NTNU

Kunnskap for en bedre verden

Kvantifisering av histomorfologiske forandringer i hjertevev forårsaket av PMCV-infeksjon i Atlantisk laks - med bruk av bildeanalyseverktøy

Quantification of histomorphological lesions in heart tissue caused by PMCV infection in Atantic Salmon by using a tool for image analysis

Bacheloroppgave

Innleveringsdato: 18.12.2020

Gradering: TKJE3001

Forfatter: Berit Rø Tjøtta

Veileder: Ina Merete Stuen

Oppdragsgiver: Mowi Genetics AS

Kontaktperson: Matt Baranski

Forord

Denne bacheloroppgaven ble skrevet ved NTNU: Institutt for materialteknologi i samarbeid med Mowi Genetics AS. Oppgaven er et resultat av et eget ønske om å skrive en annerledes oppgave relatert til oppdrettsnæringen, og velvilje fra Mowi som gav meg oppgaven

Jeg vil rette en takk til Mowi og Matt Baranski som lot meg få skrive oppgave i samarbeid med dem. Videre ønsker jeg å takke intern veileder Ina Merete Stuen for gode råd og påminnelser, samt for at jeg fikk mulighet til å utsette oppgaven til høstsemesteret. Jeg vil også takke Jinni Gu ved veterinærinstituttet i Trondheim for stor hjelp med klassifiseringen av histopatologibildene.

Jeg ønsker å takke verdens beste venner for støtte, oppmuntring og hjelp til barnepass: Dere har alltid troa, selv når jeg ikke har det selv. Jeg vil gi en spesiell takk til Inge Matshaven for hjelp med programmering og for gode faglige råd, og til Eirik Knævelsrud fordi du overtalte meg til å teste ut ConvNets/CNN.

Jeg vil takke alle ved Programvareverkstedet (PVV) ved NTNU. Her vil jeg gi en spesiell takk til Richard Alexander Haydon for hjelp til skripet som konverterer svs-filene til png-filer, og til Torstein Nordgård-Hansen fordi du sprer så mye humor og positivitet, samt fordi du lager god kaffe til alle som vil ha.

Sist men ikke minst vil jeg takke datteren min Sofia, fordi du pyntet Mac`en min med masse koselige klistremerker, fordi du har vært så tålmodig med meg når jeg har vært stresset og sliten, og fordi du alltid er så glad.

Summary

Use of machine learning as a tool for developing a high resolution phenotype of Atlantic Salmon infected with PMCV were investigated. The main problem was evaluated as a classification problem, therefor a supervised learning method were chosen. Further three different machine learning methods were investigated and evaluated.

Support vector Machine is by definition a separable hyperplane, and this type of model is primarily designed for non-linearity and binary classification problems. Use of SVM as a machine learning tool related to the issue in this thesis, were therefore rejected. Traditional neural networks seems to have problems related to image analysis, there the main reason seems to be the lack of invariance related to geometric transformations. ConvNets is a special case of neural network, specially designed for image analysis and object recognition, were the architecture maintain invariance by special features characteristic for ConvNets. Recent studies have revealed that ConvNets has low robustness for noise, and does not maintain the invariance at a high level. Despite this it seems like ConvNets is the best choice for solving the given issue.

Therefore a suggestion for a possible ConvNet model were developed. The model were tested on a dataset, consisting of histopathological images classified by seven classes. Training of the model gave little success. Probably because the weights weren't initialized properly, something that may have caused problems with updating of the gradient. The cause can also be that the network were to complex compared to the dataset. Other causes may be that the images were prone to overlap, because of the little margin between the given classes, lack of zero-mean and poor resolution of the input-data.

Implementation of machine learning as a tool for solving the main problem seems to be a real probability. Still, it seems to remain some investigation and work before a high functional model - related to the specific histopathological lesions seen in PMCV injected Salmon. Therefore there were proposed several possibilities for further work, like optimalization and tuning of hyperparameters and implementation of wavelets.

Sammendrag

Muligheten for bruk av maskinlæring til å utarbeide en høyoppløsning av fenotypen til Atlantisk laks infisert av PMCV ble undersøkt. Problemstillingen ble vurdert til å omfatte et klassifiseringsproblem, og det ble derfor valgt å benytte en overvåket maskinlæringsmetode. Videre ble tre ulike maskinlæringsmetoder undersøk og vurdert.

Support Vector Machine er definert av et separabelt hyperplan, noe som medfører at SVM fortrinnsvis er designet for ikke-lineære og binære klassifiseringsproblemer. SVM som metode for å løse problemstillingen ble derfor forkastet. Tradisjonelle nevrale nettverk kan tenkes å fungere dårlig for klassifisering av bilder, da arkitekturen i nettverket medfører at antallet parametere raskt eskalerer, samt at tradisjonelle nettverk mangler invarians relatert til geometriske transformasjoner. ConvNets er en type nevrale nettverk designet for bildeanalyser, der den spesielle arkitekturen skal opprettholde invarians relatert til geometriske transformasjoner. På tross av at nyere studier har vist at ConvNets har lav robusthet mot støy og at invarians ikke alltid opprettholdes, ser det ut til at denne metoden er den mest optimale for å løse problemstillingen.

Det ble derfor utarbeidet et forslag til en ConvNet-modell, og modellen ble testet på et datasett bestående av manuelt klassifiserte bilder á syv klasser. Treningen av modellen gav liten suksess, årsaker til dette kan tenkes å være feil i programmeringen, og sannsynligvis ble ikke vektene initiert, noe som førte til at alle vektene startet med samme verdi. Dette medførte trolig at hverken gradienten eller vektene i modellen kunne oppdateres. Det kan også tenkes at modellen ble bygget for kompleks i forhold til datasettet som ble benyttet. Andre feilkilder kan være at bildene som ble klassifisert tenderte mot mye overlapp, manglende null-sentrering av input-data og dårlig oppløsning av input-data.

Implementering av maskinlæring som et kraftig verktøy for å løse problemstillingen i oppgaven ser ut til å være en reell mulighet. Likevel ser det ut til at det gjenstår både forskning og utvikling for å utarbeide en høy funksjonell maskinlæringsmodell, tilpasset den spesifikke histopatologien relatert til CMS. Det ble derfor utarbeidet flere forslag til veien videre, blant annet optimalisering og finstilling av nettverket, samt implementering av wavelets.

Innholdsfortegnelse

1 Innledning

Mowi regnes som et av verdens største oppdrettsselskap, og selskapet arbeider kontinuerlig for å opprettholde en bærekraftig vekst. Gjennom visjonen: *The Blue revolution – Our vision is to Lead the Blue Revolution and unlock the potential of the ocean in a way that respects our planet.* Har selskapet satt ambisiøse og bærekraftige mål for å sikre en positiv påvirkning av planeten. I praksis inkluderer dette bruk av bærekraftig for, reduksjon av klimagassutslipp, bruk av fornybare energikilder, redusering av plast i alle produksjonsledd, samt forbedring av fiskehelsen. Visjonen og ambisjonene underbygges av robuste investeringer innen forskning og utvikling internt i hele selskapet. Der hele Mowi-kjeden kontinuerlig ser etter nye innovative løsninger og en bærekraftig vekst (1).

For å opprettholde målene om en bærekraftig vekst og bedre fiskehelse står oppdrettsnæringen ovenfor flere problemstillinger. En av de største utfordringene ansees for å være hjertesykdommen kardiomyopatisyndrom (CMS), forårsaket av *Piscint myokardittvirus* (PMCV). Sykdommen rammer først og fremst stor fisk, etter at mesteparten av produksjonskostnadene er påløpt, noe som medfører store økonomiske tap for næringen (2).

Utvikling av en fungerende vaksine mot CMS har så langt ikke vist positive resultater. Det kan derfor synes at selektiv avl er det mest reelle tiltaket for bekjempelse og kontroll av sykdommen. Resistens mot PMCV-infeksjon er derfor et viktig avlsmål i Mowi sitt avlsprogram for Atlantisk laks (3). Kontrollerte smitteforsøk har vist at Atlantisk laks kan deles inn i to hovedgrupper, dette ut i fra den individuelle vertsresponsen ved PMCV-infeksjon (4). Ved å beregne korrelasjonen mellom fenotypen og genotypen kan arvbarheten beregnes og benyttes i avlsarbeid (5). Fenotypen som benyttes i dag er basert på diagnostikk av histopatologibilder av hjertevevet til laks med PMCV-infeksjon, der alvorlighetsgraden av CMS-relaterte endringer i hjertevevet vurderes etter en skala mellom 0 (ingen skade) og 3 (alvorlig grad av skade) (3, 4). Da denne kvantifiseringen er relativt unøyaktig, legger den begrensninger for hvor nøyaktig de genetiske parameterne kan estimeres, en bedre oppløsning av fenotypen kan derfor synes å forbedre nøyaktigheten i avlsarbeidet (3).

De siste årene har ny teknologi bidratt til å revolusjonere praktisk histopatologi, og i dag er bruk av Whole Slide Imaging (WSI) en vanlig prosedyre ved diagnostisering av histopatologibilder (6). WSI er en virtuell mikroskoperingsmetode, noe som gjør det mulig å konvertere fullstendige snitt av vevsprøver om til digitale filer (7). Sett i lys av den raske utviklingen innen kunstig intelligens og maskinlæring, kan det tenkes at det er mulig å optimalisere fenotypen med maskinlæring som verktøy.

Mowi ønsker derfor at det utarbeides en høyoppløsning av fenotypen til Atlantisk laks smittet av PMCV, dette ved hjelp av maskinlæring som verktøy. Der en høyoppløsning skal utarbeides ved å løse en bildeanalyse bestående av histopatologibilder med CMS-relaterte endringer (3). Herunder er tanken at bildeanalysen skal løses ved å utarbeide en lengre og mer detaljert klassifiseringsskala ved hjelp av maskinlæring. Denne klassifiseringsskalaen vil da tilsvare en høyoppløsning av fenotypen.

Problemstillingen for denne oppgaven er derfor å undersøke mulighetene for bruk av maskinlæringsmetoder til å lage en høyoppløsning av fenotypen til Atlantisk laks, smittet med CMS – samt å utarbeide et forslag til en maskinlæringsmodell som kan klassifisere histopatologibilder med CMS-relaterte endringer.

2 Teori

2.1 Kardiomyopatisyndrom (CMS)

Kardiomyopatisyndrom (CMS), også kjent som hjertesprekk, er en infeksiøs hjertelidelse som først og fremst rammer atlantisk oppdrettslaks (8). Da CMS svekker hjertemuskulaturen, fortrinnsvis forkammeret og tilhørende blodkar, regnes sykdommen som et krevende fiskehelseproblem. Svekkelsen av hjertemuskulaturen gjør fiskehjertet skjørt, og i verste fall kan ruptur av hjerteveggen inntreffe – derav navnet hjertesprekk (9). Etiologien til sykdommen var ukjent fram til 2009 (8), det ble da gjort funn av en patogen agens som med stor sannsynlighet kunne knyttes til CMS. I 2011 ble et nytt virus tilknyttet CMS indentifisert; *Piscine myocarditis virus* (PMCV). PMCV ansees fremdeles for å være den forårsakende agensen til CMS (10).

CMS forekommer ofte som en kronisk lidelse med en langsom utvikling. Der sykdommen gir en generell forhøyet dødelighet, samt lavere vekstrate over en lengre tidsperiode. Smittet laks kan vise kliniske tegn til å være smittet, men i all hovedsak vil smittet laks vise få kliniske tegn, noe som kan gjør sykdommen vanskelig å oppdage (11). Sykdommen oppdages da ved akutt dødelighet av tilsynelatende frisk og hurtigvoksende laks (normalt utseende, næringsopptak og normal vektskurve) (11). Den akutte dødeligheten opptrer som regel i forbindelse med stressende håndteringer, som avlusning eller overføring til slaktemerder. Eller som følge av ugunstige miljøforhold (9). Diagnostisering av CMS gjøres på grunnlag av klinisk undersøkelse, obduksjon og ved histopatologisk undersøkelse (2).

2.1.1 Laksehjertets normale anatomi og funksjon

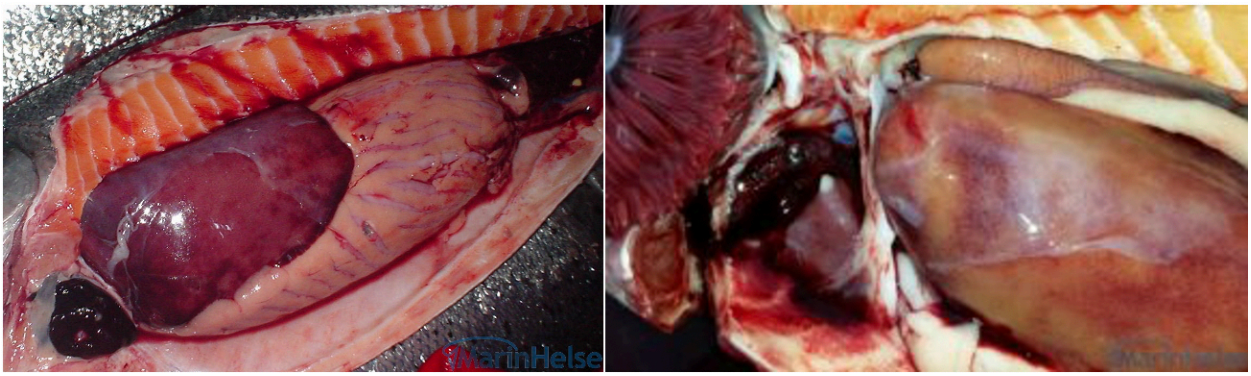
Laksen har et relativt enkelt sirkulasjonssystem. Laksehjertet består av *sinus venosus* (venesekken), *bulbus arteriosis*, samt hjertekamrene; atrium (forkammer) og ventrikkelen (hjertekammer). Ved normal hjertefunksjon pumpes blodet en runde rundt i fiskekroppen, der laksen via gjellene kvitter seg med CO_2 i vannet og tar opp oksygen til blodet. Det oksygenfattigblodet ankommer venesekken og atrium, hvor det så pumpes videre til ventrikkelen gjennom bulbus og inn i gjellene. Et jevnt trykk i blodstrømmen opprettholdes ved hjelp av bulbus (12).

Hjerteveggen består av tre ulike lag, herunder endokard, myokard og epikard. Endokard består av et enkelt cellelag som kler innsiden av hjertet. Myokard defineres som muskelvevet i hjertet. Epikard er et cellelag som kler utsiden av myokard, og epikarden danner det indre laget i hjerteposen (perikard) (12). Muskelveggen i ventrikkelen består av et kompakt muskellagg kalt, kalt kompaktum. Det indre muskellaget i ventrikkelen er mer spongiøst og svampaktig (spongiøs myokard). Atrium mangler kompaktum og består hovedsakelig av spongiøst myokard (8).

Figur 2.1: Laksehjertet - normal anatomi, bilde til venstre viser retningen på blodstrømmen gjennom de fire hjertekamrene, bilde til høyre viser kompaktum i ventrikkel (markert med røde stjerner), og spongiøst myokard i atrium (markert med blå prikk) (13).

2.1.2 Kliniske symptomer og funn – relatert til CMS

Kliniske symptomer som kan opptre ved PMCV-infeksjon er tegn til sirkulasjonssvikt. Hvor dette kan observeres som sløvhet hos laksen - såkalte «svimere», blødninger i huden (fortrinnsvis under buken), utstående øyne (exopthalmia) og skjellommeødem. Skjellommeødem kjennetegnes ved at laksen får et grålig og rufsete preg i merden. Ved obduksjon gjøres det ofte funn av unormale forandringer i hjertet, herunder forstørret og utspilt atrium og ventrikkel. Ved ekstreme tilfeller kan atrium revne, og det gjøres funn av blodkoagel i hjertesekken (hjertetamponade). Andre unormale funn under obduksjon er ascites-væske i bukhulen, mørkfarget lever med et fibrinøst belegg på overflaten og forandringer i milten (9).

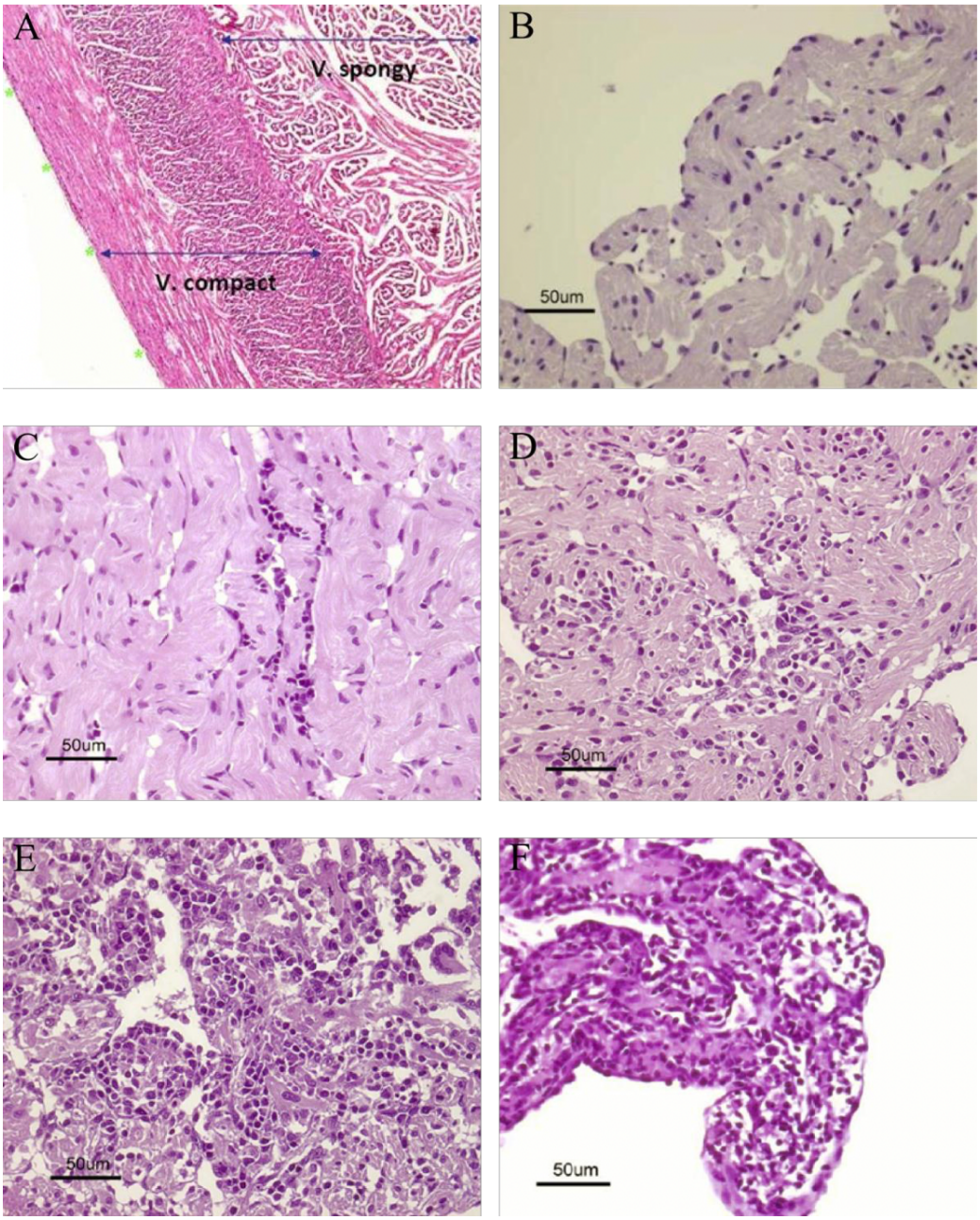


Figur 2.2: Laks med kliniske symptomer til CMS - Bildet til venstre viser størknet blod i bukhulen til laks. Bildet til høyre viser moderate endringer av leveren, der denne er dekket av et fibrinslør (13)

2.1.3 Karakteristisk histopatologi i hjertet til Atlantisk laks – relatert til CMS

Død laks kan ha betydelige histopatologiske forandringer i hjertet, uten at det gjøres makroskopiske funn ved klinisk undersøkelse og obduksjon. Diagnosen avgjøres derfor i hovedsak med tanke på histopatologiske endringer, funnet i atrium og ventrikkelen. (10). CMS karakteriseres av betennelsesendringer og fortykning av det endokarde, og nekrose av muskelceller i spongiøst myokard (9). Sparsommelige og moderate funn av betennelsesinfiltrater i epikardiet rundt atrium og ventrikkel, er også vanlig (8). Kompaktum i ventrikkelen er som regel ikke affisert, men i enkelte alvorlige tilfeller kan det utvikles en cellerik epikarditt (betennelse) på ytterflaten av hjertet. Betennelsesceller fra epikarditt (12) kan da spre seg innover i ventrikkelens kompaktum-lag via blodkar (8).

Forandringer og betennelsesinfiltratet opptrer først i atrium, deretter sprer det seg videre til ventrikkelen. Som oftest vil endringene være mest omfattende og alvorlige i atrium (8). Betennelsesinfiltratet består av mononukleære celler, som mest sannsynlig er lymfocytter og makrofager (10). I tidlige stadier av sykdommen vil endringene opptre som multifokale betennelsesforandringer i atrium. Betennelsesvevet vil være omgitt av, og tydelig separert fra friskt muskel vev (Figur 3C) (10). Etterhvert som sykdommen utvikler seg vil det multifokale betennelsesvevet gradvis gå over til mer diffuse og omfattende forandringer. Dette medfører at antall lymfocytter også vil øke (figur 3D). I senere stadier vil man se en utbredt inflammasjon og store skader, der betennelsesvevet fortrenger det normale muskelvevet (figur 3E) (8).



Figur 2.3: A: kompaktum og myokard i ventrikkel, B: Normal myokard i atrium, C: multifokale endringer i atrium, D: gradvis mer diffuse endringer i atrium, E og F: Betennelsesinfiltrat erstatter nesten alt normalt vev i atrium (10).

Affiserte muskelceller viser tap av normal struktur og en overproduksjon av eosinofile celler (type hvite blodceller), og de eosinofile cellene kan skille ut enzymer som skader muskelvevet ytterligere. I noen tilfeller er det også observert hypertrofiske cellekjerner og fibrose, der årsaken mest sannsynlig er for å kompensere for den pågående sirkulasjonssvikten (10). Alvorlige tilfeller av CMS med omfattende betennelsesendringer kan medføre at atrium eller ventrikkelveggen brister (9).

Ved diagnostisering av CMS vurderes både atrium og ventrikkel. Det gis en individuell score mellom 0-3, relatert til utbredelsen av betennelsesvev. Score 0 tilsvarer normal myokarditt (figur 3B). Score 1 tilsvarer noen få multifokale endringer, tydelig omsluttet av frisk vev (figur 3C). Score 2 tilsvarer mer utbredte og diffuse endringer (figur 3D). Score 3 gis for tilfeller der det spongiøse vevet nesten er erstattet av betennelsesinfiltrat (figur 3E og 3F) (10).

2.1.4 CMS – utvikling og utfall på fiskegruppenivå

Det finnes noe usikkerhet rundt den eksakte inkubasjonstiden for smitteutsatt laks i sjø. Tidligere feltstudier og forsøk har vist at PMCV kan være tilstede i oppdrettsanlegg over lang tid, uten at det påvises klinisk symptomer relatert til CMS i fiskegruppa (9). Som nevnt tidligere kan det i imidlertid se ut til at CMS tenderer mot å være en kronisk sykdom. Der laksen kan leve med sykdommen over flere måneder og år, før kliniske symptomer og død inntreffer (11). Det antas at laksens hjerte- og sirkulasjonssystem har en betydelig reservekapasitet, noe som kan maskere sykdommen slik at kliniske tegn oppstår sent i sykdomsforløpet (9). Kliniske tegn og akutt død relatert til CMS har primært blitt sett hos hurtigvoksende og stor laks, like før de når slaktevekten (2-5 kg) (14), og som regel i løpet av det andre året etter sjøsetting. De siste årene er det derimot observert tendenser til at små laks (100-300g) også rammes, ofte få måneder etter sjøsetting (2).

Smitteforsøk har vist at viralt RNA oppformeres i flere organer og kan påvises i laksen fra 4 uker etter smitte, samt at histopatologiske endringer kan påvises fra 6 uker etter smitte. Det er påvist korrelasjon mellom konsentrasjonstopp for viralt RNA i hjertet, milt og nyrer, og alvorlighetsgraden av histopatologiske endringer i organene. Herunder med signifikant korrelasjonen mellom konsentrasjon av viralt RNA i hjertet, alvorlighetsgraden av hjertepatologien, hjertespesifikk T-celle respons og aktivering av komplementsystemet i milten. Disse korrelasjonene kan tyde på at den primære agensen til den spesifikke hjertepatologien, er den cytopatogene effekten (4).

Det ser ut til at den individuelle vertsresponsen tenderer mot to ulike hovedtyper, noe som gjør at smittet laks kan deles inn i to ulike hovedgrupper; «high responders (HR)» og «low responders (LR)» (4). Ved hjelp av transkripsjons-analyse er det også gjort funn av ulike

karaktertrekk og genmarkører relatert til underliggende og molekylære mekanismer i vertsresponsen (4). Resultatene ble funnet ved å sammenligne ulike individer ved tre ulike stadier; 2-4 uker etter smitte (tidlig stadiet), 6 uker etter smitte (mellom stadiet) og 8-10 uker etter smitte (sent stadiet).

Begge gruppene viste tilnærmet lik respons ved det tidlig stadiet, der de hadde tilsvarende nivåer av viralt RNA og lik histopatologi score (0-1). Ved mellom stadiet viste begge gruppene like nivåer av viralt RNA, men HR-gruppen viste signifikante endringer i histopatologien sammenlignet med LR-gruppen. Ved sene stadier av smitteresponsen hadde HR-gruppen både høyere nivå av viralt RNA og mer uttalte histopatologiske endringer, enn LR-gruppen (4).

Ved mellom stadiet ble det identifisert 12 gener som kun var indusert hos HR-gruppen, noe som kan tyde på at disse genene er markører for tidlig patologi. Det ble også identifisert 9 gener som kun var indusert hos LR-gruppen, noe som da kan tyde på at dette er markører for rekonvalesens. Genene identifisert hos HR-gruppen kan alle relateres til immunologiske funksjoner. Genene identifisert hos LR-gruppen ser derimot ikke ut til å være relatert til immunologiske funksjoner (4).

Ved 8-10 uker etter smitte ble det identifisert 116 gener som kun var oppregulert hos HR-gruppen, der alle kan relateres til det spesifikke immunforsvaret. Det antas derfor at disse er markører for alvorlig patologi. Hos LR-gruppen ble det identifisert 33 ulike gener, der alle hovedsakelig kodet for enzymer som kan knyttes til metabolske reaksjonsveier eller til celle-respirasjon. Noe som kan tyde på at genene identifisert i LR-gruppen er markører for rekonvalesens og overlevelse (4).

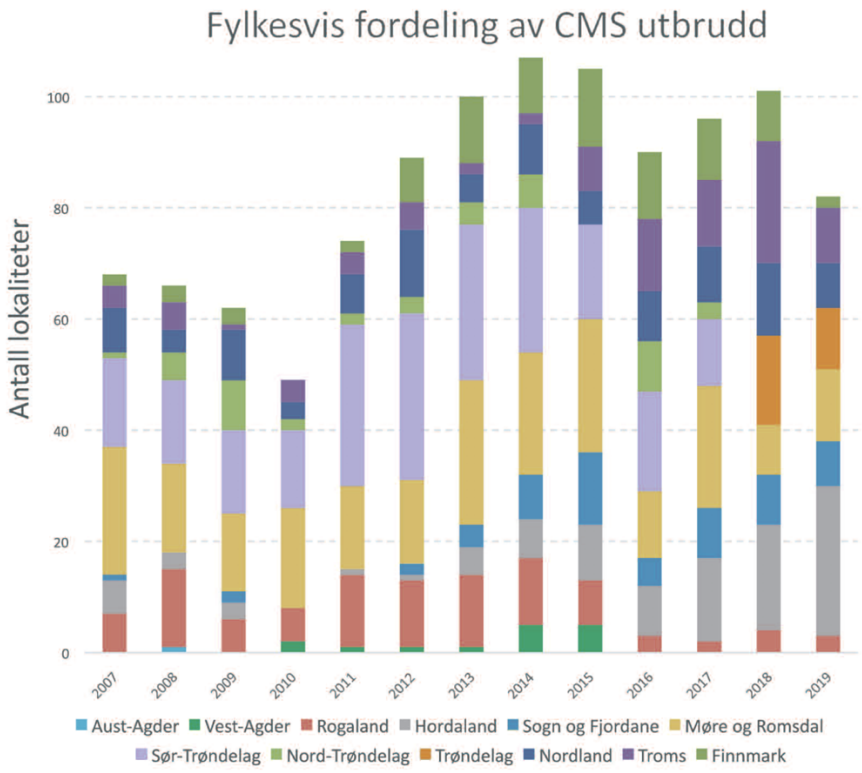
2.1.5 Utbredelse og omfang av CMS-smitte hos Atlantisk laks

De første tilfellene av CMS hos oppdrettslaks ble oppdaget på Nordmøre i 1985 (11). Likevel ble ikke CMS-smitte i Norske oppdrettsanlegg beskrevet vitenskapelig før i 1988. Senere ble det også påvist tilfeller på Færøyene (1992), i Skottland (1995) og Irland (2012). Etter 2000 er det funnet tilfeller med lignende patologiske endringer hos både vill Atlantisk laks, og vill Chinook laks i Canada (10).

CMS forekommer i dag hos oppdrettslaks langs hele norskekysten (2). CMS-utbrudd kan påvises hele året, men sykdommen tenderer mot å ha økt forekomst ved senvinter/tidlig vår, eller på høsten. Generelt ser det ut til å være lav forekomst av utbrudd i løpet av sommermånedene (10). Hovedsetet for CMS-utbrudd ansees å være i Midt-Norge (Trøndelag-fylkene og Møre og Romsdal) (11).

CMS så lenge ut til å være svakt progredierende, deretter så det i noen år ut til at sykdommen var avtagende. Fra påvisning om utbrudd ved 80 ulike lokaliteter i 2006, til 47 ulike lokaliteter i 2010. Fra 2010 har sykdommen igjen hatt en progredierende tendens. Mellom årene 2012-2019 ble det hvert år påvist utbrudd ved mellom 89-107 ulike lokaliteter i Norge (2). Tall fra veterinærinstituttet fra de tre siste årene, viser en nedgang i antall utbrudd i de tre nordligste fylkene. Derimot viser Hordaland en sterk økning i antall utbrudd for samme tidsperiode (15).

I 2019 påviste veterinærinstituttet CMS-utbrudd ved 82 ulike lokaliteter i Norge. Det ble også rapportert om 155 utbrudd fra private laboratorier. Tallene viser en kraftig økning fra 2018, der det ble påvist smitte ved 125 forskjellige lokaliteter. Det kan tenkes at det finnes noe overlapp mellom påvisning utført av private laboratorier og påvisninger gjort av veterinærinstituttet. Dette fordi lokalitetenes identitet ikke oppgis, samt at CMS ikke er meldepliktig, noe som kan medføre underregistrering. Nevnte faktorer medfører vanskeligheter med å vurdere den helhetlige utviklingen de siste årene. Likevel ser det ut til å være en tendens til at antall påvisninger holder seg totalt, eller er svakt økende (15).



Figur 2.4: Figuren viser en fylkesvis oversikt over CMS-utbrudd fra 2007-2019. Her kan man se at hovedsete for CMS-utbrudd fortsatt kan ansees å være i Midt-norge/Møre og romsdal. Og at Hordaland tar kraftig innpå, med en sterk økning av utbrudd de siste tre årene(15).

2.1.6 Økonomiske konsekvenser i oppdrettsnæringen – forårsaket av CMS-utbrudd

CMS-utbrudd er relatert til omfattende økonomiske tap på fiskegruppenivå. Sykdommen er per i dag, en av de største tapsfaktorene for norsk oppdrettsnæring. Det antas at akutte dødsfall av slakteklar laks etter at mesteparten av produksjonskostnadene er påløpt, medfører størst økonomiske tap. Kroniske utbrudd med langsom utvikling gjennom nesten hele produksjonssyklusen ansees også for å gi store konsekvenser, både for drift og økonomi (15).

Det finnes relativt få analyser over de totale økonomiske konsekvensene forårsaket av CMS. I 2003 ble det estimert at Norske oppdrettsanlegg hadde et totalt tap på € 28 millioner (årlige og direkte kostander) (16). Helseøkonomiske beregninger utført av Marine Harvest i 2007, estimerte at det totale CMS-relaterte produksjonstapet var på rundt 200 millioner for næringen. Foruten om de direkte kostandene som følge av tapt biomasse, kommer det i tillegg andre kortsiktige utgifter som kan forbindes med CMS-utbrudd, eksempler på dette er

allokering av ressurser og ekstraordinære arbeidsoppgaver. Det kan også tenkes at stor dødelighet hos stamfisk kan medføre tap av verdifull genetikk (15).

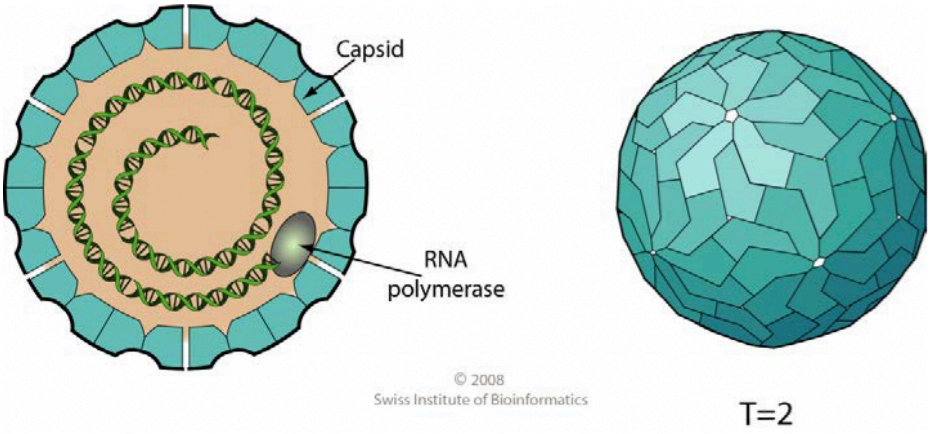
2.2 *Piscine Myocarditis virus (PMCV)*

Genomet til *Piscint myokardittvirus* (PMCV) har mange likhetstrekk med virus tilhørende *Totiviridae* familien. Denne familien består av fem kjente genus med virus, som alle forårsaker vedvarende infeksjoner hos ulike parasitter og sopp (17). Det er nylig oppdaget flere virus med likhetstrekk til *Totiviridae* familien. Der de fleste forårsaker vedvarende infeksjoner i reker, mygg, bananfluer og andre insekter. Det er også gjort funn av et PMCV-lignende virus i karpefisken Golden shiner (9).

De registrerte *Totiviridae* virusene overføres til nye celler ved celledeling, sporogenese (produksjon av sporer), eller ved cellefusjon. PMCV og flere av de nylig kjente toti-lignende virusene har derimot ekstracellulær overføring, viruset overføres altså via cellens omgivelser. De har dessuten en ekstra proteinkodende sekvens i genomet, og man antar at denne sekvensen kan knyttes til hvordan vertscellen skiller ut, og/eller tar opp virus fra omgivelsene (17).

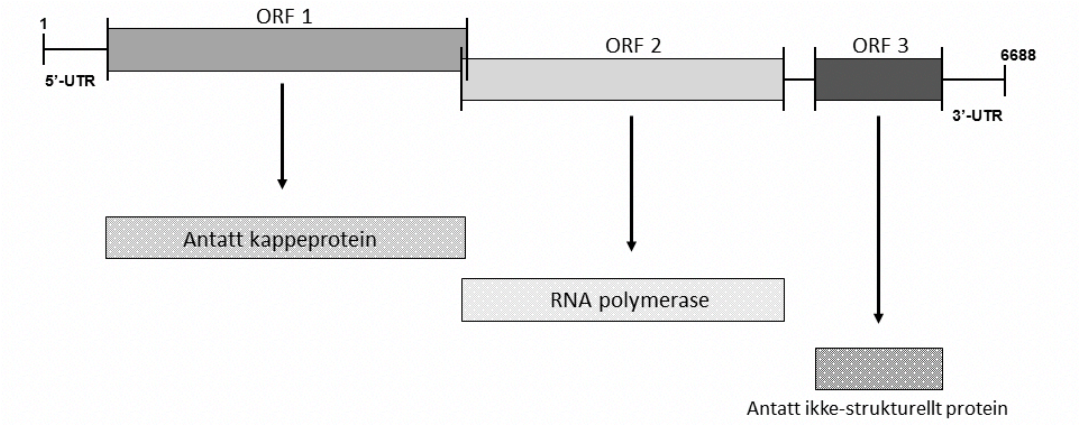
2.2.1 Struktur og oppbygning av PMCV

PMVC er sfæriske virioner med en diameter på rundt 50 nm. Det antas at partiklene er relativt enkle uten en ytre membran og at nukleinsyrekjernen er omsluttes av et proteinskall (kapsid), bestående av mange identiske kopier av et kappeprotein med form som et ikosaeder. Nukleinsyrekjernen er et enkelt segment av et dobbeltrådet RNA (ds-RNA) med en størrelse på 6,688 basepar. Den består av tre åpne leserammer; ORF1, ORF2 og ORF3 (9).



Figur 2.5: Oppbygning av virus i Totiviridae familien, et segment med dobbelttrådet RNA er pakket inn i et enkelt proteinskall - bestående av et mangefoldig kappeprotein (19)

Det antas at ORF1 representerer proteinkappen til virionet, og at ORF 2 koder for RNA-avhengig polymerase. Funksjonen til det uttrykte ORF3-proteinet er derimot mer usikker. Men, det kan se ut til at sekvensen er relatert til infisering av multi-cellulære organismer ved ekstracellulær overføring. En hypotese er derfor at sekvensen koder for fiber-lignende proteiner som deltar i den ekstracellulære overføringen. Det antas også at ORF3 kan kode for ustrukturerte proteiner som kun er tilstede i infiserte celler, og at proteinene har en rolle relatert til den celle-lytiske mekanismen i cella. En siste og interessant hypotese er at ORF3 kan relateres til den karakteristiske celle- og vevsdøden som man finner i hjertet til laks med alvorlig grad av CMS (17).



Figur 2.6: Figuren viser tre åpne leserammer; ORF1, ORF3 og ORF3. Der funksjonen til ORF3 ikke er kjent, men det antas at dette ikke bidrar til struktur. Det kan også tenkes at genet kan relateres til den karakteristiske histopatologien som man kan finne hos laks med CMS (17).

2.2.2 PMCV – Mulige reservoarer og smitteoverføring

Det finnes ulike studier hvor det er undersøkt hvorvidt villaks kan fungere som smittekilde og reservoar for PMCV . I en studie utført i Norge mellom 2007-2009 ble villaks fra 35 ulike elver testet for PMCV. Resultatene viste funn av PMCV hos 0.25% av de testede individene (18). Ved en annen studie foretatt i Norge ble det gjort funn av PMCV hos 0,22% av de testede individene. Totalt fra begge studiene ble det gjort funn av PMCV i bare tre villaks, der alle hadde Ct-verdier mellom 24.9 og 29.2 (9). Det er også påvist at virussekvenser isolert fra villaks har tilnærmet samme arvestoff som virussekvenser isolert fra oppdrettslaks (17). Dette kan tyde på at det er en smitteoverføring mellom villaks og oppdrettslaks, men siden forekomsten av PMCV i villaks er relativt lav, kan det tenkes at PMCV hos villaks stammer fra rømt oppdrettslaks. Det antas derfor at villaks hverken er en reell smittekilde eller et fungerende reservoar for PMCV (9).

Eventuell forekomst av PMCV i 32 ulike fiskearter ble undersøkt, dette i forbindelse med en studie av marine fiskearter. Studien viste funn av PMCV ved 6.6% av prøvene fra vassild (*Argentina silus*), vassild kan relateres til oppdrettslaks via foret. Sekvenseringen av arvematerialet til PMCV funnet i vassilda viste ikke noe nært slektskap til PMCV funnet i oppdrettslaks. Det ser derfor ut til at heller ikke vassild fungerer som et reservoar, eller som smittekilde av PMCV til oppdrettslaks (9). Miljøprøver tatt av sediment, plankton, biofilm og ulike organismer (bunnlevende, frittlevende og fast-sittende) fra oppdrettsanlegg, har kun vist forekomst av PMCV i prøver som stammet fra selve laksen. Det kan derfor tenkes at det viktigste reservoaret for PMCV er oppdrettslaksen selv, samt miljøet den lever i (9).

Det ser ut til at den mest vanlige smitteoverføringen av PMCV skjer ved horisontalsmitte, der PMCV kan overføres mellom laks i tanker, mellom merder og mellom oppdrettsanlegg (17). *In vivo* eksperimenter bygger opp under dette, da det er påvist at IP-injisert laks kan overføre smitte til andre oppdrettslaks innad i merden, samt at individer som smittes innad i merden utvikler patologiske forandringer som er karakteristiske for CMS (17).

Det er per i dag ikke rapportert utbrudd av CMS i yngel, smolt eller stamfisk under ferskvannsfasen. Dog er det funnet PMCV i stamfisk etter overføring fra sjø til ferskvann, og i enkelte grupper er det funnet PMCV hos opp til 70-90% av stamfisken. Det er også funnet lave

nivåer av viralt RNA fra PMCV i yngel. Et nylig utbrudd av CMS på Færøyene kobles mot egg importert fra Norge. Sett i sammenheng med den høye tilstedeværelsen av PMCV hos stamfisk, samt at man har funnet PMCV hos yngel like etter overføring til sjø. Kan det tenkes at PMCV kan overføres fra stamfisk til avkom, altså at vertikal smitte også spiller en vesentlig rolle (19).

2.3 Kontroll og bekjempelse av CMS i oppdrettsnæringen

2.3.1 Risikofaktorer for CMS-utbrudd i oppdrettsanlegg

Ved en studie gjort i regi av veterinærinstituttet fra 2013, ble ulike risikofaktorer for CMS i norsk oppdrettsnæring analysert. Dette ved bruk av en kohort studie, kohort i denne sammenhengen er definert som en gruppe av smolt-laks som sjøsettes i samme oppdrettsanlegg, i løpet av en periode på noen måneder. Smolt-laksen i samme kohort må også oppdrettes sammen frem til de er slakteklare. Smolten kan komme fra ulike leverandører, men de er alltid innen samme årsklasse. Produksjon av laks i kohorter regnes som den dominerende produksjonsmetoden i norsk oppdrettsnæring. Videre ble det brukt data basert på innrapporterte CMS-saker fra kjente oppdrettsanlegg i løpet av perioden 2004 til 2012. Datamaterialet inkluderte alle CMS-relaterte funn, uavhengig av om det kun var mistanke om CMS, eller konkludert CMS-utbrudd. Det ble også benyttet tilgjengelig data over andre vanlige sykdommer, for eksempel hjerte- og skjelettmuskelbetennelse (HSBM), infeksiøs pankreasnekrose (IPN) og pankreassykdom (PD) (20)

For å beregne infeksjonspresset ble det antatt at kohortene kan utsettes for CMS-infeksjoner fra alle hypotetiske kohort med infisert laks, og at smittepresset avhenger av sjøavstanden mellom tilgjengelige og infiserte kohorter. Ut i fra dette ble infeksjonspresset kalkulert etter følgende formel (formel 1) (20).

$$iP_i(t) = I_i^s(t) \sum_{jEN_i(t)} \frac{I_j^I(t) x_j(t)}{d_{ij}} \qquad (1)$$

Der *i* defineres som kohort og iP_i er kalkulert i måneder (t). *I(t) er* satt til 1 når kohort j har smitte ved tidspunktet t, *I(t)* er lik 0 i tilfeller uten forekomst av smitte. Fordi smitteavstandene til CMS ikke er kjent ble smitteavstanden testet med ulike og antatte verdier; 5, 10, 25, 50 og 100 kilometer (20)

Resultatene fra studien viste at de største risikofaktorene for utvikling av alvorlig CMS i oppdrettsanleggene er kohort tid, CMS utbrudd i tidligere kohort, HSBM i samme kohort, infeksjonspress og kohort størrelsen (20).

2.3.2 Utvikling av vaksiner mot PMCV

Det er i dag en utbredt bruk av vaksiner innen oppdrettsnæringen, og vaksiner ansees som et viktig verktøy for å forbedre fiskehelsen, samt for å bekjempe ulike bakterie- og virusinfeksjoner. Dette har ført til at det nå brukes langt mindre antibiotika, enn hva som ble brukt på 90-tallet. Dog ser ut til å være noe problemer knyttet til bruk av olje-adjuvans, da olje-adjuvans i vaksiner kan redusert appetitt og hemme veksten (10). Det er også observert at vaksiner kan forårsake skader mellom organer plassert i buken, samt gi pigmentforandringer i muskulaturen, noe som medfører dårligere slaktekvalitet. Enkelte laks kan også utvikle autoimmune reaksjoner mot olje-adjuvansen, noe som kan redusere slaktekvaliteten ytterligere (10).

De positive sidene ved bruk av vaksiner innen oppdrettsnæringen ser ut til å overgå de negative bieffektene (10). Utvikling av en vaksine mot PMCV er derfor en pågående prosess. Dessverre har det så langt vist seg å være noe problematisk, da det ikke finnes cellelinjer for *in vivo* virusreplikasjoner (17). Det antas at det kan være mulig å utvikle en velfungerende vaksine, da sekvensering av PMCV-genomet funnet hos oppdrettslaks har vist få genetiske variasjoner (17).

2.3.3 Stressreduserende tiltak og alternative metoder for å bedre den generelle fiskehelsen

Per i dag eksisterer det ingen behandling mot CMS, og forebyggende tiltak mot sykdommen kan derfor tenkes å være viktig for å redusere dødeligheten. Som nevnt viser erfaringer at ulike former for stress kan bidra til å utløse og/eller øke dødeligheten. Oppdrettere gis råd om

å behandle laksen så forsiktig som mulig, men dette kan være utfordrende å gjennomføre i praksis. Et av hovedproblemene i oppdrettsnæringen kan tenkes å være problematikken rundt lakselus og ulike parasitter, samt utvikling av resistens mot behandlingsmidlene. Dette fører til at laksen ofte må behandles med metoder som genererer mye stress, noe medfører et økt oksygenbehov for laksen. Gjellelidelser kan også bidra til økt dødelighet, særs for laks med CMS. Erfaringer viser at utbredt formalinbehandling mot gjelleparasitter og sopp kan redusere CMS-relatert dødelighet (9)

2.3.4 Selektiv avl av Atlantisk laks

Ved omfattende avlsarbeid utnytter man variasjonene som allerede finnes i naturen, slik at produksjonsegenskapene til arten forbedres. I oppdrett fjernes den naturlige konkurransen mellom individene og erstattes av menneskebestemt seleksjon (5). En forutsetning for fremgang innen avlsarbeidet er at de ønskede egenskapene som det selekteres for har genetisk variasjon. Da dette gjør det mulig å måle arvbarheten, en verdi mellom 0-1 (21).

Egenskapene som skal endres må kunne registreres eller måles *in vivo* på levende dyr som er avlskandidater, og/eller på levende og nære slektninger av avlskandidaten. Egenskapene endres generelt ved bruk av kvantitativ genetikk, der man baserer seg på at det finnes en genetisk variasjon i de egenskapene som man ønsker å endre. Egenskaper kan også endres ved seleksjon for egenskaper som er genetisk korrelert til ønskede egenskaper (5).

Tre ulike kriterier må oppfylles for at egenskaper skal kunne tas med i avlsmålet for nye arter, og disse er følgende: De avlsrelaterte kostandene for å fremme eller undertrykke den bestemte egenskapen må være innenfor rimelighetens grenser, egenskapen må medføre betydelige økonomiske tap for oppdretter og/eller næringen som helhet, og det må være mulig å måle arvbarheten til egenskapen. Med økende antall egenskaper i avlsmålet reduseres den genetiske fremgangen per egenskap. Det er derfor vanlig å starte med noen få grunnegenskaper, og deretter addere flere (5).

2.3.4.1 Avlsarbeid som fremmer egenskaper for sykdomsresistens

Avlsarbeid som fremmer sykdomsresistens er viktig innen oppdrettsnæringen, og ansees som et viktig forskningsfelt. Laks viser i noen tilfeller svært tydelige genetiske variasjoner for enkelte sykdommer, og i enkelte tilfeller er variasjonene så tydelige at egenskapen kan brukes som kriterium i seg selv. Omfattende kunnskap om de genetiske parameterne er en forutsetning for et vellykket avlsarbeid. Da de genetiske parameterne danner grunnmuren for utarbeidelse av avls-mål og systemer basert på registrering og indekser. Herunder er det essensielt å kjenne til de genetiske og miljømessige sammenhengene mellom ønskede egenskaper, dette for å fremme ønsket genetisk fremgang i populasjonen (5).

Innen akvakultur har det de siste årene blitt vanlig å benytte funksjonell genomforskning til å skaffe kunnskap om de genetiske parameterne. Ved funksjonell genomforskning undersøker man hvilke gener som uttrykkes i ulike livsstadier og/eller ved ulike miljøforhold. Metoden kan gi genetisk informasjon mellom individer og populasjon ved at ulike gener uttrykkes selv under identiske miljøforhold. Det ansees derfor å være langt enklere å oppnå en mer omfattende forståelse for sammenhengen mellom genotypen og fenotypen ved bruk av denne metoden (5).

2.4 Moderne histopatologi i praksis

Histopatologi kan defineres som læren om sykelige endringer i celler og vev. Ved patologiske undersøkelser stilles en diagnose på grunnlag av sykdommens årsak, patogenese og forandringer i struktur og funksjon (22). Generelt stilles diagnoser ved at en patolog, veterinær eller bioteknolog analyserer vevsprøver ved hjelp av mikroskopering. Diagnostiseringen kan derfor være en tidkrevende prosess, men de siste årene har utvikling av ny teknologi bidratt til å effektivisere og forenkle histopatologi i praksis (6). I dag er derfor mesteparten av analyse/diagnostiserings-prosessen digitalisert ved hjelp av Whole-slide imaging (WSI) (6).

2.4.1 Whole Slide Imaging (WSI) – et revolusjonerende hjelpemiddel innen histopatologi

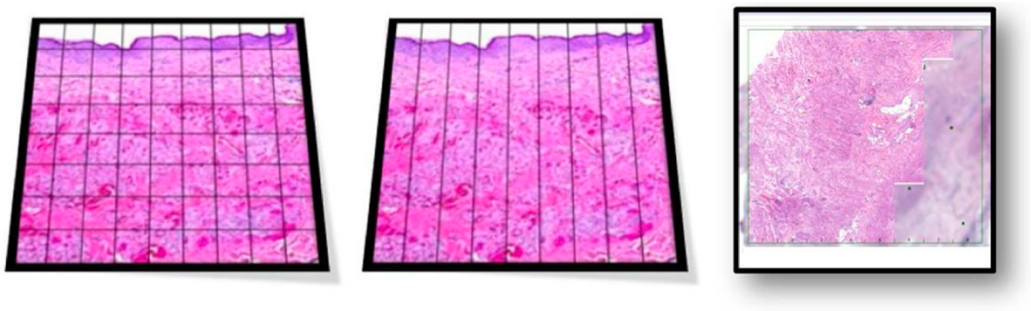
Whole Slide Imaging (WSI) er per definisjon en virtuell mikroskoperingsmetode, noe som gjør det mulig å konvertere fullstendige patologiske snitt av vevsprøver til digitale filer. Dette med

en oppløsning på lik linje med mikroskopiske standarder. WSI kan deles inn i to ulike hovedprosesser. Den første prosessen involverer bruk av en spesialisert maskinvare (en skanner) som skanner hele lysbilder, skannede bilder av vevsprøver lagres da som en digital slide (digital fil med høy oppløsning). Den andre prosessen involverer bruk av spesifikke programvarer for å lese og analysere den digitale filen (7).

Maskinvaren består av en WSI-skanner som i all hovedsak er et automatisert mikroskop, kontrollert av datateknologi. Det finnes et bredt utvalg av skannere med ulik funksjonalitet, alt fra enkle skannere til relativt komplekse skannere som skanner hundrevis av lysbilder på sekunder. Komplekse skannere har ofte en egen «oppbevaringsdel» bestående av flere brett eller karuseller, tilpasset lagring av vevsprøver som microarrays (7).

Generelt består skanneren av seks ulike komponenter; 1) et mikroskop med en linse; 2) en lyskilde (klart og eller fluoriserende; 3) en robot som kontrollerer glassplaten med lysbilde; 4) et eller flere digitale og spesialiserte kameraer med optiske sensorer; 5) en data/prosessor; 6) en programvare som manipulerer, opererer og viser det digitale lysbilde (23). Aperio-skanneren, er en av få skannere som kan brukes til både tørr-skanning og skanning med immersjonsolje. Den automatiserte robotdelen i Aperio skannere kan skifte glassplatene med en hastighet på 180 mm/s, og roboten ansees for å være nøkkelen til vellykkede skanninger med stor nøyaktighet. (7).

Den digitale dataen konverteres videre til et virtuelt lysbilde ved bruk av spesielt tilpassede programvarer. Programvaren er hovedsakelig enten tile-basert, eller linje-basert. Ved mikrobiologiske skanninger benyttes generelt den tile-baserte metoden. Da dette er den mest omfattende metoden, som potensielt gir digitale filer/bilder med høyest oppløsning (23). Tile-baserte skanninger består av et automatisert steg der det opprettholdes et stort antall kvadratiske miniatyrbilder (tiles). Tilene blir videre satt sammen i et mosaikk mønster, med et overlapp på 2%-5% per tile. Til slutt blir alle tilene auto-korrigert, noe som gjør at de passer perfekt sammen, slik at de kan «limes» og danne et helt sømløst bilde (7).



Figur 2.7: Viser skanning og konvertering av digital data til virtuelle lysbilder. Bildet til venstre viser skanning ved bruk av en tile-basert metode, der flere tiles er satt sammen i ett «perfekt mosaikk mønster». Bildet til venstre viser skanning ved bruk av linje-basert metoden. Bildet til høyre viser det ferdige bildet, der enten tiles, eller linjer er "limt" sammen, slik at de danner et sømløst bilde (26).

WSI-bilder har en massiv størrelse og en rådatafil kan ofte inneholde 1600 megapiksler, noe som krever rundt 4.8 GB minne. Filstørrelser av denne rangordenen er relativt uhåndterbare, og størrelsen minimeres derfor ved at det benyttes en kompresjons-dekompresjons-metode som filformat. Denne typen filformat konstruerer en slags «pyramide-fil» bestående av flere lag, satt sammen av små deler av det originale bilde (tiles). Pyramidestrukturen produserer et konservativt felt, noe som gjør det mulig å analysere bildefilene digitalt ved flere oppløsninger. Bildene kan altså mikroskoperes digitalt ved standard oppløsning (x4, x20, x40, x100, etc) (7).

Eksempler på vanlige brukte filformat som benyttes ved denne pyramide-strukturen er JPEG2000, TIFF og SVS. Innenfor medisin og mikrobiologi tenderer svs-filer mot å være de mest anvendte filene, dette fordi filformatet er implementert som standard i de fleste konvensjonelle medisinske skannere, som for eksempel Aperio (24). Svs-filene består av flere versjoner av det originale WSI-bilde, der disse legges oppå hverandre slik at de danner en pyramide-struktur (25).

Pyramiden består av et grunnbilde med full oppløsning, som er inndelt i tiles med en størrelsesorden på 240x240 piksler. Deretter følger ofte et miniatyrbilde med lav oppløsning, lagret som png eller jpg. Videre følger det et eller flere bilder som danner resten av pyramiden, der disse er komprimert og delt inn i tiles på en tilsvarende måte, som grunnbilde. En sjelden

gang vil det kunne være et lite etikkbilde på slutten av svs-filen, dette bilde har lav oppløsning og er alltid strippet (26) (27).

2.4.2 Implementering av maskinlæring i moderne patologi

Flere studier har vist at det finnes minimale forskjeller på kvaliteten mellom manuell diagnostikk utført ved bruk av høy oppløste og digitale filer, og manuell diagnostikk utført ved hjelp av tradisjonell mikroskopering. Selv om mye av arbeidet innen histopatologi er effektivisert, gjenstår det mest sannsynlig mye forskning og arbeid for å optimalisere og digitalisere prosessen. Den hellige gral innen moderne histopatologi ansees å være utviklingen av selvstendige hjelpemidler for patologen. Der bildene kan analyseres og diagnostiseres med høy nøyaktighet ved bruk av kunstig intelligens og maskinlæring (6).

2.5 Maskinlæring

Maskinlæring er en gren innenfor kunstig intelligens og termen maskinlæring omfatter flere ulike programmeringsteknikker, beregnet for analyser av store mengder data (28). Maskinlæring defineres som en vitenskap der man får en maskin til å forstå en oppgave, uten eksplisitt programmering (29). *Mer spesifikt; et dataprogram som kan lære fra en erfaring E, med hensyn til en oppgave O og en handling H - hvis programmet som utfører O, måles av H, og utvikles med erfaring E* (Tom Mitchell, 1997)

Teknologien er basert på prinsipper fra flere ulike fagfelt. Herunder biologi, matematikk, statistikk og datavitenskap (30). I motsetning til klassiske algoritmer, defineres maskinlærings-algoritmer i en modell med generelle retningslinjer (29). Den spesielle arkitekturen bak algoritmene og modellene, er et resultat av mange års forskning og utvikling (30). Arkitekturen gjør det mulig å koble og justere modellene til ønsket data. Videre beregnes manglende informasjon ved hjelp av statistiske teknikker. Hovedmålet er da at algoritmene i modellen, skal kunne ekstrahere nyttig informasjon selvstendig. Og deretter benytte ekstrahert informasjon til å lære spesifikke mønstre fra datamaterialet. Den tillærte kunnskapen benyttes videre til å produsere en output-verdi, i form av en score eller klasse. Med andre ord – modellen løser et ønsket problem ved at den «trenes» på et tilgjengelig datasett (29).

2.5.1 Historie, bakgrunn og utvikling av maskinlæring

1957: Den Amerikanske psykiateren, Fran Rosenblatt, presenterte «the perceptron» og perceptron-algoritmen. Perceptroner er kunstige nevroner som kan lære fra treningsdata, ved hjelp av perceptron-algoritmen. Perceptron-algoritmen kan kun benyttes til binær-klassifisering av lineært separable klasser. Da funksjonen er lineært avhengig av input, x – noe som underbygges ved at ligningen for perceptron-algoritmen tilsvarer ligningen for et hyperplan (formel 2 og 3)(31).

$$\sigma(W^T x + b) = \begin{cases} 1 hvis\ w^T\ x + b \geq 0, \\ 0\ hvis\ ikke. \end{cases} \quad (2)$$

$$w_i(t+1) = w_i(t) + \alpha\left(d_j - w_j(t)\right) x_{j,i} \quad (3)$$

1959: Hubel & Wiesel publiserte artikkelen, *«Receptive fields of single neurones in the cat`s striate cortex».* Hubel & Wiesel oppdaget at nervecellene i netthinnen hadde reseptive felt, som kan deles inn i separable induserende og inhiberende områder. Det visuelle systemet ble delt inn i tre ulike nivåer, og det ble vist at de reseptive feltene kan induseres eller inhiberes, ved hvert nivå, der dette avhenger av type visuell stimuli cellene utsettes for (32).

1960: Widrow and Hoff utviklet Adaline/Madaline. Hvor lineære perceptron-lag ble stablet sammen, slik at de dannet et perceptron-nettverk, bestående av flere lag (33).

1968: Huber og Wiesel videreførte studien gjort på kattene, til aper. Der de oppdaget at cellene i det visuelle systemet har en horisontal organisering som korresponderer til lagene i cortex. Cellene er segregert i en hierarkisk struktur. Der laveste rang/orden består av enkle celler som responderer til lysorientering, neste orden består av komplekse celler, som responderer til lysorientering og bevegelse. Cellene med høyest rang/orden ble kalt hyperkomplekse celler, og disse responderer til bevegelse med et endepunkt. Viser til artikkelen: «Receptive fields and functional architecture of monkey striate cortex» (34).

1986: Rumelhart fremstilte back propagation, en prinsipiell metode for å trene nevrale nettverk. Metoden ansees fremdeles for å være essensen i treningen av nevrale nettverk (35).

1998: LeCun utviklet et nevralt nettverk som ved hjelp av gradient-basert læring, og back propagation kunne gjenkjenne enkle zip-koder. Dette nettverket ble benyttet til kommersielt bruk i postverket (50)

2012: Krizhevsky, Sutskever, Hinton presenterte AlexNet som regnes for å være den moderne inkarnasjonen av konvolusjonelle nettverk. AlexNet utnyttet parallell datakraft fra GPU, samt store mengder tilgjengelig data (50).

2014: K. Simonyan og A. Zisserman presenterte det konvolusjonelle nettverket VGG16. Arkitekturen ble benyttet til å vinne konkurransen ILSVR(Imagement) i 2014 (36).

2015: ResNet introdusert av K. He, X. Zhang, S. Ren og J. Sun, og GooLeNet bidro til et paradigmeskifte innen konvolusjonelle nettverk (37).

2.5.2 Maskinlæringsmetoder relatert til klassifiseringsproblematikk

Klassifisering av bilder ved hjelp av maskinlæring kan ansees som et klassifiseringsproblem. Denne type problemstillinger løses generelt ved bruk av overvåketlæring. Da målet er å få en modell til å forutse en klasse (y), for korresponderende inputdata (x). For klassifisering av store datasett, særs der mye av dataen ikke inneholder emneknagger kan semi-overvåketlæring ofte være et bedre valg (38).

Overvåketlæring er per i dag den mest brukte metoden for å trene maskinlærings-modeller. Hovedprinsippet bak metoden er at datamaterialet defineres av markerte egenskaper. For eksempel defineres inputdata som variabelen (x), og output som variabelen (y). Algoritmene i modellen lærer en kartleggingsfunksjon; $Y = f(x)$ (38). Dette ved at egenskapene, (x) i datamaterialet sorteres manuelt etter bestemte emneknagger, (y) - før det benyttes som input i modellen (29). Intensjonen er at modellen selv skal finne mønstre som kan overføres

til analytiske prosesser. (29). Slik at modellen ved tilførsel av ny inputdata, i form av (x), selv vil kunne beregne og fastslå en verdi for variabelen (y) (38).

Ikke-overvåketlæring benyttes i tilfeller hvor man har et datasett, bestående av variabelen (x), uten en korresponderende (y)-verdi. I slike tilfeller eksisterer det ofte ikke et korrekt svar og modellen har heller ingen «lærer», derav navnet ikke-overvåketlæring. Algoritmene i modellen blir da overlatt til å finne interessante strukturer i datasettet på egenhånd. Generelt deles problemstillingene innenfor u-overvåketlæring inn i to hovedgrupper; «grupperinger» og «assosiasjoner». U-overvåket læring kan for eksempel benyttes til å finne skjulte strukturer i et datasett, oppdage interne grupperinger, eller beregne korrelasjoner internt i datasettet (38).

Semi-overvåketlæring er en blanding mellom to nevnte læringsmetodene. Metoden benyttes gjerne til problemstillinger hvor man har et ulikt forhold, mellom inputdata (x) og output (y). Eksempler på dette kan være store datasett, hvor bare noe av datasettet er markert – noe som ofte kan være tilfelle i store bildearkiver. Mange praktiske maskinlæringsoppgavene havner innenfor denne kategorien (38).

Klassifiseringsproblemer deles generelt inn i to hovedkategorier; binærklassifisering og multi-klassifisering (39). Det objektivet målet med klassifisering ved hjelp av maskinlæring, er å lage en modell som klarer å maksimerer utførelsen gjort på treningsdataen slik at generaliseringsevnen økes (40). Det finnes flere ulike klassifiseringsalgoritmer og modeller, og det kan være vanskelig å utpeke en bestemt algoritme/modell som viser signifikant bedre egenskaper enn andre. Dette fordi valg av algoritme og modell avhenger av flere faktorer, for eksempel hvilken type og hvor stort datasett man skal klassifisere, hvorvidt det gjelder et binært- eller multi-klassifiseringsproblem, og om klassene er lineært separable, eller ikke-lineær separable. Eksempler på type klassifiseringsalgoritmer/modeller er; Decision Tree Support vector machine (SVM) og «nevrale nettverk» (39).

2.5.3 Klassisk maskinlæring ved bruk av Support vector machine (SVM)

Support Vector Machine (SVM) er en lineær modell og en overvåket maskinlæringsmetode. Modellen benyttes fortrinnsvis til å løse lineære klassifisering- og regresjonsproblemer (40). SVM ble introdusert av Vapnik i 1995, og har siden høstet mye oppmerksomhet og interesse (41). SVM er i dag en av de mest kjente og brukte maskinlæringsteknikkene, og ansees som et kraftig verktøy for å løse klassiske klassifiseringsproblemer (41). Modellen er kjent for å ha gode generaliserings og optimaliseringsevner, samt at den har gode diskriminerende ferdigheter (40).

2.5.3.1 Klassifisering via hyperplan og hjelpevektorer

Formelt defineres SVM av et separabelt hyperplan (41). Hovedmålet bak klassifiseringsstrategien er å maksimere generaliseringsevnen til modellen. Dette ved å maksimalisere den eksisterende separasjonsmarginen, mellom grensesnittene i et høy-dimensjonalt rom, kalt «feature space». Herunder er «feature space» definert som en observert samling av egenskaper, relatert til objekter/hendelser/fenomener. Altså en samling av egenskaper som beskriver et gitt datasett, for eksempel; alder, høyde, kjønn og vekt (40). Separasjonsmarginen maksimeres ved å separere klassene i settet med en overflate. (40). Herunder er denne overflaten definert som et hyperplan i et to-dimensjonalt rom. Der nevnte hyperplan er definert av en linje, som deler et plan i to ulike deler. Da vil punktene for hver klasse, ideelt sett befinne seg på hver side av denne linja (41).

For lineært separable klasser finnes det et uendelig antall hyperplan, og den geometriske marginen kan optimaliseres ved å sette funksjonsmarginen, *kappai = 1.* Når et treningssett bestående av par, defineres som (formel 4), og den lineære klassifiseringen defineres som (formel 5). Der w definerer det optimale separasjons hyperplanet og b er bias (40).

$$X = [x_i, y_i]_{i=1}^{n}, \; x_i \in R^d \; og \; y_i \in (1, -1) \quad (4)$$

$$y_i = 1 \; (5)$$

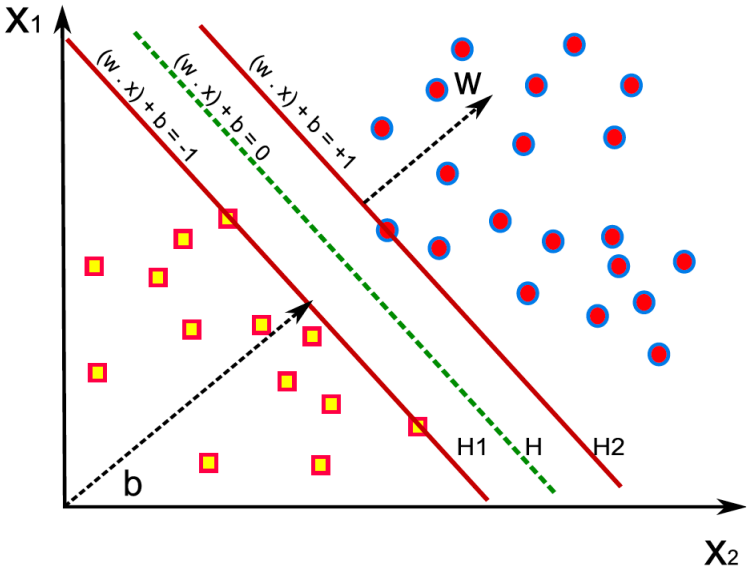
$$\left.\begin{aligned} w * x^+ + b = 1 \\ w * x^- + b = 1 \end{aligned}\right\} \; (5)$$

$$y_i(\langle w * x_i \rangle + b \geq 1 \; \forall \; i) \quad (5)$$

$$\gamma_i = \frac{1}{2}\left(\left\langle \frac{w}{\|w\|} * x^{+} \right\rangle - \left\langle \frac{w}{\|w\|} * w^{-} \right\rangle\right) \quad (5)$$

$$\gamma_i = \frac{1}{\|w\|} \quad (5)$$

For alle tilfeller der formel 2 er gyldig, kan den geometriske marginen optimaliseres. Dette ved å beregne to hjelpevektorer, (H_1 og H_2) og det optimale hyperplanet – via *«The Lagrangian multipler method».* Da avstanden mellom hjelpevektorene og det optimale planet tilsvarer den maksimale marginen. *«Lagrangian multipler method»* en kvadratisk løsningsmetode, der overflaten danner en polaroide med kun et globalt minimum (40). Løsningen for H_1 og H_2 tilsvarer de datapunktene fra hver klasse som ligger nærmest hverandre. Datapunktene som befinner seg under H_1, og/eller over H_2, kan fritt fjernes eller endres, uten at planene H_1 og H_2 krysses. Disse punktene påvirker derfor ikke generaliseringsevnen til modellen (41)



*Figur 2.8: Figuren visert et plott av det optimale hyperplanet for et binært klassifiseringsproblem, der hyperplanet er definert av, y = (w*x)+b=0. Hyperplanet beregnes ved hjelp av hjelpevektorene H_1 og H_2, der hjelpevektorene er definert av y = (w*x)+b = +1/-1*

2.5.3.2 Svakheter med SVM – relatert til ikke-lineære datasett

For problemstillinger der datasettet ikke er lineært separabelt, tenderer SVM-modellen mot å ha en dårlig generaliseringsevne. Generaliseringsevnen kan forbedres ved å innføre en kernel funksjon og dele læringsprosessen inn i to steg; 1. Ikke-lineær transformasjon, 2. SVM-modell som klassifiserer inputdata lineært i det høydimensjonale «feature space» (40):

Der den ikke lineære transformasjonen gjøres ved å transformere input-vektorer: $x \in R^n$ til vektorene: $\Phi(x)$ i et høydimensjonalt «feature space», når $\Phi(x)$ representerer kartleggingen $R^n \rightarrow R^f$ (formel 6 og 7) (40).

$$x \in R^n \rightarrow \phi(x) = [\phi_1(x)\phi_2(x) \dots \dots \phi_n(x)]^T \epsilon R^f \quad (6)$$

$$f(x) = \sum_{i=1}^{l} w_i \phi_i(x) + b \qquad (7)$$

Den ikke lineære transformasjonen er bare gyldig hvis formel 7 kan uttrykkes som en lineær kombinasjon av punktene i inputdata. Dette kan valideres ved å direkte bestemme prikkproduktene av $\Phi(x_i)*\Phi(x_i)$, som en funksjon av originalt datainput (formel 8) (40).

$$f(x) = \sum_{i=1}^{l} \alpha_i y_i \langle \phi(x_i) * \phi(x) \rangle + b \quad (8)$$

I denne sammenhengen defineres en kernel funksjon som formel 9, når $(x,z) \in X$. Kernel funksjonen må innfri Mecer teoremet, se artikkel for utdypende informasjon (40). Funksjonen må også oppfylle følgende kriterier (formel 10,11,12,13), når $x,y,z \in X$ og $\alpha \in R$ (40).

$$K(x,z) = \langle \phi(x) * \phi(z) \rangle \quad (9)$$

$$1.\ x * x = \begin{cases} 0, \text{når } x = 0, \\ > 0, \text{når } x \neq 0 \end{cases} \quad (10)$$

$$2.\ x * y = y * x. \qquad (11)$$

$$3.\ (\alpha x * y) = \alpha(x * y) \quad (12)$$

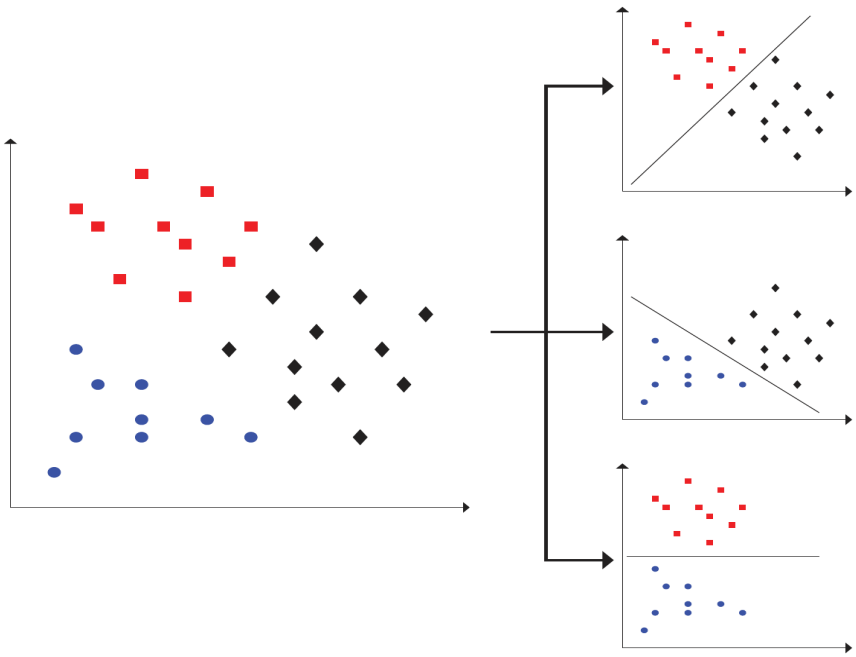
$$4.\ (z + x) * y = (z * y) + (x * y) \quad (13)$$

2.5.3.3 Svakheter ved SVM – relatert til datasett bestående av flere klasser

Klassifisering av datasett som består av flere enn to klasser kalles multi-klassifisering. Denne typen datasett ansees som ikke-lineært separable, da klassene kan defineres på flere ulike måter. Denne type analyser er derfor langt mer mer komplekse, sammenlignet med binærklassifisering (42). For å analysere multi-klasseproblemer ved bruk av SVM, må problemet i de fleste tilfeller dekomponeres til et enklere binært problem (43). Herunder gjøres dette fortrinnsvis via to ulike metoder; «one versus one» (OvO) og «one versus all» (OvA) (40).

OvO-metoden består av to ulike steg. I det første steget dekomponeres klassene ved at det konstrueres *k*(k-1)/2*, individuelle og binære klassifiseringsmodeller for k klasser. Det originale datasettet splittes da til flere binære sub-datasett (43). For eksempel i et datasett med k=3 klasser; svart, blå og rød. Lager algoritmen da, *k*(k-1)/2=3* binære modeller. De binære modellene trenes til å skille mellom klassene (C_i,C_j) ved hjelp av en konfidensgrad r_{ij} $\in$[0,1] i favør av C_i, samt en konfidensgrad i favør av C_j - som beregnes ved $r_{ji} = 1-r_{ij}$ (43)

I det andre steget forenes klassene, og den endelige klassen for inputvariabelen *X* beregnes. Dette gjøres normalt ved bruk av strategien «VOTE» (43). Der hver k*(k-1)/2 modell anvendes til et usett eksempel av input *X*. Hvis modellen bestemmer at *X* tilhører en bestemt klasse, adderes en stemme til denne klassen for gitt *X*-verdi. Til slutt summeres alle stemmene, og *X* plasseres i den klassen med flest stemmer. Ulempen med denne strategien er at det for enkelte tilfeller, der klassene har relativt store likheter og eventuelle overlapp, ofte er noe vanskelig å plassere *X* innenfor en bestemt klasse. Dette fordi inputvariabler av *X* i nevnte tilfeller kan oppnå et likt antall stemmer (40).



Figur 2.9: Viser klassifisering av tre ulike klasser ved bruk av "one vs one"-metoden. Der multi-problemet degraderes til tre ulike binære-modeller; 1. rød vs svart, 2. blå vs svart, 3. rød vs blå. Klassene vil så forenes ved hjelp av «VOTE»-metoden (43).

OvA-metoden dekomponerer multi-klassifiseringen til et lignende binært problem forklart i OvO-metoden. Det konstrueres k-binære modeller, for k-klasser. Hovedforskjellen er at hver av de k-binære modellene, trenes med en bestemt klasse mot de resterende klassene (40). For eksempel vil sub-datasettene for klassene grønn, blå og rød utformes på følgende måte; 1. grønn vs blå og rød, 2. blå vs grønn og rød, 3. rød vs blå og grønn (42). Videre bestemmes klassen for inputvariabelen *X* ved hjelp av en kernel funksjon (definert i 2.5.3.2), samt en bestemmelsesfunksjon (40).

2.5.4 Tradisjonelle nevrale nettverk

Kunstige nevrale nettverk er datamodeller, bestående av flere noder som kobles sammen i ulike lag, slik at de tilsammen danner et nettverk med en dybdekomponent. Den spesielle arkitekturen er inspirert av biologiske nervesystemer fra naturen. Fortrinnsvis hvordan nevronene i hjernen er strukturert og koblet sammen. Da den grunnleggende beregningsenheten i hjernen og nervesystemet, ansees for å være nevroner (44).

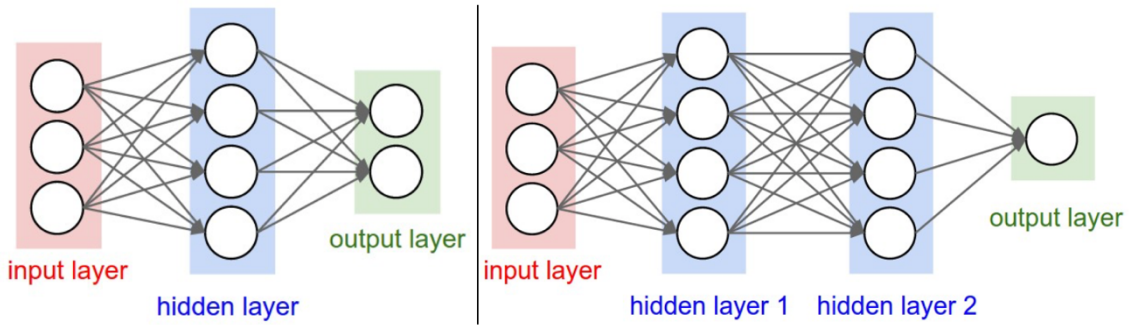
2.5.4.1 Arkitekturen i nevrale nettverk – forklart via biologiske systemer

For å forklare hvordan kunstige nettverk fungerer kan man ta utgangspunkt i det biologiske nervesystemet. Nervesystemet til et menneske består av omlag 86 billioner nevroner, som etter behov kobles sammen via 10^14-10^15 ulike synapser. I synapsen danner nervecellene koblinger med dendrittene til andre nerveceller. Nevronene i hjernen mottar et innkommende signal fra dendrittene og signalet stimulerer nevronene til å produsere et utkommende signal som sendes ut langs aksonet til andre nevroner (56).

Grunnprinsippet for signaloverføring er dermed følgende; dendrittene leder signaler inn til nervecellen, hvis den totale summen av signaler overstiger en gitt terskel: «Fyrer nervecellen av»/aktiveres og sender ut et elektrisk signal langs aksonet, som videre overføres i synapsen til andre nerveceller via deres dendritter. Det ser ut til at synapsestyrken kan forbedres ved læring, samt at økende synapsestyrke korrelerer med økende påvirkningsevne hos nevronene (56).

I nevrale nettverk kobles nevronene sammen via trenbare vekter slik at de danner en asyklisk graf. Her viser vektene tilkoblingen mellom et nevron *M*, til et nevron *N*. koblingsvektene beregnes ved å finne skalarproduktet av vektorene i nettverket. Bias er antallet individuelle noder i det totalet nettverket (minus nodene i innkommende lag). Intensjonen med vektene er at de skal simulerer bindingene(e) til synapsen (44). Ved hjelp av en aktiveringsfunksjon stimuleres nevronene til å sende ut signaler til korresponderende nevroner. Det vil si at utgående data fra noen nevroner konverteres, til innkommende data for andre nevroner (44).

Sykliske koblinger kan ikke benyttes, da dette fører til uendelige løkker i den fremre delen av nettverket. Nevronene arrangeres i en fastsatt struktur, bestående av flere ulike fullstendige-lag. Nevroner innenfor samme lag har ingen direkte kobling. Istedenfor kobles nodene parvis sammen med noder fra parallelle lag, slik at de danner trenbare vekter og bias (44).



Figur 2.1: Venstre side viser et nettverk á tre lag; input, skjult og output. Høyre side viser et nettverk á fire lag; input, to skjulte og output. Rundingene i hvert av lagene simulerer nevronene, og pilene viser koblingene mellom dem (vekter og bias) (44).

2.5.4.2 Generelle metoder og effekter relatert til trening av nevrale nettverk

Hovedmålet med en maskinlæringsmodell er at den skal tilfredsstille prinsippene i definisjonen av maskinlæring, nevnt innledningsvis i kapitel 2.5. For at nevrale nettverk skal kunne lære må modellen kunne beregne treningsparameterne (vekter og bias). Nevnte parametere beregnes under trening av modellen ved hjelp av en loss-funksjon (taps-funksjon), samt metodene forward propagation (fremdrift), og back propagation (tilbakeføring) (35, 45).

Treningen deles inn i flere treningsepoker, og for hver epoke blir modellen vist et bestemt datasett (treningssett) med merkede etikker (klassene). Deretter testes den aktuelle klassifiseringsevnen til modellen på et annet datasett (treningssettet), og feilberegningene beregnes ved hjelp av loss-funksjonen og back propagation (45).

Forward propagation (fremdrift) benyttes for å føre datainput gjennom nettverket. Dette gjøres for at nettverket skal kunne generere output som sendes videre til neste lag. Innføring eller flyt av datainput i motsatt retning vil føre til sykliske koblinger, slik at output ikke kan genereres. Under forward propagation beregnes og lagres verdiene av mellomprodukter til beregningsgrafen, definert av nettverket (46).

Back propagation (tilbakeføring) benyttes til å oppdatere treningsparameterne i nettverket mellom hver treningsepoke, der loss-funksjonen gjør modellen i stand til å evaluere og redusere feilberegninger i nettverket (45). Det totalet tapet fra hver treningsepoke integreres tilbake i modellen via back propagation, noe som gjør at modellen kan detektere hvilke

nevroner som er ansvarlig for mest tap, internt i hvert lag. Videre justeres så treningsparameterne slik at de «dårligste» nevronene gis lavere treningsparametere, samtidig som de «beste» nevronene gis høyere treningsparametere (35). I praksis medfører dette at treningsparameterne i nettverket kan finstilles, med tanke på feilraten fra tidligere treningsepoker. (35).

Treningen og antall treningsepoker er essensielt for at nevrale nettverk skal kunne generaliseres til annen data. Prosessen er relativt kompleks og i enkelte tilfeller kan modellen trenes «for mye», noe som gjør at den tilpasses for godt til datamaterialet og får dårlig generaliseringsevne, altså at den ikke kan interferere med annen data. Denne effekten kalles overfitting (for mye tilpasset), og dette oppstår generelt når modellen er for kompleks. Det vil si at den inneholder for mange variabler og egenskaper, sammenlignet med antall observasjoner og størrelsen på datamaterialet (47).

Modeller som tenderer mot overfitting vil vise stor nøyaktighet for treningsdata, men vil mest sannsynlig vise relativt lav nøyaktighet for testsettet og særs ny data. Hovedårsaken til effekten antas å være at modellen tilegner seg kunnskap relatert til spesifikt støy i et bestemt treningssettet, fremfor å lage relasjoner mellom variablene i datamaterialet. Da den spesifikke støyen mest sannsynlig ikke vil være en del av et nytt datasett, vil modellen mest sannsynlig ikke kunne generaliseres, en slik modell har som regel ingen mening eller verdi (47).

Underfitting (under-tilpasset) er annen effekt som kan oppstå under trening hvis modellen er for dårlig trent. Den vil da ikke være tilpasset datasettet den er trent på, og modellen vil ikke klarer å fange opp tendenser og korrelasjoner mellom egenskaper i datamaterialet. Underfitting medfører også at modellen ikke kan generaliseres til annen data. Årsaken til effekten er som regel at modellen er for simpel, altså inneholder for få uavhengige variabler. Andre årsaker til underfitting er bruk av feil type modell for feil type data, for eksempel bruk av en lineær modell til å løse et ikke-lineært datasett (47).

2.5.4.3 Tradisjonelle nettverk og bildeanalyser

Fordi tradisjonelle nevrale nettverk består av fullstendig sammenkoblede lag er ikke de ikke spesielt godt egnet til bildeanalyser (48). Da fullstendig sammenkoblede lag medfører at inputdata strekkes ut til en stor vektor, som videre multipliseres med nevronene i laget. For eksempel vil et lite inputbilde med en størrelse på 32x32x3, strekkes ut til vektoren 3072x1. Hvis nettverket består av for eksempel 100 nevroner i det første laget, vil nettverket raskt inneholde flere millioner parametere, særs for inputdata av en hvis størrelse. Videre vil et stort antall parametere kreve større treningssett, lang treningstid og tenderer ofte mot «overfitting (44).

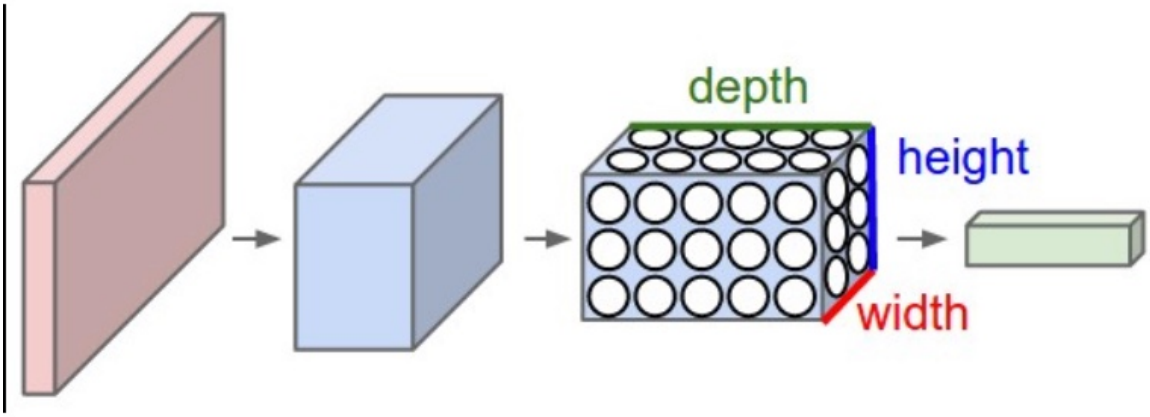
Hovedproblemet med tradisjonelle nevrale nettverk relatert til bildebehandling, ser likevel ut til å være en manglende innebygget invarians. Der invariansen er relatert til skalering og geometrisk forvrenging av input-data. Manglende invarians gjør systemet/nettverket følsomt for svært små endringer. Eksempler på endringer er farge, rotering, sentrering av objektet og naturlige variasjoner innenfor en gitt klasse (48).

2.5.5 Konvolusjonelle nevrale nettverk (ConvNets)

Konvolusjonelle nettverk (ConvNets) er en spesiell type nevrale nettverk, som hovedsakelig benyttes til ulike former for bildeanalyser (49). På samme måte som tradisjonelle nevrale nettverk, består ConvNets av flere lag av sammenkoblede nevroner, med trenbare parametere (vekter og bias) (44). Hovedforskjellen fra tradisjonelle nettverk er de karakteristiske egenskaper i arkitekturen, der disse egenskapene bidrar til at det opprettholdes invarians - relatert til bildetransformering av inputdata. Invariansen er nøkkelen til at ConvNets kan gjenkjenne mønstre, på tross av geometriske forvrenginger internt i bildene (50)

Egenskapene som opprettholder invarians er inspirert av hierarkiske strukturer som finnes i synsbarken hos mamaler, oppdaget av Hubert & Wiesel (51). Den hierarkiske strukturen simuleres ved bruk av reseptive felt, konvolusjonelle lag (Convd-lag), delte parametere og nedskalering via «pooling». Arkitekturen medfører at nettverket danner en pyramide med økende dybde og minkende romlig størrelse. Nevroner i de tidlige lagene simulerer enkle

celler, og «ser etter» enkle egenskaper, for eksempel kanter. De midtre lagene simulerer komplekse celler, og «ser etter» egenskaper som hjørner, klatter og farger. De siste lagene simulerer hyperkomplekse celler og nevronene «ser etter» mer komplekse strukturer (37).



Figur 2.11: Viser pyramidestrukturen i ConvNets. Inputbilde føres gjennom flere convd-lag med et ulikt antall filter, mellom convd-lagene nedskaleres inputbilde via pooling, den to-dimensjonale størrelsen av input vil da minke, samtidig som dybden økes.

Navnet ConvNets stammer fra konvolusjons-teoremet, definert i formel 14. Konvolusjon kan forklares som en matematisk operasjon, der to ulike funksjoner «foldes sammen», slik at de danner en tredje funksjon. Dette ved at man lar en funksjon gli over den andre, samtidig som de multipliseres og adderes med hverandre. Konvolusjonen i ConvNets er en noe løsere definisjon av konvolusjons-teoremet, og inputdata «foldes» sammen med en flippet versjon av kjernefilter-matrisen i convd-laget (37).

$$f[x,y] * g[x,y] = \sum_{n1=-\infty}^{\infty} \sum_{n2=-\infty}^{\infty} f[n_1,n_2] * g[x-n_1, y-n_2] \quad (14)$$

2.5.5.1 Hyperparametere - relatert til arkitekturen

Hyperparameterne i ConvNets kan defineres som alle variabler relatert til arkitekturen i nettverket, og/eller som alle variabler relatert til treningsalgoritmen. Disse bestemmes før

trening, men verdiene kan finstilles/endres under trening for å optimalisere nettverket (52). Hyperparameterne påvirker hvorvidt og hvor bra nettverket evner å ekstrahere ut egenskaper fra datainput. De kan derfor relateres direkte til nøyaktigheten og generaliseringsevnen til nettverket (53).

De viktigste hyperparameterne relatert til arkitekturen er følgende:

- Antall convd-lag, størrelsen på de reseptive feltene (kjernefilteret) og antall filter i hvert lag.
- Initiering av vektene
- Hvilken type nedskalerings-lag som benyttes.
- Aktiveringsfunksjonen
- Pre-prosessering av dataen
- Regulering av nettverket ved bruk av bach-normalisering og/eller dropout
- Dataaugmentasjon

(52-54).

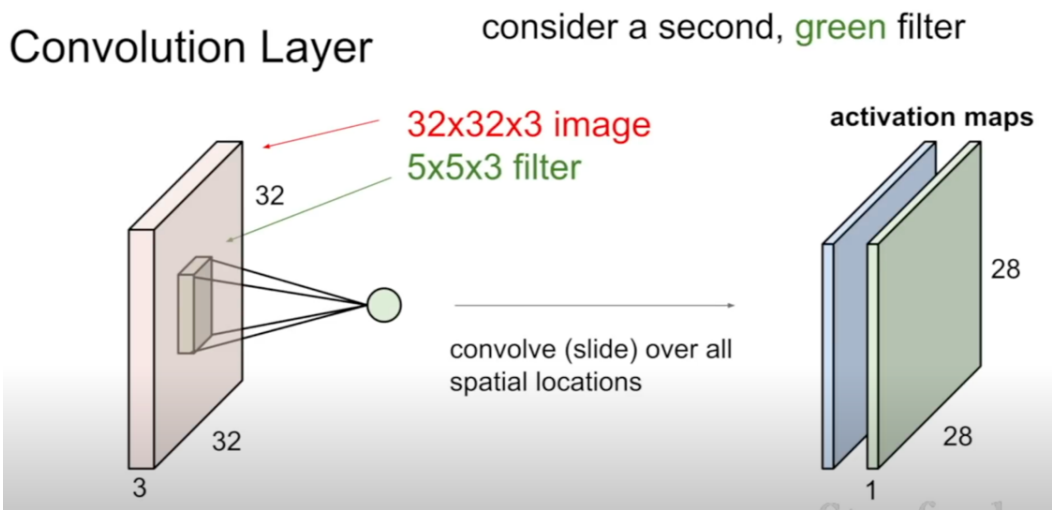
2.5.5.2 Konvolusjonelle-lag, reseptive felt og delte parametere

Konvolusjonelle-lag (convd-lag) består av flere kjernefiltre, også kjent som kernel. Kjernefilteret er alltid en matrise som definerer størrelsen på det reseptive feltet, samt dimensjonen i convd-laget. Størrelsen på kjernefilteret varierer normalt mellom (3x3), (5x5) og (7x7). Filterstørrelsen må korrespondere til størrelsen på input-bilde og stride-størrelsen, samt inneholde oddetall. Dette for å opprettholde nødvendig symmetri i outputdata. I denne sammenhengen er stride relatert til hvor mye kjernefilteret forflyttes, under hver «folding». Antall filter internt i hvert lag korrelerer med antall nevroner i laget, og kan velges fritt. Dog kan generelt bare partall benyttes, for eksempel; 32, 64, 128, 512 (37).

Hvert av kjernefiltrene i laget presenteres for samme input-data, men de «ser etter» ulike egenskaper og mønstre. Der dette kartlegges i et aktiveringskart - relatert til outputdata (37). Aktiveringskartet beregnes ved at kjernefiltermatrisen og pikslene i input-bilde «foldes» sammen. Der man lar hvert kjernefilter «gli» over input-bilde, samtidig som prikkproduktet

mellom filtermatrisen og pikslene beregnes for hvert punkt i bilde. Hvert prikkprodukt vil da representere et punkt på aktiveringskartet (37), og antall filter vil indikere dybde-dimensjonen til aktiveringskartet. Det totalet aktiveringskartet består derfor av flere sammensatte 2D-segmenter med et ulikt sett av egenskaper (50).

Filtret som benyttes i hvert enkelt 2D-plan inneholder trenbare vekter, der disse deles på tvers av alle filter i planet (50). I praksis medfører parameterdelingen til at hvert sett av egenskaper (hvert segment i aktiveringskartet) presentert i output-volumet, reflekterer parallelle transformasjoner internt i et input-bilde. En direkte konsekvens av dette er at treningstiden reduseres (50).



Figur 2.12: Folding av inputbilde ved bruk av et kjernefilter med størrelsen (5x5x3), der (5x5) representer arealet av selve filtermatrisen, og -3 representerer antall farge kanaler (RGB) – relatert til dybden i filteret. Det produseres så ulike aktivitetskart bestående av 2D-segemter -tilsvarende antall filter i convd-laget (50).

2.5.5.3 Initiering av vektene i ConvNets før trening

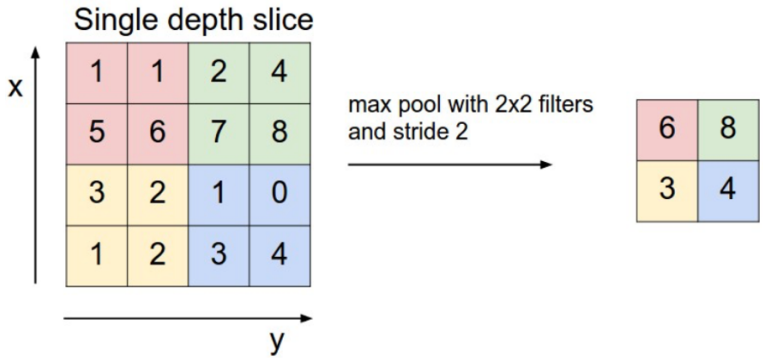
Vektene i nevrale nettverk som aldri har vært trent tidligere må gis en gitt startverdi. Deretter vil vektene oppdateres under trening ved hjelp av gradienten og back propagation. Hvis vektene ikke initieres med en gitt verdi, vil de starte å trene med verdien null. Dette medfører at alle datainput gir samme output-verdi, og alle vektene i nettverket vil da få samme verdi og

egentlig utføre samme oppgave. I praksis medfører dette at gradientoppdateringen ikke vil fungere, og nettverket vil ikke kunne trenes (54).

2.5.5.4 Nedskalering gjennom nettverket via «pooling» og funksjonell bruk av «padding»

«Pooling» er en funksjon som benyttes for å redusere den to-dimensjonale størrelsen på matrisen med representert data, samt redusere antall treningsparametere. Dette gjøres ved at en convd-matrise transformeres til en langt mindre matrise (pool) (44). Generelt adderes det pooling-lag suksessivt mellom de ulike convd-lagene. Herunder kan disse være av typen max-pooling, normal-pooling eller global-pooling. Max-pooling tenderer mot å være et klart førstevalg. Da denne teknikken skal bidra til å redusere overfitting i nettverket, samt sørge for at nettverket til en hver tid, prosesserer de mest aktive pikslene (egenskapene) (44).

Max-pooling har en fiksert stride lik 2, uavhengig om inputbilde er oddetall eller partall. Filterstørrelsen er derimot avhengig av om inputbilde består av partall eller oddetall. For partalls-bilder settes filterstørrelsen nesten alltid til en matrise lik (2x2). En filterstørrelse bestående av partall vil dog ignorere den siste matrisekolonnen i oddetallsbilder, noe som kan medføre tap av viktig informasjon fra aktivitetskartet. For inputbilder bestående av oddetall benyttes derfor filterstørrelsen (3x3) (55).



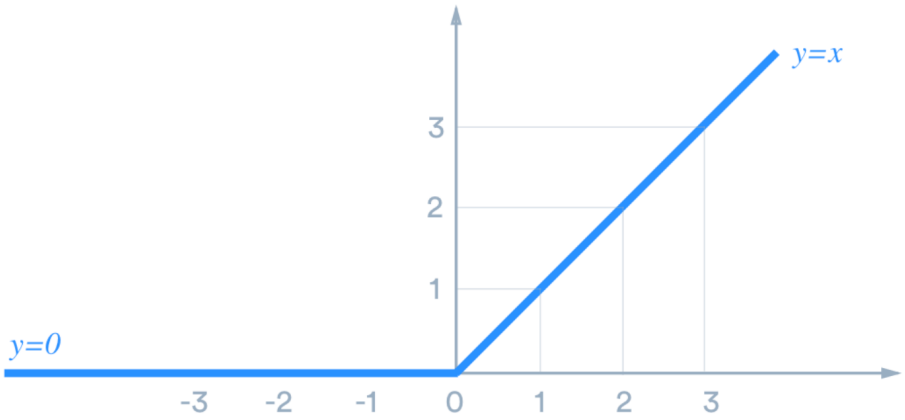
Figur 2.13: Illustrasjon av Max-pooling av et aktivitetskar, presentert av en partalls-matrise (55). Matrisen deles inn i sub-matriser (pools), og den høyeste verdien fra hver sub-matrise overføres til et en ny og nedskalert matrise, som da vil representere aktivitetskartet (56).

Under konvolusjonen/foldingen vil kjernefilteret sentrere seg rundt de midterste pikslene av filteret. Dette medfører at pikslene i hjørnene «faller vekk» og bildet nedskaleres, noe som

kan føre til tap av verdifull informasjon. Dette kan løses ved å addere «padding» til konvolusjonen. De mest vanlige padding-metodene er «zero» eller «same», og «valid». Zero- og same-padding indikerer at inputbilde polstres med null-ere, der disse innrammer bildet. Valid-padding betyr egentlig at det ikke utføres padding, og denne type padding benyttes i tilfeller der det kan være nyttig å nedskalere input-bilde ved hjelp av filterstørrelsen, fremfor nedskalering ved bruk av pooling (37).

2.5.5.5 Aktiveringsfunksjonen i ConvNets

Aktiveringslag inneholder en aktiveringsfunksjon, også kalt overføringsfunksjon. Denne funksjonen benyttes til å stimulere nevronene i nettverket, slik at de sender ut en output-verdi. Det finnes flere ulike aktiveringsfunksjoner, og disse kan hovedsakelig deles inn i to hovedgrupper; lineære- og ikke-lineære aktiveringsfunksjoner. Den mest anvendte aktiveringsfunksjonen i ConvNets ansees å være «ReLU» (56). ReLU er en transistor funksjon som øker ikke-lineariteten i nettverket. Dette sørger for at nettverket behandler bildeanalysen som et ikke-lineært problem. Funksjonen har en lineær identitet for alle positive verdier, og *x=0* for alle negative verdier. Matematisk er funksjonen definert som; *y=max(0,x)* (57).



Figur 2.14: ReLU-funksjonen vist som y=max(0,x) - grafen viser definisjonsområdet til funksjonen (57).

Fordeler med ReLU-funksjonen er at den involverer enkle beregninger, noe som gjør modellen både rask og lett å trene. Den konvergerer fort, og har begrenset aktivering. Der den

begrensede aktivering er et resultat av at funksjonen vil være null, for alle negative input (57). I denne sammenhengen er den begrensede aktiveringen relatert til modellen, og ikke manglende data (noe som mest sannsynlig hadde gitt problemer). Begrenset aktivitet i denne sammenhengen antas å være en fordel, da målet med nevrale nettverk er å simulere biologiske systemer (57). Intensjonen med ReLU-aktiveringen er derfor at modellen skal oppnå bedre generaliseringsevner, samt hindre overfitting og støy. For eksempel vil det være gunstig at nevronene i en modell, trent på katter. Ikke aktiveres hvis inputbilde er av en bygning, eller en bil (57).

Svakheter med ReLU-funksjonen er effektene «dead ReLU» og dying ReLU». «Dead ReLU`s er nevroner i nettverket som aldri aktiveres eller oppdateres under trening, disse regnes derfor som døde nevroner. Dette kan oppstå i tilfeller hvor deler dataskyen (treningsdataen) som benyttes, befinner seg i et negativt området. Da definisjonsområdet til funksjonen fører til at gradienten «drepes» for negative verdier, vil nevronene eksponert for negative dataverdier hverken aktiveres eller oppdateres. Effekten oppstår generelt som følge av dårlig initiering av vektene (54).
«Dying ReLU» er en effekt der nevronene dør under trening, noe som kan oppstå når læringsraten i nettverket er satt for høyt. Da en høy læringsrate kan medføre at gradientflyten gjennom nettverket er for høy. En høy gradientflyt kan medføre at vektene «hopper» for mye rundt under oppdateringen, noe som gjør at nevronene sparkes ut av posisjon og ut fra definisjonsområdet for treningsdataen (54).

For å løse problemene med ReLU-funksjonen er det utviklet flere ulike «datter-funksjoner» av ReLU-funksjonen. Eksempler på dette er «datter-funksjonene»; Leaky-ReLU, Parametric Rectifier (PReLU) og Exponential Linear Units (ELU) (54). Nevnte funksjoner har alle likheter med ReLU og så og si de samme fordelene, samt at de alle er definert for negative verdier. ELU-funksjonen skal i tillegg kunne bidra til at nettverket blir mer robust mot støy (54).

2.5.5.6 Pre-prosessering av datamaterialet i ConvNets

Datasettet som skal benyttes i modellen bør pre-prosesseres. Generell standard for pre-prosessering av bilde data, er å null-sentrere dataen ved at gjennomsnittet subtraheres fra

hvert datapunkt (54). Null-sentrering hindrer at datasettet ikke opptrer som enten bare positivt eller negativt. Da kun positive datainput vil medføre at gradienten også kun vil være positiv, noe som kan medføre problemer med konvergensen til gradienten. Siden gradienten kalkuleres via back propagation vil problemer i konvergensområde medføre vansker med oppdatering av vektene i nettverket (58).

Normalisering av datamaterialet gjøres for å sikre at egenskapene i datainput ligger i samme området, slik at de bidrar og påvirker like mye. For datasett bestående av bilder utføres generelt ikke normaliseringen. Dette fordi pikselverdiene generelt ligger innenfor et relativt lignende verdiområde, sammenlignet med datasett fra andre maskinlæringsproblemer (54).

2.5.5.7 Nettverksregulering i ConvNets ved bruk av Batch-normalisering

Batch-normalisering er en teknikk som kan integreres i det nevrale nettverket. Hovedmålet er å øke treningshastigheten, der dette gjøres ved å senke det interne kovariansskifte. Herunder er det interne kovarians skifte definert som endringer av nevronaktivitet, relatert til endringer av treningsparametere under trening (59). Dette kan forklares ved å ta utgangspunkt i et nettverk som trenes for å kunne gjenkjenne katter. Der naturlige variasjoner innen for arten, som for eksempel farge og lengde på pelsen, kan medføre små endringer i treningsparameterne. I praksis kan det se ut til at batch-normalisering sørger for, at ingen de aktive nevronene aktiveres for mye, eller for lite. Og dette skal bidra til å stabilisere nettverket, øke treningsraten, og eliminere behovet for et eget dropout-lag. (60). Dette underbygges av studier, som har vist at bruk av Batch-normalisering i modellen kan gi samme nøyaktighet, med 14 færre trenings-steg (59).

Batch-normalisering integreres i modellen ved at man legger til et eget lag for dette, mellom et convd-lag og korresponderende aktiveringslag. Algoritmen (figur 17) normaliserer utkommende output, ved å trekke fra batch-gjennomsnittet og dividere med batch-standarden. I tilfeller der degradering av normaliseringen er gunstig for å minimerer loss-funksjonen, vil normaliseringen degraderes av en stokastisk gradient. For å forhindre denne degraderingen adderer batch-normalisering konsekvent to ekstra trenbare parametere til hvert lag. Der dette gjøres ved å multiplisere output med en parameter for standardavvik

(gamma) og addere en parameter for gjennomsnittet (beta). Den stokastiske gradienten vil da degradere normaliseringen, ved å endre på parameterne gamma og beta, fremfor alle vektparameterne (60).

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

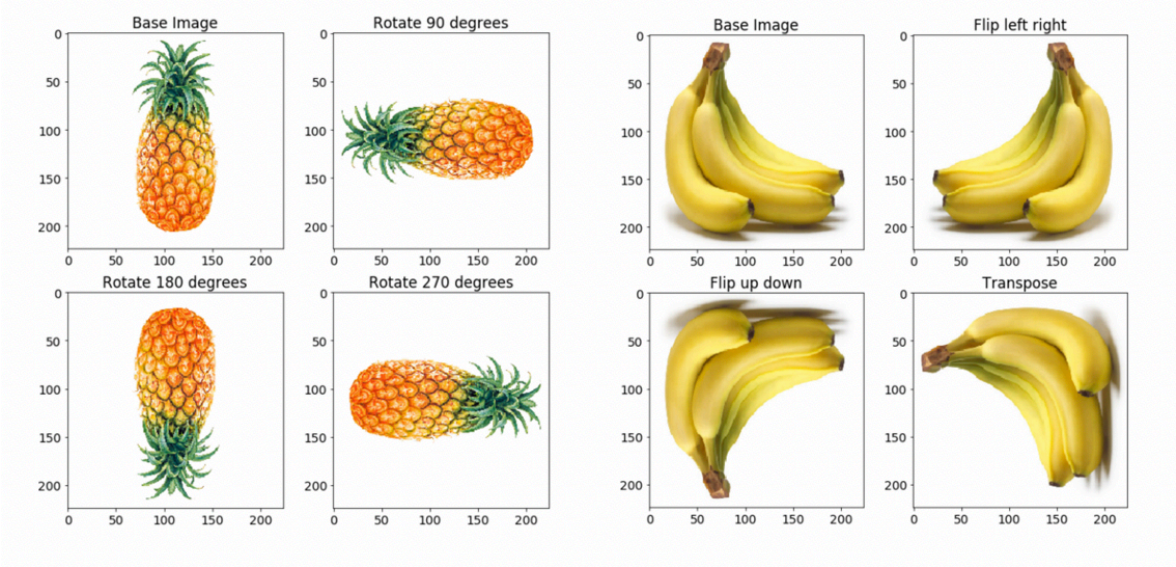
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Figur 2.15: Illustrerer oppbygningen av algoritmen som benyttes i batch-normalisering. (60).

2.5.5.8 Data augmentasjon for å øke datamengden og invariansen i nettverket

Augmentasjon av data (bilder) gjøres ved å legge til roterte, transformerte og/eller flippede versjoner av de originale bildene (61). Data augmentasjon kan implementeres i modellen ved å programmere egne koder, eller ved innebygde pakker som finnes i ulike rammeverk (61).
Data augmentasjon ansees for å være en viktig hyperparameter. Da nøyaktigheten i nevrale nettverk generelt vil øke med økende størrelse på datasettet. Dette fordi nevrale nettverk inneholder mange tusen parametere som krever et proporsjonalt datasett. I tillegg antas det at bruk av data augmentasjon bidrar til å opprettholde invariansen, noe som gjør nettverket mer robust for støy (61).



Figur 2.16: Viser bruk av data augmentasjon - Venstre side viser rotering av original bilde, slik at det dannes tre ekstra versjoner. Høyre side viser flipping av originalbilde, slik at det dannes tre ekstra bilder (61).

2.5.5.9 Klassifiseringsblokk og fullstendig sammenkoblede lag

Som nevnt tidligere danner arkitekturen i ConvNets en pyramide struktur (37). Toppen av pyramiden består av en «klassifiseringsblokk» med integrerte fullstendig sammenkoblede lag og et klassifiseringslag som forutser verdien av endelig output. Det første laget er ofte av typen «Flatten» og dette laget transformerer det nedskalerte aktiveringskartet (representert av en matrise) om til en enkel kolonne. Deretter følger ett eller flere fullstendig sammenkoblede lag av av typen «Dense». Til slutt adderes det siste klassifiseringslaget som beregner den endelige klassen til output *X.* Antall noder i klassifiseringslaget korrelerer med antall klasser i datasettet, og nodene stimuleres ved å benyttes en klassifiseringsfunksjon, her er det vanlig å benytte Softmax for multi-klassifisering (62).

2.5.5.11 Svakheter med ConvNets

ConvNets er som nevnt inspirert av det visuelle systemet som finnes hos mennesker, der den spesiell arkitekturen skal bidra til at det opprettholdes invarians gjennom hele nettverket (48). Der det hovedsakelig er bruken av konvolusjonelle- og pooling-lag gjennom hele nettverket, som ansees for å være nøkkelen til at invariansen opprettholdes. Det har også vært antatt at bruk av data augmentasjon under trening, er en viktig nøkkelfaktor for gode generaliseringsevner i nettverket. ConvNets har med dette vært revolusjonerende innen data visualisering, og regnes som kanskje den største suksessen innenfor objekt gjenkjenning (63).

I mange år har det derfor vært en utbredt antagelse om at convNets opprettholder invariansen, nylige studier viser derimot at dette ikke alltid er tilfellet (63).

Det kan se ut til at ConvNets tenderer mot å ha dårlige generaliseringsevner, samt lav robusthet mot støy (64). Der hovedårsaken til dette kan være at de fleste ConvNets-arkitekturer ikke filtrerer de ulike komponentene i aktivitetskartet før det nedskaleres. I tillegg kan det tenkes at nedskaleringen ved bruk polling, eller konvolusjon/folding med stride uten padding, kan påvirke nettverket negativt (64). Dette fordi denne type nedskalering ikke overholder «The classical sampling theorem», noe som kan medføre at det oppstår foldingsfeil (for utdypende informasjon om «the classical theorem» se kilde) (65). Herunder er foldingsfeil en effekt som nærmest gjør det umulig å skille mellom ulike signaler fra lav- og høyfrekvente komponenter (64).

For å forhindre foldingsfeil, samt for å øke støy-robustheten er det gjort forsøk på å implementere ulike typer filtrering. Eksempler på filtrering er Gaussian-, mean-og median-filtrering i en støydempende blokk (64). I tillegg er det forsøkt å øke data-augmentasjon. Dette har så langt ikke vist spesielt gode resultater. Det antas at data-augmentasjon ikke bidrar til å øke generaliseringsevne, da nettverket kun lærer å være invariant for transformasjoner av bilder som er veldig like, eller typiske bildene i treningssettet (63). Videre antas det at støy-demping ved bruk av ulike typer filtrering ikke fungerer spesielt bra, dette fordi støydempingen utføres i hele frekvensområdet. Her er det ønskelig at filtreringen kun gjøres i det høy-frekvente område, da det fortrinnsvis er støy fra komponenter i det høy-frekvente område som overføres og akkumulerer i aktivitetskartet. I tillegg vil støydemping i hele området kunne degradere den grunnleggende objekt-strukturen hos komponentene i det lav-frekvente området. (64).

2.6 Rammeverk og programvarer

2.6.1 Tensorflow – et rammeverk designet for maskinlæring

TensorFlow er et forståelig og fleksibelt rammeverk for maskinlæring, utviklet av forskere og ingeniører ved Google Brain (66). TensorFlow er en åpenkilde plattform som inneholder flere

ulike verktøy, biblioteker og kilder. Dette gjør det mulig å presse grensene for «*the-state-of-the-art»*. Det er i dag flere store selskaper som har inkorporert bruk av TensorFlow i sine selskaper. Spesifikke eksempler på dette er; Google, Coca-Cola, Intel, Airbnb og Ge Healthcare (67).

Rammeverket inneholder for eksempel Keras API, en applikasjon som gjør det relativt enkelt, både å bygge og trene modeller, relatert til nevrale nettverk. Herunder står API for «Application Programming Interface», et programmeringsgrensesnitt utviklet for datautveksling mellom ulike applikasjoner (67). API gjør det mulig å addere funksjoner eller tjenester til en applikasjon eller program, uten at brukeren må sette seg inn i de primære kildekodene for applikasjonen (68). TensorFlow benytter Python som kodespråk til front-end API, slik at man kan bygge inn applikasjoner i rammeverket, og applikasjonene uttrykkes i C++ (66).

2.6.1.1 Keras API – en applikasjon for TensorFlow

Applikasjonen Keras er spesifikt utviklet for normale mennesker, og ikke maskiner. Keras minimerer antallet bruker steg og har en relativt logisk oppbygging (69). Noe som medfører at maskinlæringsmodeller kan kodes ved bruk av relativt lite koding (66). Keras tilbyr tydelige brukermeldinger hvis feil skulle oppstå, og kodene skal være relativt enkle å «debugge» (69) (skanne for feil). Det finnes også et bredt utvalg av dokumentasjon og utviklingsguider. Det skal derfor være mulig å benytte og forstå applikasjonen ved bruk av «sunn fornuft» (69).

2.6.2 Python – et kommersielt programmeringsspråk

Python er et objektivt programmeringsspråk som inneholder flere ulike applikasjoner, biblioteker og pakker. Herunder er noen kjente; Pandas, SciPy, IPython og numpy. Python ble startet opp av Guido Van Rossum i 1989, og har siden vært under stadig utvikling, og det finnes i dag flere ulike versjoner av programmet. Den første kommersielle versjonen het Python 1.4 og ble sluppet i 1996. Den siste versjonen, Python 3.9.0 ble sluppet i oktober 2020 (70).

I dag er Python en åpenkilde som styres av Python Software foundation (psf) med visjonen; å promotere, beskytte og avansere python programspråket, samt å støtte og forenkle vekst av

et variert og internasjonalt samfunn for python-programmerere (71). Det finnes derfor et rikt utvalg av ulike kilder lett tilgjengelig, både faglitteratur og egne nettsider (70).

2.6.3 Qu-path – programvare designet for kvantitativ histopatologi

Qupath er en åpenkilde programvare spesielt designet for å møte den økende etterspørselen etter brukervennlige og åpne (gratis) programvareløsninger. Fortrinnsvis innen kvantitativ histopatologi og bioteknologiske-bildeanalyser. Programvaren prosesserer whole slide images (wsi) ved hjelp av tile-baserte metoder (72) (beskrevet i kapittel 1.4).

Programvaren er også inkorporert med omfattende verktøy, som for eksempel svulst-deteksjon, mulighet for markering av ulike celletyper og vevsforandringer (72). Qupath er skrevet i groovy, og tilbyr brukeren mulighet til å addere og/eller gjøre endringer i den opprinnelige koden. Herunder er groovy et Java-syntaks-kompatibelt objektorientert programmeringsspråk. Groovy inneholder funksjoner som ligner på Python og Ruby, og kan minne om en slags blanding mellom flere ulike programmeringsspråk (73).

3 Metode

3.1 Utstyr, programvarer, rammeverk og programmeringsspråk

Utstyr:

- MacBook Pro
- Hard-disk med WSI-bilder og informasjon over nåværende score (0-3) relatert til fiske-ID.
- Totalt datasett med 911 bilder

Programvarer:

- Qupath
- Pycharm

Rammeverk og programmeringsspråk:

- Tensorflow
- Keras API:
- Python

3.2 Fremgangsmåte

- Filene ble sortert og knyttet til fiske-ID med tilhørende histopatologiscore (0-3).

- Det ble laget en ny klassifiseringsskala á syv ulike klasser. Deretter ble Atrium klassifisert etter den nye standarden og informasjonen ble ført i et excel-skjema

- Svs-filene ble konvertert til png-filer (vedlegg 1). Deretter ble atrium klippet ut, og størrelsen på bildene ble endret slik at alle fikk størrelsen (1536x1536) (vedlegg 2)

- Bildeinformasjonen lagret i excel-filen ble konvertert til en pandasliste og lagret som en csv-fil (vedlegg 3).

- Navnet på bildefilene ble endret, slik at tilhørende klasseverdi ble implementert i starten av filnavnet (vedlegg 4).

- Bearbeidede bilder med nytt ble strukturert etter klassen og plassert i et treningssett og et testsett (vedlegg 5)

- Det ble utarbeidet en ConvNet-modell (vedlegg 6), som ble forsøkt trent på datasettet bestående av syv ulike klasser.

4 Resultat og diskusjon

4.1 Bearbeiding av datamaterialet

Fenotypen til Atlantisk laks, smittet med CMS er per i dag utarbeidet ved klassifisering av histopatologiske bilder, bestående av ventrikkelen og atrium. Bildene klassifiseres etter skalaen 0-3, der klasse 0 tilsvarer friskt vev uten endringer og klasse 3 tilsvarer alvorlig grad av histopatologiske endringer, relatert til CMS (10). Mowi ønsket en høyoppløsning av denne fenotypen, hvor denne skulle utarbeides ved bruk av maskinlæring.

Datamaterialet ble klassifisert etter en ny klassifiseringsskala og deretter bearbeidet. Datamaterialet var gitt som 911 histopatologiske bilder av atrium og ventrikkelen, fra ulike smitteforsøk. Bildene var lagret som svs-filer uten en åpenbar kobling til fiske-ID og original klasseverdi. De originale klasseverdiene med tilhørende fiske ID var vedlagt som en ekstern excel fil.

Det tok derfor noe tid å sortere datamaterialet etter fiske-ID og finne riktig original klasse. Her ble det oppdaget at fiske-ID var lagret som et etikettebilde, der etikettebilde måtte åpnes manuelt for hvert bilde. Det kan tenkes at denne operasjonen kunne vært utført mer effektivt ved å kode et eget skript designet for denne type problematikk, dette ble ikke gjort grunnen manglende kunnskap rundt Svs-filer og programmering.

4.1.1 Klassifisering

Det ble laget en ny klassifiseringsskala for klassifisering av atrium. Her ble det valgt å kun fokusere på atrium. Dette for å begrense kompleksiteten, samt fordi CMS-relaterte endringer hovedsakelig rammer atrium (10). Den nye klassifiseringsskalaen ble utarbeidet ved å ta utgangspunkt i de originale klassene og veiledning fra Jinni Gu (veterinær ved veterinærinstituttet i Trondheim).

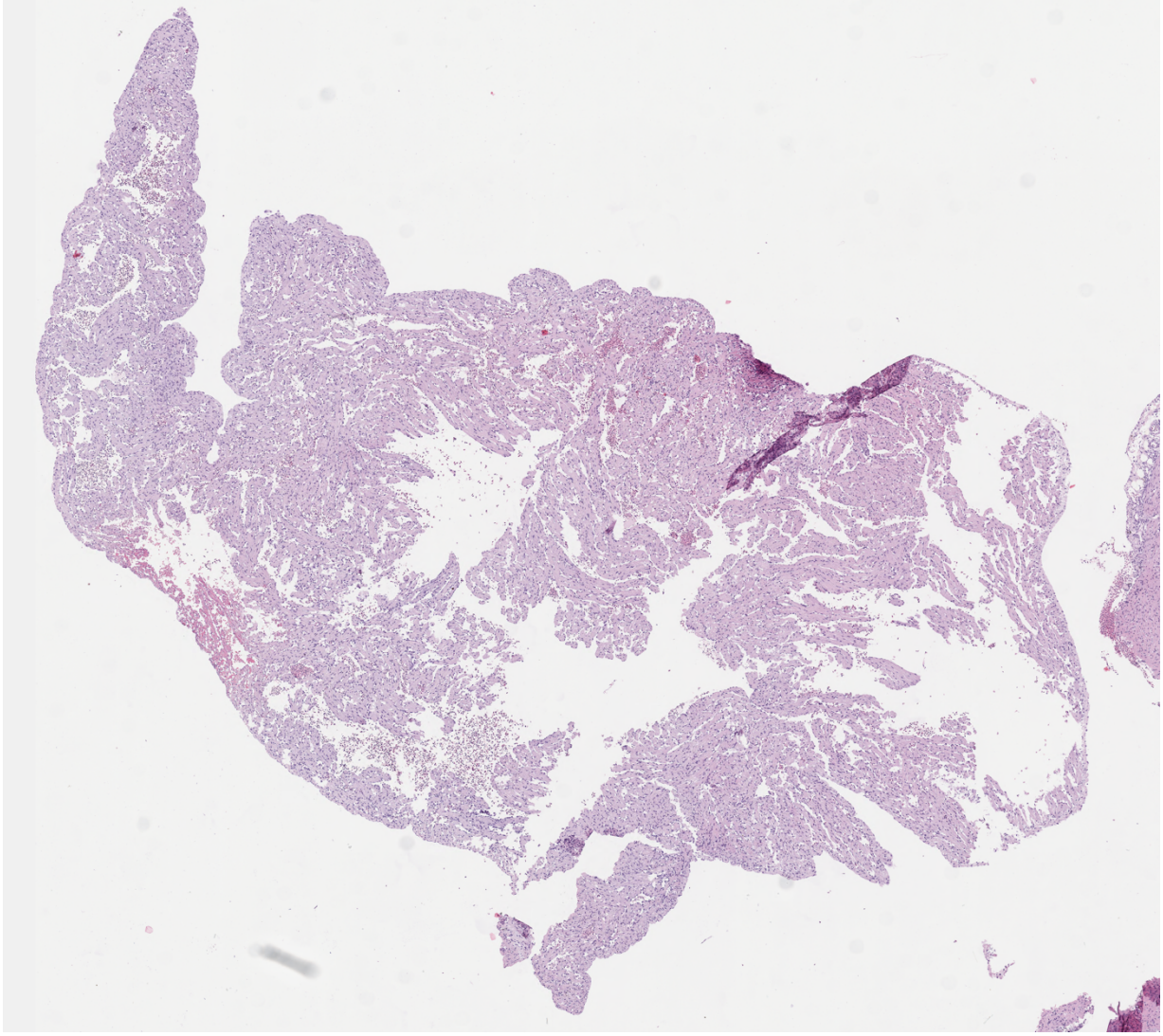
Det maksimale antall klasser for den nye skalaen ble vurdert til å være 7 klasser totalt. Dette fordi de histopatologiske endringene har relativt store likhetstrekk, noe som sannsynligvis kan medføre store overlapp ved bruk av for mange klasser. Klassifiseringsskalaen ble utarbeidet ved å legge til en klasseverdi a 0.5 mellom hver av de originale klassene. Klassene i den nye klassifiseringsskalaen ble dermed følgende: [0, 0.5, 1, 1.5, 2, 2.5, 3].

For *klasse 0* ble ikke gjort endringer for noen av bildene, og ny klasse tilsvarer derfor den original klassen (figur 4.1). *Klasse 0.5* ble bestemt til å defineres av færre enn fem multifokale endringer i vevet (figur 4.2). Klasse 1 ble bestemt til å defineres av flere enn fem, relativt små multifokale endringer i det spongiøse vevet (figur 4.3).

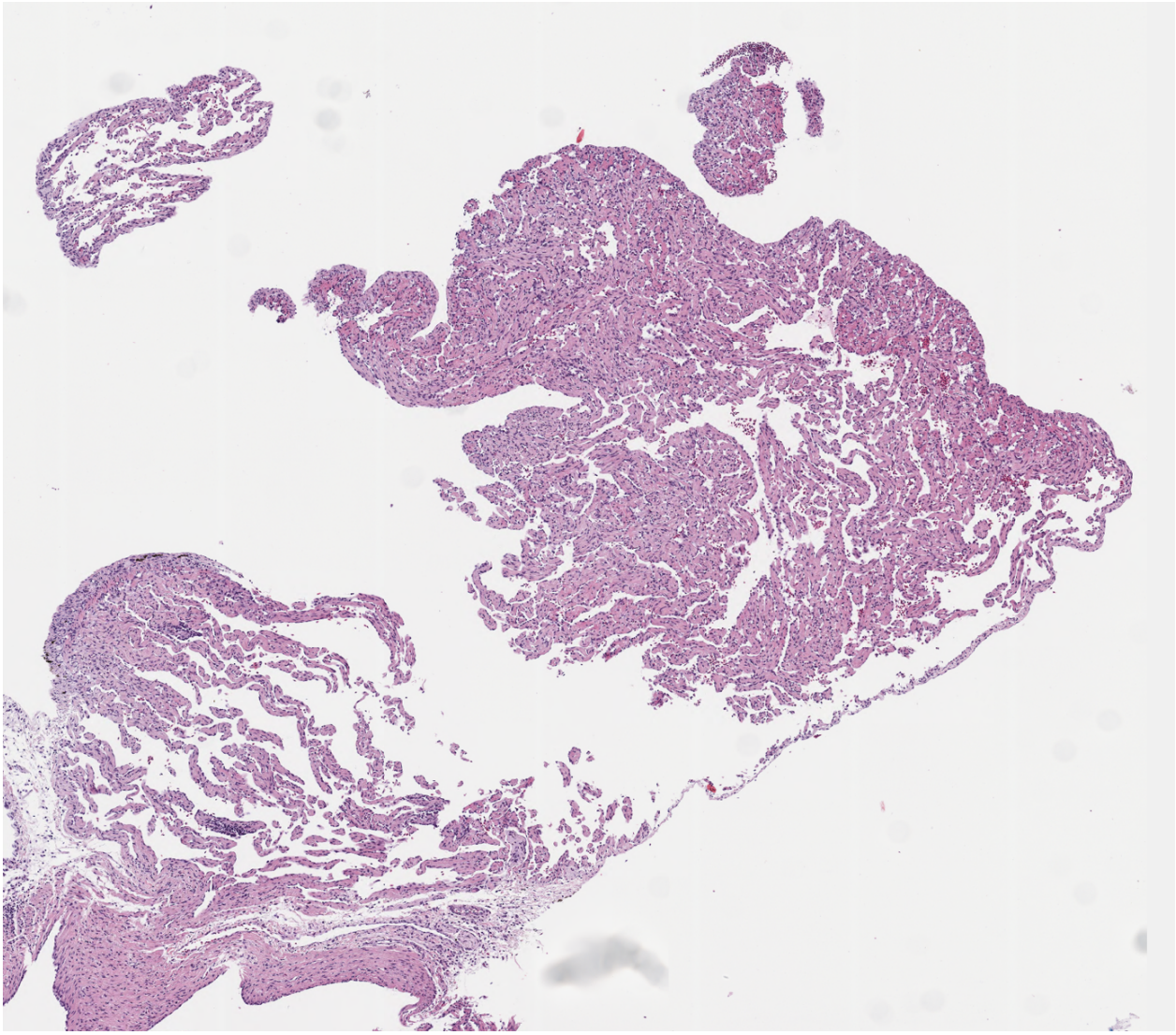
Klasse 1.5 ble utarbeidet ved å sammenligne bilder fra opprinnelig klasse 1 og opprinnelig klasse 2. Der klassen ble bestemt til å defineres av større og flere multifokale endringer enn i

klasse 1, men uten diffuse tendenser som i klasse 2 (figur 4.4). Klasse 2 ble bestemt til å defineres av flere og mer diffuse endringer i det spongiøse vevet, noe som tilsvarer definisjonen for den opprinnelige klassen (10) (figur 4.5).

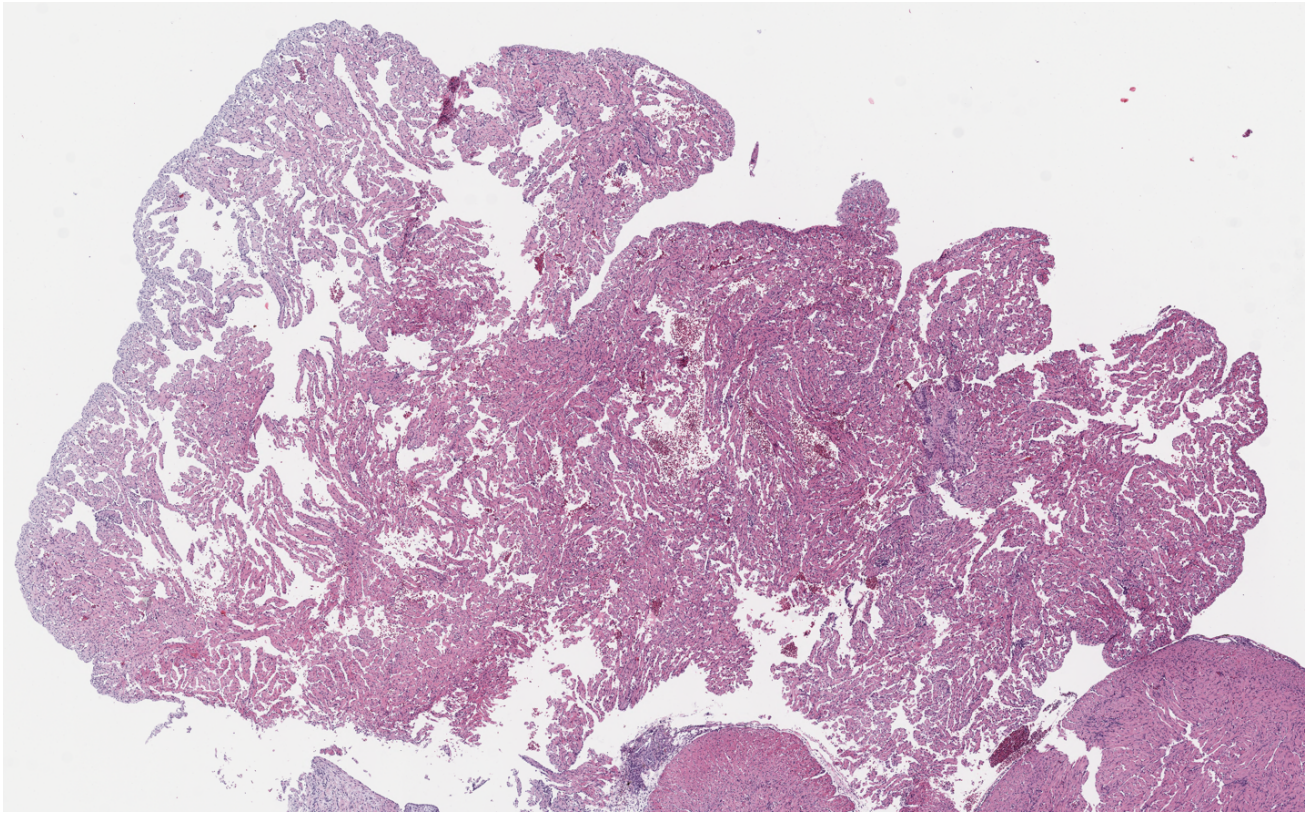
Klasse 2.5 ble bestemt til å defineres av noe mer diffuse og større endringer i det spongiøse vevet, sammenlignet med klasse 2 (figur 4.6). Klasse 3 ble beholdt som den opprinnelige klassen, der normalt vev er tilnærmet erstattet av betennelsesvev (10) (figur 4.7).



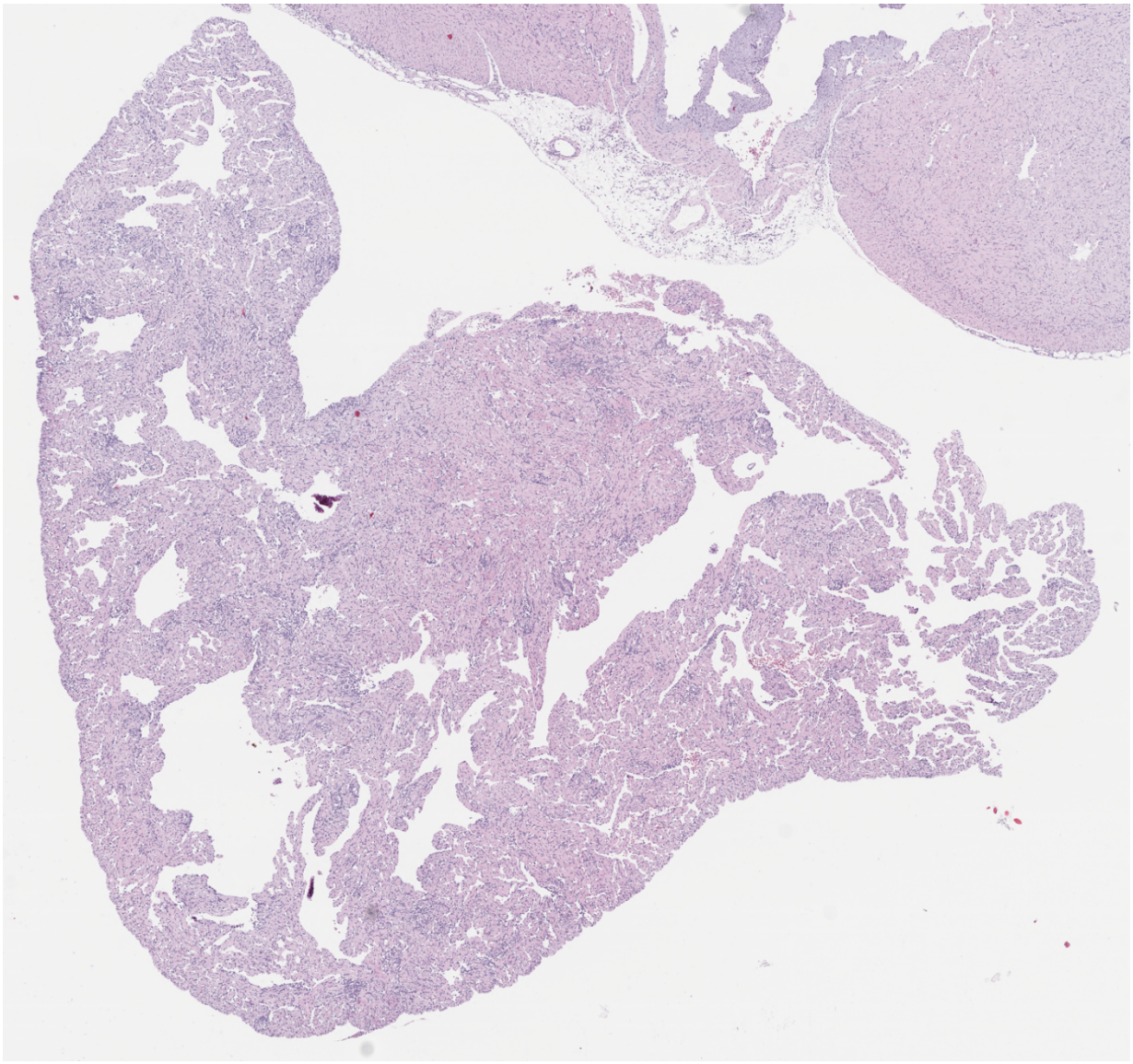
Figur 4.1: Det opprinnelige histopatologibilde er konvertert til png og atrium er klippet ut. Figuren viser ny klassifisering av klasse 0 - det spongiøse vevet i atrium er normalt og uten synlige endringer



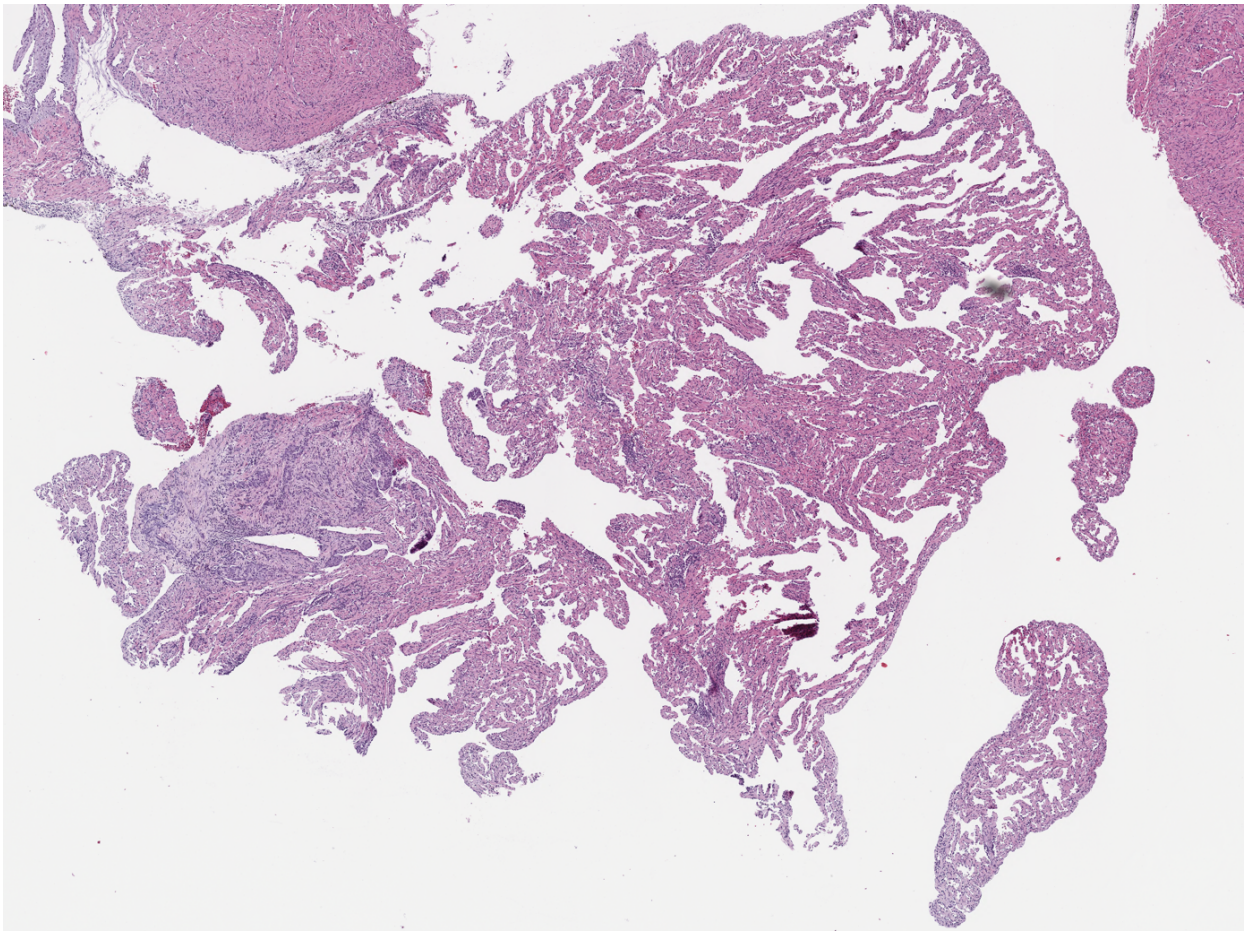
Figur 4.2: Det originale histopatologibilde er konvertert og atrium er klippet ut. Figuren viser ny klassifisering av klasse 0.5 - der det spongiøse vevet i atrium er relativt normalt, men med begynnende og relativt små multifokale endringer, det totale antallet av multifokale endringer er tre, altså færre enn fem.



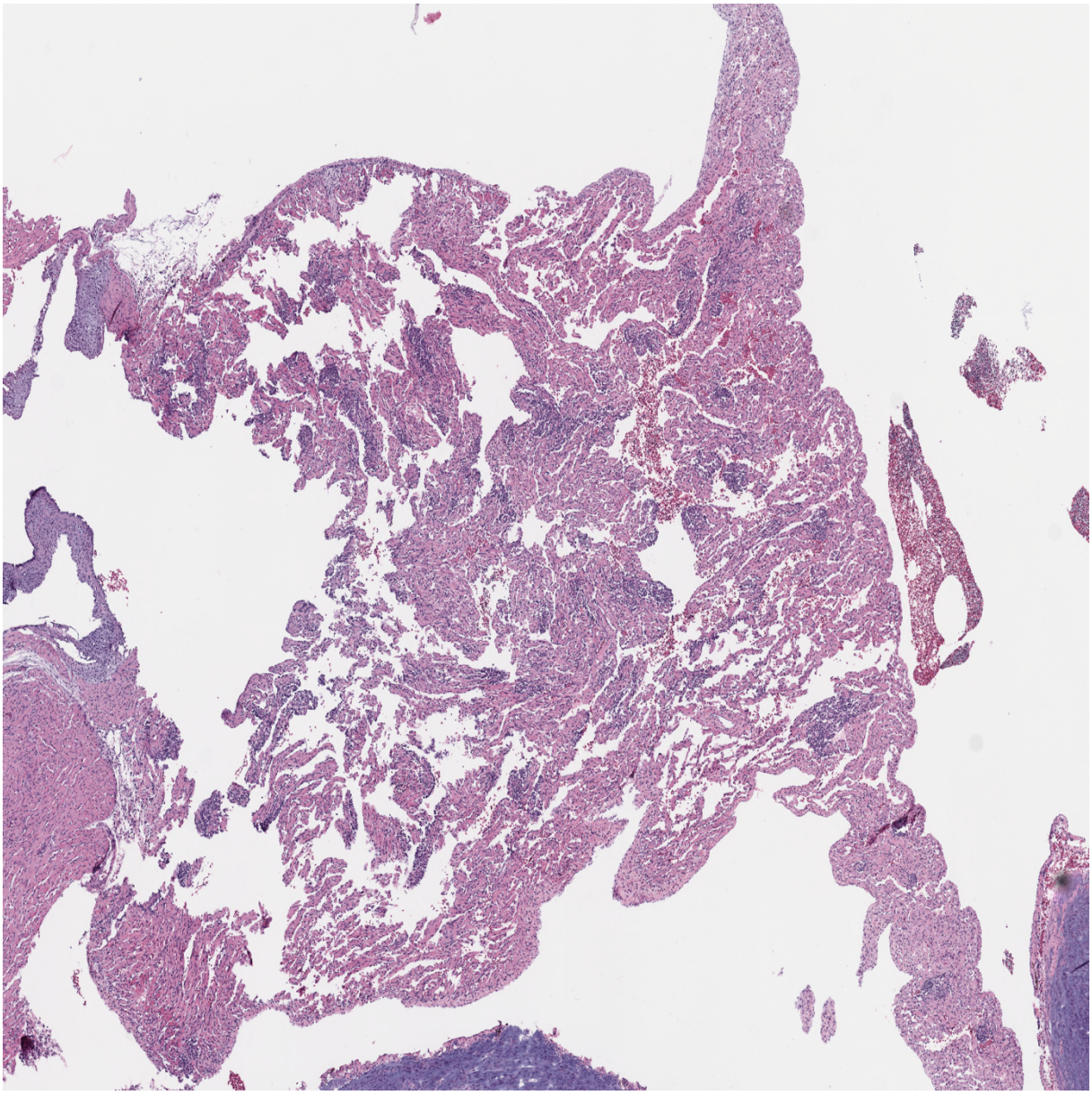
Figur 4.3: Det originale histopatologibilde er konvertert til png og atrium er klippet ut. Figuren viser ny klassifisering av klasse 1 – der de multifokale endringen i atrium er flere enn fem



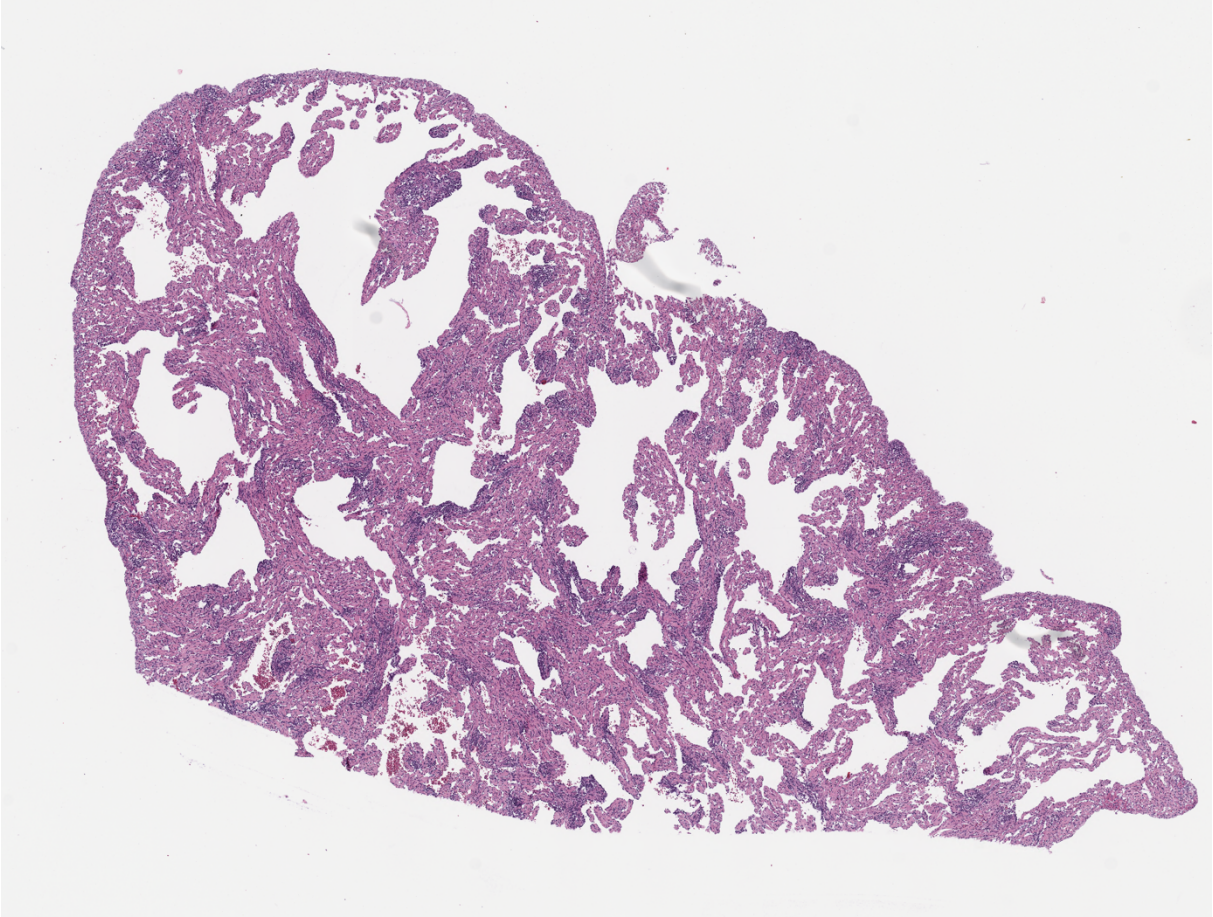
Figur 4.4: Det originale histopatologibilde er konvertert til png og atrium er klippet ut. Figueren viser ny klassifisering av klasse 1.5 - antallet multifokale endringer og størrelsen på disse er både mer utbredt og større, enn for endringene i klasse 1.



Figur 4.5: Det originale histopatologibilde er konvertert til png og atrium er klippet ut. Figuren viser ny klassifisering av klasse 2, der de multifokale endringene gradvis blir mer utbredte og diffuse.



Figur 4.6: Det opprinnelige histopatologibilde er konvertert til png og atrium er klippet ut. Figuren viser klassifisering av klasse 2.5 – der betennelsesendringene er mer diffuse og utbredt, sammenlignet med klasse 2.



Figur 4.7: Det opprinnelige histopatologibilde er konvertert til png og atrium er klippet ut. Figuren viser klassifisering av klasse 3 - betennelsesendringene er diffuse og utbredte.

4.1.2 Utfordringer knyttet til filformatet og konvertering fra svs- til png-filer.

Som nevnt var bildeformatet svs-filer da de ble klassifisert, dette er relativt store og komplekse filer (25). De fleste filene hadde en størrelse mellom 600MB og 1 GB, og det viste seg at de ikke kunne åpnes ved bruk av normale programvarer. Bildene ble derfor klassifisert i programvaren Qu.Path – en programvare tilpasset histopatologibilder med god brukervennlighet (72).

Det viste seg imidlertid at programvaren krevde mye minnekapasitet, noe som medførte at det bare kunne åpnes rundt 20 bilder samtidig. Programvaren viste også tendenser til å henge seg opp og låse seg, der dette måtte løses ved tvungen avslutning av programmet, noe som kunne medføre tap av informasjon. Klassifiseringen via Qu.Path var generelt en tidkrevende prosess. Det kan derfor tenkes at det hadde vært mer gunstig å konvertere bildefilene til png før klassifiseringen. Det kan også synes at noe av informasjonen gikk tapt

ved å konvertere filene til png, da oppløsningen ble langt dårligere, noe som gjør det vanskelig å skille de nedre klassene fra hverandre.

Det viste seg å være noe vanskelig å klassifisere histopatologibildene etter den nye standarden, da det er relativt små marginer/forskjeller som skiller de ulike klassene. Dette kan tenkes å ha bidratt til noe overlapp i klassifiseringen, som videre kan forplante seg som feil i en eventuell maskinlæringsmodell.

4.1.3 Preprosessering av datamaterialet – rettet mot maskinlæring

Det ble undersøkt og utprøvd flere muligheter for preprosessering av datamaterialet (bildene), relatert til maskinlæringsprosesser. Her ble det forsøkt å konvertere bildene til et HOG format, uten nevneverdig suksess. Det ble også forsøkt å dele alle bildene inn i tiles, da tanken bak dette var å øke størrelsen på datasettet, samt redusere størrelsen for hvert input.

Problemet med å dele bildene inn i tiles var at alle tilene tilhørende et bestemt bilde, da ble klassifisert som graden av endringer for det totalet bildet. Fordi klasseverdien gis på grunnlag av den totale graden av endringer i atrium, blir bruk av tiles mest sannsynlig rotete og feil. Dette fordi atrium klassifisert mellom klassene 0.5 og 2 kan inneholde tiles med friskt vev, avhengig av størrelsen på tilene som benyttes. Problemet kunne ha vært løst ved å dele alle bildene inn i tiles først, og deretter klassifisere hver enkelt tile manuelt, for deretter å summere disse sammen. Dog kan det tenkes at dette ville vært en relativt tidkrevende prosess med liten gevinst.

Bildene som til slutt ble brukt i maskinlæringsmodellen ble bearbeidet ved å klippe ut atrium manuelt, slik at atrium var sentrert i bilde. Formen og størrelsen på atrium var ikke konsis i de ulike bildene, noe som medførte ulike størrelser på bildene. Størrelsen til bilder med utklipp av atrium ble derfor endret, slik at alle bildene var av samme størrelse. Dette medførte enkelte merkelige transformasjoner av atrium, noe som kan ha påvirket oppløsningen.

Det ble utprøvd ulike størrelser for bildene, og endelig størrelse ble valgt til (1536x1536), da standard input for ConvNets er kvadratiske bilder (37). Det kan tenkes at den valgte størrelsen var for stor, dette fordi det generelt er vanlig å benytte bilder med størrelsen (32x32),

(128x128) og (224x224) (37). Her ble det forsøkt å trene nettverket med bilder a størrelsen (224x224), der denne størrelsen så ut til å medføre tap av informasjon i bilde.

Keras API krever at datasettet som benyttes er strukturert på en spesielt måte, ved manglende struktur vil ikke funksjonene i Keras lese datasettet. Datasettet må inneholde filer der filnavnet starter med den tilhørende klassen, videre må klassene må sorteres i ulike sub-mapper (en for hver klasse) relatert til treningssett og testsett (74). Filnavnene ble derfor tilpasset dette og bildene sortert etter nevnt struktur. En oversikt over det endelige datasettet som ble benyttet er vist i tabell 1. Datasettet ble til slutt relativt lite i forhold til det opprinnelig datasettet av ca 900 bilder tilgjengelig. Dette på grunn av et ulikt forhold mellom klassene, der klasse 0 besto av relativt få eksempler.

Tabell 1 – oversikt over datasettene som ble benyttet til maskinlæringsmodellen

Treningssett		**Testsett**	
Klasse:	**Antall:**	**Klasse:**	**Antall:**
0	37	0	10
0.5	37	0.5	10
1	37	1	10
1.5	37	1.5	10
2	37	2	10
2.5	37	2.5	10
3	37	3	10
Antall tot.	**259**	**Antall tot.**	**70**

4.2 Undersøkelse og valg av maskinlæringsmodell

Mulighetene for bruk av maskinlæring til å lage en høyoppløsning av fenotypen ble undersøkt. Her ble problemstillingen vurdert til å være et typisk klassifiseringsproblem (38), det kan

derfor synes at bruk av en overvåket læringsmetode er det mest passende alternativet. Videre ble mulighetene for klassifisering ved bruk av Support vector machine (SVM), nevrale nettverk og konvolusjonelle nettverk undersøkt.

4.2.1 Multi-klassifisering av histopatologibilder ved Support vector machine (SVM)

Det ble forsøkt å lage en klassifiseringsmodell av typen support vector machine. Da SVM ansees som et kraftig klassifiseringsverktøy med gode evner til å avgrense/skille de ulike klassene, samt god generalisering (41). Den generelle SVM-modellen ansees for å være relativt enkel og modellen har vist relativt gode resultater for også små datasett. De histopatologiske endringene i atrium er relativt like, og det kan være noe vanskelig å skille klassene fra hverandre. Det totalet datasettet inneholdt også relativt få bilder. Det kan derfor tenkes at klassifisering ved bruk av SVM kan gi en god løsning.

Problemet med SVM-modellen er at den defineres av et separabelt hyperplan, noe som gjør at modellen hovedsakelig er tilpasset lineære og binære problemstillinger. Det er vist at både de diskriminerende ferdighetene til modellen og generaliseringsevnen reduseres kraftig når datasettet er ikke-lineært (40). Det er også problemer knyttet til klassifisering av problemstillinger bestående av flere enn to klasser, da multi-klassifisering hovedsakelig er ikke-lineære problemer (42). Datasettet i denne oppgaven består av bilder fordelt mellom 7 ulike klasser, noe som gjør at generelle SVM-modeller ikke kan benyttes direkte.

Det er imidlertid mulig å designe SVM-modellen slik at den tilpasses ikke-lineære problemstillinger og multi-klassifiseringer (42). Der multi-klassifiseringsproblemet degraderes til flere binære-modeller som forenes, slik at den endelige klassen kan bestemmes. Enten ved hjelp av metoden «one vs one» og en VOTE-funksjon (stemmefunksjon) (43), eller ved bruk av metoden «one vs all» og en kernelfunksjon (40).

SVM-modeller designet for ikke-lineære, multi-klasseproblemer er relativt komplekse og inneholder mange ulike algoritmer. Denne type modeller ser derfor ut til å kreve mye programmering, noe som kan øke sannsynligheten for feil og problemer. Men, det kan tenkes at dette er løsbart for personer med høy kompetanse og kunnskap innen maskinlæring.

4.2.2 Klassifisering av histopatologibilder ved bruk av nevrale nettverk

Mulighetene for bruk av nevrale nettverk til å lage en høyoppløsning av fenotypen ble undersøkt. Her kom det relativt raskt frem at tradisjonelle nettverk ikke er spesielt godt tilpasset bildeanalyser. Dette som følge av arkitekturen og manglende invarians relatert til geometriske transformasjoner av inputdata (48). Konvolusjonelle nevrale nettverk synes derimot ut til å kunne fungere for bildeanalyser og klassifisering av bilder. Da denne typen nettverk er designet for klassifisering av bilder og objekt gjenkjenning (49). Det ble derfor valgt å forsøke og lage en prototype-modell av et konvolusjonelt nettverk (ConvNets), tilpasset tilgjengelig datasettet.

4.2.3 Utforming av arkitekturen i ConvNets-modellen

Det ble laget en ConvNet-modell ved bruk av rammeverket TensorFlow og applikasjonen Keras API, der disse programmeres ved bruk av kodespråket Python. Dette ble valgt da TensorFlow benyttes av flere store bedrifter (67), samt fordi Keras API ansees for å være svært brukervennlig, noe som reduserer antall kodesteg (66). Det finnes derfor relativt mye informasjon, veiledning og tips lett tilgjengelig.

4.2.3.1 Valg av algoritmer og hyperparametere relatert til arkitekturen

Figurene 4.8-4.14 viser valg av hyperparametere relatert til arkitekturen i nettverket og verdier av variablene. Nettverket ble bygget relativt dypt, bestående av 15 ulike convd-lag med ulike antall filter. Størrelsen på de reseptive feltene ble satt til å være konstant gjennom hele nettverket, og ReLU ble valgt som aktiveringsfunksjon. Input-data ble nedskalert ved bruk av et convd-lag med stride, uten padding - samt ved implementering av flere max-pooling lag (37). Det ble også lagt til flere lag for batch-normalisering i nettverket, med intensjon om å stabilisere nettverket og senke det interne kovariansskifte, noe som kan synes å øke treningshastigheten (59).

En fullstendig oversikt over arkitekturen og antall parametere i nettverket er presentert i tabell 2. Tabellen viser antall trenbare parametere og størrelsen av aktivitetskartene (output) for hvert lag. Det totalet antallet trenbare parameter og ikke-trenbare parametere er oppsummert nederst i tabellen. Antallet parametere er relativt høyt, noe som antas å øke kompleksiteten i nettverket (54).

```
train_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.preprocess_input) \
    .flow_from_directory(directory=train_path, target_size=(1536, 1536),
                         classes=['0.0', '0.5', '1.0', '1.5', '2.0', '2.5', '3.0'], batch_size=5, shuffle=True)

test_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.preprocess_input) \
    .flow_from_directory(directory=test_path, target_size=(1536, 1536),
                         classes=['0.0', '0.5', '1.0', '1.5', '2.0', '2.5', '3.0'], batch_size=5, shuffle=True)
```

Figur 4.8: Figuren viser bruk av funksjonen ImageDataGenerator fra Keras, en preprosseseringsfunksjon som konverter inputbildene (png-filer) til numpy arrays som kan leses av TensorFlow. Ved å kalle på funksjonen vil den returnere batches/grupper av tilfeldige x-verdier fra input (trenings eller testsettet), der bildene returneres med korresponderende y-verdi (klassenavnet) (74).

```
#blokk 1 - entry blokk
model.add(Conv2D(input_shape=(1536,1536,3),filters=(64),kernel_size=(3,3),padding="same"))
model.add(Activation("relu"))
model.add(Conv2D(filters=(64),kernel_size=(3,3),padding="valid",strides=(2,2)))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(3,3),strides=(3,3)))
```

Figur 4.9: Figuren viser den første blokka i modellen, bestående av to convd-lag. Det første laget i modellen må alltid inneholde størrelsen for input, her ble denne satt til (1536x1536x3) – altså arealet av bilde x antall fargekanaler (rgb). Begge lagene består av 64 filter hver, og størrelsen på de reseptive feltene (kjernefilteret) ble satt til å være (3x3). Det første convd-laget inneholder padding av typen zero/same, da vil stride for konvolusjonen implisitt være (1x1). I det andre convd-laget nedskaleres bilde under konvolusjonen da padding er fjernet (valid) og stride er valgt til (2x2). Nevronene aktiveres av aktiveringsfunksjonen ReLU, der denne adderes som et aktiveringslag etter hvert convd-lag.

```
#blokk 2
model.add(Conv2D(filters=(128),kernel_size=(3,3),padding="same"))
model.add(Activation("relu"))
model.add(Conv2D(filters=(128),kernel_size=(3,3),padding="same"))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(3,3),strides=(3,3)))
```

Figur 4.10: Figuren viser den andre blokka i modellen, bestående av to convd-lag á 128 filter med kjernefiltermatrise på (3x3). Det er lagt til padding av typen same med implisitt stride lik (1x1) ved begge convd-lagene. Convd-lagene aktiveres ved hjelp av aktiveringsfunksjonen

ReLU, reguleres av et batch-normaliseringslag. Output fra blokka nedskalleres ved bruk av max-pooling, dette ved en pool-matrise på (3x3) og stride (2x2) – her er er pool-matrisen 3, da input-matrisen for max-pooling laget er en oddetallsmatrise.

```
#blokk 3
model.add(Conv2D(filters=(256),kernel_size=(3,3),padding="same"))
model.add(Activation("relu"))
model.add(Conv2D(filters=(256),kernel_size=(3,3),padding="same"))
model.add(Activation("relu"))
model.add(Conv2D(filters=(256),kernel_size=(3,3),padding="same"))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(3,3),strides=(3,3)))
```

Figur 4.11: Figuren viser blokk tre i modellen, bestående av tre ulike convd-lag á 256 filter, med kjernefilter på (3x3). Det er brukt same/zero padding med implisitt stride lik (1x1) og ReLU som aktiveringsfunksjon for alle lagene. Output fra convd-lagene i blokka reguleres av et batc-normaliseringslag og nedskaleres ved bruk av max-pooling med en pool-matrise lik (3x3) og stride (2x2) .

```
#blokk 4
model.add(Conv2D(filters=(512),kernel_size=(3,3),padding="same"))
model.add(Activation("relu"))
model.add(Conv2D(filters=(512),kernel_size=(3,3),padding="same"))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(Conv2D(filters=(512),kernel_size=(3,3),padding="same"))
model.add(Activation("relu"))
model.add(Conv2D(filters=(512),kernel_size=(3,3),padding="same"))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
```

Figur 4.12: Figuren viser blokk fire i modellen, bestående av fire convd-lag á 512 filter, med kjernefilter lik (3x3). Det er benyttet same/zero padding i alle lagene, og ReLU som aktiveringsfunksjon. Convd-lagene reguleres av batc-normalisering og output nedskaleres ved bruk av max-pooling. Her er størrelsen på pool-matrisen benyttet til (2x2), da input for max-pooling laget er en partalls-matrise.

```
#blokk 5
model.add(Conv2D(filters=(728),kernel_size=(3,3),padding="same"))
model.add(Activation("relu"))
model.add(Conv2D(filters=(728),kernel_size=(3,3),padding="same"))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(Conv2D(filters=(728),kernel_size=(3,3),padding="same"))
model.add(Activation("relu"))
model.add(Conv2D(filters=(728),kernel_size=(3,3),padding="same"))
model.add(Activation("relu"))
model.add(Conv2D(filters=(728),kernel_size=(3,3),padding="same"))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
```

Figur 4.13: Figuren viser den femte blokka i modellen med fem ulike convd-lag á 728 filter og kjernefilter lik (3x3). Det er likt til same/zero padding i alle lagene, og ReLU er benyttet som aktiveringsfunksjon. Blokka reguleres av batch-normalisering og nedskaleres via max-pooling.

```
#klassifiserings blokk
model.add(Flatten())
model.add(Dense(units=4096))
model.add(Activation("relu"))
model.add(Dropout(0.5))
model.add(Dense(units=4096))
model.add(Activation("relu"))
model.add(Dense(units=7))
model.add(Activation("softmax"))
```

Figur 4.14: Figuren viser klassifiseringsblokka i modellen – der laget «Flatten» transformerer det nedskalerte aktivitetskartet (output) om til en enkel kolonne, deretter er det implementert to identiske og fullstendig sammenkoblede lag, adskilt av et dropout lag. Her er tanken at dropout-laget skal bidra til å normalisere output. Klassene beregnes i det endelige output-laget, et fullstendig sammenkoblet lag bestående av nevroner tilsvarende antallet klasser i modellen. Her er det valgt softmax som aktivering, da dette er et muli-klassifiseringsproblem.

Tabell 2: Viser en samlet oversikt over antall spesifikke lag, og alle parametere benyttet i nettverket – samt en oversikt over nedskalering og størrelsen på de ulike aktivitetskartene gjennom hele nettverket.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 1536, 1536, 64)	1792
activation (Activation)	(None, 1536, 1536, 64)	0
conv2d_1 (Conv2D)	(None, 767, 767, 64)	36928
batch_normalization (BatchNo	(None, 767, 767, 64)	256
activation_1 (Activation)	(None, 767, 767, 64)	0
max_pooling2d (MaxPooling2D)	(None, 255, 255, 64)	0
conv2d_2 (Conv2D)	(None, 255, 255, 128)	73856
activation_2 (Activation)	(None, 255, 255, 128)	0
conv2d_3 (Conv2D)	(None, 255, 255, 128)	147584
batch_normalization_1 (Batch	(None, 255, 255, 128)	512
activation_3 (Activation)	(None, 255, 255, 128)	0
max_pooling2d_1 (MaxPooling2	(None, 85, 85, 128)	0
conv2d_4 (Conv2D)	(None, 85, 85, 256)	295168
activation_4 (Activation)	(None, 85, 85, 256)	0
conv2d_5 (Conv2D)	(None, 85, 85, 256)	590080
activation_5 (Activation)	(None, 85, 85, 256)	0
conv2d_6 (Conv2D)	(None, 85, 85, 256)	590080
batch_normalization_2 (Batch	(None, 85, 85, 256)	1024
activation_6 (Activation)	(None, 85, 85, 256)	0
max_pooling2d_2 (MaxPooling2	(None, 28, 28, 256)	0
conv2d_7 (Conv2D)	(None, 28, 28, 512)	1180160

activation_7 (Activation)	(None, 28, 28, 512)	0
conv2d_8 (Conv2D)	(None, 28, 28, 512)	2359808
batch_normalization_3 (Batch	(None, 28, 28, 512)	2048
activation_8 (Activation)	(None, 28, 28, 512)	0
conv2d_9 (Conv2D)	(None, 28, 28, 512)	2359808
activation_9 (Activation)	(None, 28, 28, 512)	0
conv2d_10 (Conv2D)	(None, 28, 28, 512)	2359808
batch_normalization_4 (Batch	(None, 28, 28, 512)	2048
activation_10 (Activation)	(None, 28, 28, 512)	0
max_pooling2d_3 (MaxPooling2	(None, 14, 14, 512)	0
conv2d_11 (Conv2D)	(None, 14, 14, 728)	3355352
activation_11 (Activation)	(None, 14, 14, 728)	0
conv2d_12 (Conv2D)	(None, 14, 14, 728)	4770584
batch_normalization_5 (Batch	(None, 14, 14, 728)	2912
activation_12 (Activation)	(None, 14, 14, 728)	0
conv2d_13 (Conv2D)	(None, 14, 14, 728)	4770584
activation_13 (Activation)	(None, 14, 14, 728)	0
conv2d_14 (Conv2D)	(None, 14, 14, 728)	4770584
activation_14 (Activation)	(None, 14, 14, 728)	0
conv2d_15 (Conv2D)	(None, 14, 14, 728)	4770584
batch_normalization_6 (Batch	(None, 14, 14, 728)	2912
activation_15 (Activation)	(None, 14, 14, 728)	0
max_pooling2d_4 (MaxPooling2	(None, 7, 7, 728)	0

```
flatten (Flatten)            (None, 35672)             0
_________________________________________________________________
dense (Dense)                (None, 4096)              146116608
_________________________________________________________________
activation_16 (Activation)   (None, 4096)              0
_________________________________________________________________
dropout (Dropout)            (None, 4096)              0
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16781312
_________________________________________________________________
activation_17 (Activation)   (None, 4096)              0
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 28679
_________________________________________________________________
activation_18 (Activation)   (None, 7)                 0
=================================================================
Total params: 195,371,071
Trainable params: 195,365,215
Non-trainable params: 5,856
_________________________________________________________________
```

4.2.4 Trening og nøyaktighet av modellen

Det ble gjort forsøk på å trene modellen ved bruk av ønsket inputstørrelse på bildene (1536x1536). Dette viste seg å være relativt problematisk, da nettverket krevde alt av minnekapasitet til maskinen, noe som førte til at skriptet ble avbrutt etter noen få treningsepoker. Det ble derfor gjort forsøk på å benytte google colab, men også her kan det synes å være problemer relatert til minnekapasiteten. Det kan også tenkes at problemene skyldes bugg (feil) i skriptet, der dette kan medføre feil og problemer med beregningene internt i nettverket.

Det var mulig å trene nettverket ved bruk av relativt små bilder (224x224), dog gav ikke dette nevneverdige resultater da det så ut til at vektene ikke ble oppdatert. Det kan tenkes at input bilder av denne størrelsen medfører tap av informasjon, noe som kan medføre at nettverket ikke klarer å skille mellom de ulike klassene. Andre årsaker relatert til inputbildene kan tenkes å være overlapp mellom klassene i datasettet, da marginen mellom klassene er relativt liten.

Det svake resultatet kan også tenkes å skyldes at hyperparameterne i nettverket ikke var finstilt og at nettverket ikke var optimalisert (54). Videre kan det tenkes at nettverket var for komplekst i forhold til det relativt lille datasettet som ble benyttet. Andre årsaker kan synes å være at vektparameterne ikke ble oppdatert, som følge av «dying ReLUs» og/eller som følge av manglende essensielle funksjoner (54).

4.3.5 Evaluering av arkitekturen og hyperparametere i ConvNets-modellen

Det kan tenkes at nettverket generelt er for komplekst og består av for mange treningsparametere, slik at det vil tendere mot overfitting (47). Intensjonen med å lage et komplekst nettverk var at det skulle tilpasses kompleksiteten i histopatologibildene. Problemet er da at nettverket krever et proporsjonalt datasett (47), noe som mest sannsynlig vil ta relativt lang til å bearbeide. Det ser ut til at det hadde vært mer optimalt å lage et enklere nettverk med tanke på størrelsen av datasettet i denne oppgaven. Likevel er det ikke sikkert at et overfladisk nettverk vil evne å ekstrahere ut nødvendig informasjon fra bildene, eller å lage relasjoner mellom variablene/informasjonen relatert til betennelsesendringene i atrium.

Det ble ikke implementert en egen funksjon for initiering av vektene, noe som mest sannsynlig har medført at vektene i nettverket startet å trene med verdi 0. Når vektene i nevrale nettverk starter å trene med samme verdi, kan ikke gradienten oppdateres. Årsaken til dette er at vektparameterne i nettverket oppdateres ved å forplante det totale tapet, tilbake i modellen via back propagation. Modellen kan da finne de nevronene i nettverket som er ansvarlig for mest tap, samt hvilke som gir minst tap, slik at vektparameterne kan oppdateres (35).

Nettverket mangler en egen preprosesserings funksjon for null-sentrering av datainput ved bruk av zero-mean. Her ble det antatt at preprosesseringsfunksjonen fra Keras (figur 4.8) utfører zero-mean, da null-sentrering av inputbildene er standard prosedyre for datamaterialet i konvolusjonelle nettverk (54). Manglende null-sentrering av input-data kan medføre at datasettet presenteres som enten bare negativt, eller positivt. Der dette kan medføre at gradienten ikke konvergerer, noe igjen kan medføre vansker relatert til oppdatering av vektene i nettverket eller til effekten «dying ReLU», der nevronene dør og blir inaktive under trening (58).

Det finnes heller ikke en egen funksjon for data augmentasjon i nettverket. I og med at datasettet er relativt lite i forhold til kompleksiteten i nettverket kan se ut til at data augmentasjon bør implementeres i nettverket. Det er finnes også relativt stor geometrisk variasjon mellom atrium, noe som kan forplante støy i nettverket (64). Data augmentasjon har vært ansett som en metode for å øke invariansen i nettverket, samt for å øke støy robustheten

5 Konklusjon

Det ble utarbeidet en ny klassifiseringsskala á syv ulike klasser for klassifisering av CMS-relaterte betennelsesendringer i Atrium. Problemstillingen ble derfor vurdert til å omfatte et klassifiseringsproblem, og det ble derfor valgt å benytte en overvåket maskinlæringsmetode.

Tre ulike maskinlæringsmetoder ble vurdert, herunder Support Vector Machine, tradisjonelle nevrale nettverk og konvolusjonelle nettverk (ConvNets). Da ConvNets ble vurdert som den mest optimale løsningen for å løse problemstillingen ble det utarbeidet et forslag til en ConvNets-modell.

Trening av modellen viste liten suksess, her kan det tenkes at hovedårsaken var at vektene ikke ble initiert, noe som medførte at hverken gradienten eller vektparameterne kunne oppdateres. Det kan også tenkes at modellen ble bygget for kompleks i forhold til datasettet som ble benyttet, og/eller at den nye klassifiseringsskalaen som ble utarbeidet inneholdt mye overlapp. Andre feilkilder kan tenkes å være manglende null-sentrering og dårlig oppløsning av input-data.

Implementering av maskinlæring som et kraftig verktøy for å løse problemstillingen i oppgaven ser ut til å være en reell mulighet. Likevel ser det ut til at det gjenstår både forskning og utvikling for å utarbeide en høy funksjonell maskinlæringsmodell, tilpasset den spesifikke histopatologien relatert til CMS.

6 Veien videre

Ved å ta utgangspunkt i resultater og konklusjon fra denne oppgaven ser det ut til at det gjenstår en god del utvikling og videre studier for å løse problemstillingen som ble gitt på en effektiv på. Det ble derfor utarbeidet ulike forslag mot veien videre.

- Implementere algoritmer for å initiere vektene, da vektene mest sannsynlig ikke ble initiert i dette nettverket.

- Optimalisering av nettverket ved å finstille hyperparametere relatert både til arkitekturen og til treningsalgoritmen. Dette for å se om nøyaktigheten kan forbedres, og deretter teste generaliseringsevnen til nettverket på umerkede bilder.

- Implementere algoritmer for visualisering av aktiviteten i de ulike convd-lagene, da man ved visualisering kan få innblikk i funksjonsnivået til hvert lag. Videre kan det vurderes om eventuelle convd-lag i nettverket er overflødige og uten funksjon, der overflødige lag bør slettes for å forhindre overfitting, samt for å øke treningshastigheten.

- Implementering av algoritmer for pre-prosessering av bildene og data augmentasjon. I og med at nettverket som ble laget inneholder integrerte algoritmer fra Keras API, kan det være vanskelig å vite de eksakte funksjonene og verdiene i algoritmen. Det kan tenkes at det er en fordel å implementere algoritmer som er kodet fra start, slik at man får bedre kontroll over dette.

- Vurdere klassifiseringen som ble utført, og eventuelt lage en ny standard for ønskede antall klasser. Videre kan det tenkes å være gunstig at det klassifiseres langt flere bilder, slik at størrelsen på datasettet øker. Da klassifiseringsproblemet ser ut til å være relativt komplekst, noe som krever et komplekst nettverk med et proporsjonalt datasett, for å oppnå best mulig resultat.

- Utforske og studere mulighetene for implementering av wavelets transformasjon i nettverket. Dette kan være aktuelt da nyere studier og forskning har vist at, integrering av wavelets transformasjon i arkitekturen kan gjøre nettverket mer robust for støy. Ved å erstatte tradisjonelle nedskaleringslag med wavelets skal det være mulig å dekomponere alle komponentene og separere disse ut i fra frekvensområde. De høy-frekvente komponentene kan da filtreres ut, noe som vil kunne dempe støy og nettverket ekstraherer ut informasjonen fra de lav-frekvente komponentene (64).

- Overføre problemstillingen til Phd-stipendiater, masterstudenter innen datateknologi, eller til en bedrift med høy kunnskap og kompetanse innen maskinlæring og kunstig intelligens, for eksempel Nivero.

Litteratur

1. Mowi. Leading the blue revolution2020 10.11.2020. Available from: https://mowi.com/about/leading-the-blue-revolution/.
2. Jensen BB, Garseth ÅH. Epidemiologisk studie av Kardiomyopatisyndrom (CMS): Spredning, risikofaktorer og sykdomsforløp i norsk lakseoppdrett (CMS-Epi)2019 10.11.2020. Available from: https://www.vetinst.no/rapporter-og-publikasjoner/rapporter/2019/epidemiologisk-studie-av-kardiomyopatisyndrom-cms-spredning-risikofaktorer-og-sykdomsforlop-i-norsk-lakseoppdrett.
3. Baranski M. Forslag til bacheloroppgave - Institutt for materialteknologi, NTNU. 2019.
4. Timmerhaus G, Krasnov A, Takle H, Afanasyev S, Nilsen P, Rode M, et al. Comparison of Atlantic salmon individuals with different outcomes of cardiomyopathy syndrome (CMS). BMC genomics. 2012;13(1):205.
5. Nævdal G. Muligheter og problemer knyttet til avlsarbeid på aktuelle marine arter i oppdrett.[Oslo]: Norges forskningsråd. Området for bioproduksjon og foredling. 2003.
6. Aeffner F, Zarella MD, Buchbinder N, Bui MM, Goodman MR, Hartman DJ, et al. Introduction to digital image analysis in whole-slide imaging: a white paper from the digital pathology association. Journal of pathology informatics. 2019;10.
7. Parwani AV, Pantanowitz L, Farahani N. Whole slide imaging in pathology: advantages, limitations, and emerging perspectives. Pathology and Laboratory Medicine International. 2015.
8. Kongtorp RT, Brun E, Taksdal T, Lillehaug A. Kardiomyopatisyndrom (CMS) i Norge. National Veterinary Institute, Oslo. 2006.
9. Garseth ÅH, Svendsen J, Fritsvold C, Mikalsen AB. Kardiomyopatisyndrom (CMS) hos laks - Sykdomsutvikling - Agens - Epidemiologi2017 02.09.2020. Available from: https://www.vetinst.no/rapporter-og-publikasjoner/rapporter/2017/kardiomyopatisyndrom-cms-hos-laks.
10. Timmerhaus G. Cardiomyopathy syndrome (CMS) in Atlantic salmon, Salmo salar L.: functional genomics studies of host-pathogen responses and disease markers. 2012.
11. Ding J. Genome-wide association study of resistance to cardiomyopathy syndrome (CMS) in Atlantic salmon (Salmon salar L.): Norwegian University of Life Sciences, Ås; 2019.
12. NHI. Hjertet. 2020.
13. Helse M. Kardiomyopatisyndrom - CMS Marinhelse.no: MarinHelse; 2018 [Available from: https://marinhelse.no/cms/.
14. Løvoll M, Wiik-Nielsen J, Grove S, Wiik-Nielsen CR, Kristoffersen AB, Faller R, et al. A novel totivirus and piscine reovirus (PRV) in Atlantic salmon (Salmo salar) with cardiomyopathy syndrome (CMS). Virology Journal. 2010;7(1):1-7.
15. Fritsvold C, Jensen BB. 4.5 Hjertesprekk eller Kardiomyopatisyndrom (CMS). vetinst.no; 2019 20.09.2020.
16. Brun E, Poppe T, Skrudland A, Jarp J. Cardiomyopathy syndrome in farmed Atlantic salmon Salmo salar: occurrence and direct financial losses for Norwegian aquaculture. Diseases of aquatic organisms. 2003;56(3):241-7.
17. Garseth ÅH, Fritsvold C, Svendsen J, Bang Jensen B, Mikalsen AB. Cardiomyopathy syndrome in Atlantic salmon Salmo salar L.: A review of the current state of knowledge. Journal of fish diseases. 2018;41(1):11-26.

18. Garseth ÅH, Biering E, Tengs T. Piscine myocarditis virus (PMCV) in wild Atlantic salmon Salmo salar. Diseases of aquatic organisms. 2012;102(2):157-61.
19. Bang Jensen B, Nylund S, Svendsen JC, Ski PMR, Takle H. Indications for a vertical transmission pathway of piscine myocarditis virus in Atlantic salmon (Salmo salar L.). Journal of fish diseases. 2019;42(6):825-33.
20. Jensen BB, Brun E, Fineid B, Larssen RB, Kristoffersen AB. Risk factors for cardiomyopathy syndrome (CMS) in Norwegian salmon farming. Diseases of Aquatic Organisms. 2013;107(2):141-50.
21. aquagen. seleksjon for øket sykdomsresistens gir resultater aquagen.no: aquagen; 2005 [
22. legeforening Dn. Målbeskrivelse og gjennomføringsplan for Patologi. legeforeningen.no: den norkse legeforeningen 2006.
23. Farahani N, Post R, Duboy J, Ahmed I, Kolowitz BJ, Krinchai T, et al. Exploring virtual reality technology and the Oculus Rift for the examination of digital pathology slides. Journal of pathology informatics. 2016;7.
24. Bourke P. SVS image files2014 9.10.2020. Available from: http://paulbourke.net/dataformats/svs/.
25. Tupac.tue. Tumor Proliferation Assessment Challenge 2016 - software - reading whole-slide images tupac.tue-image.nl: Tumor Proliferation Assessment Challenge 2016; 2020 [Available from: http://tupac.tue-image.nl/node/95.
26. openslide.org. openslide - aperio format openslide.org: openslide.org; 2020 [Available from: https://openslide.org/formats/aperio/.
27. Jodogne S, Lenaerts E, Marquet L, Erpicum C, Greimers R, Gillet P, et al., editors. Open implementation of DICOM for whole-slide microscopic imaging. Proceedings,-12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (Volume 6); 2017.
28. Wehle H-D, editor Machine learning, deep learning, and ai: What's the difference? International Conference on Data scientist innovation day, Bruxelles, Belgium; 2017.
29. Lemm S, Blankertz B, Dickhaus T, Müller K-R. Introduction to machine learning for brain imaging. Neuroimage. 2011;56(2):387-99.
30. Skansi S. Introduction to Deep Learning: from logical calculus to artificial intelligence: Springer; 2018.
31. Loiseau J-C. Rosenblatt`s perceptron, the first modern neural network2019 10.11.2020. Available from: https://towardsdatascience.com/rosenblatts-perceptron-the-very-first-neural-network-37a3ec09038a.
32. Hubel DH, Wiesel TN. Receptive fields of single neurones in the cat's striate cortex. The Journal of physiology. 1959;148(3):574.
33. Widrow B, Lehr MA. 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. Proceedings of the IEEE. 1990;78(9):1415-42.
34. Hubel DH, Wiesel TN. Receptive fields and functional architecture of monkey striate cortex. The Journal of physiology. 1968;195(1):215-43.
35. Al-Masri A. How Does Back-Propagation in Artificial Neural Networks Work.
36. Thakur R. Step by step VGG16 implementation in Keras for beginners towardsdatascience.com: towardsdatascience.com; 2019 [Available from: https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c.

37. Li F-F, Johnson J, Yeung S, editors. Lecture 5 - Convolutional Neural Networks. Lecture 5-18; 2017 18.04.2017; Standford University. youtube.com: Standford University school of Engineering; 2017.
38. Brownlee J. Supervised and Unsupervised Machine Learning Algorithms 2020 15.09.2020. Available from: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/.
39. Asiri S. Machine Learning Classifiers 2018 05.11.2020. Available from: https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623
.
40. Cervantes J, Garcia-Lamont F, Rodriguez-Mazahua L, Lopez A. A comprehensive survey on support vector machine classification: Applications, challenges and trends. Neurocomputing. 2020;408:189-215.
41. Durgesh KS, Lekha B. Data classification using support vector machine. Journal of theoretical and applied information technology. 2010;12(1):1-7.
42. Band A. Multi-class Classidication-One-vs-All & One-vs-One2020 20.11.2020 . Available from: https://towardsdatascience.com/multi-class-classification-one-vs-all-one-vs-one-94daed32a87b.
43. Zhang Z, Krawczyk B, Garcìa S, Rosales-Pérez A, Herrera F. Empowering one-vs-one decomposition with ensemble learning for multi-class imbalanced data. Knowledge-Based Systems. 2016;106:251-63.
44. Github. CS231n Convolutional Neural Network for visual Recognition. Available from: https://cs231n.github.io/convolutional-networks/#overview.
45. Parmar R. Common Loss Functions in machine learning towardsdatascience.com: towardsdatascience.com; 2018 [Available from: https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23.
46. Luhaniwal V. Forward propagation in neural networks - Simlified math and code version towardsdatascience.com: towardsdatascience.com; 2019 [Available from: https://towardsdatascience.com/forward-propagation-in-neural-networks-simplified-math-and-code-version-bbcfef6f9250.
47. Bronshtein A. Train/Test split and cross Validation towardsdatascience.com2017 [Available from: https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6.
48. LeCun Y, Haffner P, Bottou L, Bengio Y. Object recognition with gradient-based learning. Shape, contour and grouping in computer vision: Springer; 1999. p. 319-45.
49. Cornelisse D. An intuitive guide to Convolutional Neural Networls2018 6.10.2020. Available from: https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/.
50. Alake R. Undertsanding Parameter Sharing (or weights replication) within Convolutional Neural Networks. 2020.
51. Kim J, Sangjun O, Kim Y, Lee M. Convolutional neural network with biologically inspired retinal structure. Procedia Computer Science. 2016;88:145-54.
52. Radhakrishnan P. What are Hyperparameters an How to tune the Hyperparameters in a Deep Neural Network towardsdatascience.com: towardsdatascience.com; 2017 [Available from: https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a.

53. Lee W-Y, Park S-M, Sim K-B. Optimal hyperparameter tuning of convolutional neural networks based on the parameter-setting-free harmony search algorithm. Optik. 2018;172:359-67.
54. Li F-F, Johnson J, Yeung S. Lecture 6 - Training Neural Networks I. youtube.com: Stanford University school of engineering; 2017.
55. Arish S, Sinha S, Smitha K, editors. Optimization of Convolutional Neural Networks on Resource Constrained Devices. 2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI); 2019: IEEE.
56. Sharma S. Activation Functions in Neural Networks: towardsdatascience.com; 2017 [Available from: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.
57. Liu D. A practical guide to ReLu medium.com: medium.com; 2017 [Available from: https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7.
58. Ashwath B. Why in preprocessing image data, we need to do zero-centered data? [Internet]. overflow s, editor. stackoverflow.com: stacoverflow.com. 2019. [cited 2020]. Available from: https://stackoverflow.com/questions/59540276/why-in-preprocessing-image-data-we-need-to-do-zero-centered-data.
59. Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:150203167. 2015.
60. D F. Batch normalization in Neural Networks: towardsdatascience.com; 2017 [Available from: https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c.
61. Pai P. Data Augmentation Techniques in CNN using Tensorflow medium.com: medium.com; 2017 [Available from: https://medium.com/ymedialabs-innovation/data-augmentation-techniques-in-cnn-using-tensorflow-371ae43d5be9.
62. Mwiti D. Convolutional Neural Networks: An Intro Tutorial Heartbeat.fritz.ai2018 [Available from: https://heartbeat.fritz.ai/a-beginners-guide-to-convolutional-neural-networks-cnn-cf26c5ee17ed.
63. Azulay A, Weiss Y. Why do deep convolutional networks generalize so poorly to small image transformations? arXiv preprint arXiv:180512177. 2018.
64. Li Q, Shen L, Guo S, Lai Z, editors. Wavelet Integrated CNNs for Noise-Robust Image Classification. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2020.
65. Benedetto JJ, Ferreira PJ. Modern sampling theory: mathematics and applications: Springer Science & Business Media; 2012.
66. Yegulalp S. What is TensorFlow? The machine learning library explained United states: InfoWorld; 2019 [Available from: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html.
67. TensorFlow.org. An end-to-end open source machine learning platform TensorFlow.org: TensorFlow.org; 2020 [Available from: https://www.tensorflow.org.
68. Rossen E. API snl.no: snl.no; 2020 [Available from: https://snl.no/API.
69. keras.io. Keras: Developer guides Keras.io: Keras.io; 2020 [Available from: https://keras.io/guides/.
70. Python.org. python Python.org: Python.org; 2020 [Available from: https://www.python.org.
71. foundation p-s. Python Software Foundation Python.org: Python.org; 2020 [Available from: https://www.python.org/psf/.

72. Salto-Tellez J, Hamilton P. QuPath: Open source software for digital pathology image analysis. Scientific reports [Internet]. 2017 25.11.2020; 7. Available from: https://www.nature.com/articles/s41598-017-17204-5.
73. groovy-lang.org. Groovy: A multi-faceted language for the Java platform groovy-lang.org: groovy-lang.org; 2020 [Available from: https://groovy-lang.org.
74. TensorFlow.org. tf.keras.preprocessing.image_dataset_from_directory tensorflow.org: tensorflow.org; 2020 [Available from: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image_dataset_from_directory.

Vedlegg

Vedlegg 1 – konvertering av SVS-filer til PNG-filer

- Dette vedlegget inneholder et skript for å konvertere svs-filer til png-filer.

```
from PIL import Image
import glob
import slideio
import sys
import os



def get_path(sti_til_mappe):
    while True:
        path = input(sti_til_mappe)
        if os.path.isdir(path):
            return path
        else:
            print("Error: Invalid file path.")
            continue



def get_downsampling_factor(sti_til_mappe):
    while True:
        try:
            value = int(input("Downsampling factor: "))
        except ValueError:
            print("Value error: Gi en verdi mellom 1 og 16")
            continue
        else:
            if value in range(1, 17):
                return value
            else:
                print("Value out of bounds: Gi en verdi mellom 1 og 16.")
                continue



def convert_svs(directory, downsampling_factor):
    for name in glob.glob(str(directory) + '/*.svs'):
```

```
        print(str("Opening slide " + str(name) + "..."), end="")
        try:
            slide = slideio.open_slide(name, 'SVS')
            print("[OK]".rjust(10))
        except:
            print("[ERROR]".rjust(10), sys.exc_info())
            continue

        print("Converting slide to PNG...", end="")
        try:
            scene = slide.get_scene(0)
            arr = scene.read_block(scene.rect,
                                   ((scene.rect[2] // downsampling_factor), (scene.rect[3] //
downsampling_factor)))
            im = Image.fromarray(arr)
            print("[OK]".rjust(10))
        except:
            print("[ERROR]".rjust(10), sys.exc_info())
            continue

        try:
            name = name.rstrip(".svs") + ".png"
            print(str("Saving as " + str(name) + "..."), end="")
            im.save(name, 'PNG')
            print("[OK]".rjust(10))
            continue
        except:
            print("[ERROR]".rjust(10), sys.exc_info())
            continue
```

Vedlegg 2 – Resizing

- Dette vedlegget inneholder et skript for å endre størrelsen på bilder fra en bestemt mappe, hvor bilder med ønsket størrelse øverføres i en ny mappe.

```
import PIL

import os

import cv2


import os.path

from PIL import Image
```

```
from os import listdir,makedirs
from os.path import isfile,join

path = (sti_til_kildemappe)
dstpath =(sti_til_nymappe)



try:
   makedirs(dstpath)
except:
   print ("Directory already exist, images will be written in asme folder")

files = [f for f in listdir(path) if isfile(join(path,f))]

for image in files:
   try:
      img = cv2.imread(os.path.join(path,image))
      img_re = cv2.resize(img, (1536,1536))
      dstPath = join(dstpath,image)
      cv2.imwrite(dstPath,img_re)

   except:
      print ("{} is not converted".format(image))
```

Vedlegg 3 – Konvertering av Excel-fil til en pandasliste lagret som csv-fil

- Dette vedlegget inneholder et skript som konverter en ønsket excel fil om til en pandasliste, der denne lagres som en csv-fil

```
import pandas as pd
import fsspec

from pandas import ExcelFile
```

```
from pandas import ExcelWriter

Excel_fil = '//Users/berittjotta/Documents/Mowi/Bildesortering.xlsx'

Laks = pd.read_excel(Excel_fil, usecols=['Bildefil', 'prøve_nr', 'atrium_new'])

print(Laks)

Laks_Ark1 = pd.read_excel(Excel_fil, index_col=0, usecols=['Bildefil', 'prøve_nr',
'atrium_new',])

print(Laks_Ark1)

df = pd.DataFrame(Laks_Ark1, columns= ['prøve_nr', 'atrium_new',])


df.to_csv('/Users/berittjotta/Documents/Mowi/Atriumdata.csv', header=True)
```

Vedlegg 4 – skript for å implementere tilhørende klasse i filnavnet

- Dette vedlegget inneholder et skript som endrer filnavnene slik at de inneholder filnavn som starter med den tilhørende klassen.

```
from pathlib import Path
import pandas as pd
from shutil import copyfile


def get_df(navn=sti_til_csv_fil):
   return pd.read_csv(navn)


df = get_df()


def finn_atrium(filsti=sti_til_mappe):
```

```
    navn = Path(filsti).stem
    nummer = str(navn)[:6]
    atrium = df.loc[df.Bildefil == nummer, 'atrium_new']
    try:
        atrium = float(atrium)
    except:
        atrium = 9.9
    return atrium


def navn_med_atrium(filsti, artium):
    navn = Path(filsti).stem
    utnavn = f'{artium:3.1f}_{navn[:6]}.png'
    return utnavn


def kopier_fil(innfil=sti_til_mappe_inn, utmappe=sti_til_mappe_ut):
    atrium = finn_atrium(innfil)
    navn = navn_med_atrium(innfil, atrium)
    utfil = Path(utmappe) / navn

    innfil = Path(innfil).resolve()
    utfil = Path(utfil).resolve()
    copyfile(innfil, utfil)


if __name__ == '__main__':

    print('Hellow!')
    mappeinn = sti_til_mappe_inn
    mappeut = sti_til_mappe_ut
```

```
    innfiler = list(Path(mappeinn).glob('*.png'))
    print(f'fant {len(innfiler)} filer!')
    for fil in innfiler:
        kopier_fil(fil, mappeut)
        print(f'Kopierte {fil}')
```

Vedlegg 5 – Skript for å strukturere datasettet

- Dette vedlegget inneholder skript for å strukturere datasettet i mappene testsett og trensett, med tilhørende sub-mapper for hver av klassene.

```
from pathlib import Path
import pandas as pd
from shutil import copyfile
import random
import glob
import os


os.chdir(sti_til_mappe)
if os.path.isdir('train/0.0') is False:
    os.makedirs('train/0.0')
    os.makedirs('train/0.5')
    os.makedirs('train/1.0')
    os.makedirs('train/1.5')
    os.makedirs('train/2.0')
    os.makedirs('train/2.5')
    os.makedirs('train/3.0')
    os.makedirs('test/0.0')
    os.makedirs('test/0.5')
    os.makedirs('test/1.0')
```

```
    os.makedirs('test/1.5')
    os.makedirs('test/2.0')
    os.makedirs('test/2.5')
    os.makedirs('test/3.0')

klasser = [
    '0.0',
    '0.5',
    '1.0',
    '1.5',
    '2.0',
    '2.5',
    '3.0',
    # '9.9',
    ]


def tell_per_klasse():
    for k in klasser:
        filer = [f for f in Path(sti_til_mappe).glob('*.png')]
        antall = len([f for f in filer if f.stem.startswith(k)])
        print(f'Det er {antall:4d} filer i "{k}".')


def finn_n_per_klasse(n):
    dictut = {}
    for k in klasser:
        filer = [f for f in Path(sti_til_mappe).glob('*.png') if f.stem.startswith(k)]
        dictut[k] = random.sample(filer, n)
    return dictut
```

```
def kopier_nm_per_klasse(n, m):
    # n treningsfiler og m testfiler
    dictut = {}
    for k in klasser:
        filer = [f for f in Path(sti_til_mappe).glob('*.png') if f.stem.startswith(k)]
        dictut[k] = random.sample(filer, n + m)

    for k in klasser:
        tren = dictut[k][:n]
        test = dictut[k][n:]

        for f in tren:
            f = f.resolve()
            utf = Path('train') / k / f.name
            copyfile(f, utf)
            print(f'Kopierte {utf}')
        for f in test:
            f = f.resolve()
            utf = Path('test') / k / f.name
            copyfile(f, utf)
            print(f'Kopierte {utf}')




if __name__ == '__main__':

    print('Hellow!')
    #lag_mapper()
    tell_per_klasse()
    for k, v in finn_n_per_klasse(3).items():
```

```
        print(k, v)


    kopier_nm_per_klasse(36, 10)
```

Vedlegg 6 – Skript for ConvNets-modellen som ble utarbeidet

- Dette vedlegget inneholder skript for ConvNets modellen som ble utarbeidet.

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from keras.models import Sequential
from keras.layers import Activation, Dense, Flatten, BatchNormalization, Conv2D,
MaxPooling2D, Dropout
from keras.metrics import categorical_crossentropy
from keras_preprocessing.image import ImageDataGenerator
from keras.applications.vgg16 import preprocess_input
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import confusion_matrix
import itertools
import os


import matplotlib.pyplot as plt
import os
import numpy as np
import PIL
from numpy import asarray



train_path = (sti_til_trensett)
test_path = (sti_til_testsett)


train_batches =
ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.preprocess_input)
\
    .flow_from_directory(directory=train_path, target_size=(1536, 1536),
                         classes=['0.0', '0.5', '1.0', '1.5', '2.0', '2.5', '3.0'], batch_size=5, shuffle=True)

test_batches =
ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.preprocess_input)
\
```

```
    .flow_from_directory(directory=test_path, target_size=(1536, 1536),
                         classes=['0.0', '0.5', '1.0', '1.5', '2.0', '2.5', '3.0'], batch_size=5, shuffle=True)
```

```
model = Sequential()

#blokk 1 - entry blokk
model.add(Conv2D(input_shape=(1536,1536,3),filters=(64),kernel_size=(3,3),padding="sam
e"))
model.add(Activation("relu"))
model.add(Conv2D(filters=(64),kernel_size=(3,3),padding="valid",strides=(2,2)))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(3,3),strides=(3,3)))



#blokk 2
model.add(Conv2D(filters=(128),kernel_size=(3,3),padding="same"))
model.add(Activation("relu"))
model.add(Conv2D(filters=(128),kernel_size=(3,3),padding="same"))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(3,3),strides=(2,2)))

#blokk 3
model.add(Conv2D(filters=(256),kernel_size=(3,3),padding="same"))
model.add(Activation("relu"))
model.add(Conv2D(filters=(256),kernel_size=(3,3),padding="same"))
model.add(Activation("relu"))
model.add(Conv2D(filters=(256),kernel_size=(3,3),padding="same"))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(3,3),strides=(2,2)))


#blokk 4
model.add(Conv2D(filters=(512),kernel_size=(3,3),padding="same"))
model.add(Activation("relu"))
model.add(Conv2D(filters=(512),kernel_size=(3,3),padding="same"))
model.add(BatchNormalization())
model.add(Activation("relu"))
```

```
model.add(Conv2D(filters=(512),kernel_size=(3,3),padding="same"))
model.add(Activation("relu"))
model.add(Conv2D(filters=(512),kernel_size=(3,3),padding="same"))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))


#blokk 5
model.add(Conv2D(filters=(728),kernel_size=(3,3),padding="same"))
model.add(Activation("relu"))
model.add(Conv2D(filters=(728),kernel_size=(3,3),padding="same"))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(Conv2D(filters=(728),kernel_size=(3,3),padding="same"))
model.add(Activation("relu"))
model.add(Conv2D(filters=(728),kernel_size=(3,3),padding="same"))
model.add(Activation("relu"))
model.add(Conv2D(filters=(728),kernel_size=(3,3),padding="same"))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))



#klassifiserings blokk
model.add(Flatten())
model.add(Dense(units=4096))
model.add(Activation("relu"))
model.add(Dropout(0.5))
model.add(Dense(units=4096))
model.add(Activation("relu"))
model.add(Dense(units=7))
model.add(Activation("softmax"))


model.summary()


epochs=50
batch_size=5
callbacks = [keras.callbacks.ModelCheckpoint("save_at_{epoch}.h5", save_best_only=True,
save_weights_only=False)]
model.compile(optimizer=keras.optimizers.Adam(learning_rate=1e-
5),loss="categorical_crossentropy",metrics=["accuracy"])
```

```
model.fit(x=train_batches,
    steps_per_epoch=train_batches.samples // batch_size,
    epochs=epochs,
    validation_data=test_batches,
    validation_steps=test_batches.samples // batch_size,
    verbose=True)
```

Vedlegg 7 – populærvitenskapelig artikkel

Er kunstig intelligens fremtidens patologer?

Oppdrettsselskapet Mowi ønsker å benytte maskinlæring for å kvantifisere histomorfologiske endringer i hjertetvevet til Atlantisk laks med hjertesprekk.

Maskinlæring er en gren innenfor kunstig intelligens og kan defineres som *et dataprogram som kan lære fra en erfaring E, med hensyn til en oppgave O og en handling H - hvis programmet som utfører O, måles av H, og utvikles med erfaring E* (Tom Mitchell, 1997)

Hjertesprekk medfører store økonomiske konsekvenser for oppdrettsnæringen

Hjertesprekk (CMS) er en alvorlig hjertelidelse som rammer norsk oppdrettslaks. Lidelsen rammer først og fremst stor og slakteferdig laks, noe som medfører betydelige økonomiske tap for næringen. Hjertesprekk er forårsaket av viruset PMCV, og lidelsen gir karakteristiske og sykelige celleendringer i hjertevevet til smittet laks.

Selektiv avl som tiltak for å bekjempe sykdommen

Resistens mot hjertesprekk er en viktig del av avlsmålet i Mowi sitt avlsprogram for Atlantisk laks. Informasjon om egenskaper relatert til resistens mot viruset skaffes ved å utføre kontrollerte smitteforsøk av laksen, der kontrollerte smitteforsøk utføres med en påfølgende analyse av hjertevevet til smittet laks. Sykelige endringer relatert til hjertesprekk vurderes i dag etter en skala mellom 0 (ingen skade) og 3 (alvorlig skade). Det har vist seg at denne analysemetoden har en relativt dårlig nøyaktighet, noe som begrenser påliteligheten av de genetiske parametere som benyttes i avlsarbeidet. Det er derfor ønskelig å optimalisere analysen, da dette sannsynligvis medfører mer nøyaktige genetiske parametere.

Er maskinlæring den hellige gral ?

Histomorfologiske analyser utføres i dag ved bruk av Whole Slide Imaging, en virtuell mikroskoperingsmetode som konverterer fullstendige utsnitt av vevsprøver om til digitale filer. Det har imidlertid vist seg å være minimale forskjeller på om patologen ved vevsanalyser og diagnostikk - benytter Whole Slide Imaging eller tradisjonell mikroskopering. Implementering av maskinlæring innen patologi kan øke nøyaktigheten av arbeidet og kan derfor synes å være den hellige gral innen fagfeltet.

Mulig løsning eller fjern fremtid ?

ConvNets er en spesiell type nevralt nettverk, designet spesifikt for bildeanalyser. Arkitekturen i nettverket er inspirert av det visuelle systemet som finnes hos mennesker, og teknologien ansees for å være revolusjonerende innenfor bildeanalyser og gjenkjenning av objekter. Likevel ser det ut til at det enda gjenstår både forskning og utvikling for å utarbeide en høy funksjonell maskinlæringsmodell, tilpasset den spesifikke histopatologien relatert til hjertesprekk (CMS).