne masteroppgaven fra den statistiske testen som indikerte at påført forskyvning førte til økt ytelse. Påføres inputbilder derimot en drastisk forskyvning, der bildet plasseres delvis utenfor rammen, kan viktig informasjon forvrenges eller gå tapt, noe som kan ha negativ innvirkning på modellens ytelse.

Augmenteringsteknikken flipp har i tidligere studier [26, 27] vist lovende resultater for forbedring av ytelsesmål og det er også tilfelle i denne masteroppgaven. Fra Tabell 5.3 indikerte den statistiske testen signifikant innvirkende effekt, der effektstørrelsen ble beregnet til moderat. Sammenlignet med de andre affine transformasjonene, hvor effektstørrelsen var liten, utgjør dermed flipp en avgjørende forskjell på segmenteringsytelsen. Dette ble videre bekreftet i jitter-plottene i Figur 5.4 som viste at den todelte fordelingen av Dice-score i Figur 5.3a., Figur 5.3b. og Figur 5.3c. skyldes parameteren flipp. Det kan dermed konstateres at modellene som trener på speilede bilder om *x*-aksen, vertikal flipp, har positiv effekt på ytelsesmålet. Horisontal flipp, speiling om *y*-aksen, har imidlertid ikke blitt testet i denne masteroppgaven ettersom et slikt bak-frem-bilde ikke avdekker nye varianter av de medisinske bildene og heller ikke naturlig ville forekommet i datasettet. I klassifiseringsoppgaver av for eksempel hunder og katter ville det vært mer naturlig å foreta en horisontal flipp, speiling om *y*-aksen, ettersom modellen ikke nødvendigvis ville hatt noe nytte av å trene på en opp-ned-katt. Innen medisin vil derimot en svulst som speiles om *x*-aksen fremdeles se ut som en troverdig svulst og dyplæringsmodellen vil dermed ha nytte av en slik speiling [26].

6.3 Punkt- og filteroperasjoner

Kontrast, oppløsning og støy er faktorer som kan påvirke kvaliteten til et CT-bilde [35]. I tillegg vil avstanden mellom pasienten og PET/CT-skanneren, bevegelse under skanningen og bruk av forskjellige apparater på forskjellige institusjoner, kunne skape variasjoner i PET/CT-bildene [35, 56]. Punkt- og filteroperasjoner ble derfor benyttet som augmenteringsteknikker i eksperimentplan 2 for å kunne gjenskape slike variasjoner.

Ifølge den statistiske testen vist i Tabell 5.5, var det ingen signifikant forskjell på innvirkningen av Dice-score når punkt- og filteroperasjonsteknikkene ble påført inputbildene i treningsdatasettet. En faktor som kan ha hatt betydning for resultatet kan være intervallene for de ulike teknikkene definert i konfigurasjonsfilen. Augmenteringsfaktoren bør ligge innenfor et intervall som endrer bildene til troverdige nye eksempler. Er faktoren for stor eller for liten kan teknikken generere unaturlige treningseksempler som kan føre til feilklassifiseringer. Genererte treningseksempler som skiller seg for lite fra det originale bildet, eller genererte treningseksempler hvor sammenhengen mellom inputbildet og den sanne inntegningen har gått tapt, vil kunne innvirke negativt på segmenteringsytelsen.

Augmenteringsteknikken kontrast og lysstyrke kan være gunstig i tilfeller der det er vanskelig å skille piksler fra hverandre. Ved å øke kontrasten i slike FDG-PET/CT-bilder kan kreft-piksler, hvor det radioaktive stoffet FDG akkumuleres grunnet høy metabolsk aktivitet, tydeligere sees. Endres dog kontrasten eller lysstyrken i bildet med en for stor eller for liten faktor kan det bli vanskelig å skille piksler som indikerer friskt-vev fra piksler som indikerer ikke-friskt vev. Ettersom fiolinplott Figur 5.5a. og Figur 5.5b. indikerte at ytelsen ble redusert for eksperimentmodeller hvor augmenteringsteknikkene ble benyttet på bildene, kan det tyde på at modellene har hatt negativ effekt av augmenteringsteknikkene.

I medisinske bilder er støy og uskarphet vanlige fenomener som kan forekomme [35]. Fra fiolinplott Figur 5.5c. og Figur 5.5d. så det ut til at augmenteringsteknikkene støy og uskarphet hadde positiv effekt på Dice-score. Likevel førte ikke teknikkene til signifikant økt segmenteringsytelse, til tross for at teknikken uskarphet i tidligere studie har vist å gi forbedring av ytelsesmålet [26]. Teknikken støy har imidlertid vist å være mindre effektiv enn andre augmenteringsteknikker [23, 26], som kan samsvare med funn fra denne masteroppgaven.

6.3.1 Effekten av augmenteringsteknikker i tidligere arbeid

Bruk av augmenteringsteknikker på inputdata har i flere tidligere studier vist å kunne forbedre ytelsen til dyplæringsmodeller [26-29]. Teknikkene brukes blant annet for å forhindre overtilpasning og for å håndtere ubalanserte klasser, som ofte er en utfordring i medisinske datasett [30]. Ved å utføre forskjellige transformasjoner på treningsdata samtidig som forholdet til den sanne inntegningen holdes konstant, skapes nye varianter av treningseksempler [29]. Nettverkes evne til å memorere detaljer i treningsdataene reduseres på denne måten, samtidig som treningsdataene og modellens kompleksitet kan utnyttes mer

effektivt. Dype nevrale nettverk trent på et variert datasett vil kunne ignorere små forandringer i inputbilder, som ofte kan forekomme hvis bildene er tatt med forskjellige bildeteknikker eller apparater. Generaliseringsevnen til modellen kan dermed økes og dyplæringsmodellen kan potensielt benyttes på datasett innsamlet med andre prosedyrer [29].

Augmenteringsteknikkene som er undersøkt i denne masteroppgaven er alle tradisjonelle augmenteringsteknikker. Disse teknikkene har i flere studier vist seg å være effektive og populære teknikker, som i noen klassifiseringsoppgaver har utkonkurrert andre metoder [23, 27]. Ifølge studien til Mikołajczyk og Grochowsk [23] er de tradisjonelle augmenteringsteknikkene noen av de mest populære teknikkene å bruke, ettersom de både er raske og enkle å implementere sammenlignet med andre nyere teknikker. Dette underbygges videre i studien til Nalepa et al. [56] som undersøkte ulike augmenteringsteknikker benyttet i sammenheng med hjerne-tumorsegmentering i MR-bilder. Studien konkluderte med at tradisjonelle teknikker, som affine transformasjoner, er de mest brukte teknikkene, ettersom de lager anatomisk realistiske hjerne-avbildninger og er enkle å implementere. I tillegg viste studien at augmenteringsteknikkene flipp og rotasjon ga spesielt økt effekt [56]. Dette samsvarer med funn i denne masteroppgaven hvor den beste kombinasjonen av augmenteringsteknikker bestod av affine transformasjoner, og hvor teknikken flipp hadde størst innvirkning på segmenteringsytelsen.

Tradisjonelle augmenteringsteknikker tilføyer imidlertid ikke mye ny informasjon til datasettet sammenlignet med GANs, som genererer syntetiske treningseksempler med mye variabilitet [23, 28]. I studien til Frid-Adar et al. [28] ble det vist at klassifiseringsytelsen av leverlesjoner til CNN-modeller økte ytterligere ved å kombinere bildene påført tradisjonelle augmenteringsteknikker med bildene generert av GANs. I studien til Perez og Wang [27] oppnådde imidlertid bruk av GANs lavere klassifiseringsytelse enn bruk av tradisjonelle augmenteringsteknikker. Som nevnt i delkapittel 3.8.4 er GANs både krevende å implementere og krever mye prosesseringskraft og tid [23]. I tillegg kan teknikken i noen tilfeller generere for like treningseksempler som gjør at modellens generaliseringsevne ikke øker [56]. GANs har dermed ikke blitt benyttet i denne masteroppgaven, men bør være en del av videre arbeid for å undersøke om kombinasjonen av augmenterte inputbilder og kunstig fremstilte inputbilder kan øke ytelsen, som vist i studien til Frid-Adar et al. [28].

En annen augmenteringsteknikk som har vist lovende resultater innenfor biomedisinske segmenteringsoppgaver, men som heller ikke har blitt testet ut i denne masteroppgaven, er elastisk deformasjon. Som tidligere nevnt ble elastisk deformasjon påført bilder tatt med

mikroskop i studien til Ronneberger et al. [50] for å skape et større og variert datasett. Teknikken kan imidlertid skape anatomisk ukorrekte treningseksempler og er i tillegg tidkrevende og vanskelig å implementere sammenlignet med andre augmenteringsteknikker [56]. Teknikken bør likevel være del av videre arbeid for å undersøke om elastisk deformasjon kan overgå ytelsen oppnådd med augmenteringsmodellen trent på datasett augmentert med de affine transformasjonene.

Ved å kombinere en rekke augmenteringsteknikker kan størrelsen på datasettet øke betraktelig. Et stort datasett er ikke alltid en fordel og kan i noen tilfeller føre til en forverring av modellens ytelse hvis datasettet ikke inneholder representative treningseksempler. I tillegg øker krav til minne, prosesseringskraft og treningstid [30]. Kvaliteten på treningseksemplene er i mange tilfeller viktigere enn kvantiteten [77]. I studien til Kleppe et al. [29] ble det vist at ved å foreta tilfeldige fargemodifikasjoner på inputdata kan augmentasjon i noen tilfeller kompensere for små og begrensede datasett, men ved å videre øke fargeforandringene gikk relevant informasjon tapt og ytelsen ble forverret [29]. Det er derfor viktig å evaluere om augmenteringsteknikkene skaper representative treningseksempler. Som beskrevet tidligere, så det ut til at augmenteringsteknikkene lysstyrke og kontrast førte til redusert ytelse. Ytelsen ble også redusert når teknikkene støy og uskarphet ble kombinert med de affine transformasjonene i eksperimentplan 3, vist i Tabell 5.8. Det kan derfor være at teknikkene skapte treningseksempler som ikke var representative for datasettet og som dermed førte til feilklassifiseringer, slik som i studien til Kleppe et al. [29]. Videre arbeid bør av den grunn undersøke andre nivåer av punkt- og filteroperasjonene.

Valg av augmenteringsteknikker avhenger av klassifiseringsoppgaven modellen står ovenfor. Ut ifra oppgaven, vil det involverte datasettet tolerere ulike mengder av augmenteringsteknikker før sammenhengen mellom inputbildene og sann inntegning forsvinner [29]. I studien til Hussain et al. [26] ble forskjellige augmenteringsteknikker undersøkt for klassifiseringsoppgaver av medisinske avbildninger. Studien fant blant annet ut at teknikker som bevarte egenskaper fra originalbildet oppnådde høyere ytelse. Fra de testede teknikkene oppnådde augmenteringsteknikken flipp og justering av uskarpheten høyest ytelsesmål, mens teknikken støy hadde negativ innvirkning på ytelsesmålet [26]. Funn fra studien til Hussain et al. [26] samsvarer med funn for augmenteringsteknikkene flipp og støy. Teknikken uskarphet førte imidlertid ikke til signifikant økt ytelse.

Ifølge Shijie et al. [78] kan valg av augmenteringsstrategier være mer avgjørende enn valg av nettverkets struktur. Det er imidlertid ikke gjort studier, fra forfatterens kunnskap, som har

kommet frem til en generell valgstrategi av augmenteringsteknikk. Augmenteringsmetoder som øker ytelsen i ett tilfelle, kan forverre ytelsen i et annet tilfelle. Valg av augmenteringsteknikker må derfor tilpasses til den individuelle oppgaven og datasett modellen står ovenfor [29].

6.4 Begrensninger som gjelder valg av augmenteringsmodellen

Valget av augmenteringsmodellen (tilfeldig rotasjon mellom -90° og 90°, zoomfaktor mellom 0,5 og 1,5, et antall pikselforskyvning mellom -10 og 10 i *x*- og *y*-retning og flipping om *x*-aksen) ble basert på resultatene fra de innledende forsøkene og de statistiske testene. Det er imidlertid flere faktorer som kan ha hatt innvirkning på valg av beste modell og derfor medfører en viss usikkerhet.

6.4.1 Statistiske tester

Den opprinnelige planen var å bruke N-veis ANOVA for å analysere resultatdataene. N-veis ANOVA ser på parameterne hver for seg og interaksjonene dem imellom. Ettersom betingelsen for normalfordeling ikke var oppfylt, ble Friedmantest etterfulgt av Nemenyi post-hoc test og Wilcoxon signed rank-test benyttet for evaluering. Friedmantest og Wilcoxon signed rank-test tar ikke hensyn til interaksjoner. Ettersom alle eksperimentmodellene utført i denne masteroppgaven bestod av kombinasjoner av ulike augmenteringsteknikker, burde interaksjonene blitt tatt hensyn til i valg av beste modell. Videre arbeid bør derfor undersøke interaksjonene mellom de ulike augmenteringsteknikknivåene.

6.4.2 Ytelsesmål

Overlappsmålet Dice-score, benyttet i denne oppgaven, er ifølge studien til Guo et al. [8] gullstandarden innenfor segmenteringsoppgaver, men kan i noen tilfeller gi misvisende resultater, som beskrevet i delkapittel 3.9.2. Avstandsmålene HD_{95} og MSD ble derfor i tillegg benyttet som ytelsesmål på testsettene for å korrekt kunne evaluere modellens totale kreftsvulstinntegning. Avstandsmålene ble imidlertid ikke benyttet på valideringssettet, og valg av beste modell, augmenteringsmodellen, er derfor bare basert på overlappsmålet som medfører en viss usikkerhet i modellens totale ytelse.

Valg av augmenteringsmodellen ble i tillegg basert på Dice-score per tverrsnitt og ikke Dice-score per pasient, som er ytelsesmålet oppgitt for baselinemodellen i både studiene til Moe et al. [21] og Groendahl et al. [11] som augmenteringsmodellen sammenlignes med. Dice-score per pasient kan gi et mer helhetlig bilde av kreftsvulstinntegningen utført av modellen fremfor å se på tumorinntegningen i hvert tverrsnitt. Bruk av Dice-score per pasient er i tillegg lik

evalueringsmåten brukt i andre medisinske studier som undersøker interobservasjonsvariabiliteter [79]. Fremtidig arbeid bør derfor undersøke om ytelsesmålene Dice-score per pasient og avstandsmålene samsvarer med resultatene oppnådd med Dice-score per tverrsnitt i denne masteroppgaven.

6.4.3 Kjøring av eksperimenter

Inputbildene i treningsdatasettet ble i løpet av treningsfasen påført augmenteringsteknikker med en gitt sannsynlighet (beskrevet i delkapittel 4.6.3). Dette tilsier at ikke nødvendigvis alle eksperimentene med like augmenteringsteknikknivåer trente på datasett med samme fordeling av augmenterte bilder. Den tilfeldige faktoren som velges fra et definert område for hver augmenteringsnivå kan i tillegg gjøre at noen modeller inneholder mer ekstreme augmenteringer enn andre, som kan skape variasjoner i ytelsesmål til de kjørte eksperimentene og dermed påvirke evalueringen av augmenteringsparameterne.

Antall ganger eksperimentmodellene kjøres vil føre til variasjoner i oppnådd ytelsesscore, som kan komme av de tilfeldige vektene satt i starten av treningsfasen og rekkefølgen på inputbildene [25]. Hvert eksperiment ble i utgangspunktet kjørt en gang hver, ettersom tiden og prosesseringskraften var begrenset. Eksperimentmodellen uten påført augmenteringsteknikker ble imidlertid kjørt to ganger, en gang i eksperimentplan 1 og en gang i eksperimentplan 2, der eksperimentene oppnådde gjennomsnittlig Dice-score per tverrsnitt på henholdsvis 0,61 og 0,62 på valideringssettet, vist i Tabell B.1 og Tabell B.2. Dette viser at naturlige variasjoner i ytelsen vil forekomme for hver gang modellen trener. Ideelt burde hvert eksperiment blitt kjørt flere ganger for å få et bedre estimat på eksperimentmodellenes ytelsesmål. Fremtidige eksperimenter bør derfor kjøres flere ganger, eller bruke forhåndsbestemte vekter.

6.5 Begrensninger som gjelder datasettet

Begrensninger ved det benyttede datasettet kan ha innvirkning på modellenes segmenteringsytelse. Til tross for at de sanne inntegningene i PET/CT-bildene er utført og verifisert av flere erfarne spesialister, vil det alltid være en usikkerhet tilknyttet posisjonen av den sanne inntegningen. Intervariabiliteter av inntegning på tvers av spesialister, samt intravariabiliteter hos spesialistene, er faktorer som i studien til Weiss og Hess [14] har vist å gi større unøyaktigheter i inntegning av tumorvolum enn unøyaktigheten tilknyttet pasientposisjon og organbevegelse. Inntegningsvariabiliteten blant spesialister ble videre bekreftet i studien til Gudi et al. [79] hvor overlapp av tumorvoluminntegning i PET/CT-

bilder av hode- og halskreft mellom tre radiologer ble beregnet til 69 %. Modellen som trener på datasettet kan dermed ikke bli noe bedre enn den sanne inntegningen utført av spesialistene.

Tilgang på et variert treningsdatasett vil også ha innvirkning på segmenteringsytelsen. Datasettet benyttet i denne masteroppgaven er hentet fra Oslo universitetssykehus. Modellene har dermed trent på PET/CT-bilder fra bare en institusjon, noe som vil kunne gjøre at modellen blir bias mot institusjonenes bildeteknikk og prosedyre, og som kan føre til at modellen generaliserer dårlig på datasett innsamlet fra andre institusjoner. Dette kan videre underbygges fra ytelsen augmenteringsmodellen oppnådde på det eksterne Maastro-testsettet som var lavere enn ytelsen oppnådd på OUS-testsettet, vist i Figur 5.20. Selv om dataaugmenteringsteknikker bidrar til å øke mengden treningseksempler med nye variasjoner, vil trening på innsamlet data fra andre institusjoner være gunstig for modellens generaliseringsevne. I tillegg kreves det, ifølge studien til Kleppe et al. [29], et eksternt testsett som inneholder et representativt utvalgt av data fra andre institusjoner for å korrekt kunne evaluere om modellen er bias ovenfor datasettet modellen trente på.

6.6 Evaluering av augmenteringsmodellen

6.6.1 Treningskurve og valideringskurve

Treningskurven og valideringskurven i Figur 5.2 viste at augmenteringsteknikkene hadde en positiv innvirkning på overtilpasning sammenlignet med treningskurven og valideringskurven til eksperimentmodellen uten augmentering, ettersom avstanden mellom kurvene var mindre for eksperimentmodellene med augmentering. Dette stemmer med tidligere studier som tyder på at augmenteringsteknikker reduserer overtilpasning [50, 80, 81]. Avstanden er likevel stor i Figur 5.2a. og Figur 5.2b. som kan tyde på at de augmenterte modellene fremdeles kan være overtilpasset.

Fra Figur 5.2a. og Figur 5.2b. ble det observert noen tydelige nedoverpigger i valideringskurven som tyder på at modellen kan være sensitivt for visse vektoppdateringer. Mulig har modellen blitt eksponert for augmenterte treningseksempeler som ikke er representative for datasettet. De augmenterte treningseksemplene kan dermed ha innvirket negativt på modellen. Likevel gjennomførte augmenteringsmodellen (eksperimentmodell 46) flere epoker enn eksperimentmodellen uten augmentering, som tyder på at ytelsen har økt ved å inkludere augmenterte treningseksempler til treningsdatasettet.

For å ytterligere øke generaliseringsevnen til nettverket bør videre arbeid sette søkelys på å inkludere regulariseringsteknikker som utelatelse, regulariseringsvekter eller andre augmenteringsteknikker. *Transfer learning* kan også bidra til å øke nettverkets generaliseringsevne [29]. Ved bruk av *transfer learning* starter nettverket med allerede optimaliserte vekter fra andre oppgaver og introduserer dermed biaser som kan bidra til å forhindre overtilpasning [29].

6.6.2 Valideringsprediksjoner og testprediksjoner

Fra Figur 5.20 oppnådde augmenteringsmodellen høyere ytelse på OUS-testsettet, sammenlignet med OUS-valideringssettet. Som regel oppnår en dyplæringsmodell høyere ytelsesscore på et valideringssett sammenlignet med et usett testsett, ettersom informasjon fra valideringssettet kan lekke over til modellen underveis når modellen trenes og valideres [17]. En innvirkende faktor på ytelsen oppnådd kan være antall pasienter i de ulike settene. Ettersom valideringssettet bestod av 15 pasienter og OUS-testsettet bestod av 40 pasienter, vil en lav Dice-score påvirke gjennomsnittsverdien til valideringssettet i større grad enn gjennomsnittet til OUS-testsettet.

En annen innvirkende faktor på den oppnådde ytelsen kan være bildene i datasettene. Fra Figur 5.10 oppnådde to pasienter betydeligere lavere Dice-score sammenlignet med de andre pasientene i OUS-valideringssettet. Den ene pasienten hadde et veldig lavt antall kreftvoxler, mens den andre hadde et veldig høyt antall kreftvoxler, som vist i Figur 5.11. For pasienten med et lavt antall kreftvoxler kan det være vanskelig for augmenteringsmodellen å detektere små tumorer og påvirkede lymfeknuter. Videre undersøkelser fra eksempeltverrsnittene i Figur 5.13 og Figur 5.14 viste at modellen tegnet inn typiske lyse områder (falske positive) i PET/CT-bildene som ikke var klassifisert som sann inntegning. Augmenteringsmodellen kan ha lært at lyse områder i PET-kanalen indikerer høy sannsynlighet for kreft, som kan føre til feilaktige klassifiseringer hos pasienter med høy SUV, en hypotese formulert i Huynh [22] sin masteroppgave. Andre feilaktige inntegninger ble observert i snitt hvor den sanne inntegningen inneholder lavt PET-signal og atypiske Hounsfield-verdier. Slike sanne inntegninger kan være basert på klinisk informasjon eller funn fra tidligere undersøkelser som modellen ikke har tilgang på, som kan gjøre at modellen oppnår lav segmenteringsytelse. Feilaktige inntegninger vil gi en lav Dice-score som vil trekke ned den gjennomsnittlige scoren på valideringssettet.

Til tross for at augmenteringsmodellen oppnådde høy ytelse for de fleste pasientene i OUS-testsettet, var det likevel noen få pasienter med relativ lav oppnådd ytelse. Dette gjaldt spesielt en pasient, hvor overlappsmålet ble beregnet til 0,00, som i likhet med pasienten i valideringssettet hadde et veldig lavt antall kreftvoxler (observert i Figur 5.16). Videre observasjoner av feilaktige inntegninger i OUS-testsettet samsvarer med observasjoner fra valideringssettet.

Ytelsen oppnådd på det eksterne Maastro-testsettet er lavere enn ytelsen oppnådd på OUS-settene (observert i Figur 5.20). Dette kan komme av at augmenteringsmodellen, som har trent og predikert på OUS-settet, kan være bias mot bildekarakteristikkene i datasettet og derfor generaliserer dårlig til Maastro-settet med andre bildekarakteristikker, som beskrevet i delkapittel 6.5. Eksempler på inntegninger hvor modellen oppnådde lav ytelse viser samme trender som de observert i OUS-settene. Videre arbeid bør derfor fokusere på å fjerne falske positive strukturer i treningssettet ved hjelp av post-prosessering, noe som kan bidra til å øke modellens ytelse. I tillegg bør videre arbeid inkludere noen Maastro-pasienter i OUS-treningssettet for kalibrering, som kan bidra til å øke ytelsen på Maastro-testsettet.

Det ble observert stor spredning i Dice-score for pasienter med et lavt antall kreftvoxler (Figur 5.11, Figur 5.16 og Figur 5.19). De lave Dice-scorene kan skyldes lymfeknuter som ikke tar opp FDG og derfor ikke lyser opp på PET-bildet, FDG-negative lymfeknuter, mens de høye Dice-scorene kan skyldes FDG-positive lymfeknuter [63]. Fra Figur 5.11, Figur 5.16 og Figur 5.19 kan en felles trend i spredningsplottene tyde på at modellen har lettere for å tegne inn påvirkede områder hos pasienter med et stort tumorvolum og at et økende antall kreftvoxler vil kunne føre til økt segmenteringsytelse. Likevel gjelder ikke dette for alle pasienter og korrelasjonen må derfor sees på som en indikasjon.

6.7 Sammenligning

6.7.1 Augmenteringsmodellen sammenlignet med baselinemodellen

Studiene til Moe et al. [21], Groendahl et al. [11] og masteroppgaven til Huynh [22] har alle undersøkt 2D U-net-modellens segmenteringsytelse av kreftsvulster og påvirkede lymfeknuter i PET/CT-bilder av hode- og halskreftpasienter fra Oslo universitetssykehus, uten bruk av augmenteringsteknikker på treningsdata. Den gjennomsnittlige ytelsen per pasient til baselinemodellen benyttet i studiene til Moe et al. [21] og Groendahl et al. [11] er gitt i Tabell 6.1 sammen med ytelsen oppnådd med augmenteringsmodellen (tilfeldig rotasjon mellom

-90° og 90°, zoomfaktor mellom 0,5 og 1,5, et antall pikselforskyvning mellom -10 og 10 i *x*- og *y*-retning og flipping om *x*-aksen).

Tabell 6.1: Tabellen gir en oversikt over gjennomsnittlig ytelse med standardavvik oppnådd med baselinemodellen i studiene til Moe et al. [21] og Groendahl et al. [11], og med augmenteringsmodellen undersøkt i denne masteroppgaven.

	Dice-score	HD_{95} [mm]	MSD [mm]
Moe et al. [21]	0,71 ± 0,16	21,0 ± 17,1	1,80 ± 4,00
Groendahl et al. [11]	0,75 ± 0,09	5,79 ± 4,60	-
Augmenteringsmodellen	0,75 ± 0,16	11,4 ± 11,9	1,09 ± 3,95

Sammenlignes ytelsen med resultatene fra studien til Moe et al. [21] i Tabell 6.1, oppnådde augmenteringsmodellen bedre ytelse for alle ytelsesmålene. Overlapp mellom sann inntegning og predikert inntegning økte, og avstandsmålene HD_{95} og MSD ble betydelig redusert. Dette tyder på at augmenteringsmodellen utførte mindre alvorlige inntegningsfeil og at den totale inntegningen ble forbedret sammenlignet med baselinemodellen. Bortsett fra at bildene i denne masteroppgaven er normalisert, er datasettets innhold og splitting det samme som det benyttes i studien til Moe et al. [21]. Ved å inkludere dataaugmentering på treningsdatasettet ser det ut til at modellens segmenteringsytelse forbedres.

Sammenlignes augmenteringsmodellens ytelse med ytelsen oppnådd med CNN-modellen brukt i studien til Groendahl et al. [11] oppnådde augmenteringsmodellen samme Dice-score med større standardavvik og høyere HD_{95}. Dette kan tyde på at augmenteringsmodellen utførte flere alvorlige inntegningsfeil enn baselinemodellen. Det er viktig å merke seg at datasettet benyttet i studien til Groendahl et al. [11] ikke er det samme som datasettet benyttet i denne masteroppgaven. I studien til Groendahl et al. [11] ble et vindussenter på 60 HU benyttet, i motsetning til et vindussenter på 70 HU brukt i augmenteringsmodellen. I tillegg ble bildene i studien til Groendahl et al. [11] beskjært slik at bildene bare inneholdt det ønskede undersøkelsesområdet. Inputbildene hadde bildestørrelse på 176×176 i motsetning til bildestørrelsen 191×256 benyttet i denne masteroppgaven. Beskjæringen gir mindre størrelse som kan føre til færre feilklassifiseringer [54]. Forskjeller i ytelsene oppnådd kan også komme av treningsprosessen. I studien til Groendahl et al. [11] ble *k*-fold kryssvalidering benyttet. Teknikken splitter den tilgjengelige treningsdataen inn i *k* deler, hvor modellen trener på $k - 1$ deler og evaluerer på den gjenværende delen. Etter at modellen har evaluert på hver og en del, brukes gjennomsnittet av ytelsen oppnådd på hver av de *k*-delene [17]. Groendahl et al. [11] benyttet også en læringsrate på 10^{-5}, mens augmenteringsmodellen benyttet 10^{-4} som kan ha innvirkende effekt på oppnådd segmenteringsytelse.

6.7.2 Augmenteringsmodellen sammenlignet med lignende studier

Flere studier har forsøkt å automatisere den tidkrevende og utfordrende kreftsvulstinntegningen og har oppnådd lovende resultater med høy grad av overlapp mellom den predikerte og den sanne inntegningen [8, 15, 82]. Studien til Lin et al. [15] undersøkte bruk av 3D CNN, VoxResNet, for automatisk inntegning av krefttumorvolum i neseområdet («nasopharyngeal cancer») på MR-bilder med størrelse 96×96×32, hvor modellen oppnådde en median Dice-score per pasient på 0,79. Studien til Huang et al. [82] benyttet 2D CNN, inspirert av FCN og U-Net-arkitektur, for automatisk inntegning av hode- og halskrefttumorvolum i PET/CT-bilder med størrelse 512×512. Modellen oppnådde en gjennomsnittlig Dice-score per pasient på 0,74. Dataaugmenteringsteknikker som rotasjon, speiling og bildeskalering ble benyttet på treningsdatasettet for å øke mengden treningsdata [82]. Guo et al. [8] segmenterte både GTV og store påvirkede lymfeknuter ved hjelp av 3D CNN, Dense-Net, i PET, CT og PET/CT-bilder med størrelse 128×128×48. Modellen oppnådde en gjennomsnittlig Dice-score per pasient på 0,71 ved bruk av PET/CT-modaliteten.

I 2020 ble konkurransen HECKTOR (HEad and neCK TumOR) arrangert for automatisk segmentering av hode- og halskrefttumorvolum i FDG-PET/CT-bilder med størrelse 144×144×144, på den internasjonale konferansen *Medical Image Computing and Computer Assisted Intervention* (MICCAI) [19]. Flere av de høyest rangerte modellene benyttet dataaugmenteringsteknikker på treningsdataen. Studien til Xie and Peng [80] utførte automatisk segmentering ved hjelp av en 3D U-Net variant og oppnådde den gjennomsnittlige Dice-scoren 0,74 per pasient. Tradisjonelle dataaugmenteringsteknikker som tilfeldig rotasjon, tilfeldig skalering og speiling i tillegg til tilfeldig elastisk deformasjoner ble benyttet på treningsdataene for å forhindre overtilpasning [80]. I studien til Naser et al. [81] ble en variant av 2D- og 3D U-Net for segmentering av kreftsvulster benyttet, som oppnådde den gjennomsnittlige Dice-scoren 0,64 per pasient. Modellen trente på augmentert datasett påført standard augmenteringsteknikker som rotasjon, zoom og flipp for å forhindre overtilpasning [81].

Ytelsen oppnådd med augmenteringsmodellen på OUS-settet (Dice-score per pasient lik 0,75) er noe høyere enn ytelsen oppnådd i tidligere nevnte studier, mens ytelsen oppnådd på det eksterne Maastro-settet (Dice-score per pasient lik 0,65) er noe lavere enn de andre nevnte studiene, med unntak av studien til Naser at al. [81]. Sammenligningen bør imidlertid brukes som en indikasjon fremfor en presis sammenligning, ettersom flere faktorer som kan ha

innvirkning på segmenteringsresultatet er ulike fra denne masteroppgaven. I studien til Lin et al. [15] og Huang et al. [82] segmenterte modellene tumorvolum og inkluderte ikke påvirkede lymfeknuter. Automatiseringen av kreftsvulstsegmentering undersøkt i HECKTOR-konkurransen var begrenset til munnsvelgområdet hvor en avgrensningsboks ble benyttet for inntegning av tumorvolum [19]. Ettersom augmenteringsmodellen i denne masteroppgaven ikke benyttet avgrensningsbokser, benyttet andre inputstørrelser og i tillegg tegnet inn både tumorvolum og påvirkede lymfeknuter, vil ytelsesscoren ikke kunne sammenlignes på en rettferdig måte.

Segmentering av tumorvolum og påvirkede lymfeknuter i PET/CT-bilder ved bruk av konvolusjonsnettverk med VoxResNet-arkitektur og Dense-Net-arkitektur har også blitt undersøkt i masteroppgavene til henholdsvis Krogstie [83] og Gjengedal [84]. Slik som utført i denne masteroppgaven, ble nettverkene trent på OUS-datasettet og videre testet på OUS-testsettet og Maastro-testsettet. Kombinasjonen av augmenteringsteknikker funnet i denne masteroppgaven ble benyttet for å undersøke augmenteringsteknikkenes innvirkning på konvolusjonsnettverkenes segmenteringsytelse. For VoxResNet-arkitekturen økte nettverkets ytelse ved bruk av dataaugmentering på alle settene, hvor nettverket oppnådde de gjennomsnittlige Dice-scorene per pasient på 0,73 og 0,71 på OUS-testsettet og 0,64 og 0,62 på Maastro-testsettet, henholdsvis med og uten augmentering [83]. For Dense-Net-arkitekturen økte ytelsen ved bruk av augmentering på OUS-settene, men forble den samme på Maastro-settet. Ytelsene oppnådd, var de gjennomsnittlige Dice-scorene per pasient på 0,74 og 0,73 på OUS-testsettet henholdsvis med og uten augmentering, og 0,64 på Maastro-testsettet både med og uten augmentering [84]. Sammenlignes segmenteringsytelsene med resultater fra denne masteroppgaven, oppnådde U-Net-arkitekturen i augmenteringsmodellen noe høyere ytelse enn både VoxResNet-modellen og Dense-Net-modellen. Funn tyder videre på at augmenteringsteknikkene kan ha en positiv innvirkning på segmenteringsytelsen til alle nettverkene, men at effekten av innvirkning varierer.

6.8 Dyp læring i radiologi

Manuell inntegning av kreftsvulster og påvirkede lymfeknuter er en utfordrende og tidkrevende prosess utsatt for inter- og intravariabiliteter, samt unøyaktigheter [8, 15]. Målet er å skape en dyplæringsmodell som på kort tid gjennomfører presis og nøyaktig segmentering. Funn fra studien til Lin et al. [15] viser at både inntegningstid og inter- og intraobservasjonsvariabiliteter betydelig reduseres når CNN-basert inntegning blir brukt som

et hjelpemiddel under manuell inntegning. Dette tyder på at automatisk inntegning vil kunne ha stor nytteverdi i klinisk sammenheng.

Konvolusjonsnettverk som trener på medisinske bilder krever mye minne og stor beregningskraft og kan dermed bruke lang tid [17]. Tiden hver eksperimentmodell brukte på å trene i denne masteroppgaven rangerte fra 4–16 timer. Augmenteringsmodellen som oppnådde høyest Dice-score på valideringssettet ble videre kjørt på testsett der modellen brukte litt over 6 minutter på OUS-testsettet, bestående av 40 pasienter, og ca. 30 minutter på det eksterne Maastro-testsettet, bestående av 114 pasienter. Undersøkes en enkel pasient vil modellen dermed bruke mye mindre tid, noe som kan være tidsbesparende for spesialistene hvis modellen tas i bruk i klinisk sammenheng.

Hode- og halsområdet består av kompleks og vital anatomi hvor konsekvensene av feil strålebehandling kan være store. Hvis en dyplæringsmodell skal brukes i klinisk praksis må prediksjonene til modellen være nøyaktig og presise. Lovlige aspekter ved bruk av dyplæringsmodeller i helsesektoren må i tillegg tas i betraktning. Ifølge *The Ethics guidelines for trustworthy AI* [85] kreves det menneskelig interaksjon underveis som dyplæringsmodellen utvikles og benyttes. De etiske retningslinjene krever i tillegg en reserveplan hvis noe går galt med modellen. Dyplæringsmodellen må dermed evalueres av spesialister før radioterapibehandling fastsettes. Den CNN-baserte automatiske inntegningen kan derfor bare brukes som et hjelpemiddel, fremfor en erstatning av den manuelle inntegningen.

Helseregisterloven [86] må også tas i betraktning når kunstig intelligens benyttes på sensitiv helsedata. Ifølge loven må behandling av helsedata skje på en etisk forsvarlig måte som fremmer helse. Det stilles derfor strenge krav til informasjonssikkerhet, taushetsplikt og hvem som har innsyn til helseopplysningene. Forskning på automatisert inntegning ved hjelp av CNN-modeller kan derfor utføres så lenge behandling av datasettet gjøres på en forsvarlig måte som ivaretar pasientenes personvern.

Augmenteringsmodellen oppnådde den gjennomsnittlige Dice-scoren 0,75 per pasient på OUS-testsettet og den gjennomsnittlige Dice-scoren 0,65 per pasient på det eksterne Maastro-testsettet. Ifølge definisjonen til Zijdenbos et al. [75] kan dermed augmenteringsmodellen klassifiseres som god segmentering (over 0,70) på OUS-settet, men ikke på det eksterne Maastro-datasettet. Som tidligere nevnt, er inter- og intravariabiliteter av kreftsvulstinntegning blant spesialistene en stor utfordring, hvor studien til Gudi et al. [79]

beregnet overlapp av tumorvoluminntegning mellom tre radiologer til 69 % ved bruk av PET/CT-bilder. Augmenteringsmodellen oppnådde Dice-score på lik linje med det rapporterte overlappet mellom de tre radiologene og har dermed potensialet til å kunne brukes som et assisterende hjelpemiddel i klinisk sammenheng. Likevel har augmenteringsmodellen noen svakheter som må tas i betraktning. I noen tverrsnitt tegner modellen inn små strukturer som ikke er av klassen ikke-friskt vev. Dette kan føre til unødvendig skade av vev og organer ved strålebehandling. Augmenteringsmodellen har i noen tverrsnitt ikke tegnet inn den sanne inntegningen. Dette er av større bekymring ettersom alle tumorer eller påvirkede lymfeknuter må ha tilstrekkelig strålebehandling for at pasienten skal kunne bli erklært frisk. Fremtidig arbeid bør derfor fokusere på å fjerne små strukturer i PET/CT-bildene ved hjelp av post-prosessering og finne metoder som forbedrer modellens inntegning av sanne strukturer.

6.9 Videre arbeid

6.9.1 Augmenteringsteknikker

Augmenteringsteknikkene benyttet i denne masteroppgaven er alle standard teknikker, som i tidligere studier har vist å kunne forbedre modellytelsen [23, 27, 56]. Ettersom det finnes mange augmenteringsteknikker og ettersom valg av augmenteringsteknikker bør tilpasses modellens oppgave og datasett, bør videre arbeid undersøke effekten av andre teknikker eller undersøke ulike nivåer av augmentering. Eksempler på andre augmenteringsteknikker kan være elastisk deformasjon, benyttet i studien til Ronneberger et al. [50], eller *Generative Adversarial Networks* som i studien til Frid-Adar et al. [28] oppnådde lovende resultater.

6.9.2 Interaksjoner og utførelse av eksperimenter

Interaksjoner mellom augmenteringsteknikknivåene ble ikke tatt i betraktning når augmenteringsparameterne ble evaluert. Interaksjoner kan ha en innvirkning på ytelsesscoren til modellen. Fremtidig arbeid bør derfor basere seg på å undersøke parameterinteraksjonene.

Ettersom de statistiske testene utført på eksperimentplan 2 ga ikke-signifikant effekt, kan modellen med de beste kombinasjonene fra eksperimentplan 2 benyttet i eksperimentplan 3 være feil. Videre arbeid kan derfor undersøke alle mulige kombinasjoner av augmenteringsnivåene i eksperimentplan 1 og 2 for å finne kombinasjonen av teknikker som gir høyest ytelsesscore.

Det kan også være interessant å se hvordan augmenteringsmodellen vil prestere med treningsprosessen *k*-fold kryssvalidering, som ble benyttet i studien til Groendahl et al. [11].

6.9.3 Variert datasett

Til tross for at augmenteringsteknikker kan se ut til å forbedre modellenes ytelse, oppnådde augmenteringsmodellen en lavere score på det eksterne Maastro-settet, noe som kan komme av at modellen bare har trent på data hentet fra samme institusjon. Videre arbeid bør derfor sette søkelys på å trene en dyplæringsmodell på datasett bestående av bilder hentet fra flere institusjoner, der bildene er tatt med forskjellige bildeteknikker og prosedyrer. På denne måten kan modellen hindres i å bli bias. I tillegg kan andre regulariseringsteknikker som utelatelse, regulariseringsvekter eller *transfer learning* undersøkes i fremtidig arbeid for å øke generaliseringsevnen til nettverket.

6.9.4 Forbedring til klinisk bruk

Til tross for at augmenteringsmodellen oppnådde ytelsesscore som kan klassifiseres som god segmentering på OUS-testsettet, tegnet modellen likevel inn falske positive og falske negative strukturer i noen av tverrsnittene, som kan gi store konsekvenser for utfallet av pasientens strålebehandling. Augmenteringsmodellen må derfor videre forbedres for å kunne brukes som et assisterende hjelpemiddel. Videre arbeid bør av den grunn undersøke bruk av post-prosessering som fjerner små feilklassifiserte strukturer og metoder for å forhindre falske negative strukturer. I tillegg bør en spesialist undersøke modellens inntegninger før modellen kan brukes i klinisk sammenheng.

Kapittel 7: Konklusjon

I denne masteroppgaven har 2D U-Net-arkitekturen blitt undersøkt for segmentering av kreftsvulster i PET/CT-bilder av hode- og halskreftpasienter fra Oslo universitetssykehus. Ved å benytte tradisjonelle augmenteringsteknikker på inputbilder som øker mengden treningsdata med nye variasjoner, har masteroppgaven satt søkelys på å forbedre 2D U-Net-modellens kreftsvulstinntegning.

65 eksperimentmodeller med 2D U-Net-arkitektur ble trent på treningsdatasett påført ulike kombinasjoner av augmenteringsteknikknivåer innenfor kategoriene affine transformasjoner og punkt- og filteroperasjoner. Affine transformasjoner hadde signifikant innvirkning på segmenteringsytelsen, i motsetning til punkt- og filteroperasjonsteknikkene som i noen tilfeller hadde negativ effekt på modellens ytelse. Augmenteringsteknikken med den mest markante innvirkningen var den affine transformasjonen flipp. Videre tydet de statistiske testene på at kraftig augmentering oppnådde de høyeste Dice-scorene.

Modellen som bestod av tilfeldig rotasjon mellom -90° og 90°, zoomfaktor mellom 0,5 og 1,5, et antall pikselforskyvning mellom -10 og 10 i x- og y-retning og flipping om x-aksen, oppnådde høyest ytelse på valideringssettet av alle eksperimentmodellene. Modellen ble derfor videre kjørt på testsettet fra Oslo universitetssykehus og det eksterne Maastro-settet, der augmenteringsmodellen oppnådde den gjennomsnittlige Dice-scoren per pasient på henholdsvis 0,75 ± 0,16 og 0,65 ± 0,19. Tilsvarende modell kjørt uten augmentering på OUS-testsettet oppnådde en Dice-score per pasient på 0,71 ± 0,16.

Til tross for høy oppnådd ytelse, utførte modellen feilinntegninger i både valideringssettet og testsettene. Observasjoner fra visualiserte inntegninger viste at modellen hadde vanskeligheter med å tegne inn små tumorvolum og sanne inntegninger med atypiske verdier. I tillegg inkluderte modellen falske positive inntegninger i typiske lyse områder som ikke var tegnet inn av spesialistene.

Funn fra denne masteroppgaven kan dermed tyde på at affine transformasjoner kan forbedre segmenteringsytelsen til 2D U-Net-arkitekturen. Augmenteringsmodellen viste lovende resultater for automatisk kreftsvulstinntegning, men må videre forbedres for å kunne brukes som et assisterende hjelpemiddel i klinisk sammenheng. Videre arbeid bør derfor undersøke post-prosessering, innhenting av treningsdata fra forskjellige institusjoner, samt undersøke andre augmenteringsteknikker for å kunne ytterligere forbedre modellens segmenteringsytelse.

Kapittel 8: Referanser

[1] World Health Organization. "All cancers fact sheet." Hentet fra: https://gco.iarc.fr/today/data/factsheets/cancers/39-All-cancers-fact-sheet.pdf (lest 09.04.21).

[2] Kreftforeningen. "Hva er kreft?" Hentet fra: https://kreftforeningen.no/om-kreft/hva-er-kreft/ (lest 09.04.21).

[3] Kreftregisteret, "Cancer in Norway 2019," Institute of Population-based Cancer Research, Faktarapport okt. 2020. [Online]. Hentet fra: https://www.kreftregisteret.no/globalassets/cancer-in-norway/2019/cin_report.pdf

[4] Kreftforeningen. "Hode- og halskreft." Hentet fra: https://kreftforeningen.no/om-kreft/kreftformer/hode-og-halskreft/ (lest 09.04.21).

[5] Oslo universitetssykehus. "Hode- og halskreft." Hentet fra: https://oslo-universitetssykehus.no/behandlinger/hode-og-halskreft (lest 09.04.21).

[6] Kreftforeningen. "PET/CT." Hentet fra: https://kreftforeningen.no/om-kreft/undersokelser/pet-ct/ (lest 10.04.21).

[7] R. L. Wahl, *Principles and Practice of PET and PET/CT*, 2. utg. Philadelphia: Wolters Kluwer, 2009.

[8] 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.

[9] Kreftforeningen. "Strålebehandling." Hentet fra: https://kreftforeningen.no/om-kreft/behandling/stralebehandling/ (lest 10.04.21).

[10] C. F. Njeh, "Tumor delineation: The weakest link in the search for accuracy in radiotherapy," *Journal of medical physics*, Online vol. 33, no. 4, s. 136-140, 2008, doi: 10.4103/0971-6203.44472.

[11] 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.

[12] V. Grégoire, J. A. Langendijk, and S. Nuyts, "Advances in Radiotherapy for Head and Neck Cancer," *Journal of Clinical Oncology*, Online vol. 33, no. 29, s. 3277-3284, okt. 2015, doi: 10.1200/JCO.2015.61.2994.

[13] P. M. Harari, S. Song, and W. A. Tomé, "Emphasizing Conformal Avoidance Versus Target Definition for IMRT Planning in Head-And-Neck Cancer," *International Journal of Radiation Oncology, Biology, Physics*, Online vol. 77, no. 3, s. 950-958, april 2010, doi: 10.1016/j.ijrobp.2009.09.062.

[14] 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.

[15] 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.

[16] S. Raschka and V. Mirjalili, *Python Machine Learning*, 3. utg. Birmingham: Packt Publishing Ltd., 2019.

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

[18] J. Kleesiek, J. M. Murray, C. Strack, S. Prinz, G. Kaissis, and R. Braren, "Artificial intelligence and machine learning in oncologic imaging," *Der Pathologe*, Online vol. 41, no. 6, s. 649-658, nov. 2020, doi: 10.1007/s00292-020-00827-3.

[19] 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 Challange*, Cham, jan. 2020: Springer, in Head and Neck Tumor Segmentation, s. 1-21, doi: 10.1007/978-3-030-67194-5 1.

[20] 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.

[21] 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.

[22] 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.

[23] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," i *2018 international interdisciplinary PhD workshop (IIPhDW)*, Poland, 2018: IEEE, s. 117-122, doi: 10.1109/IIPHDW.2018.8388338.

[24] M. D. Bloice, P. M. Roth, and A. Holzinger, "Biomedical image augmentation using Augmentor," *Bioinformatics*, Online vol. 35, no. 21, s. 4522-4524, nov. 2019, doi: 10.1093/bioinformatics/btz259.

[25] 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.

[26] Z. Hussain, F. Gimenez, D. Yi, and D. Rubin, "Differential data augmentation techniques for medical imaging classification tasks," i *AMIA Annual Symposium Proceedings*, april 2018: American Medical Informatics Association, s. 979-984.

[27] L. Perez and J. Wang, "The effectiveness of Data Augmentation in Image Classification using Deep Learning," 2017. [Online]. Hentet fra: arXiv: 1712.04621 [cs.CV].

[28] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification," *Neurocomputing*, Online vol. 321, s. 321-331, des. 2018, doi: 10.1016/j.neucom.2018.09.013.

[29] A. Kleppe, O.-J. Skrede, S. De Raedt, K. Liestøl, D. J. Kerr, and H. E. Danielsen, "Designing deep learning studies in cancer diagnostics," *Nature Reviews Cancer*, Online vol. 21, no. 3, s. 199-211, mars 2021, doi: 10.1038/s41568-020-00327-9.

[30] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, Online vol. 6, no. 60, juli 2019, doi: 10.1186/s40537-019-0197-0.

[31] J. K. Shultis and R. E. Faw, *Fundamentals of Nuclear Science and Engineering*, 3. utg. Boca Raton: CRC Press, 2016.

[32] J. G. Webster, *Medical Instrumentation Application and Design*, 4. utg. John Willey & Sons, INC., 2009.

[33] J. Lilley, *Nuclear Physics: Principles and Applications*, 1. utg. Chichester: John Willey and Sons Ltd, 2001.
[34] D. J. Bell and M. M. Nadrljanski. "Computed tomography." Hentet fra: https://radiopaedia.org/articles/computed-tomography?lang=us (lest 01.02.21).
[35] S. A. Kane and B. A. Gelman, *Introduction to Physics in Modern Medicine*, 3. utg. Boca Raton: CRC Press, 2020.
[36] C. M. Moore and P. Mudgal. "Characteristic radiation." Hentet fra: https://radiopaedia.org/articles/characteristic-radiation?lang=us (lest 03.02.21).
[37] F. Deng and M. M. Nadrljanski. "Attenuation coefficient." Hentet fra: https://radiopaedia.org/articles/attenuation-coefficient?lang=us (lest 01.02.21).
[38] A. Johnston and A. Murphy. "Iodinated contrast media." Hentet fra: https://radiopaedia.org/articles/iodinated-contrast-media-1?lang=us (lest 03.02.21).
[39] I. K. Espe, "Kvalitetskontroll av ikke-dosimetriske parametre ved CT-basert planlegging av stråleterapi," Statens strålevern, Strålevernrapport 2006. [Online]. Hentet fra: https://dsa.no/publikasjoner/stralevernrapport-13-2006-kvalitetskontroll-av-ikke-dosimetriske-parametre-ved-ct-basert-planlegging-av-straleterapi/StralevernRapport_13_2006.pdf
[40] A. Murphy. "Windowing (CT)." Hentet fra: https://radiopaedia.org/articles/windowing-ct?lang=us (lest 03.02.21).
[41] J. McKenzie and S. Goergen. "Computed Tomography (CT)." Hentet fra: https://www.insideradiology.com.au/computed-tomography/ (lest 04.02.21).
[42] M. Saber and A. Shetty. "Positron emission tomography." Hentet fra: https://radiopaedia.org/articles/positron-emission-tomography?lang=us (lest 01.02.21).
[43] M. E. Phelps, *PET physics, instrumentation, and scanners*, 1. utg. Los Angeles: Springer, 2006.
[44] S. Wang and R. M. Summers, "Machine learning and radiology," *Medical Image Analysis*, Online vol. 16, no. 5, s. 933-951, juli 2012, doi: 10.1016/j.media.2012.02.005.
[45] V. Zocca, G. Spacagna, D. Slater, and P. Roelants, *Python Deep Learning*, 1. utg. Birmingham: Packt Publishing, 2017.
[46] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation," in *2016 fourth international conference on 3D vision (3DV)*, Standford California, juni. 2016: IEEE, s. 565-571, doi: 10.1109/3DV.2016.79.
[47] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 2015. [Online]. Hentet fra: arXiv:1412.6980.
[48] W. Burger and M. J. Burge, *Digital Image Processing: An Algorithmic Introduction Using Java*, 2. utg. London: Springer, 2016.
[49] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Online vol. 39, no. 4, s. 640-651, april 2017, doi: 10.1109/tpami.2016.2572683
[50] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional Networks for Biomedical Image Segmentation," i *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Munich, Germany, nov. 2015: Springer, s. 234-241, doi: 10.1007/978-3-319-24574-4_28.
[51] Q. Ren and R. Hu, "Multi-scale deep encoder-decoder network for salient object detection," *Neurocomputing*, Online vol. 316, s. 95-104, nov. 2018, doi: 10.1016/j.neucom.2018.07.055.
[52] 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.
[53] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," 2016. [Online]. Hentet fra: arXiv: 1603.07285 [stat.ML].
[54] B. N. Huynh, "Personlig kommunikasjon," 2021.
[55] E. Castro, J. S. Cardoso, and J. C. Pereira, "Elastic deformations for data augmentation in breast cancer mass detection," i *2018 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, Las Vegas, mars 2018: IEEE, s. 230-234, doi: 10.1109/BHI.2018.8333411.
[56] J. Nalepa, M. Marcinkiewicz, and M. Kawulok, "Data Augmentation for Brain-Tumor Segmentation: A Review," *Frontiers in Computational Neuroscience*, Online vol. 13, des. 2019, doi: 10.3389/fncom.2019.00083.
[57] 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.
[58] D. Zhang and G. Lu, "Review of shape representation and description techniques," *Pattern recognition*, Online vol. 37, no. 1, s. 1-19, jan. 2004, doi: 10.1016/j.patcog.2003.07.008.
[59] 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.
[60] Y. M. Moe, "Deep learning for automatic delineation of tumours from PET/CT images," Masteroppgave, Faculty of Science and Technology, Norwegian University of Life Sciences, Ås, 2019.
[61] UICC global cancer control. "What is the TNM cancer staging system?" Hentet fra: https://www.uicc.org/resources/tnm (lest 23.03.21).
[62] A. Collette, *Python and HDF5: Unlocking Scientific Data*, 1. utg. Sebatpool: O'Reilly Media, Inc., 2013.
[63] A. R. Grøndahl, "Personlig kommunikasjon," 2021.
[64] M. Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," 2015. [Online]. Hentet fra: arXiv:1603.04467 [cs.DC].
[65] NMBU-SUPPORT. "NMBU Orion HPC Cluster." Hentet fra: https://support.nmbu.no/it-dokumentasjon/nmbu-orion-regneklynge/ (lest 03.03.21).
[66] CIGENE. "CIGENE computational unit." Hentet fra: https://cigene.no/lab-and-infrastructure/cigene-computational-unit/ (lest 03.03.21).
[67] NMBU Orion user support. "Partitions." Hentet fra: https://nmbu-orion-support.readthedocs.io/en/latest/partitions.html (lest 10.05.21).
[68] NMBU Orion user support. "Connecting to Orion." Hentet fra: https://nmbu-orion-support.readthedocs.io/en/latest/connect.html (lest 10.05.21).
[69] N. B. Huynh. "Data." Hentet fra: https://deoxys.readthedocs.io/en/latest/data.html#module-deoxys.data.preprocessor (lest 05.03.21).
[70] D. C. Montgomery, *Design and Analysis of Experiments*, 8. utg. Arizona: John Wiley & Sons, Inc., 2013.
[71] 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.
[72] Alan Grafen and R. Hails, *Modern Statistics for the Life Sciences*, 1. utg. United States: Oxford University Press, 2002.

[73] 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.
[74] M. Hollander, D. A. Wolfe, and E. Chicken, *Nonparametric Statistical Methods*, 3. utg. Hoboken: John Wiley & Sons, Inc., 2014.
[75] 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.
[76] E. Kauderer-Abrams, "Quantifying translation-invariance in Convolutional Neural Networks," 2017. [Online]. Hentet fra: arXiv:1801.01450 [cs.CV].
[77] Datatilsynet, "Kunstig intelligens og personvern," Datatilsynet, Faktarapport jan. 2018. [Online]. Hentet fra: https://www.datatilsynet.no/globalassets/global/dokumenter-pdfer-skjema-ol/rettigheter-og-plikter/rapporter/rapport-om-ki-og-personvern.pdf
[78] J. Shijie, W. Ping, J. Peiyi, and H. Siping, "Research on data augmentation for image classification based on convolution neural networks," i *2017 Chinese automation congress (CAC)*, Jinan, China, okt. 2017: IEEE, s. 4165-4170, doi: 10.1109/CAC.2017.8243510.
[79] 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, juni 2017, doi: 10.1016/j.jmir.2016.11.003.
[80] J. Xie and Y. Peng, "The head and neck tumor segmentation using nnU-Net with spatial and channel 'squeeze & excitation'blocks," i *3D Head and Neck Tumor Segmentation in PET/CT Challenge*, Cham, jan. 2020: Springer, s. 28-36, doi: 10.1007/978-3-030-67194-5_3.
[81] M. A. Naser, L. V. van Dijk, R. He, K. A. Wahid, and C. D. Fuller, "Tumor segmentation in patients with head and neck cancers using deep learning based-on multi-modality PET/CT images," i *3D Head and Neck Tumor Segmentation in PET/CT Challenge*, Cham, jan. 2020: Springer, s. 85-98, doi: 10.1007/978-3-030-67194-5_10.
[82] 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*, Online vol. 2018, 2018, doi: 10.1155/2018/8923028.
[83] 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.
[84] M. E. Gjengedal, "Segmentering av hode- og halskreft i PET/CT-bilder ved bruk av dype nevrale nettverk," Masteroppgave, Fakultet for realfag og teknologi, Norges miljø- og biovitenskaplige universitet, Ås, 2021.
[85] 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
[86] Regjeringen. "Helseregisterloven." Hentet fra: https://www.regjeringen.no/no/tema/helse-og-omsorg/innsikt/helseregisterloven/id2413825/ (lest 20.04.21).

Vedlegg A

Vedlegg A inneholder en detaljert oversikt over U-Net arkitektur fordelt i tabeller.

U-Net-arkitekturen

Tabell A.1: Tabellen gir en oversikt over sammentrekningsveien (nedsamplingsveien) i arkitekturen U-Net benyttet i eksperimentmodellene og baselinemodellen. Veien består av inputlag, konvolusjonslag, batchnormaliseringslag og makssamlingslag.

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

Tabell A.2: Tabellen gir en oversikt over flaskehalsen i U-Net arkitekturen benyttet i eksperimentmodellene og baselinemodellen. Flaskehalsen består av to konvolusjonslag og to batchnormaliseringslag.

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.3: Tabellen gir en oversikt over en del av den ekspanderende veien, også kalt oppsamplingsvei, i U-Net arkitekturen benyttet i eksperimentmodellene og i baselinemodellen. Veien består av konvolusjonslag, transponerte konvolusjonslag, sammenkoblinger og batchnormaliseringslag.

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

Tabell A.4: Tabellen gir en oversikt over den resterende delen av den ekspanderende veien i U-Net arkitekturen benyttet i eksperimentmodellene og baselinemodellen.

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

Vedlegg B

Vedlegg B inneholder detaljert oversikt over eksperimentene utført i tre separate eksperimentplaner med gjennomsnittlig Dice-score per tverrsnitt på valideringssettet. Eksempel på en konfigurasjonsfil er også lagt ved.

Eksperimentplan 1

Tabell B.1: Tabellen gir en oversikt over eksperimentene utført i eksperimentplan 1. Parameternivåene med tilhørende gjennomsnittlig Dice-score per tverrsnitt på valideringssettet er oppgitt i tabellen.

Eksperiment	Rotasjon	Zoom	Forskyvning	Flipp	Dice-score
1	0	1	null	no	0,619
2	0	1	[-10, 10]	no	0,624
3	0	[0,5, 1,5]	null	no	0,623
4	0	[0,5, 1,5]	[-10, 10]	no	0,625
5	0	[0,8, 1,2]	null	no	0,628
6	0	[0,8, 1,2]	[-10, 10]	no	0,624
7	0	1	null	yes	0,643
8	0	1	[-10, 10]	yes	0,650
9	0	[0,5, 1,5]	null	yes	0,647
10	0	[0,5, 1,5]	[-10, 10]	yes	0,667
11	0	[0,8, 1,2]	null	yes	0,647
12	0	[0,8, 1,2]	[-10, 10]	yes	0,647
13	30	1	null	no	0,625
14	30	1	[-10, 10]	no	0,629
15	30	[0,5, 1,5]	null	no	0,638
16	30	[0,5, 1,5]	[-10, 10]	no	0,634
17	30	[0,8, 1,2]	null	no	0,640
18	30	[0,8, 1,2]	[-10, 10]	no	0,635
19	30	1	null	yes	0,658
20	30	1	[-10, 10]	yes	0,659
21	30	[0,5, 1,5]	null	yes	0,658
22	30	[0,5, 1,5]	[-10, 10]	yes	0,658
23	30	[0,8, 1,2]	null	yes	0,658
24	30	[0,8, 1,2]	[-10, 10]	yes	0,668
25	60	1	null	no	0,642
26	60	1	[-10, 10]	no	0,642
27	60	[0,5, 1,5]	null	no	0,635
28	60	[0,5, 1,5]	[-10, 10]	no	0,635
29	60	[0,8, 1,2]	null	no	0,627
30	60	[0,8, 1,2]	[-10, 10]	no	0,630
31	60	1	null	yes	0,659
32	60	1	[-10, 10]	yes	0,658
33	60	[0,5, 1,5]	null	yes	0,659
34	60	[0,5, 1,5]	[-10, 10]	yes	0,656
35	60	[0,8, 1,2]	null	yes	0,648
36	60	[0,8, 1,2]	[-10, 10]	yes	0,660
37	90	1	null	no	0,636
38	90	1	[-10, 10]	no	0,629

39	90	[0,5, 1,5]	null	no	0,648
40	90	[0,5, 1,5]	[-10, 10]	no	0,648
41	90	[0,8, 1,2]	null	no	0,640
42	90	[0,8, 1,2]	[-10, 10]	no	0,634
43	90	1	null	yes	0,660
44	90	1	[-10, 10]	yes	0,658
45	90	[0,5, 1,5]	null	yes	0,666
46	90	[0,5, 1,5]	[-10, 10]	yes	0,676
47	90	[0,8, 1,2]	null	yes	0,661
48	90	[0,8, 1,2]	[-10, 10]	yes	0,671

Eksperimentplan 2

Tabell B.2: Tabellen gir en oversikt over eksperimentene utført i eksperimentplan 2. Augmenteringsteknikknivåene og gjennomsnittlig Dice-score på valideringssettet er oppgitt i tabellen.

Eksperiment	Lysstyrke	Kontrast	Støy	Uskarphet	Dice-score
1	no	no	no	no	0,624
2	yes	no	no	no	0,628
3	no	yes	no	no	0,625
4	yes	yes	no	no	0,622
5	no	no	yes	no	0,623
6	yes	no	yes	no	0,622
7	no	yes	yes	no	0,623
8	yes	yes	yes	no	0,628
9	no	no	no	yes	0,627
10	yes	no	no	yes	0,621
11	no	yes	no	yes	0,622
12	yes	yes	no	yes	0,623
13	no	no	yes	yes	0,636
14	yes	no	yes	yes	0,623
15	no	yes	yes	yes	0,623
16	yes	yes	yes	yes	0,624

Eksperimentplan 3

Tabell B.3: Tabellen gir en oversikt over eksperimentplan 3 hvor augmenteringsteknikknivåer og gjennomsnittlig Dice-score på valideringssettet er gitt i tabellen.

Rotasjon	Zoom	Forskyvning	Flipp	Lysstyrke	Kontrast	Støy	Uskarphet	Dice-score
90	[0,5, 1,5]	[-10, 10]	yes	no	no	yes	yes	0,666

Konfigurasjonsfilen for modellen i eksperimentplan 3


```
{
    "dataset_params": {
        "class_name": "H5Reader",
        "config": {
            "filename": "/home/work/modegaar/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
                        ]
                    }
                }
            ],
            "augmentations": {
                "class_name": "ImageAugmentation2D",
                "config": {
                "rotation_range": 90,
```

```
                    "zoom_range": [
                        0.5,
                        1.5
                    ],
                    "shift_range": [
                        10,
                        10
                    ],
                    "flip_axis": 0,
                    "brightness_range": 1,

                    "contrast_range": 1,

                    "noise_variance": 0.05,

                    "noise_channel": 1,

                    "blur_range": [
                        0.5,
                        1.5
                    ]
                    ,
                    "blur_channel": 1

                }
            }
        }
    },
    "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": "Unet",
        "layers": [
            {
                "class_name": "Conv2D",
                "config": {
                    "filters": 64,
                    "kernel_size": 3,
                    "activation": "relu",
                    "kernel_initializer": "he_normal",
                    "padding": "same"
                },
                "normalizer": {
                    "class_name": "BatchNormalization"
                }
            },
            {
                "name": "conv2",
                "class_name": "Conv2D",
                "config": {
                    "filters": 64,
                    "kernel_size": 3,
                    "activation": "relu",
                    "kernel_initializer": "he_normal",
                    "padding": "same"
                },
                "normalizer": {
                    "class_name": "BatchNormalization"
```

```
            }
        },
        {
            "class_name": "MaxPooling2D"
        },
        {
            "class_name": "Conv2D",
            "config": {
                "filters": 128,
                "kernel_size": 3,
                "activation": "relu",
                "kernel_initializer": "he_normal",
                "padding": "same"
            },
            "normalizer": {
                "class_name": "BatchNormalization"
            }
        },
        {
            "name": "conv4",
            "class_name": "Conv2D",
            "config": {
                "filters": 128,
                "kernel_size": 3,
                "activation": "relu",
                "kernel_initializer": "he_normal",
                "padding": "same"
            },
            "normalizer": {
                "class_name": "BatchNormalization"
            }
        },
        {
            "class_name": "MaxPooling2D"
        },
        {
            "class_name": "Conv2D",
            "config": {
                "filters": 256,
                "kernel_size": 3,
                "activation": "relu",
                "kernel_initializer": "he_normal",
                "padding": "same"
            },
            "normalizer": {
                "class_name": "BatchNormalization"
            }
```

```
            },
            {
                "name": "conv6",
                "class_name": "Conv2D",
                "config": {
                    "filters": 256,
                    "kernel_size": 3,
                    "activation": "relu",
                    "kernel_initializer": "he_normal",
                    "padding": "same"
                },
                "normalizer": {
                    "class_name": "BatchNormalization"
                }
            },
            {
                "class_name": "MaxPooling2D"
            },
            {
                "class_name": "Conv2D",
                "config": {
                    "filters": 512,
                    "kernel_size": 3,
                    "activation": "relu",
                    "kernel_initializer": "he_normal",
                    "padding": "same"
                },
                "normalizer": {
                    "class_name": "BatchNormalization"
                }
            },
            {
                "name": "conv8",
                "class_name": "Conv2D",
                "config": {
                    "filters": 512,
                    "kernel_size": 3,
                    "activation": "relu",
                    "kernel_initializer": "he_normal",
                    "padding": "same"
                },
                "normalizer": {
                    "class_name": "BatchNormalization"
                }
            },
            {
                "class_name": "MaxPooling2D"
```

```
            },
            {
                "class_name": "Conv2D",
                "config": {
                    "filters": 1024,
                    "kernel_size": 3,
                    "activation": "relu",
                    "kernel_initializer": "he_normal",
                    "padding": "same"
                },
                "normalizer": {
                    "class_name": "BatchNormalization"
                }
            },
            {
                "class_name": "Conv2D",
                "config": {
                    "filters": 1024,
                    "kernel_size": 3,
                    "activation": "relu",
                    "kernel_initializer": "he_normal",
                    "padding": "same"
                },
                "normalizer": {
                    "class_name": "BatchNormalization"
                }
            },
            {
                "name": "conv_T_1",
                "class_name": "Conv2DTranspose",
                "config": {
                    "filters": 512,
                    "kernel_size": 3,
                    "strides": 1,
                    "kernel_initializer": "he_normal",
                    "padding": "same"
                }
            },
            {
                "class_name": "Conv2D",
                "config": {
                    "filters": 512,
                    "kernel_size": 3,
                    "activation": "relu",
                    "kernel_initializer": "he_normal",
                    "padding": "same"
                },
                "normalizer": {
```

```
                "class_name": "BatchNormalization"
            },
            "inputs": [
                "conv8",
                "conv_T_1"
            ]
        },
        {
            "class_name": "Conv2D",
            "config": {
                "filters": 512,
                "kernel_size": 3,
                "activation": "relu",
                "kernel_initializer": "he_normal",
                "padding": "same"
            },
            "normalizer": {
                "class_name": "BatchNormalization"
            }
        },
        {
            "name": "conv_T_2",
            "class_name": "Conv2DTranspose",
            "config": {
                "filters": 256,
                "kernel_size": 3,
                "strides": 1,
                "kernel_initializer": "he_normal",
                "padding": "same"
            }
        },
        {
            "class_name": "Conv2D",
            "config": {
                "filters": 256,
                "kernel_size": 3,
                "activation": "relu",
                "kernel_initializer": "he_normal",
                "padding": "same"
            },
            "normalizer": {
                "class_name": "BatchNormalization"
            },
            "inputs": [
                "conv6",
                "conv_T_2"
            ]
```

```
            },
            {
                "class_name": "Conv2D",
                "config": {
                    "filters": 256,
                    "kernel_size": 3,
                    "activation": "relu",
                    "kernel_initializer": "he_normal",
                    "padding": "same"
                },
                "normalizer": {
                    "class_name": "BatchNormalization"
                }
            },
            {
                "name": "conv_T_3",
                "class_name": "Conv2DTranspose",
                "config": {
                    "filters": 128,
                    "kernel_size": 3,
                    "strides": 1,
                    "kernel_initializer": "he_normal",
                    "padding": "same"
                }
            },
            {
                "class_name": "Conv2D",
                "config": {
                    "filters": 128,
                    "kernel_size": 3,
                    "activation": "relu",
                    "kernel_initializer": "he_normal",
                    "padding": "same"
                },
                "normalizer": {
                    "class_name": "BatchNormalization"
                },
                "inputs": [
                    "conv4",
                    "conv_T_3"
                ]
            },
```

```
            {
                "class_name": "Conv2D",
                "config": {
                    "filters": 128,
                    "kernel_size": 3,
                    "activation": "relu",
                    "kernel_initializer": "he_normal",
                    "padding": "same"
                },
                "normalizer": {
                    "class_name": "BatchNormalization"
                }
            },
            {
                "name": "conv_T_4",
                "class_name": "Conv2DTranspose",
                "config": {
                    "filters": 64,
                    "kernel_size": 3,
                    "strides": 1,
                    "kernel_initializer": "he_normal",
                    "padding": "same"
                }
            },
            {
                "class_name": "Conv2D",
                "config": {
                    "filters": 64,
                    "kernel_size": 3,
                    "activation": "relu",
                    "kernel_initializer": "he_normal",
                    "padding": "same"
                },
                "normalizer": {
                    "class_name": "BatchNormalization"
                },
                "inputs": [
                    "conv2",
                    "conv_T_4"
                ]
            },
```

```
            {
                "class_name": "Conv2D",
                "config": {
                    "filters": 64,
                    "kernel_size": 3,
                    "activation": "relu",
                    "kernel_initializer": "he_normal",
                    "padding": "same"
                },
                "normalizer": {
                    "class_name": "BatchNormalization"
                }
            },
            {
                "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 tester utført på resultatdataene med tilhørende script fra RStudio og diagnoseplott.

RStudio-script for undersøkelse av krav om normalfordeling, samt påførte transformasjoner

```
# Undersøker kravet om normalfordelte residualer er oppfylt for resultatdataene i
# eksperimentplan 1 og eksperimentplan 2. Hvis normalitet er oppfylt kan ANOVA benyttes.


# Eksperimentplan 1: ----------------------------------------------------

#Laster inn resultatdataene
data <- aff_resultat_tabell_eksperimentnr

# Lager uavhengige variabler
data$Rotation <- factor(data$Rotation,
                        levels = c(0,30, 60, 90),
                        labels = c("R_0", "R_30", "R_60", "R_90"))

data$Zoom <- factor(data$Zoom,
                    levels = c("1","[0.5 - 1.5]", "[0.8 - 1.2]"),
                    labels = c("Z_1", "Z_0.5", "Z_0.8"))


data$Shift <- factor(data$Shift,
                     levels = c("none","[10, 10]"),
                     labels = c("no", "yes"))

data$Flip <- factor(data$Flip,
                    levels = c("yes", "no"),
                    labels = c("yes", "no"))


# ANOVA med interaksjoner
AOV.1 <- aov(f1_score ~Rotation*Zoom + Rotation*Shift + Rotation*Flip + Zoom*Shift +
               Zoom*Flip + Shift*Flip, data = data)

summary(AOV.1)

# Diagnoseplott for å undersøke kravet om normalfordelte residualer.
# 1) Residuals vs. fitted
plot(AOV.1, 1)
# Den røde streken, gjennomsnitt til residualene, skal være rett og nær null.

# 2) Normality of residuals (QQ-plott)
plot(AOV.1, 2)
# Q-Q plot skal følge den stiplede linjen.

#Foretar Anderson-Darling test for å sjekke om dataene er normalfordelte.
install.packages("nortest")
library(nortest)
ad.test(AOV.1$residuals)

#Ettersom residualene ikke er normalfordelte foretas ulike transformasjoner.

#Logaritmisk-transformasjon:
data$f1_scoreTransformedlog = log10(max(data$f1_score+1) - data$f1_score)
#Kvadratrot-transformasjon:
data$f1_scoreTransformedsqr = sqrt(max(data$f1_score+1) - data$f1_score)
#Invers-transformasjon:
data$f1_scoreTransformedinv = 1/(max(data$f1_score+1) - data$f1_score)

#Her byttes f1_scoreTransformed ut med den tilhørende transformasjonen (f1_scoreTransformedinv,
#f1_scoreTransformedsqr, f1_scoreTransformedlog) som lager en ANOVA-modell.

AOV.2 <- aov(f1_scoreTransformedinv ~Rotation*Zoom + Rotation*Shift + Rotation*Flip + Zoom*Shift +
               Zoom*Flip + Shift*Flip, data = data)

summary(AOV.2)

```

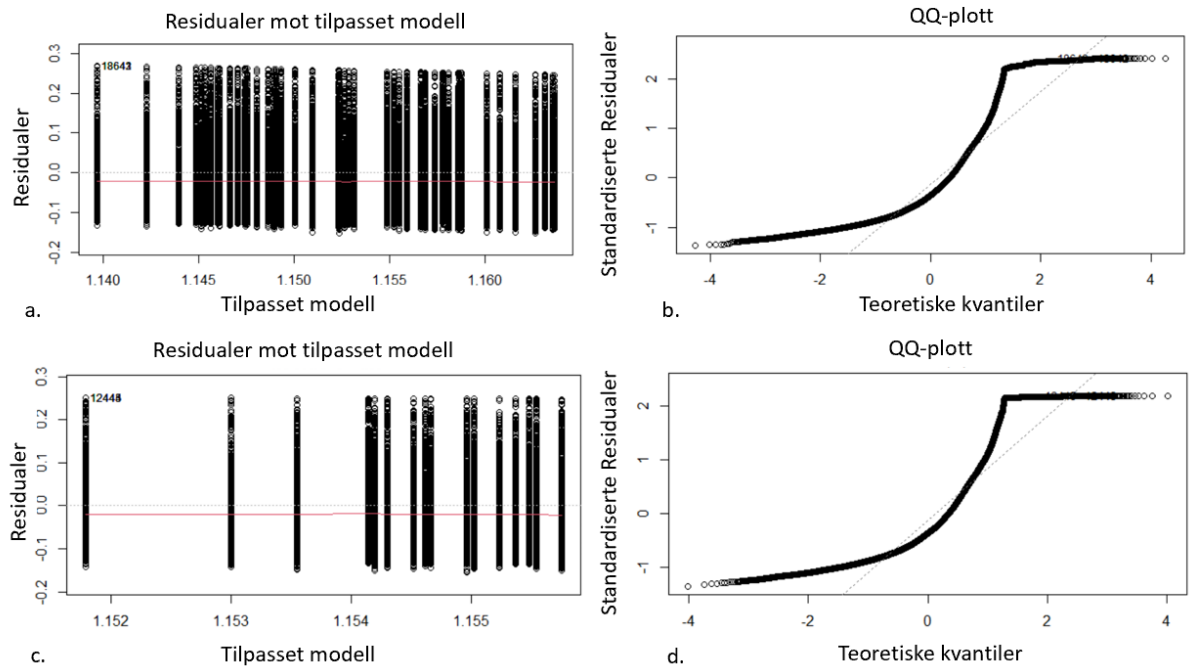
```r
# Diagnoseplott for å undersøke kravet om normalfordelte residualer.
# 1) Residuals vs. fitted
plot(AOV.2, 1)
# Den røde streken, gjennomsnitt til residualene, skal være rett og nær null.

# 2) Normality of residuals (QQ-plott)
plot(AOV.2, 2)
# Q-Q plot skal følge den stiplede linjen.

#Foretar Anderson-Darling test for å sjekke om de transformerte dataene er normalfordelte.
library(nortest)
ad.test(AOV.2$residuals)

# Boxcox transformasjon
library(MASS)

y <- data$f1_score + 0.5 #Legger på 0.5 for å få positive verdier.

AOV.1 <- aov(y ~Rotation*Zoom + Rotation*Shift + Rotation*Flip + Zoom*Shift +
               Zoom*Flip + Shift*Flip, data = data)


#Lambda (3.22) velges ut ifra boxcox-plottet som lages i linjen over.
boxcox(AOV.1, lambda = seq(3.1, 3.3, by = 0.05), plotit = TRUE)

model_cox = aov((((y ^ 3.22) - 1) / 3.22) ~ Rotation*Zoom + Rotation*Shift + Rotation*Flip +
                  Zoom*Shift + Zoom*Flip + Shift*Flip, data = data)

#Diagnoseplott:
#Lager Fitted vs. residulas
plot(fitted(model_cox), resid(model_cox), col = "dodgerblue",
     pch = 20, cex = 1.5, xlab = "Fitted", ylab = "Residuals")
abline(h = 0, lty = 2, col = "darkorange", lwd = 2)

#Lager QQ-plot
plot(model_cox, 2)

#Foretar Anderson-Darling test for å sjekke om de transformerte dataene er normalfordelt.
ad.test(model_cox$residuals)


#Trasformasjonene oppnådde ikke kravet om normalitet --> Friedmantest etterfulgt av
#Nemenyi post-hoc test og Wilcoxon signed-rank test benyttes på dataene.


```

```r
# Eksperimentplan 2: ------------------------------------------------------
#Gjentar prosessen over, men nå for eksperimentplan 2

#Laster inn resultatadataene for eksperimentplan 2.
data2 <- points_resultat_tabell_cnn_template


# Lager uavhengige variabler.
data2$brightess <- factor(data2$brightess,
                          levels = c("no","yes"),
                          labels = c("no", "yes"))

data2$contrast <- factor(data2$contrast,
                         levels = c("no","yes"),
                         labels = c("no","yes"))


data2$noise <- factor(data2$noise,
                      levels = c("no","yes"),
                      labels = c("no", "yes"))

data2$blur <- factor(data2$blur,
                     levels = c("yes", "no"),
                     labels = c("yes", "no"))

# ANOVA with simple interactions
AOV.1 <- aov(f1_score ~brightess*contrast + brightess*noise + brightess*blur + contrast*noise +
               contrast*blur + noise*blur, data = data2)

summary(AOV.1)

# Diagnoseplott for å undersøke kravet om normalfordelte residualer.
# 1) Residuals vs. fitted
plot(AOV.1, 1)
# Den røde streken, gjennomsnitt til residualene, skal være rett og nær null.

# 2) Normality of residuals (QQ-plott)
plot(AOV.1, 2)
# Q-Q plot skal følge den stiplede linjen.

#Foretar Anderson-Darling test for å sjekke om dataene er normalfordelte.
#install.packages("nortest")
library(nortest)
ad.test(AOV.1$residuals)

#Ettersom residualene ikke er normalfordelte foretas ulike transformasjoner.

#Logaritmisk-transformasjon:
data2$f1_scoreTransformedlog = log10(max(data2$f1_score+1) - data2$f1_score)

#Kvadratrot-transformasjon:
data2$f1_scoreTransformedsqr = sqrt(max(data2$f1_score+1) - data2$f1_score)

#Invers-transformasjon:
data2$f1_scoreTransformedinv = 1/(max(data2$f1_score+1) - data2$f1_score)

#Her byttes f1_scoreTransformed ut med den tilhørende transformasjonen (f1_scoreTransformedinv,
#f1_scoreTransformedsqr, f1_scoreTransformedlog) som lager en ANOVA-modell.
AOV.2 <- aov(f1_scoreTransformedsqr ~brightess*contrast + brightess*noise + brightess*blur +
               contrast*noise + contrast*blur + noise*blur, data = data2)

summary(AOV.2)

# Diagnoseplott for å undersøke kravet om normalfordelte residualer.
# 1) Residuals vs. fitted
plot(AOV.2, 1)
# Den røde streken, gjennomsnitt til residualene, skal være rett og nær null.
```

```r
# 2) Normality of residuals (QQ-plott)
plot(AOV.2, 2)
# QQ plot skal følge den stiplede linjen.

#Foretar Anderson-Darling test for å sjekke om de transformerte dataene er normalfordelte.
library(nortest)
ad.test(AOV.2$residuals)

#Boxcox-transformasjon:
library(MASS)

y <- data2$f1_score + 0.5 #Legger på 0.5 for å få positive verdier.

AOV.1.p <- aov(y ~brightess*contrast + brightess*noise + brightess*blur + contrast*noise +
                 contrast*blur + noise*blur, data = data2)

boxcox(AOV.1.p, lambda = seq(2.8, 3.3, by = 0.05), plotit = TRUE)

#Lambda (3.06) velges ut ifra boxcox-plottet som lages i linjen over.
model_cox.p = aov((((y ^ 3.06) - 1) / 3.06) ~ brightess*contrast + brightess*noise +
                    brightess*blur + contrast*noise + contrast*blur + noise*blur, data = data2)

#Diagnoseplott:
#Lager Fitted vs. residuals
plot(fitted(model_cox.p), resid(model_cox.p), col = "dodgerblue",
     pch = 20, cex = 1.5, xlab = "Fitted", ylab = "Residuals")
abline(h = 0, lty = 2, col = "darkorange", lwd = 2)

#Lager QQ-plot:
plot(model_cox, 2)

#Foretar Anderson-Darling test for å sjekke om de transformerte dataene er normalfordelt.
ad.test(model_cox$residuals)

#Trasformasjonene oppnådde ikke kravet om normalitet -->  Wilcoxon signed-rank test
#benyttes på resultatdataene.
```

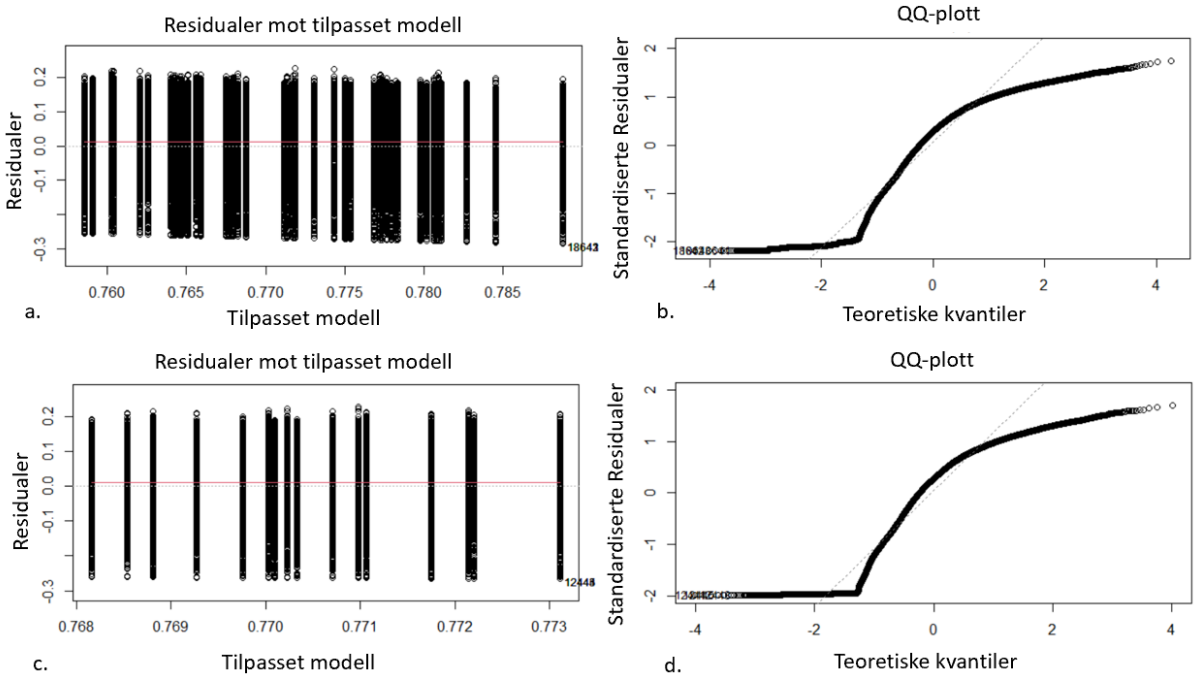
Diagnoseplott

Kvadratrottransformasjon



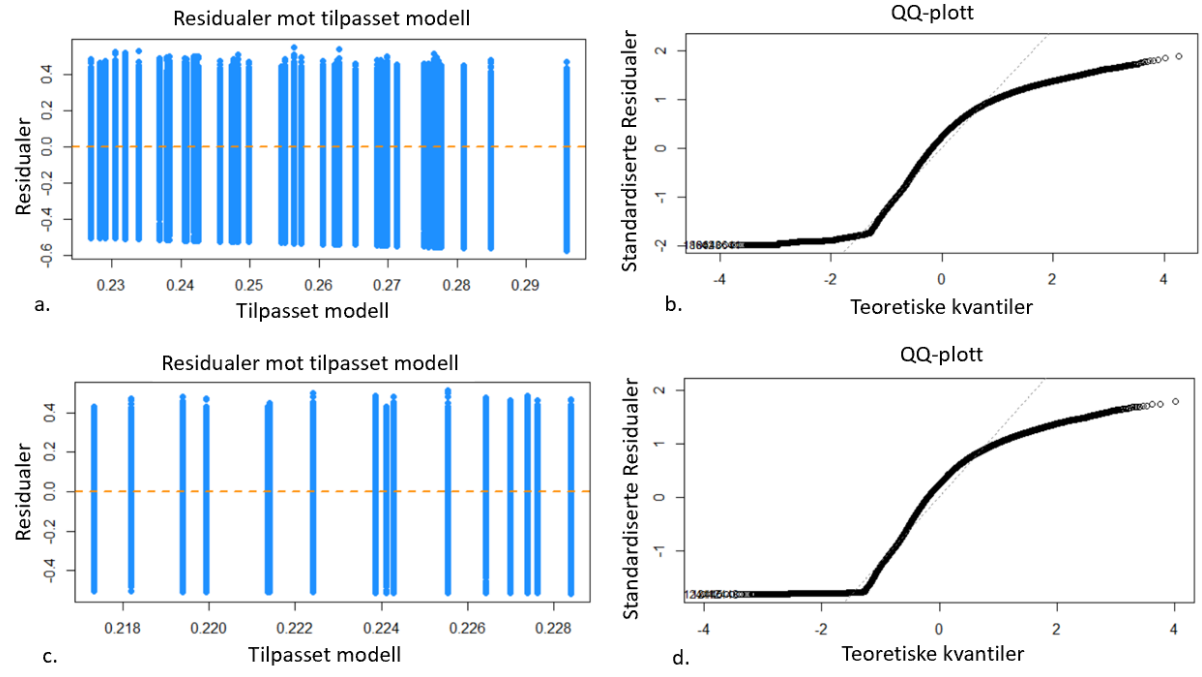
Figur C.1: Figuren viser diagnoseplott for kvadratrot-transformert resultatdata for eksperimentplan 1 (a. og b.) og 2 (c. og d.) Residualer mot tilpasset modell er vist til venstre (a. og c.), mens QQ-plottet er vist til høyre (b. og d.).

Invers-transformasjon



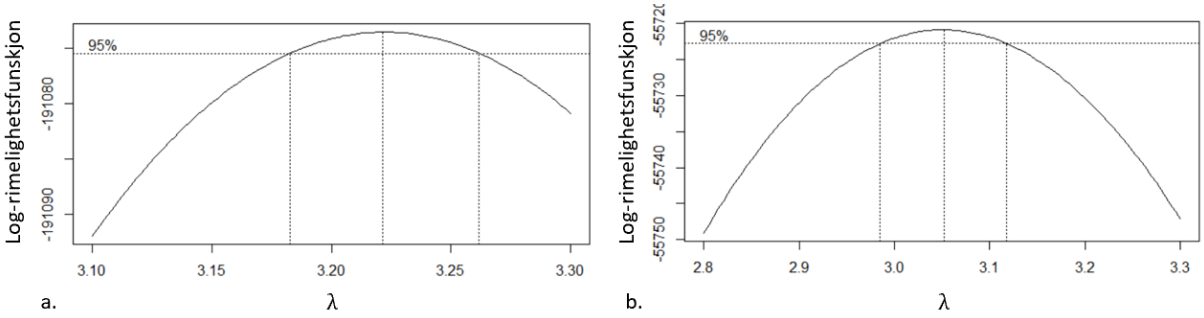
Figur C.2: Figuren viser diagnoseplott for invers-transformert resultatdata for eksperimentplan 1 (a. og b.) og 2 (c. og d.) Residualer mot tilpasset modell er vist til venstre (a. og c.), mens QQ-plottet er vist til høyre (b. og d.).

Boxcox-transformasjon



Figur C.3: Figuren viser diagnoseplott for boxcox-transformert resultatdata for eksperimentplan 1 (a. og b.) og 2 (c. og d.) Residualer mot tilpasset modell er vist til venstre (a. og c.), mens QQ-plottet er vist til høyre (b. og d.).

Log-rimelighetsfunksjon for valg av lambda til boxcox-transformasjon



Figur C.4: Figuren viser Log-rimelighetsfunksjon («Log-likelihood») til eksperimentplan 1 (a.) og eksperimentplan 2 (b.) som brukes for å velge lambda i boxcox-transformasjonen.

Oversikt over statistiske tester utført på de forskjellige parameterne

Tabell C.1: Tabellen viser hvilke statistiske tester som ble utført på de ulike parameterne i eksperimentplan 1.

Parameter	Antall nivåer	Test
Rotasjon	4	Friedman + Nemenyi post-hoc
Zoom	3	Friedman + Nemenyi post-hoc
Forskyvning	2	Wilcoxon signed rank-test
Flipp	2	Wilcoxon signed rank-test

Tabell C.2: Tabellen viser hvilke statistiske tester som har blitt utført på de ulike parameterne i eksperimentplan 2.

Parameter	Antall nivåer	Test
Lysstyrke	2	Wilcoxon signed rank-test
Kontrast	2	Wilcoxon signed rank-test
Støy	2	Wilcoxon signed rank-test
Uskarphet	2	Wilcoxon signed rank-test

RStudio-script for statistiske tester utført på eksperimentplan 1

```r
#Friedmantest etterfulgt av Nemenyi post-hoc test og Wilcoxon signed-rank test utført
#på resultatdataene til eksperimentplan 1.

#Laster inn nødvendige pakker
install.packages("PMCMRplus")
library(PMCMRplus)
library(tidyverse)
library(ggpubr)
library(rstatix)

# Parameteren rotasjon ------------------------------------------------------------
#Parameteren rotasjon har fire nivåer --> Friedmantest

#Laster inn datasettet uten replikasjoner for parameteren rotasjon.
data_rot <- mean_Rotation

y_rot <- c(data_rot$f1_score)
block_rot <- 1:1033
blocks_rot <- rep(block_rot,times=4)
groups_rot <- factor(c(rep(0, times=1033), rep(30, times=1033), rep(60, times=1033),
                       rep(90, times=1033)))

#Boksplott for paramteren rotasjon
ggboxplot(data_rot, x = "Rotation", y = "f1_score", add = 'jitter')
ggboxplot(data_rot, x = "Rotation", y = "f1_score")

#Foretar Friedmantest:
friedman.t.chidist_rot <- friedmanTest(y_rot, groups_rot, blocks_rot)
friedman.t.chidist_rot[["p.value"]]

#Effektstørrelse:
data_rot %>% friedman_effsize(f1_score ~ Rotation |Teller)

#Foretar Nemenyi post-hoc test:
nemenyi.all.comp <- frdAllPairsNemenyiTest(

  y_rot, groups_rot, blocks_rot, alternative = "two.sided")

nemenyi.all.comp

# Parameteren zoom ----------------------------------------------------------------
#Parameteren zoom har tre nivåer --> Friedmantest

#Laster inn datasettet uten replikasjoner for parameteren.
data_zoo <- mean_Zoom

y_zoo <- c(data_zoo$f1_score)
block_zoo <- 1:1033
blocks_zoo <- rep(block_zoo,times=3)
groups_zoo <- factor(c(rep("1", times=1033), rep("[0.5 - 1.5]", times=1033), rep("[0.8 - 1.2]",
                                                                                 times=1033)))

```

```r
#Foretar Friedmantest:
friedman.t.chidist_zoo <- friedmanTest(y_zoo, groups_zoo, blocks_zoo)
friedman.t.chidist_zoo[["p.value"]]

#Boksplott:
ggboxplot(data_zoo, x = "Zoom", y = "f1_score")
ggboxplot(data_zoo, x = "Zoom", y = "f1_score", add = 'jitter')

#Effektstørrelse:
data_zoo %>% friedman_effsize(f1_score ~ Zoom |Teller)

#Nemenyi post-hoc test:
nemenyi.all.comp <- frdAllPairsNemenyiTest(

  y_zoo, groups_zoo, blocks_zoo, alternative = "two.sided")

nemenyi.all.comp

# Parameteren forskyvning ("shift") ------------------------------------------------------------
#Parameteren forskyvning har to nivåer --> Wilcoxon signed-rank test

#Laster inn datasett
data_shi <- mean_Shift

#Boksplott:
ggboxplot(data_shi, x = "Shift", y = "f1_score")
ggboxplot(data_shi, x = "Shift", y = "f1_score", add = 'jitter')


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

#Effektstørrelse:
data_shi  %>%
  wilcox_effsize(f1_score ~ Shift, paired = TRUE)


# Parameteren flipp ("Flip") ------------------------------------------------
#Parameteren flipp har to nivåer --> Wilcoxon signed-rank test

#Laster inn datasett:
data_fli <- mean_Flip

#Boksplott:
ggboxplot(data_fli, x = "flip", y = "f1_score")
ggboxplot(data_fli, x = "flip", y = "f1_score", add = 'jitter')

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

#Effektstørrelse:
data_fli  %>%
  wilcox_effsize(f1_score ~ flip, paired = TRUE)
```

RStudio-script for statistiske tester utført på eksperimentplan 2

```r
#Wilcoxon signed-rank test utført på resultatdataene til eksperimentplan 2.
#Alle parameterne har to nivåer --> Wilxocon signed-rank test


# Nødvendige pakker ----------------------------------------------------
library(PMCMRplus)
library(tidyverse)
library(ggpubr)
library(rstatix)

# Lysstyrke ("Brightness") ---------------------------------------------------------------
#Laster inn datasett
data_bri <- mean_Brightness

#Boksplott
ggboxplot(data_bri, x = "Brightness", y = "f1_score")
ggboxplot(data_bri, x = "Brightness", y = "f1_score", add = 'jitter')

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

#effektstørrelse
data_bri  %>%
  wilcox_effsize(f1_score ~ Brightness, paired = TRUE)

# Kontrast ("Contrast") ------------------------------------------------------------------
#Laster inn datasett
data_con <- mean_Contrast

#Boksplott:
ggboxplot(data_con, x = "Contrast", y = "f1_score")
ggboxplot(data_con, x = "Contrast", y = "f1_score", add = "Jitter")

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

#Effektstørrelse:
data_con  %>%
  wilcox_effsize(f1_score ~ Contrast, paired = TRUE)

# Støy ("Noise") -------------------------------------------------------------
#Laster inn datasett
data_noi <- mean_Noise


# Boksplott:
ggboxplot(data_noi, x = "Noise", y = "f1_score")
ggboxplot(data_noi, x = "Noise", y = "f1_score", add = "jitter")

```

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

#Effektstørrelse:
data_noi  %>%
  wilcox_effsize(f1_score ~ Noise, paired = TRUE)


# Uskarphet ("Blur")  ------------------------------------------------------------
#laster inn datasett:
data_blur <- mean_Blur

# Boksplott:
ggboxplot(data_blur, x = "Blur", y = "f1_score")
ggboxplot(data_blur, x = "Blur", y = "f1_score", add = "jitter")

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

#Effektstørrelse:
data_blur  %>%
  wilcox_effsize(f1_score ~ Blur, paired = TRUE)
```

Vedlegg D

Vedlegg D inneholder en oversikt over gjennomsnittlig Dice-score, HD_{95} og MSD per pasient oppnådd på testsettet til OUS-datasettet og det eksterne Maastro-testsettet. Fiolinplott som visualiserer fordeling av Dice-score til tverrsnittene for hver pasient i Maastro-testsettet ligger også vedlagt.

OUS-testsettet

Tabell D.1: Tabellen viser beregnet Dice-score, HD_{95}-verdier og MSD-verdier per pasient på testsettet oppnådd med augmenteringsmodellen.

Pasient-ID	Dice-score	HD_{95} [mm]	MSD [mm]
5	0,820	26,1	1,00
8	0,777	7,00	0,000
13	0,772	19,5	0,000
16	0,407	21,0	3,61
18	0,821	3,74	0,000
21	0,677	31,5	1,00
36	0,811	2,45	0,000
44	0,789	3,16	0,000
52	0,818	3,00	0,000
55	0,787	3,00	0,000
60	0,782	5,83	1,00
61	0,727	3,61	0,000
67	0,728	6,00	1,00
73	0,737	14,4	1,00
74	0,845	3,16	0,000
77	0,870	3,00	0,000
82	0,797	6,16	1,00
91	0,854	4,00	0,000
93	0,361	30,0	1,41
99	0,901	2,00	0,000
110	0.000	42,6	25,0
116	0,715	19,6	1,00
120	0,853	3,61	0,000
130	0,761	5,00	0,000
140	0,789	5,66	1,00
148	0,791	3,74	0,000
153	0,672	3,95	0,000
154	0,736	3,61	0,000
162	0,788	6,00	1,00
164	0,656	10,3	1,41
169	0,862	43,5	0,000
184	0,880	3,16	0,000
191	0,823	5,39	0,000
194	0,764	5,39	0,000
209	0,844	6,52	1,00
217	0,832	7,35	0,000
223	0,642	20,3	1,00
233	0,912	1,41	0,000
242	0,787	38,9	1,00

249	0,852	22,0	0,000

Maastro-testsettet

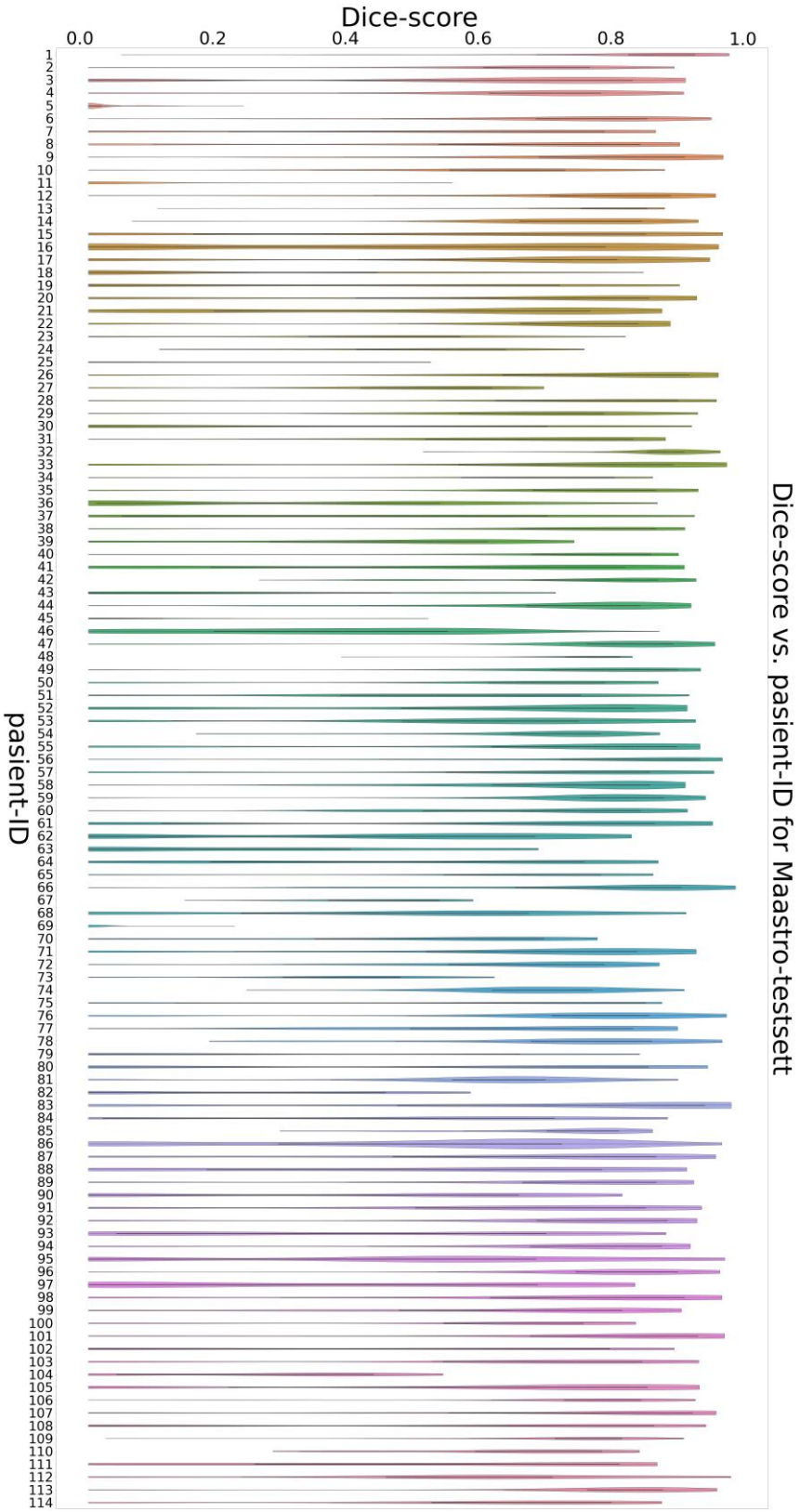
Tabell D.2: Tabellen viser beregnet Dice-score, HD_{95}-verdier og MSD-verdier per pasient på Maastro-testsettet oppnådd med augmenteringsmodellen.

Pasient-ID	Dice-score	HD_{95} [mm]	MSD [mm]
1	0,868	8,54	0,000
2	0,626	19,4	1,00
3	0,763	47,0	0,000
4	0,689	15,2	0,000
5	0,03	29,7	10,5
6	0,764	4,00	1,00
7	0,656	55,8	1,00
8	0,705	33,8	1,00
9	0,801	4,47	1,00
10	0,657	20,0	1,00
11	0,039	26,4	14,4
12	0,782	17,0	0,000
13	0,793	2,00	0,000
14	0,765	6,32	0,000
15	0,758	19,0	0,000
16	0,549	17,3	3,32
17	0,756	4,00	1,00
18	0,320	66,8	3,61
19	0,588	43,8	1,41
20	0,745	21,9	1,00
21	0,637	6,56	1,41
22	0,750	45,8	1,00
23	0,466	5,39	1,00
24	0,551	9,22	1,41
25	0,268	2,00	0,000
26	0,869	4,00	1,00
27	0,526	8,77	2,00
28	0,796	23,4	0,000
29	0,686	10,3	1,00
30	0,387	24,2	4,12
31	0,740	5,00	0,000
32	0,870	4,12	0,000
33	0,819	24,5	1,00
34	0,702	33,5	1,00
35	0,803	8,06	0,000
36	0,393	43,2	2,45
37	0,454	9,11	1,41
38	0,789	9,43	1,00
39	0,454	20,5	3,00
40	0,781	10,2	1,00
41	0,562	12,0	1,00

42	0,763	8,06	1,00
43	0,312	7,21	1,00
44	0,760	12,5	1,00
45	0,167	31,1	21,4
46	0,387	10,6	5,39
47	0,793	4,90	1,00
48	0,760	1,73	0,000
49	0,836	12,9	0,000
50	0,691	33,5	1,00
51	0,611	7,21	1,41
52	0,781	5,00	1,00
53	0,657	7,78	1,00
54	0,726	5,00	1,00
55	0,825	4,00	1,00
56	0,844	3,00	0,000
57	0,738	7,21	0,000
58	0,753	4,69	1,00
59	0,801	3,32	0,000
60	0,744	6,00	0,000
61	0,741	26,7	1,00
62	0,546	13,0	1,00
63	0,362	41,9	8,06
64	0,524	18,4	1,41
65	0,684	2,45	0,000
66	0,831	4,12	0,000
67	0,446	6,71	1,41
68	0,519	15,2	1,00
69	0,014	60,0	22,9
70	0,519	36,4	2,24
71	0,791	3,16	0,000
72	0,703	23,1	1,00
73	0,408	8,06	1,41
74	0,696	6,32	1,00
75	0,680	2,83	0,000
76	0,762	5,00	1,00
77	0,730	14,4	1,00
78	0,730	3,61	1,00
79	0,377	18,3	1,00
80	0,742	40,2	1,00
81	0,624	23,6	1,41
82	0,288	22,6	1,00
83	0,869	3,00	0,000
84	0,594	13,9	1,00
85	0,744	5,83	1,41
86	0,523	8,31	1,41
87	0,735	5,74	1,00
88	0,693	31,1	0,000
89	0,813	12,8	0,000
90	0,567	4,36	1,41

91	0,669	10,2	0,000
92	0,793	6,32	0,000
93	0,582	28,9	2,00
94	0,729	12,1	0,000
95	0,507	19,0	1,00
96	0,799	4,00	1,00
97	0,538	10,9	1,00
98	0,813	2,83	0,000
99	0,724	6,32	1,00
100	0,630	5,00	1,00
101	0,845	6,08	0,000
102	0,502	11,0	1,00
103	0,710	2,83	0,000
104	0,353	8,72	4,24
105	0,785	11,2	1,00
106	0,793	2,83	0,000
107	0,871	12,1	0,000
108	0,688	12,0	0,000
109	0,754	4,00	0,000
110	0,711	5,00	1,00
111	0,614	5,00	1,00
112	0,599	5,83	1,00
113	0,799	3,00	0,000
114	0,684	6,32	0,000

Fiolinplott for Maastro-testsettet



Figur D.1: Visualisering av fordeling av Dice-score til tverrsnittene for hver pasient i Maastro-testsettet oppnådd med den beste modellen, augmenteringsmodellen. Fiolinplottets form indikerer fordelingen av oppnådde Dice-score, der tykkelsen representerer antall tverrsnitt som har oppnådd den gitte Dice-scoren, mens lengden representerer variansen av Dice-score for hver pasient.

Vedlegg E

Vedlegg E inneholder eksempler på et PET/CT-bilde påført ulike kombinasjoner av augmenteringsteknikkene testet i denne masteroppgaven.

Figur E.1: Figuren viser et PET/CT-bilde påført ulike kombinasjoner av augmenteringsteknikkene rotasjon inntil 90 grader, zoom [0,8, 1,2], forskyvning [-10,10], vertikal flipp, lysstyrke [0,8, 1,2], kontrast [0,7, 1,3], støy [0, 0,05] og uskarphet [0,5, 1,5]. Den gule inntegningen representerer kreftsvulst tegnet inn av spesialister.

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

Postboks 5003
NO-1432 Ås
Norway