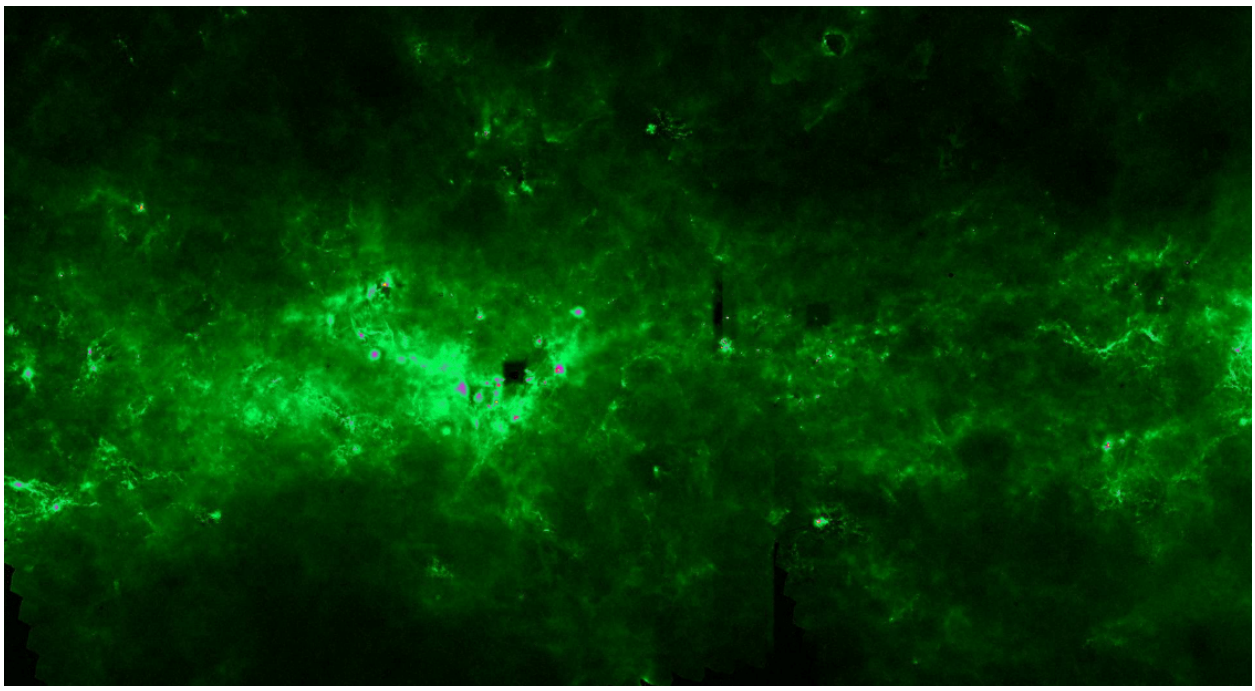


Stjärnornas vaggor: morfologi och klassificering av interstellära molekylmoln

En undersökning med bildanalysmetoder och maskininlärning

Kandidatarbete inom Rymd-, geo- och miljövetenskap

EINAR CARLSSON, WILLIAM ERIKSSON KLING, WILLIAM HILLARD, ARVID TOFT, ALMA WALLIN, VICTOR WIDÉEN

INSTITUTIONEN FÖR RYMD-, GEO-, OCH MILJÖVETENSKAP

CHALMERS TEKNISKA HÖGSKOLA
Göteborg 2024
www.chalmers.se

Kandidatarbete 2024

Stjärnornas vaggor: morfologi och klassificering av interstellära molekylmoln

En undersökning med bildanalysmetoder och maskininlärning

EINAR CARLSSON, WILLIAM ERIKSSON KLING, WILLIAM HILLARD, ARVID TOFT, ALMA WALLIN, VICTOR WIDÉEN

CHALMERS

Institutionen för Rymd-, geo- och miljövetenskap
Chalmers Tekniska Högskola
Göteborg 2024

Stjärnornas vaggor: morfologi och klassificering av interstellära molekylmoln
En undersökning med bildanalysmetoder och maskininlärning
EINAR CARLSSON, WILLIAM ERIKSSON KLING, WILLIAM HILLARD,
ARVID TOFT, ALMA WALLIN, VICTOR WIDÉEN

Handledare: Jouni Kainulainen, Institutionen för Rymd-, geo- och miljövetenskap
Examinator: Magnus Thomasson, Institutionen för Rymd-, geo- och miljövetenskap

Kandidatarbete 2024
Institutionen för Rymd-, geo- och miljövetenskap
Chalmers Tekniska Högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Omslagsbild: PROMISE-data för $l \approx 10{,}5° - 14{,}5°$ och $b < \pm 1$ enligt det galaktiska koordinatsystemet [1]. (Bild försedd med tillstånd från Jouni Kainulainen).

Skriven i LaTeX
Göteborg 2024

Stjärnornas vaggor: morfologi och klassificering av interstellära molekylmoln
En undersökning med bildanalysmetoder och maskininlärning
EINAR CARLSSON, WILLIAM ERIKSSON KLING, WILLIAM HILLARD,
ARVID TOFT, ALMA WALLIN, VICTOR WIDÉEN
Institutionen för Rymd-, geo- och miljövetenskap
Chalmers Tekniska Högskola

Sammandrag

I det ändlösa kosmoset finns struktur. Denna struktur skapas bland annat av så kallade molekylmoln i det interstellära mediet; materia mellan stjärnorna. Vid studie av dessa moln och de strukturer och fenomen som de exemplifierar och ger upphov till kan det vara praktiskt att kategorisera dem baserat på deras egenskaper och morfologi. Detta kandidatprojekt fokuserade på just denna aspekt av astronomiska studier, i syfte att undersöka huruvida det är möjligt att göra sådana klassifikationer. Molekylmoln har extraherats ur ett dataset från det opublicerade PROMISE-projektet, vilket är en kombination av data från satelliterna Herschel och Spitzer. Kombinationen resulterar i ett dataset med både Herschel-datans goda kalibrering och känslighet och Spitzer-datans höga upplösning. Molnen har bearbetats för att framhäva deras form med hjälp av ett antal bildhanteringsmetoder såsom Otsus tröskelvärdesmetod och erosion. Två olika maskininlärningsmodeller baserade på YOLO-arkitekturen tränades på de bearbetade bilderna, och grundläggande formbeskrivning och bildanalysmetoder har använts för att undersöka morfologin varefter egenskaper beskrevs i form av parametrar. Utifrån detta gjordes försök till morfologiskt baserade klassifikationer av molnen. Analys av de olika parametrarna visade dock på att det inte fanns några naturliga grupperingar vad gäller de egenskaper som undersöktes utan snarare kontinuerliga fördelningar, och att det därför kan vara mer intressant att studera relationen mellan olika parametrar. Klassifikation baserat på morfologiska parametrar krävde då subjektiva bedömningar. Detsamma gällde för övervakad maskininlärning där klassificering visade sig vara en inexakt process där subjektiva bedömningar starkt påverkade valet av klasser samt uppdelningen av molnen under dataannoteringen.

Nyckelord: molekylmoln, mörka moln, interstellärt medium, morfologi, bildanalys, maskininlärning, YOLO, PROMISE-data, klassificering.

Abstract

In the limitless cosmos one can observe structure. One example of the origin for this structure can be found among the molecular clouds of the interstellar medium; the matter between the stars. When studying these clouds, and the structures and phenomena that they exemplify and give rise to, it can be practical to categorize them based on their properties and morphology. This bachelor's thesis focused on this exact aspect of astronomical studies, with the purpose of exploring whether it is possible to make such classifications. Molecular clouds have been extracted from a dataset included in the unpublished PROMISE project, which is a combination of data from the Herschel and Spitzer satellites. This combination results in a dataset made up of both the Herschel-data's good calibration and sensitivity, and the Spitzer-data's high resolution. All clouds were processed to highlight their form with the use of image processing methods such as Otsu's method for thresholding and erosion. Two machine learning models based on the YOLO-architecture were trained on the preprocessed pictures, and basic form description and image analysis methods have been used to examine morphology, upon which select attributes were described as parameters. Following this, an attempt was made to classify clouds based on their morphology. Analysis of the different parameters showed that there was no natural groupings of the attributes but rather continuous distributions, and therefore it can be more interesting to study the relations between different parameters. Classification based on morphological parameters consequently required subjectivity. The classification made with supervised machine learning also proved to be heavily influenced by subjective assessments when creating classes and annotating data.

Keywords: molecular clouds, dark clouds, interstellar medium, morphology, image analysis, machine learning, YOLO, PROMISE-dataset, classification.

Förord

Vi vill tacka Jouni Kainulainen, vår handledare för projektet. Tack för snabb och effektiv kommunikation och det stöd vi fått genom hela arbetsprocessen, men framförallt det roliga projektet vi har fått arbeta med de senaste månaderna.

Vi vill även tacka katten Bingo för hennes bidrag som fotomodell i de fina bilderna.

Einar Carlsson, William Eriksson Kling, William Hillard, Arvid Toft,
Alma Wallin och Victor Widéen, Göteborg, maj 2024

"It depends a bit on how much you believe in spiral arms"

Innehåll

1
Introduktion

Fundamentalt så består universum av en varierad samling organiserade strukturer som gör sig själva tydliga vid olika storleksskalor. Det tydligaste exemplet av en sådan struktur framgår vid anblick mot himlen en klar natt, där stjärnor gör sig synbara. Dessa stjärnor ger i sin tur upphov till ytterligare en övergripande struktur i form av den galax som vi kallar Vintergatan [2]. Det gäller dock att stjärnor inte är de enda strukturer av intresse som en galax består av.

1.1 Bakgrund

I mellanrummen mellan stjärnorna i en galax befinner sig det interstellära mediet, eller ISM, som består av materia, elektromagnetisk strålning, magnetiska fält och gravitationsfält [3]. Med avseende på materian gäller det att cirka 90% utgörs av väte i antingen sin neutrala, molekylära, eller joniserade form. Resterande materia är mestadels helium samt små mängder av stoft, kosmiska strålpartiklar och spårelement, såsom kol, syre, och järn med flera [4].

Det är möjligt att dela upp ISM i olika faser beroende på dess beståndsdelar och andra egenskaper. En modell för denna uppdelning delar upp ISM i tre faser: ett hett joniserat medium (temperatur $\gtrsim 10^5$ K och densitet $\sim 10^{-3}$ cm^{-3}), ett varmt neutralt medium (temperatur $\sim 10^4$ K och densitet $0{,}1 - 10^2$ cm^{-3}) och ett kallt neutralt medium (temperatur $10 - 50$ K och densitet 30 till > 1000 cm^{-3}). Den kalla fasen med högre densitet brukar även kallas för 'molekylmoln' [3][4]. Dessa är större strukturer bestående av tät, kall, molekylgas [2]. 'Tät' i detta fall beskriver densiteter från 10^6 atomer per kubikmeter längs molnens kanter till runt 10^{12} atomer per kubikmeter i deras centrum [2], vilket trots de stora potenserna är mycket låga densiteter. De kan jämföras med jordens atmosfär vid havsnivå, vars densitet är runt 10^{20} gånger större än dessa värden [4].

Figur 1.1: Bild av molekylmolnet Taurus 1 tagen från Knight Observatory, Tomar, Portugal [5].

Molekylmoln är dynamiska system där nya stjärnor ständigt föds. När molekylmoln kontraherar på grund av nedkylning eller andra processer ökar densiteten inom dem, vilket i sin tur ger upphov till ytterligare kontraktion på grund av den ökande koncentrationen av massa och i sin tur även dess associerade tyngdkraft. Detta ger upphov till en snöbollseffekt som i slutänden kan resultera i tillräckliga densiteter för stjärnbildning. Det är just genom sådana händelseförlopp som majoriteten av stjärnorna i universum har bildats [2][4]. Molekylmoln är alltså en mycket viktig beståndsdel av det kosmiska ekosystemet, då de i galaxer agerar som födelseplatser för nya stjärnor.

Ett annat ord för molekylmoln är 'mörka moln', detta eftersom deras innehåll och koncentration av större stoftpartiklar gör dem opaka för synligt ljus [2][3], vilket vid observationer från jorden i det synliga spektrat får dem att se ut som svarta fläckar som blockerar ljuset från bakomliggande ljuskällor [4]. Ett exempel på detta illustreras i figur 1.1, där molekylmolnet Taurus 1 från konstellationen Taurus är avbildat [5]. Molekylmoln är dock inte opaka för infrarött ljus [3]. Nyckeln till att undersöka dessa moln ligger alltså hos detektionen av denna sortens strålning. Här visar sig de varmare faserna av ISM vara relevanta, då dessa avger infraröd strålning via termisk emission. Denna strålning absorberas och emitteras i sin tur av molekylmolnen, vars emission kan jämföras med bakgrundsemissionen för att bestämma egenskaper som exempelvis kolumndensitet hos molnen [3][4][1]. För detta projekt var det just dessa kolumndensiteter som undersöktes.

1.2 Syfte och mål

Syftet med detta projekt var att studera morfologin hos mörka moln i en del av det galaktiska planet sett från vårt solsystem, samt undersöka huruvida dessa går att klassificera med hjälp av olika bildanalysmetoder. Projektet hade en undersökande natur. Målet var att med hjälp av grundläggande formbeskrivning och analys samt övervakad maskininlärning se om det går att definiera sådana klasser utifrån molekylmolnens egenskaper och morfologi, och i så fall ta fram en klassifikationsprocess.

1.3 Avgränsningar

Projektet avgränsar sig till analys av PROMISE-datasetet; mer specifikt baserat på en sedan tidigare maskerad version av datan som grovt skilt intressanta områden från bakgrund. De metoder som tagits fram är anpassade för just detta dataset, och hade med största sannolikhet krävt modifikation för applikation på annan data. För utveckling och träning av en maskininlärningsmodell är projektet avgränsat till endast övervakad inlärning med manuell annotering. Utöver detta gäller att endast programbiblioteket Ultralytics [6] och den tillhörande modellen YOLOv8 använts för att träna klassifikationsmodeller på molndatan. Denna modell har i sig begränsningen att endast kunna hantera bilder med 8 bitars färgdjup.

2
Teori

I detta avsnitt beskrivs teori som anses relevant för projektets utförande och diskussion. Teorin inkluderar en astronomisk bakgrund, databeskrivning, metoder för bildhantering samt maskininlärning.

2.1 Molekylmoln och det interstellära mediet

Med avseende på molekylmoln och deras strukturer gäller det att dessa inte är homogent utspridda över det interstellära mediet. Deras massor och densiteter kan variera stort mellan varandra och även över enskilda moln, och även deras storlekar kan ha en stor variation. Exempelvis kan deras massor variera mellan $\sim 10^2$ $M_\odot$ och $\sim 10^7$ $M_\odot$ där $M_\odot$ är en solmassa [3], medan deras diametrar kan variera från $\sim$ 0,1pc till $\sim$ 100pc [2].

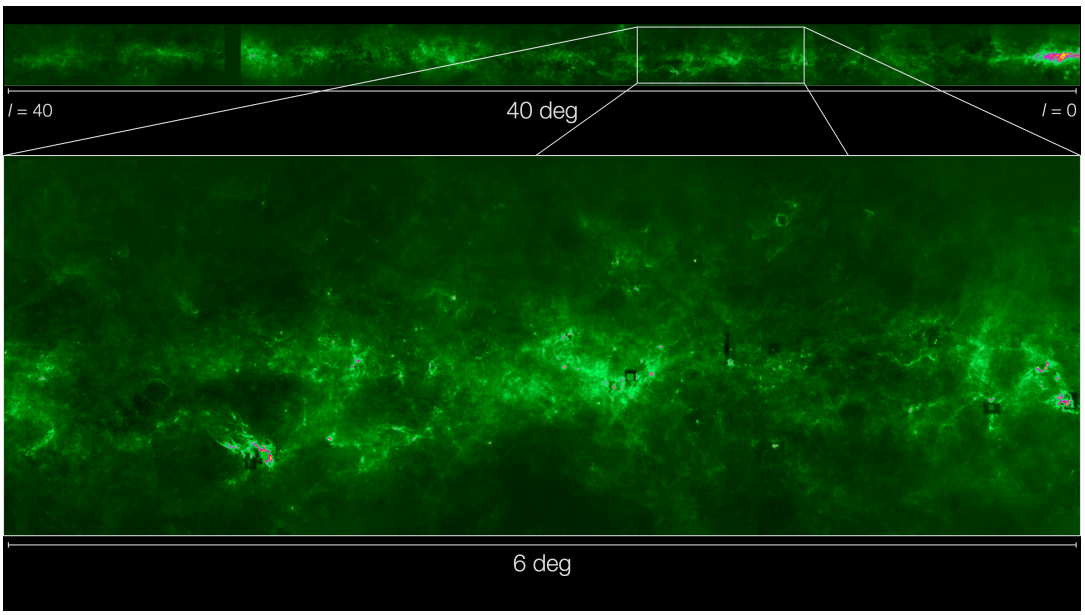
Ursprunget för denna omfattande variation kan tillskrivas de processer som ger upphov till molnens diverse strukturer. Dessa inkluderar bland annat stjärnbildning som bland annat regleras av Jeans instabilitet, vilken beskriver hur en potentiell störning i några egenskaper hos moln kan tillintetgöra den hydrostatiska jämvikten mellan gravitation och gastryck för att få molnen att kollapsa. Exempelvis så gäller det för ett moln av väte vid temperatur 10 K och atomdensitet $n \sim 10^4$ cm^{-3} att Jeans radie är 0.2 pc och Jeans massa är 1.6 $M_\odot$ [4], vilket således är de ungefärliga mått på när ett sådant moln börjar kollapsa. Stjärnbildningsprocessen kan extrahera massa från molnen och i sin tur bilda hålrum eller områden med lägre densitet [3][4].

Molnen påverkas även av strålningstrycket från bildade stjärnor, som stöter undan närliggande molnstrukturer ytterligare. När stjärnor sedan når slutet på sina livscykler och diffuserar eller exploderar i novor och supernovor tillför dessa processer materia till och stöter undan närliggande moln ytterligare. Detta skapar komprimerade områden av högre densiteter som i sin tur återigen kan ge upphov till fortsatt stjärnbildning. Även turbulens i molnens flöde har en viktig påverkan som konstant bearbetar och berör molekylmolnens utvecklande strukturer och fördelningar [3][4][7]. Utöver detta gäller det att de ständigt närvarande magnetiska fälten och gravitationsfält i helhet är väsentliga för molnstrukturen, där magnetfälten i dessa fall är en kraft som motstår kollaps [2].

Det slutgiltiga resultatet av dessa och andra processer är att de makroskopiska strukturerna och nätverken av molekylmoln i ISM kan beskrivas som något 'svampliknande' [4], med ungefärligt cirkulära strukturer av högre densiteter som ofta kallas för 'klumpar' eller 'kärnor' som är kopplade till varandra genom nätverk [2]. 'Klumpar' och 'kärnor' är specifik nomenklatur, och syftar på stora molnstrukturer som i sig kan bilda hela stjärnkluster respektive mindre strukturer som formar antingen enstaka stjärnor eller små multipla stjärnsystem [3][4].

2.2 PROMISE-projektet

Datan som användes under denna studie kommer från det opublicerade PROMISE-projektet, och förseddes via privat kommunikation med projektets huvudutredare. PROMISE-datan består av en datafil som är en kombination av data från två olika satelliter: Herschel, som samlade in kolumndensitetdata på våglängderna 100-500µm, och Spitzer, som samlade in kolumndensitetdata på våglängden 8µm. Kombinationen resulterar i ett dataset med både Herschel-datans goda kalibrering och känslighet och Spitzer-datans höga upplösning. En bild av detta dataset är givet i figur 2.1, med inzoomning på en mindre del av datasetet. Datan beskriver kolumndensiteten, det vill säga mängden materia mellan en observatör och en referensstrålningskälla, hos gasen i det interstellära mediet från vårt solsystems perspektiv. Detta täcker tusentals mörka moln inom $1.5° < \ell < 40°$ och $-1° < b < 1°$ enligt det galaktiska koordinatsystemet, med en upplösning på $2''$. För detta dataset finns även en färdig maskering som exkluderar all data förutom de tidigare identifierade molekylmolnen [1].



Figur 2.1: Färgkodad bild av det använda datasetet från PROMISE-projektet. Ljusare färger representerar högre kolumndensiteter. (Bild försedd med tillstånd från Jouni Kainulainen)

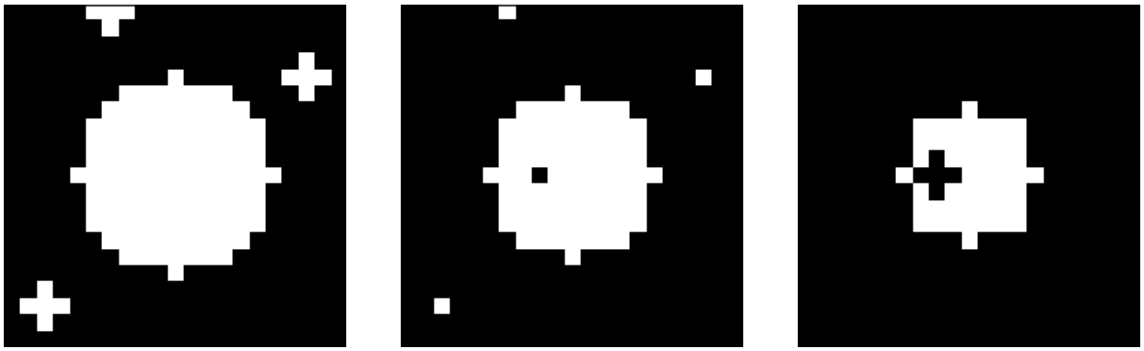
Datan från PROMISE-projektet är i formen av en FITS-fil. Filsystemet FITS (Flexible Image Transport System) är vanligt förekommande inom det astronomiska vetenskapsområdet, vars huvudsakliga ändamål är att ge upphov till effektiv transport, analysering, och arkivering av astronomisk data, specifikt för data i form av multidimensionella matriser (bilder) och 2-dimensionella värdestabeller [8]. Datan från PROMISE-projektet består bland annat av information och specifikationer gällande hur den togs fram, under vilka omständigheter, samt var och när den samlades in. Den huvudsakliga delen av datafilen består dock av en 2-dimensionell matris av storleken 120000×7000, där varje element innehåller 32-bitars flyttal som representerar kolumndensiteten i den pixeln [1]. Det är denna del av datan som används i projektet.

2.3 Bildhanteringsmetoder

Vid utvecklingen av datan i FITS-filen till tolkbara och simplifierade bilder användes ett flertal olika bildhanteringsoperationer. Nedan följer beskrivningar av några sådana operationer, specifikt 'erosion', 'dilation', 'öppning', 'stängning', Gaussisk utjämning, och Otsus tröskelvärdesmetod.

2.3.1 Erosion och dilation

Erosion och dilation är två av de mest grundläggande bildbehandlingsprocesserna. Fundamentalt så agerar erosion som en 'förtunning' av bildelement medan dilation agerar som en 'förtjockning' av bildelement, där båda metoder i slutänden leder till en minsking av överflödiga detaljer och brus [9]. Digitalt så sker erosion respektive dilation genom att varje pixel får dess värde ersatt med det lägsta respektive det högsta av angränsande pixlars värden [10].



(a) **(b)** **(c)**

Figur 2.2: Exempel på erosion och dilation på en pixelapproximation av en cirkel med några defekter. I (b) visas den ursprungliga figuren medan (a) visar ursprungsbilden med dilation och (c) visar ursprungsbilden med erosion. För både erosionen och dilationen har angränsande pixlar definierats som de fyra närmsta. Bilden är egengenererad.

Hur 'angränsande pixlar' definieras påverkar i sin tur resultaten som dessa metoder ger. De vanligaste definitionssätten är att det är fyra eller åtta närliggande pixlar som räknas som en pixels närmaste grannar. Dessa två definitionsmetoder kan illustreras enligt följande

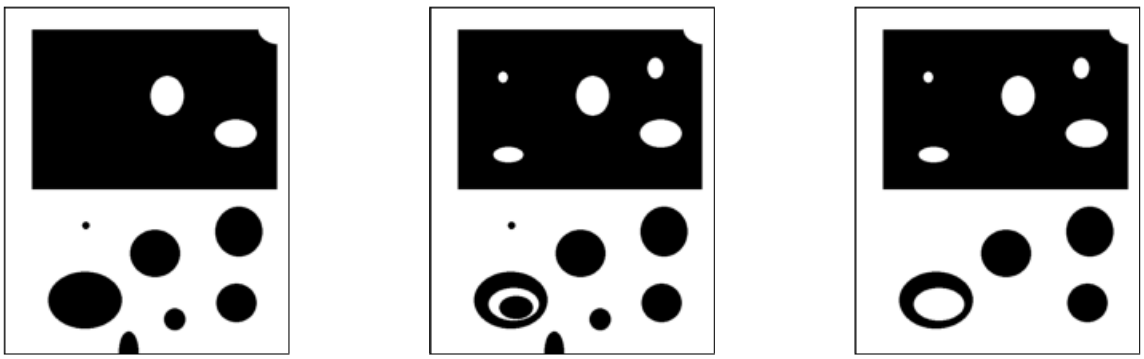
$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & \mathrm{P} & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 \\ 1 & \mathrm{P} & 1 \\ 0 & 1 & 0 \end{bmatrix},$$

där P är den pixel vars värde ska ändras, 1 är de pixlar som klassas som angränsande till P, och 0 är de pixlar som inte klassas som angränsande till P [10]. I figur 2.2 har de fyra närmaste pixlarna klassats som angränsande pixlar.

2.3.2 Öppning och stängning

Öppning och stängning innebär inom bildhantering att ta bort mindre 'öar' respektive sluta mindre 'hål' i en bild. Huvudsakligen så sker dessa operationer i en binär bild, vilket illustreras nedan i figur 2.3. Att sluta ett hål innebär i detta fall att ett område av 0:or med tillräckligt liten pixel-area som är tillräckligt omringad av 1:or fylls i med 1:or, medan borttagning av öar innebär att ett område av 1:or med tillräckligt liten pixel-area som är tillräckligt omringad av 0:or fylls i med 0:or [9][10]. Vad som menas med 'tillräckligt liten pixel-area' samt 'tillräckligt omringad' definieras båda av storleken på de underliggande operationselementen, vilket bestäms av användaren [9].

Utöver att ta bort hål respektive öar har dessa båda operationer den ytterligare effekten att 'jämna ut' former i den binära bilden. För öppning kan detta leda till att olika delar av bilden tycks 'smälta isär', medan det för stängning kan leda till att olika delar av bilden tycks 'smälta samman' [9].



(a) **(b)** **(c)**

Figur 2.3: Exempel på öppning och stängning på en binär bild där (b) representerar den ursprungliga bilden, (a) visar en öppning där små ljusa områden tagits bort och (c) visar en stängning där små mörka områden tagits bort. Bilden är egengjord.

2.3.3 Otsus tröskelvärdesmetod

Tröskelvärdesmetoder används för att dela in en bilds samtliga pixlar i två klasser, objektspixlar och bakgrundspixlar, beroende på om något värde associerat med varje pixel är större eller mindre än ett bestämt tröskelvärde. Objektpunkter är de pixlar som sägs vara 'del av' det undersökta föremålet i bilden, medan bakgrundspunkter är de resterande pixlarna. Exempelvis i en bild i gråskala, där varje pixel endast består av ett värde för att representera dess intensitet, kan samtliga pixlar delas upp i de vars intensiteter överstiger ett visst tröskelvärde och de pixlar vars intensiteter befinner sig under det, där den ena gruppen blir objektpunkter medan den andra blir bakgrundspunkter. Resultaten av en sådan kategorisering är att man får en version av den undersökta bilden som framstår som en binärbild: en bild med endast två färger, exempelvis svart och vitt, eller i datatermer som en matris med endast två olika värden utspridda bland samtliga matriselement, exempelvis 0:or och 1:or. Det gäller dock att en simpel tröskelvärdesmetod kan vara känslig för bland annat brus [9].

Otsus tröskelvärdesmetod används för att ta fram detta tröskelvärde med hjälp av ett normerat intensitetshistogram för bilden, tillsammans med sannolikhetsberäkningar. Metoden tar hänsyn till fördelningen av värden i bilden och maximerar variansen mellan de två grupperna som resulterar från indelningen. De exakta stegen och detaljerna utnyttjade för denna metod är mycket ingående, men resulterar i slutändan i en indelning som kan anses vara optimal för en majoritet av fallen [9]. Masken som skapas kan sedan faltas med originalbilden för att erhålla en bild där bakrundspixlarna tagits bort men objektspixlarna bevarats. Ett exempel på denna metod applicerad på en bild kan ses i figur 2.4.

(a) **(b)** **(c)**

Figur 2.4: Exempel på Otsus tröskelvärdesmetod på en egengjord bild. I (a) kan den ursprungliga bilden ses. I (b) visas den binära masken där områdena som klassas som objektspunkter är markerade i vitt och de i bakgrunden med svart. Slutligen visas i (c) den ursprungliga bilden med den binära masken pålagd så att de områden som klassas som bakgrund inte tas med medan de som klassas som objekspunkter bevarats med sin ursprungliga intensitet.

2.3.4 Gaussisk utjämning

Utjämningsfilter är en sorts filter i bildhantering som har i ändamål att jämna ut intensitetsövergångar. Detta kan användas för att lindra några av de problem som uppstår på grund brus och låg upplösning, exempelvis mycket skarpa intensitetsövergångar mellan enskilda pixlar, vilket kan göra det svårt att urskilja data från erhållet brus. I princip gör utjämning bilder 'suddigare' genom att jämna ut likväl detaljer som brus, vilket i vissa fall kan göra den bearbetade bilden mer lätthanterlig [9].

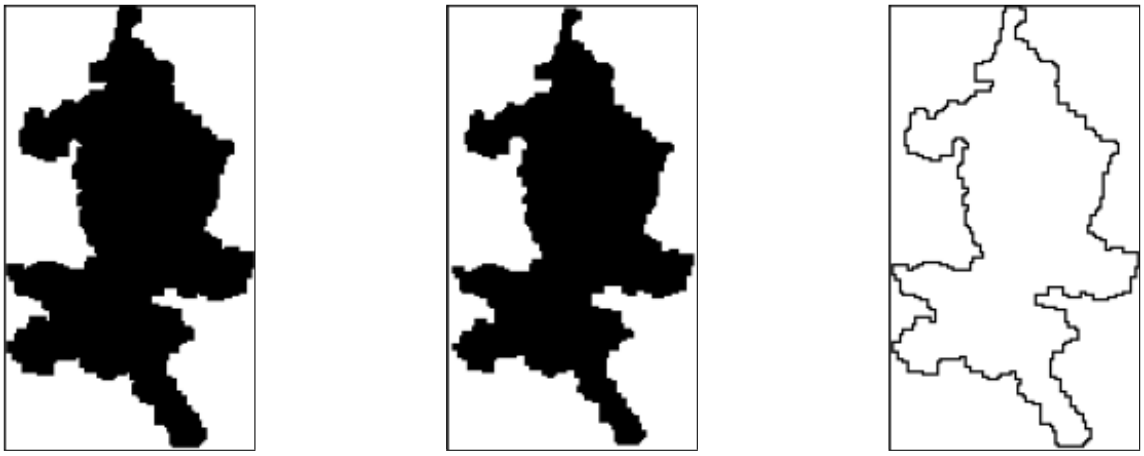
Utjämning utförs i regel genom att en så kallad 'kärna', vilket är en matris bestående av värden genererade av något typ av funktion, faltas över varje pixel i bilden. Detta ger varje pixel ett nytt genomsnittsvärde beroende av värdena hos närliggande pixlar. Ett Gaussiskt filter är i sin tur ett utjämningsfilter som använder sig av värden approximativt fördelade enligt en Gaussfördelning med ett valt värde på dess standardavvikelse. Storleken på kärnan beror i sin tur på denna standardavvikelse. En större standardavvikelse innebär en större matris och därmed en bredare utjämning, det vill säga en suddigare bild [9]. Exempel på detta kan ses i figur 2.5.

(a) **(b)**

Figur 2.5: Exempel på Gaussisk utjämning av en bild. I (a) kan originalbilden ses och i (b) kan en Gaussiskt utjämnad bild ses där många av de mindre formerna smält samman. Bilden är egengjord.

2.3.5 Omkretsapproximation

En relevant egenskap som undersöktes var omkretsar för föremål i bilder. Dessa kan exempelvis bestämmas med hjälp av programbiblioteket Scikits [10] inbyggda 'perimeter'-funktion. Funktionen är baserad på en approximationsmetod formulerad av Benkrid och Crookes i deras rapport 'Design and FsPGA Implementation of a Perimeter Estimator' [11][12]. Denna metod identifierar först en forms randpixlar genom att använda erosion, där de pixlar som eroderas identifieras som del av formens rand. Exempel på detta kan ses i figur 2.6. Sedan dras linjer som binder samman mittpunkten på alla omkretspixlar med sina grannar vars sammanlagda längd motsvarar omkretsen av föremålet [12].



(a) **(b)** **(c)**

Figur 2.6: Exempel på framtagning av omkretspixlar med hjälp av erosion. I delfigur (a) visas den binära formen som i (b) har eroderats där angränsande pixlar klassats som de åtta närmsta. I (c) visas (a) med (b) borttagen så att bara omkretsen återstår. Bilden är egengjord.

2.4 Maskininlärning och YOLO

När det gäller databehandling och -tolkning kan situationer uppstå där manuell hantering av datan eller manuell programmering av datahanteringsalgoritmer blir orimligt mödosamt och tidskrävande. Sortering och klassificering av hundratals bilder kan anses vara en av dessa. Det är under sådana omständigheter som maskininlärning kan visa sig vara det ideala valet.

2.4.1 Grundläggande principer för maskininlärning

Den maskininlärning som har använts i detta projekt är i princip baserat på inlärningen som sker inom biologiska hjärnor. Metoden går ut på att konstruera ett neuralt nätverk av noder uppdelade i olika efterföljande lager med mellanliggande viktade kopplingar. Vad som följer är en beskrivning av en grundläggande variant av ett sådant system. Varje nod i nätverket producerar utsignaler till noder i nästkommande lager baserat på insignalerna från det tidigare lagret. Det första lagret i ett sådant nätverk agerar som ett indatalager där datan som undersöks matas in, och det sista lagret agerar som utdatalager där de slutgiltiga utsignalerna erhålls. Mellanliggande lager kallas 'gömda lager', och är de lager där majoriteten av alla beräkningar utförs [13][14].

Maskininlärning delas främst upp i tre kategorier: övervakad, oövervakad och förstärkt inlärning. För detta arbete lades fokuset vid användningen av övervakad inlärning. För att ett neuralt nätverk ska kunna ge rimliga resultat måste det tränas. För övervakad maskininlärning sker detta med träningsset av annoterad data, det vill säga data där slutresultatet som ska erhållas för en datasamling redan är känt. Inlärningen i övervakad maskininlärning sker sedan baserat på jämförelser mellan

den resulterande utdatan med den förväntade utdatan. Beroende på hur korrekt nätverket är i att producera det förväntade resultatet tilldelas denna någon form av anpassningsgrad baserat på dess prestanda. Ovanstående process kallas för en 'epok'. Vid nästkommande epok ändras i sin tur vikterna på kopplingarna och noderna justeras utifrån värdet för den tidigare epoken, varefter jämförelsen görs igen. Om anpassningsgraden har minskat förkastas det nya nätverket till förmån för det tidigare. Om anpassningsgraden istället har ökat förkastas det tidigare nätverket i förmån för det nya. På så sätt kan nätverket utvecklas över epoker för att gradvis öka anpassningsgraden tills det neurala nätverket ger tillfredställande resultat med nöjaktig frekvens och säkerhet [13][14].

Problem kan dock uppstå beroende på vilken data som används för inlärningen och på vilket sätt. Exempelvis kan det resulterande nätverket vara överanpassat (för komplext) eller underanpassat (för enkelt). Överanpassning innebär att en modell har tränats för mycket på en sorts data och har anpassats på de molnens specifika egenskaper. Detta resulterar i en modell som har svårt att generaliseras och hantera ny okänd data. Underanpassning innebär istället att modellen inte har lyckats lära sig datan tillräckligt bra och resulterar i en modell som inte är tillräckligt utvecklad. För att minimera risken för sådana problem delas datan in i tre exklusiva kategorier: träningsdata, valideringsdata och testdata. Träningsdatan används för att träna modellen enligt den tidigare beskrivningen. Valideringsdatan används kontinuerligt under denna process för att se hur väl nätverket behandlar okänd data [15]. När tillräcklig utveckling har skett i modellens förutsägelseförmåga för valideringsdatan används till slut testdatan som en slutgiltig kontroll att nätverket kan appliceras och generaliseras på ny otestad data [13][14].

För att kontrollera hur väl en maskininlärningsmodell fungerar är tränings- och valideringsförluster viktiga begrepp. Förlusterna indikerar hur väl modellen förhåller sig till de två datamängderna genom att jämföra modellens gissning med verkligheten. Mindre förluster innebär att modellen förhåller sig väl till motsvarande dataset. Dessa förluster är även bra indikatorer för huruvida modellen är antingen över- eller underanpassad. Om till exempel träningsförlusten är liten men valideringsförlusten stor är modellen förmodligen överanpassad, medan om båda är stora tyder det på underanpassning. Överanpassning kan även visas genom att valideringsförlusterna vänder och börjar växa under träningen. Målet är alltså att hitta en modell där både tränings- och valideringsförlusterna är små. Detta kan till exempel göras genom att öka antal träningsepoker eller ändra modellens interna komplexitet [14].

2.4.2 You Only Look Once

Med avseende på användningsområden för maskininlärningsalgoritmer gäller det att det mest relevanta för detta arbete är detektionen och klassificeringen av föremål och former i bilder. YOLO (You Only Look Once) är en familj algoritmer med just denna specialisering. YOLO använder sig av en variant av ett 'convolutional neural network' (CNN) för detta ändamål [16]. CNN är specialiserade för att hantera data som kan delas upp i rutnät, en kategori som bland annat inkluderar bilder. Med hänsyn till bilder använder dessa nätverk faltningar mellan pixlar och deras grannar som indata, vilket resulterar i att information från varje pixel förenas med information från närliggande pixlar [14]. YOLOs specifika arkitektur är dock en mycket intrikat och komplicerad följd av matematiska operationer baserade på dessa principer vars ingående detaljer överskrider nödvändigheterna för denna rapport [6][17]. Slutresultatet är dock att det neurala nätverket konstruerat i ramverket för YOLO över tid lär sig att särskilja bilder baserat på försedda kategorier av tränings- och valideringsdata.

Varje YOLO-modells träning kan formas genom att bestämma så kallade hyperparametrar. Dessa parametervärden, som bestäms innan träningsprocessen har börjat, ställer in modellens träningsprocess för att följa bestämda regler. Exempel på hyperparametrar som bör bestämmas är data, epoker, *imgsz*, *batch* samt de som modifierar inlärningshastigheten såsom *lr0*, *lrf*, eller *cos_lr*. Data är den filväg som används för datan som modellen tränas på, epoker är antal gånger modellen tränas på hela datan, *imgsz* är den storlek bilderna ska skalas ner till för träning, och *batch* beskriver hur många bilder modellen bearbetar innan den uppdateras. Andra parametrar som kan vara värda att bestämma är exempelvis *device* eller *optimizer*. *Device* är den processor som används för träning, exempelvis om man vill träna en modell med ett grafikkort, och *optimizer* är en algoritm som används av modellen. Alla möjliga hyperparametrar finns tillgängliga i Ultralytics dokumentation för YOLO [18].

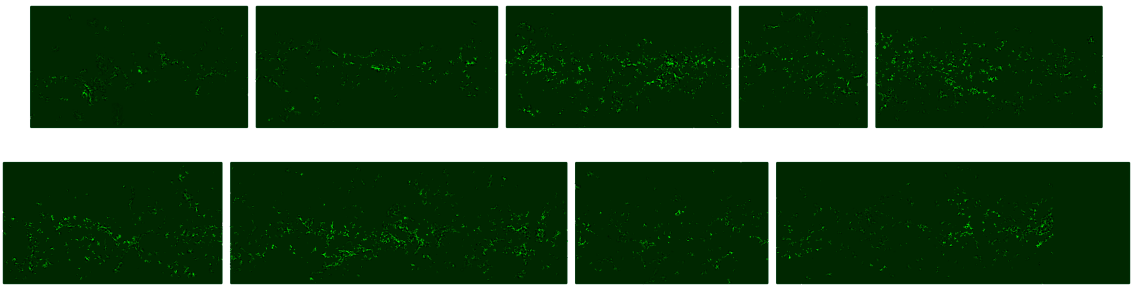
3
Metod

Projektet var indelat i tre huvudsakliga delar: uppdelning av datan och bildbehandling för framtagande av enskilda moln; undersökning och försök till klassifikation med hjälp av parametrar; samt klassifikation med maskininlärningsmodeller. Arbetet utfördes i Python och utnyttjar ett antal programbibliotek. För en komplett lista över de programbibliotek som använts, se appendix A.

Arbetet skedde iterativt. Båda grenarna av klassificeringsmetoderna byggde på samma förprocessering av datan, som resulterade i filer med enskilda moln vars struktur framhävts. Vad som anses vara ett moln kan vara en mycket subjektiv fråga, och under arbetet användes grafisk visualisering flitigt för att göra bedömningar av hur väl olika processeringsmetoder framhävde eller förvrängde de former och strukturer som annars kunde urskiljas.

3.1 Molnseparation och bildbehandling

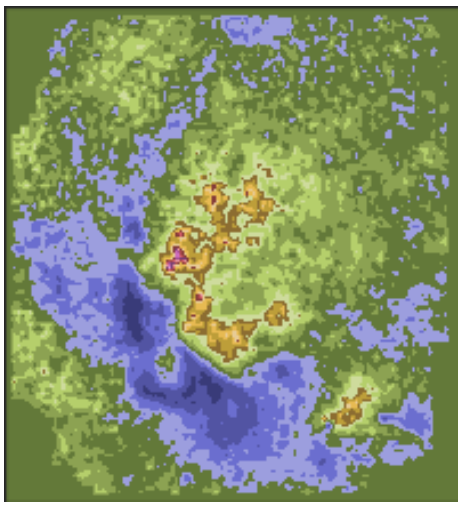
Inledningsvis genomsöktes den 2-dimensionella matrisen av datapunkter för att lokalisera samtliga vertikala sträckor av pixlar som är 'tomma', det vill säga inte innehåller några pixlar av maskeringen. Därefter valdes 8 av dessa ut med någorlunda jämna intervall för att användas som gränser för en grov uppdelning av datafilen, vars uppdelning kan ses i figur 3.1. Detta resulterade i 9 mindre FITS-filer.



Figur 3.1: I bilden visas hur det maskerade datasetet delades upp i 9 sektioner.

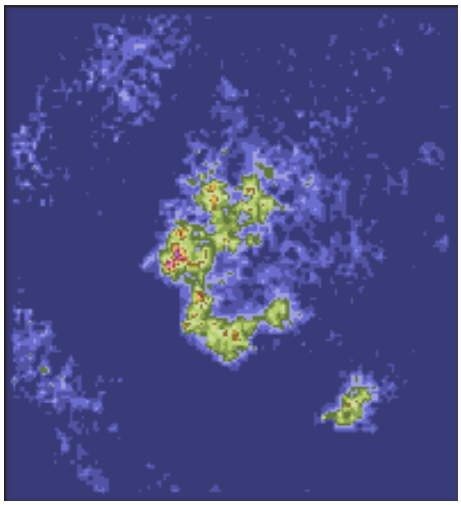
Nästa steg var att extrahera samtliga moln från dessa sektioner. Detta utfördes genom att först identifiera och sedan markera samtliga separata maskeringar i datan, varefter de isolerades för att erhålla deras dimensioner. Baserat på dessa skars mindre separata FITS-filer ut, vilket resulterade i att varje molnkluster isolerades i sin egen fil. Det var huvudsakligen på dessa filer som bildbehandling utfördes.

Innan vidare bildbehandling sållades i den slutgiltiga versionen av koden små datafiler bort. Dessa innehöll ofta mycket små samlingar av pixlar med höga värden, som vid visuell analys ibland också tycktes vara delar av ett större moln som separerats av maskeringen. Deras ringa storlek och därmed högt pixelerade natur gjorde att användbar struktur gick förlorad i senare behandlingssteg, varför dessa filer ansågs vara ointressanta för projektet. Utifrån manuell undersökning beslutades det om en något arbiträr gräns för sidlängder under 50 pixlar. Utöver detta togs även samtliga pixlar i kvarstående bilder vars värden överskred 150 bort, då dessa kan anses vara opålitliga eftersom Spitzer-teleskopet som utgör en grundläggande del av datasetet inte kunde undersöka kolumndensiteter högre än detta [7]. Ett exempel på ett moln utan vidare bildprocessering kan ses i figur 3.2, där en diskret färgskala använts för att visualisera ett moln. Denna färgskala används i samtliga steg men de exakta värdena som färgerna representerar kommer bero på det högsta och minsta pixelvärdet i bilden och kan därmed variera.



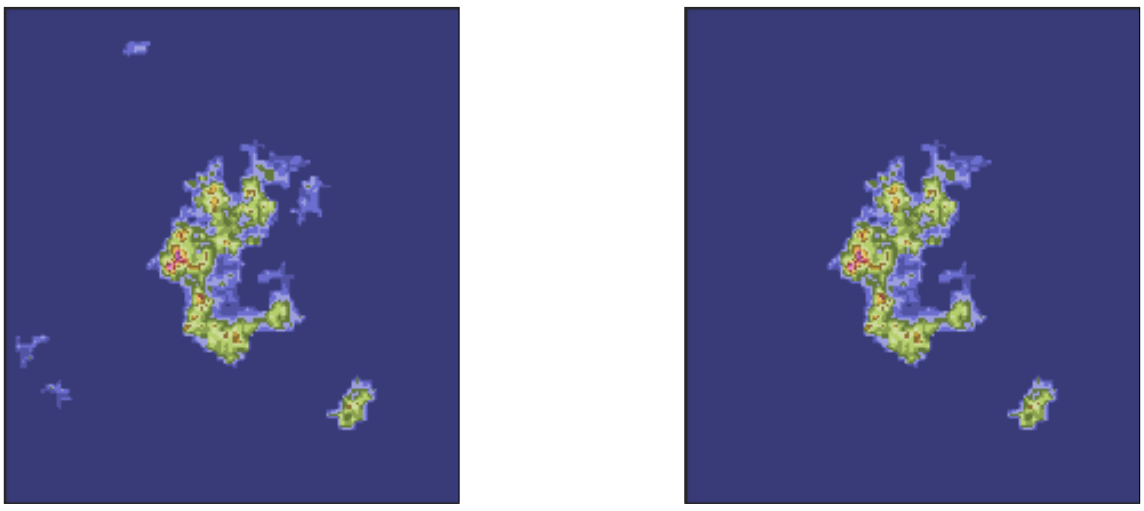
Figur 3.2: I figuren visas ett exempel på ett moln utskuret av masken efter att eventuella pixlar med värde över 150 har tagits bort. Färgskalan är diskret för enklare visualisering av molnet.

Efter detta påbörjades bearbetningen för att ta ut separata moln ur kluster. Vad som är ett moln och vad som är bakgrund är ofta väldigt godtyckligt, och kan bero på referenspunkt och omgivande objekt. För att göra denna uppdelning användes Otsus tröskelvärdesmetod. De pixlar som klassades som bakgrundspunkter sattes därefter till 0 i en binär mask som faltades med datan, som på så sätt ignorerades vid fortsatt behandling, se figur 3.3.



Figur 3.3: Figuren visar molnet i figur 3.2 efter att Otsus tröskelvärdesmetod har använts för att ta bort bakgrundspixlar. Färgskalan är diskret för enklare visualisering av molnet.

Under nästa steg togs små öar och hål bort genom en kombination av erosion med öppning och stängning, applicerat på den binära masken. Detta resulterade i ett eller flera isolerade områden per fil, vilket kan ses i figur 3.4a. Därefter beräknades värdet på den 95:e percentilen för varje område och den högsta av dessa togs fram. Områden vars 95:e percentil ej nådde hälften av det värdet sållades bort. Detta kan ses i figur 3.4b.

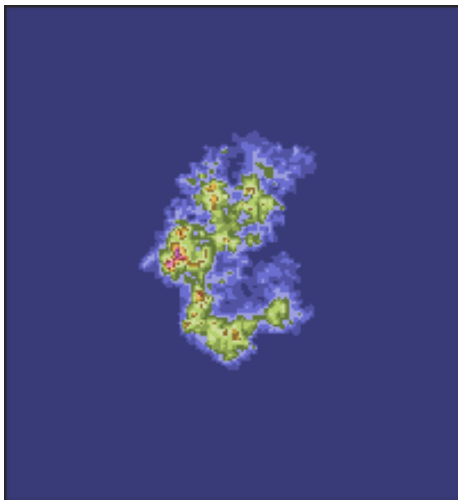


(a) **(b)**

Figur 3.4: I figurerna visas vidare processering på molnen i figur 3.3. I (a) visas molnet efter att erosion samt öppning och stängning tagit bort mycket av det som inte anses tillhöra molnet medan (b) visar molnet efter att isolerade lågintensiva områden tagits bort. Färgskalan är diskret för enklare visualisering av molnet.

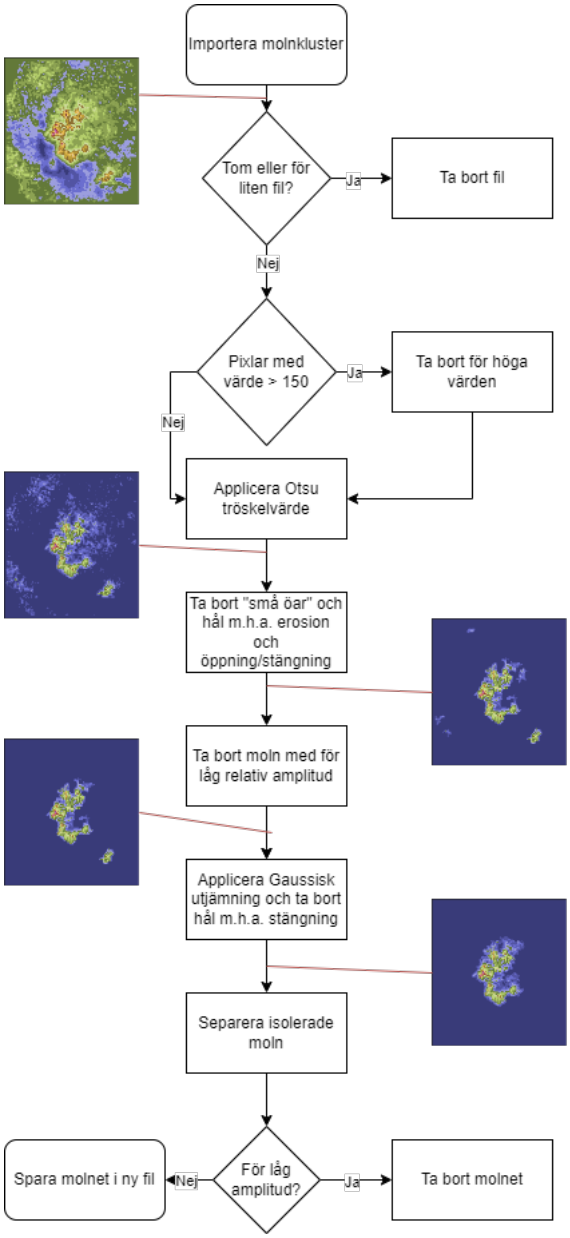
Därefter användes Gaussisk utjämning på datan med en standardavvikelse på 0,4 i kombination med ytterligare en öppning och stängning av den binära masken för att

ta bort eventuella små öar eller hål som kvarstod. De kvarvarande isolerade molnen separerades sedan, varefter de återigen sållades utefter värdet på den 99:e percentilen. Enbart de som uppnådde ett värde på 5 eller högre sparades. Detta värde valdes då moln som inte uppfyllde det kravet generellt inte genomgick bildbehandlingen väl, framförallt eftersom Otsus tröskelvärdesmetod kräver större spridning bland värdena för att ge önskade resultat i bildhanteringen. Slutresultatet av detta kan ses i figur 3.5.



Figur 3.5: I figuren visas den slutliga bilden utifrån vilket de enskilda molnen kommer isoleras och sparas. Denna erhålls av molnet i figur 3.4b efter att det utjämnats Gaussiskt och för små områden tagits bort. Färgskalan är diskret för enklare visualisering av molnet.

I figur 3.6 visas ett flödesdiagram över de huvudsakliga stegen för den inledande databehandlingen efter den ursprungliga separationen av den maskerade datan i mindre klusterfiler.



Figur 3.6: Flödesdiagram som visar bildprocesseringen av de moln som är uttagna från den maskerade datan. Som komplement finns även exempelbilder som visar hur ett moln kan se ut i de olika stegen av processering.

3.2 Bildanalys med klassiska metoder

En av analysmetoderna som användes på de framtagna molnen var bestämningen av olika egenskaper och karaktärsdrag med hjälp av grundläggande formbeskrivning och analys, även kallat 'klassiska metoder'. De attribut som undersöktes var area, omkrets, cirkularitet, total kolumndensitet, genomsnittlig kolumndensitet, utsträckning samt en specifik formfaktor. Efter framtagning sammanställdes parametrarna hos ett stort antal moln och trender undersöktes grafiskt.

Arean, A, beräknades genom att summera antalet pixlar som fastställdes utgöra de undersökta molnen efter den preliminära bildhanteringen i de binära representationerna. Omkretsen, S, beräknades med hjälp av programbiblioteket Scikits [10] inbyggda 'perimeter'-funktion, som beskrevs i delkapitel 2.3. Utifrån detta bestämdes formernas cirkularitet $C = 4\pi A/S^2$ vilket ger ett värde mellan 0 och 1, där 1 representerar en perfekt cirkel.

Den totala kolumndensiteten, TKD, erhölls genom att summera värdena på alla pixlar i ett moln. Utifrån detta beräknades den genomsnittliga kolumndensiteten, GKD, genom att dividera detta värde med den tidigare beräknade arean.

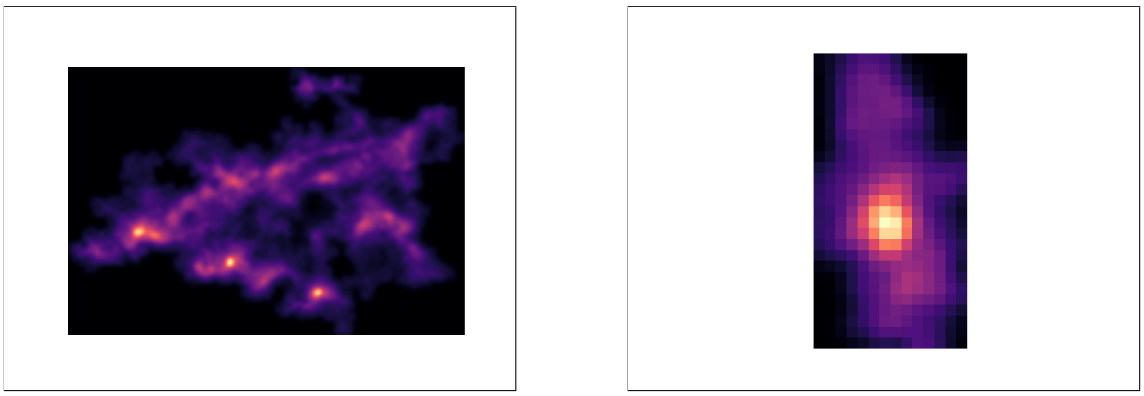
Molnens utsträckning, U togs fram genom att först identifiera de pixlar som befann sig längst ifrån varandra i den undersökta bilden. Detta skedde genom att ta fram molnets samtliga omkretspixlar, varefter avståndet mellan varje par beräknades. Utsträckningen bestämdes som det maximala avståndet mellan två pixlar. Slutligen beräknades en specifik formfaktor, $F = U/S$, för molnen. Denna formfaktor avsågs vara ett komplement till cirkulariteten, för att kvalitativt kunna skilja på raka, smala moln, vilka skulle få ett större värde, och breda eller krokiga moln som skulle få ett mindre värde.

3.3 Klassificering med hjälp av maskininlärning

De maskininlärningsmodeller som användes under detta arbete var förtränade versioner av en YOLOv8-modell. De använda modellerna valdes med hänsyn till deras komplexitet. Ultralytics erbjuder fem olika klassificeringsmodeller i stigande komplexitetsordning för YOLOv8: *n*, *s*, *m*, *l*, *x*. En ökning i komplexitet innebär även en förbättring av modellens träffsäkerhet på bekostnad av träningstiden, som är betydligt större för de mest komplexa modellerna. Tidsåtgången bedömdes dock vara irrelevant i denna undersökning och var inte en viktig faktor i valet av den förtränade modellen.

Träningen som utfördes på de framtagna molnbilderna krävde att ett antal hyperparametrar för YOLO behövde specificeras. Detta inkluderade antal epoker som modellen skulle tränas över samt storleken på bilderna som den skulle bearbeta. Även antalet bilder som modellen bearbetade innan den uppdateras kunde förändras med parametern *batch*. Antal epoker valdes för att minska risken för överanpassning

och bildstorleken *imgsz* sattes till den maximala 224x224 pixlar. I undersökning har samtliga använda bilder samma storlek på 640x480 pixlar, vilka YOLO alltså behöver skala ner. Se figur 3.7 nedan för exempel på bilderna som YOLO blir matad med.



(a) **(b)**

Figur 3.7: Exempel på bilder som YOLO har hanterat. Bilderna består av ett moln i mitten omgivet av en vit bakgrund för att justera bilderna till rätt storlek.

Datauppdelningen i träningsdata, valideringsdata och testdata skedde genom slumpmässig utplockning. Ungefär 60% av datan valdes ut som träningsdata medan resten av datan avsattes som validerings- respektive testdata.

3.3.1 Etablering av modellprestanda

Ett prestandaverktyg som erhålls efter träningen av en modell är en så kallad 'förvirringsmatris' (confusion matrix) som kan användas för att identifiera klassificeringar av bilder som är för lika varandra och ofta misstas. Från denna kan två prestandamått bestämmas, som kallas för 'precision' respektive 'recall'. 'Precision' är ett mått på modellens exakthet, och anges som den andel av en klass av data som modellen gjorde en korrekt gissning på över samtliga tillfällen då modellen gissade att data tillhörde den klassen. Denna ekvation kan skrivas enligt

$$\text{Precision}_{\text{klass}} = \frac{\text{TP}_{\text{klass}}}{\text{TP}_{\text{klass}} + \text{FP}_{\text{klass}}} \tag{3.3.1a}$$

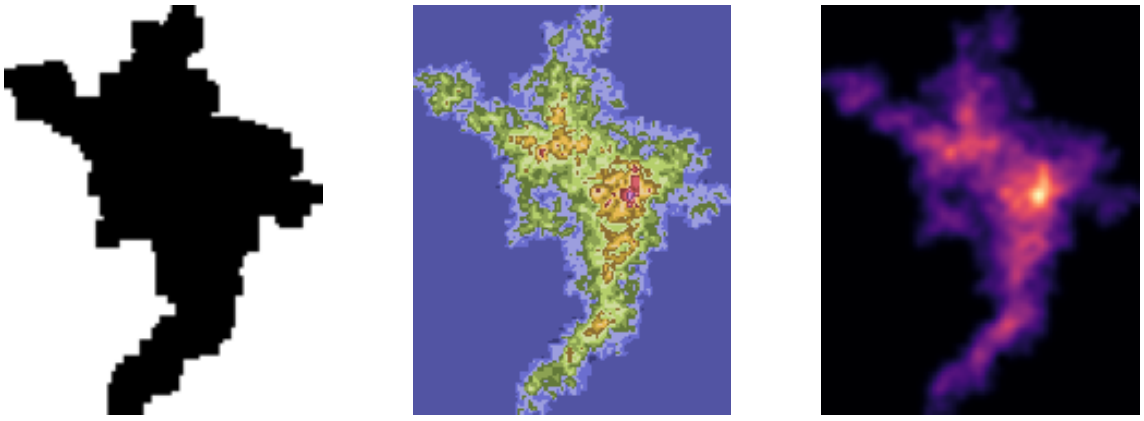
där TP_{klass} betecknar antalet korrekta gissningar på den givna klassen (True Positive) och FP_{klass} står för antalet inkorrekta gissningar på den givna klassen (False Positive) [15]. 'Recall' är i sin tur ett mått på modellens förmåga att förutsäga klasser. Detta definieras som andelen korrekta gissningar på en klass över antalet gånger som en bild tillhörande den klassen visades. Detta uttryck ges enligt

$$\text{Recall}_{\text{klass}} = \frac{\text{TP}_{\text{klass}}}{\text{TP}_{\text{klass}} + \text{FN}_{\text{klass}}} \tag{3.3.1b}$$

där FN_{klass} står för antalet gånger en bild av den givna klassen felaktigt identifierades tillhöra en annan klass (False Negative) [15].

3.3.2 Bildformatering för inlärningsmodell

För att kunna utnyttja maskininlärningsmodellen krävdes att de bilder som användes innehöll all relevant information som var nödvändig för klassificering. Detta innebar att parametrar som molnens form och massfördelning behövde vara tillräckligt tydliga för att kunna urskiljas, både för inlärningsmodellen och för annotering av datan. Detta behov resulterade i bildhanteringsmetoden som presenterades i avsnitt 3.1. I figur 3.8 nedan visas tre exempel på bilder där olika färgskalor har använts för att visualisera molndatan på ett effektivt sätt.



(a) **(b)** **(c)**

Figur 3.8: Jämförelse mellan olika datavisualiseringar inför användning inom maskininlärning. I delfigur (a) visas en binär mask, där pixlar som hör till molnet har värde 1 (svart) och övriga värde 0 (vitt). Delfigur (b) och (c) har båda skapats ur samma process med diskret respektive kontinuerlig färgskala. Datan i delfigur (c) har dessutom genomgått ytterligare en Gaussisk utjämningsprocess.

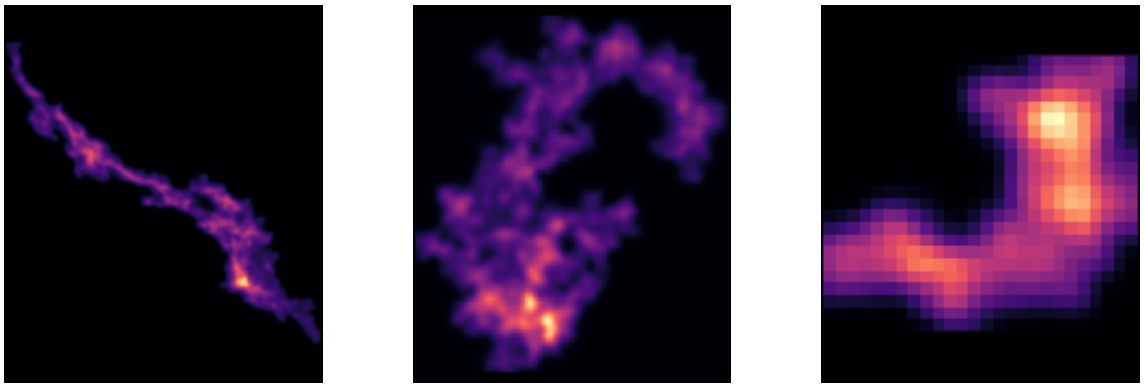
Figur 3.8a visar en binär mask för ett moln, där molnets form är tydligt urskiljbart på bekostnaden av inre detaljer. Varje pixel i figuren har antingen värdet 1 eller 0 baserat på huruvida pixelns intensitetsvärde översteg 0 och motsvaras av färgerna svart respektive vitt i figuren. Resterande figurer 3.8b och 3.8c visar båda icke-binära bildhanteringar där molnets intensitetsfördelning går att urskilja med hjälp av olika färgskalorna. Bilden i figur 3.8c har däremot genomgått ytterligare en Gaussisk utjämning för att ytterligare jämna ut datan och visualiseras genom en kontinuerlig färgskala, till skillnad från den diskreta färgskalan i figur 3.8b, för att lättare urskilja mönster inuti molnet. Det är just bildstrukturen i figur 3.8c som kommer appliceras för maskininlärningsprocessen.

3.3.3 Klassuppdelning och bildannotering

Eftersom YOLOv8 bygger på övervakad maskininlärning krävdes att klassificeringen av molnen skedde innan inlärningen. Detta resulterade i två huvudgrupper av klasser: en för molnens generella form och sammansättning, och en för molnens inre struktur, till exempel toppar i intensitet.

3.3.3.1 Molnens form och sammansättning

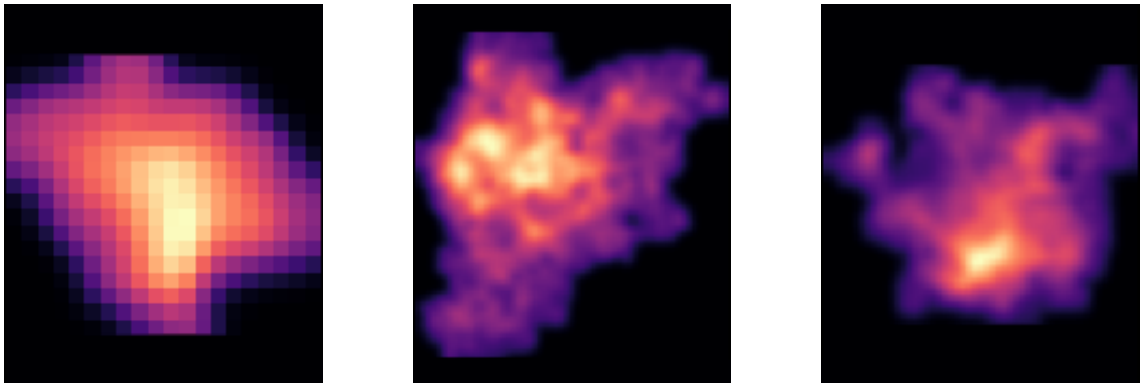
Utifrån formen och sammansättningen på molnen skapades tre stycken underklasser: *Filament*, *Samlingar*, och *Kluster*. Den första av dessa, *Filament*, innehöll moln med trådliknande egenskaper. För att förtydliga denna klass och skilja den från resterande valdes att moln med utstickande trådelement värdesattes högt för *Filament*. Några moln som ansågs tillhöra denna klass visas i figur 3.9 nedan.



(a) **(b)** **(c)**

Figur 3.9: Exempel på tre moln som placerades inom klassen *Filament*. I (a) visas ett moln med en fiberlik trådstruktur medan (b) och (c) visar ett något tjockare moln med viss vridning.

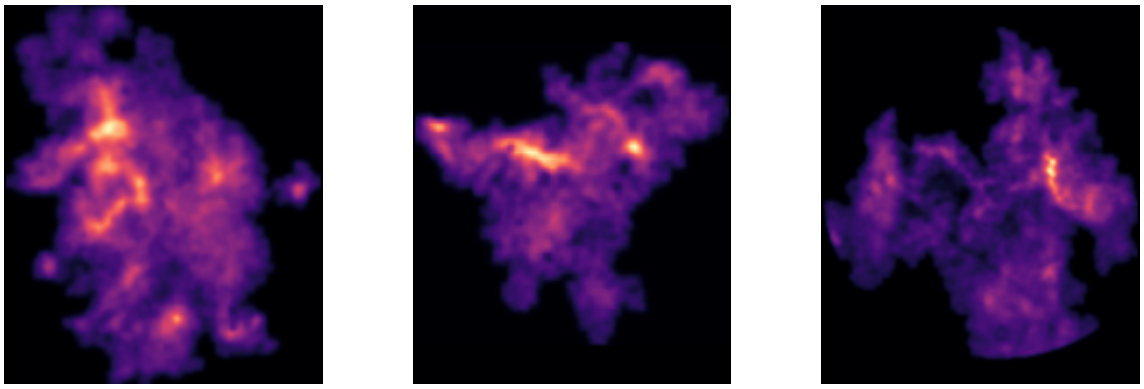
Nästa underklass, *Samlingar*, innehöll moln med en huvudsakligt centrerad massfördelning. Detta inkluderade främst moln utan utstickande delar som var rundare i form. I figur 3.10 nedan visas några exempel på moln tillhörande denna klass. Generellt klassades moln som inte hade fler än en topp till denna klass, förutom då dessa toppar befann sig mycket nära varandra. Ett exempel på detta undantag kan ses i figur 3.10b nedan.



(a) **(b)** **(c)**

Figur 3.10: Exempel på tre moln som placerades inom klassen *Samlingar*. I (a) visas ett relativt cirkulärt moln med en tydlig topp. I (b) och (c) visas mer utspridda moln där en tydlig mittpunkt fortfarande kan urskiljas.

Den tredje och sista underklassen i denna klassificering var *Kluster*. Till denna klass hörde moln som ansågs se ut som flera sammansatta moln, till exempel de med flera utspridda toppar eller oregelbundna former som inte passade in i *Filament*. De moln som varken passade in i *Filament* eller *Samlingar* kategoriserades även i denna klass. Denna klass agerade alltså delvis även som en kategori för moln som skulle kunna beskrivas som övriga. Nedan i figur 3.11 visas några exempel på moln tillhörande denna klass.



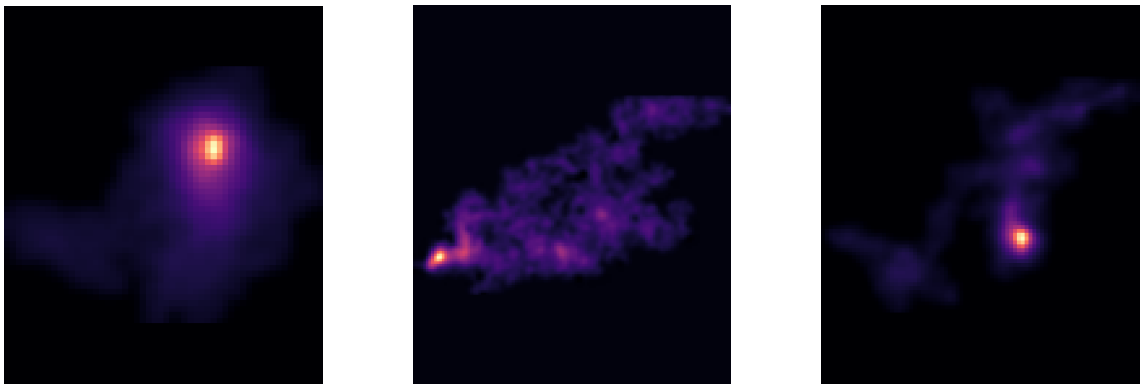
(a) **(b)** **(c)**

Figur 3.11: Exempel på tre moln som placerades inom klassen *Kluster*. I (a) och (b) visas moln med två eller flera relativt tydliga toppar med större avstånd mellan varandra. I (c) syns endast en tydlig topp men molnet har ingen tydlig mittpunkt på grund av dess utspridning.

3.3.3.2 Molnens inre struktur

Den andra huvudgruppen som undersöktes byggde på inre intensitetsskillnader. Molnen klassificerades utifrån relativa intensitetskillnaderna inom respektive moln, vilket innebär att intensitetsskillnaderna inte är jämförbara mellan olika moln. För kommande beskrivningar samt klassnamn kommer höjdskillnader beskrivas istället för intensitet. Exempel på detta är hur en mycket intensiv punkt räknas som hög kontra hur en lågintensiv del av molnet räknas som låg.

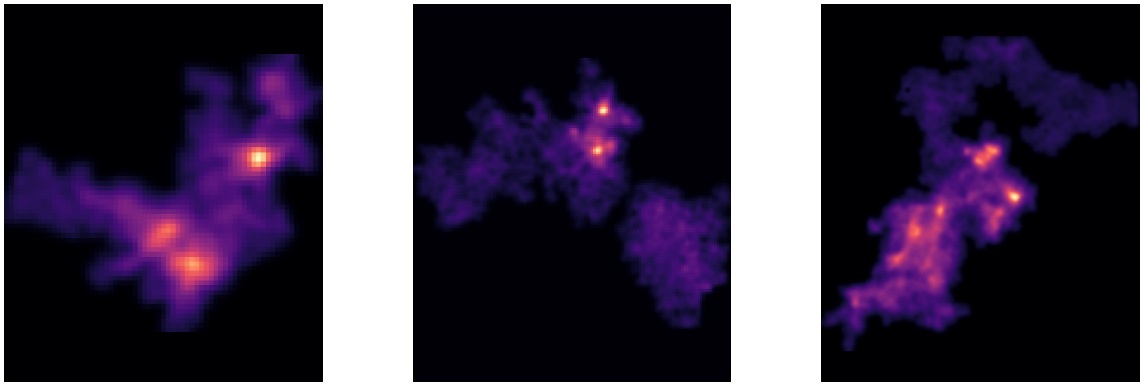
Den första underklassen kallades för *Low_one_peak*. Dessa var moln med en enskild tydlig topp, där resterande delar av molnen hade låga intensiteter relativt denna topp. Exempel är givna i figur 3.12.



(a) **(b)** **(c)**

Figur 3.12: Exempel på moln som klassades som *Low_one_peak*.

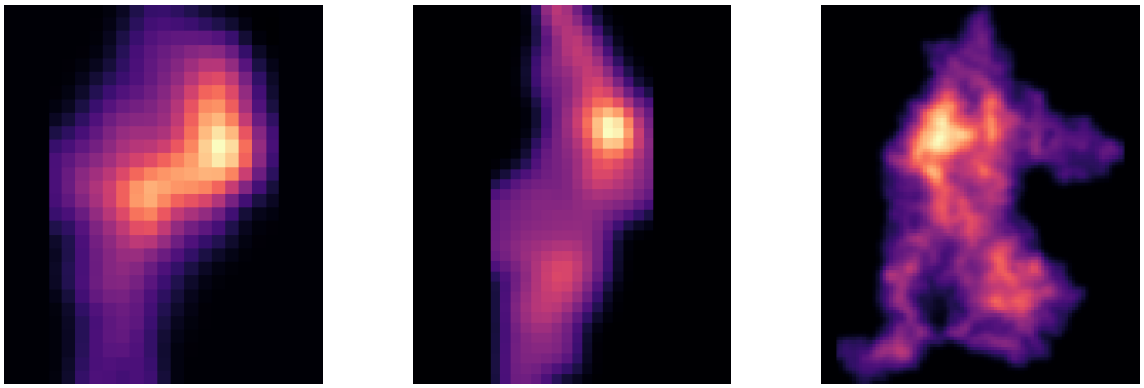
Den andra underklassen kallades för *Low_with_peaks*. Moln i denna klass liknar de i *Low_one_peak*, förutom att flera tydliga toppar behöver förekomma istället för endast en. Se exempel för klassen i figur 3.13.



(a) **(b)** **(c)**

Figur 3.13: Exempel på moln som klassades som *Low_with_peaks*.

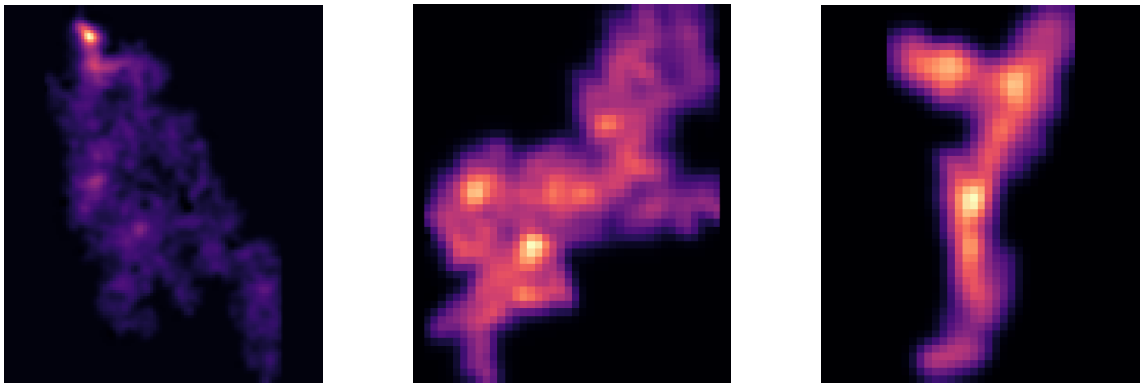
Den tredje underklassen kallades för *High_one_peak*. Dessa moln hade också en tydlig topp. Dock till skillnad från klassen *Low_one_peak* hade resterande delar av molnen fortfarande hög intensitet i relation till denna topp. Exempel visas i figur 3.14 nedan.



(a) **(b)** **(c)**

Figur 3.14: Exempel på moln som klassades som *High_one_peak*.

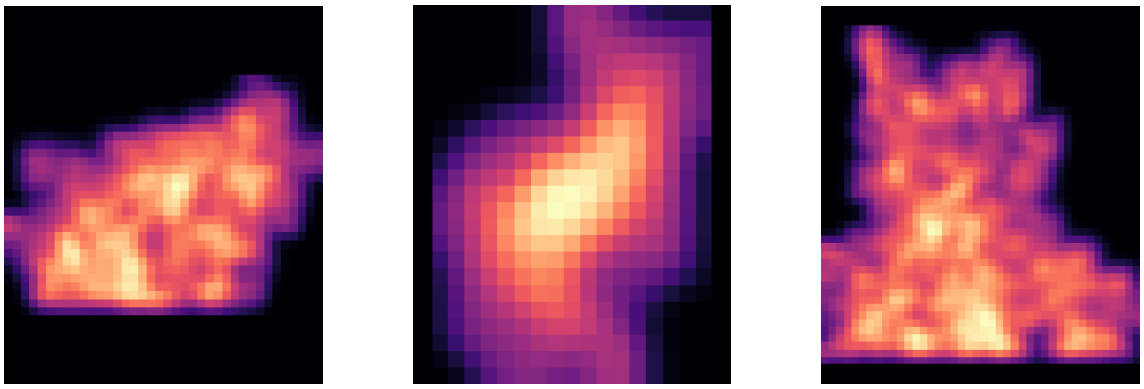
Den fjärde underklassen kallades för *High_with_peaks*. Beskrivningen för denna klass är densamma som för *High_one_peak*, med skillnaden att flera tydliga toppar var närvarande. I figur 3.15 ges exempel.



(a) **(b)** **(c)**

Figur 3.15: Exempel på moln som klassades som *High_with_peaks*.

Den femte och sista underklassen kallades endast för *High*. Denna underklass innehöll de moln vars generella höjd är hög och där det inte finns några relevanta utstående toppar. För exempel, se figur 3.16.



(a) **(b)** **(c)**

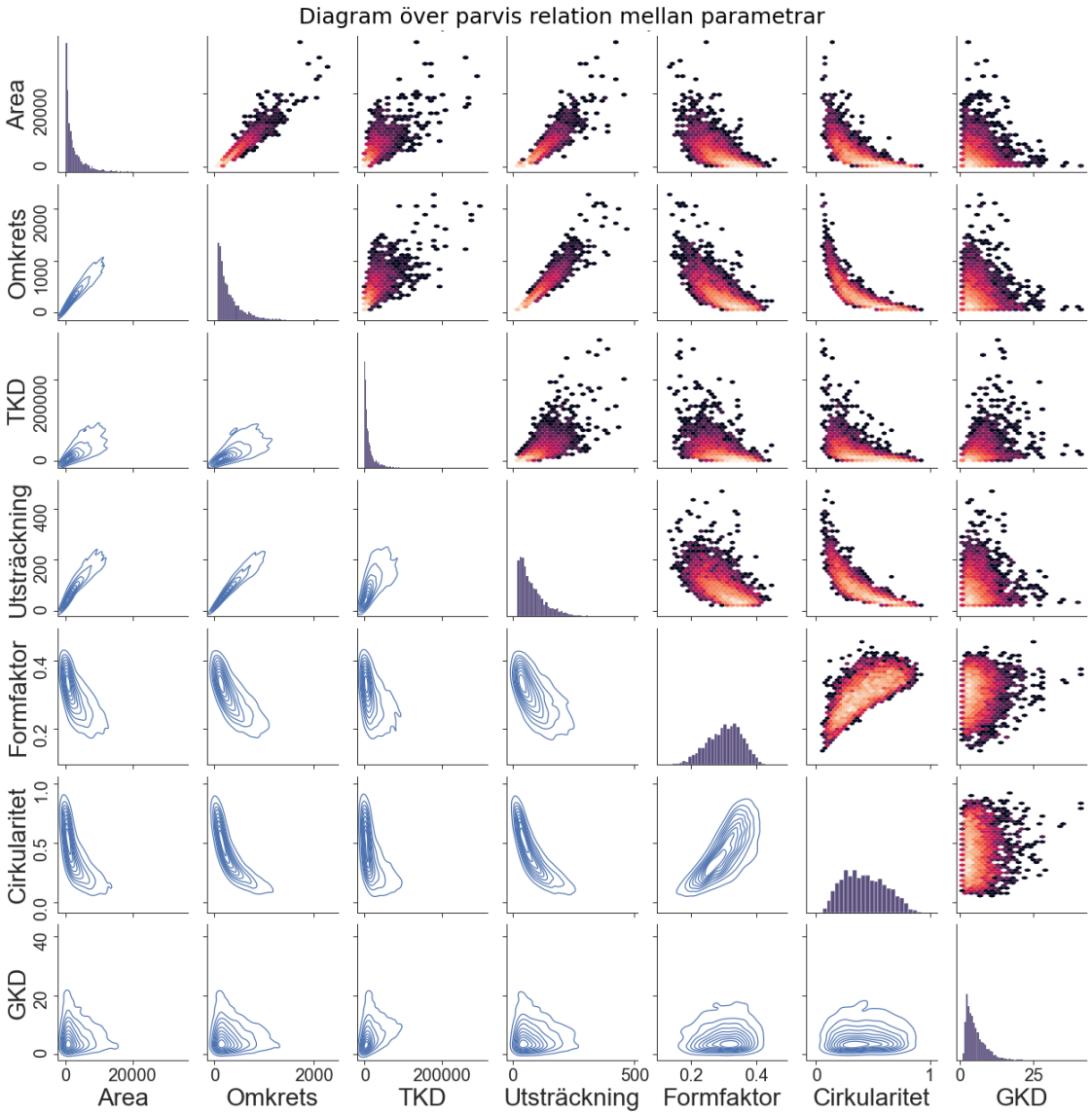
Figur 3.16: Exempel på moln som klassades som *High*.

4
Resultat och diskussion

Nedan följer erhållna resultat från de två separata tillvägagångssätten för försök till klassificering beskrivna ovan, samt diskussion angående dessa resultat.

4.1 Bildanalysmetoder

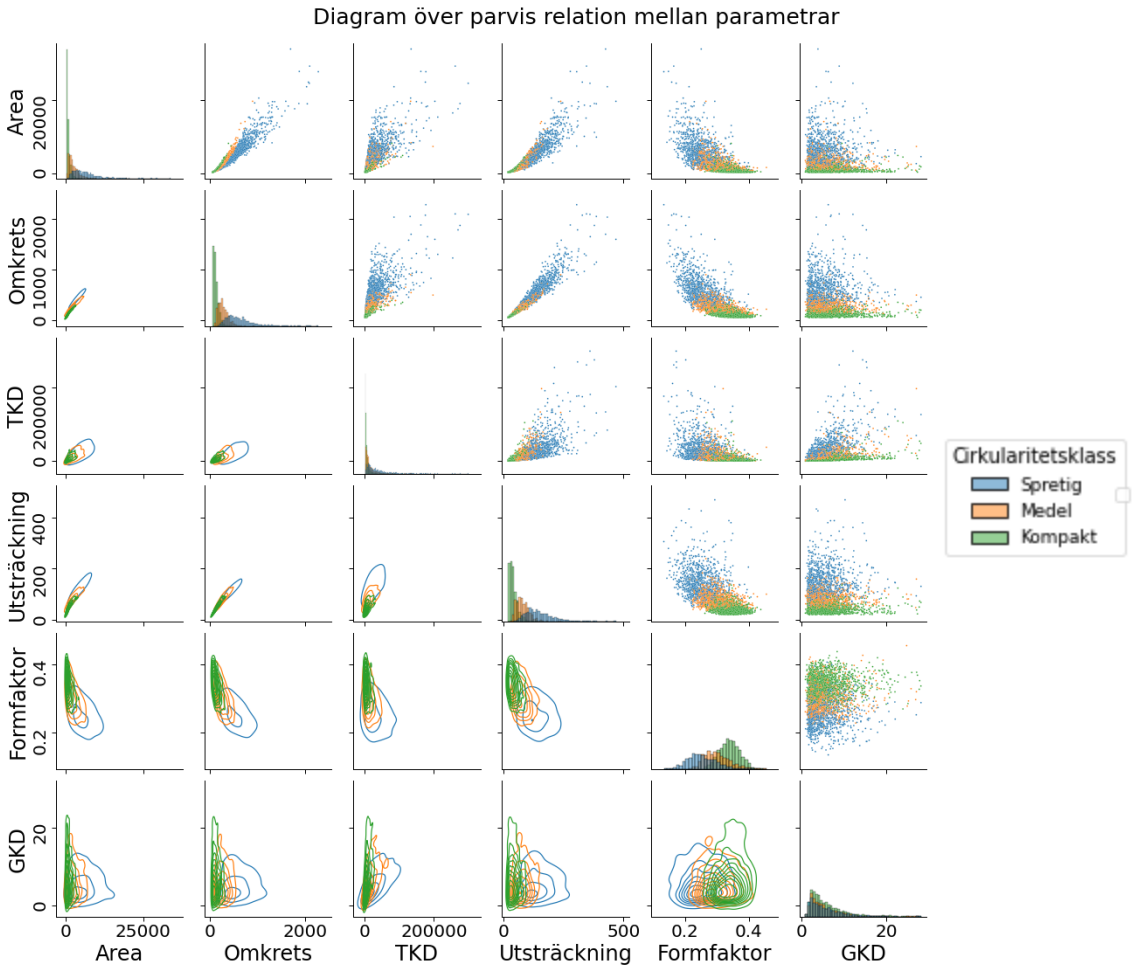
Sammanställning av de resultat som erhölls kan ses i figur 4.1. Däri kan fördelningarna ses för de olika parametrarna, samt parvisa jämförelser mellan dem. Som förväntat korrelerar area, omkrets, total kolumndensitet och utsträcking starkt positivt med varandra. Detta är rimligt eftersom alla dessa borde korrelera med större moln. Vidare korrelerar både formfaktorn och cirkulariteten negativt mot dessa parametrar. Detta kan dock delvis förklaras som ett resultat av bildhanteringsmetoden. Exempelvis har erosion använts för att jämna ut, ta bort, och separera små områden. Denna process påverkar inte större, jämna objekt så mycket, men kan helt ta bort tunna filamentdelar. Då förekomsten eller förlusten av en smal arm påverkar formen hos ett litet moln mer än ett stort kommer detta generellt leda till att mindre moln tycks utjämnas mer än stora, vilket påverkar bland annat deras cirkulariteter och formfaktorer. Detta kan vara en bidragande faktor till varför dessa parametrar korrelerar negativt. Cirkulariteten och formfaktorn korrelerar även positivt med varandra, en bidragande orsak kan dock vara att båda korrelerar med molnstorlek. Den genomsnittliga kolumndensiteten GKD saknar korrelation med någon annan parameter.



Figur 4.1: I figuren presenteras de parvisa förhållandena mellan parametrarna. På axlarna finns, i ordning uppifrån och ner på y-axeln och från vänster till höger på x-axeln, variablerna area, omkrets, total kolumndensitet TKD, utsträckning, formfaktor, cirkularitet och genomsnittlig kolumndensitet GKD. På diagonalen finns histogram över varje parameters interna fördelning, och på sidorna om den presenteras datan i form av densitetsdiagram. Den övre halvan är ett hexagonalt spridningsdiagram där högre förekomst representeras av ljusare färg, och den undre är en så kallad KDE-plot, som presenterar datan i formen av en kontinuerlig sannolikhetsdensitet. Enstaka extrempunkter har tagits bort för att förbättra visibiliteten.

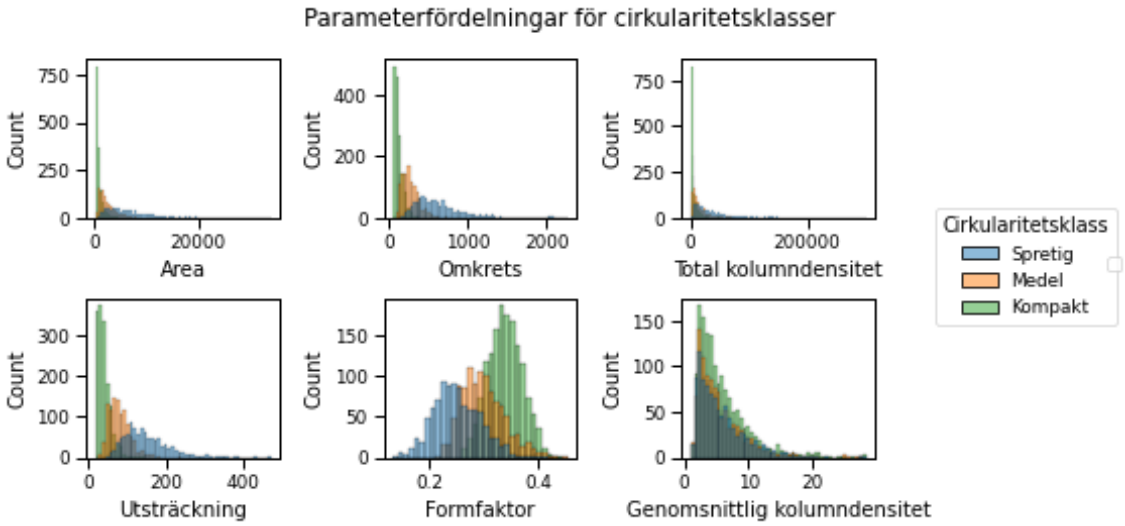
Utifrån de resultat som erhållits är det svårt att bestämma tydliga klasser, då det inte finns någon omedelbar gruppering utan snarare en kontinuerlig fördelning. Det är möjligt att subjektivt göra gränsdragningar och bestämma klasser utifrån observationer; till exempel går det att utifrån cirkulariteten dela in molnen i klasserna *Spretig*, halvkompakt eller *Medel*, och *Kompakt*, se figur 4.2. Fördelningen för cirkulariteten i figur 4.1 är kontinuerlig utan tydliga avbrott. Gränserna, som i figur

4.2 dras vid $C = 0{,}29$ och $C = 0{,}45$, baseras där inte på data utan bestämdes genom att observera cirkularitetsvärdena för ett hundratal moln. I figuren kan man se att dessa cirkularitetsklasser till stor del överlappar för de flesta parametrar, utöver genomsnittlig kolumndensitet där det är en jämn fördelning mellan klasserna.



Figur 4.2: I figuren presenteras de parvisa förhållandena mellan parametrarna, med färgkodning utifrån cirkularitetsklass där blå är *Spretig*, orange är *Medel* och grön är *Kompakt*. På axlarna finns, i ordning uppifrån och ner på y-axeln och från vänster till höger på x-axeln, variablerna area, omkrets, total kolumndensitet TKD, utsträckning, formfaktor, cirkularitet och genomsnittlig kolumndensitet GKD. På diagonalen finns histogram över varje parameters interna fördelning, och på sidorna om den presenteras datan i form av spridningsdiagram. Enstaka extrempunkter har tagits bort för att förbättra visibiliteten.

Spridningen är dock större för parametrarna hos spretigare moln, vilket kan ses i figur 4.3 där histogram över samtliga parametrars fördelning visas med färgkodning enligt cirkularitetsklass. Det går att urskilja en trend vad gäller parametrarna som berör storlek, då mindre moln klassas som kompakta till betydligt högre grad än större moln.



Figur 4.3: Histogram över parametrarna area, omkrets, total kolumndensitet, utsträckning, formfaktor och genomsnittlig kolumndensitet med färgsättning beroende på deras cirkularitetsklass, där blått är *Spretig*, orange är *Medel* och grönt är *Kompakt*.

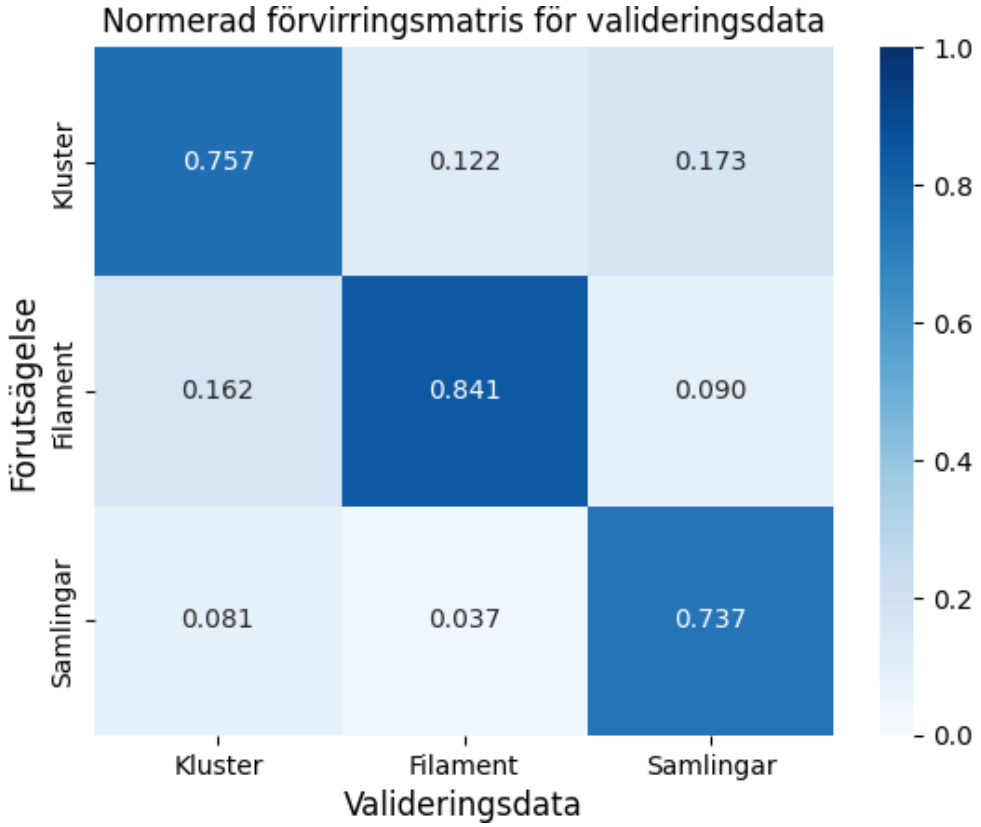
4.2 Maskininlärning

Den andra metoden som undersöktes för klassificering var maskininlärningen. Denna metod utfördes för de två olika klassystemen av moln givna i delkapitel 3.3.3. Resultaten från detta, diskussion rörande dem samt maskininlärningens användbarhet som ett klassificeringsverktyg redovisas i detta avsnitt.

4.2.1 Molnens form och sammansättning

För att undersöka maskininlärningen på molnens form och sammansättning valdes den förtränade modellen YOLOv8s-cls att användas. Denna modell valdes efter prövning av de olika modellerna som Ultralytics erbjuder där överanpassning av datan försökte undvikas och träffsäkerheten hållas så hög som möjligt. Under annoteringen placerades totalt 1602 moln i klassen *Filament*, 770 moln i klassen *Samlingar* och 1113 moln i klassen *Kluster*. Detta motsvarar ungefär 45,8%, 22,0% respektive 31,8% av alla framtagna moln. Av dessa moln avsattes cirka 40% av datan jämnt mellan validerings- och testdata medan resterande 60% blev modellens träningsdata. Träningen resulterade i en förvirringsmatris enligt figur 4.4 där modellens förutsägelser jämförs med den annoterande valideringsdatan. Matrisen visar till exempel hur modellen gissar när den förutsäger ett moln av en specifik klass. Högre värden på modellens träffsäkerhet motsvaras där och i resten av rapporten av en mörkare blå färg. Modellen tränades på träningsdata bestående av 2099 bilder under 20 epoker för att inte överträna modellen och riskera överanpassning. För klassen *Kluster* lyckades modellen förutsäga rätt klass cirka 75,7% av gångerna medan den misstar molnen för *Filament* och *Samlingar* med ungefär 16,2% respektive 8,1% träffsäkerhet. För *Filament* gissade modellen rätt cirka 84,1% av tiden medan den har fel

ungefär 12,2% och 3,7% av gångerna då den gissade på *Kluster* respektive *Samlingar*. Till sist för *Samlingar* gissar modellen rätt ungefär 73,7% av gångerna medan den tror att molnen tillhör *Kluster* och *Filament* cirka 17,3% respektive 9,0% av tiden. Modellens totala träffsäkerhet hamnade på ungefär 79,7% för hela valideringsdatan.



Figur 4.4: Förvirringsmatris som visar resulterande modell på den inmatade valideringsdatan för inlärningsprocessen. Detta visar att modellen gissar rätt ungefär 75,7%, 84,1%, respektive 73,7% av gångerna för klasserna *Kluster*, *Filament*, och *Samlingar* på den användna valideringsdatan.

Det finns däremot problem med att visualisera datan endast på detta sätt. Förvirringsmatrisen har nackdelen att en gissning där modellen är 51% säker på sitt svar resulterar i ett absolut svar oavsett om nästkommande gissning ligger på 49%. På denna data har modellen gjort sin säkraste gissning med under 65% strax under en sjättedel av gångerna, cirka 15,5%. Från andra hållet har modellens bästa gissning legat över 90% ungefär 59,3% av tiden. Modellen innehåller alltså en del osäkerheter men verkar överlag vara relativt säker i sina gissningar.

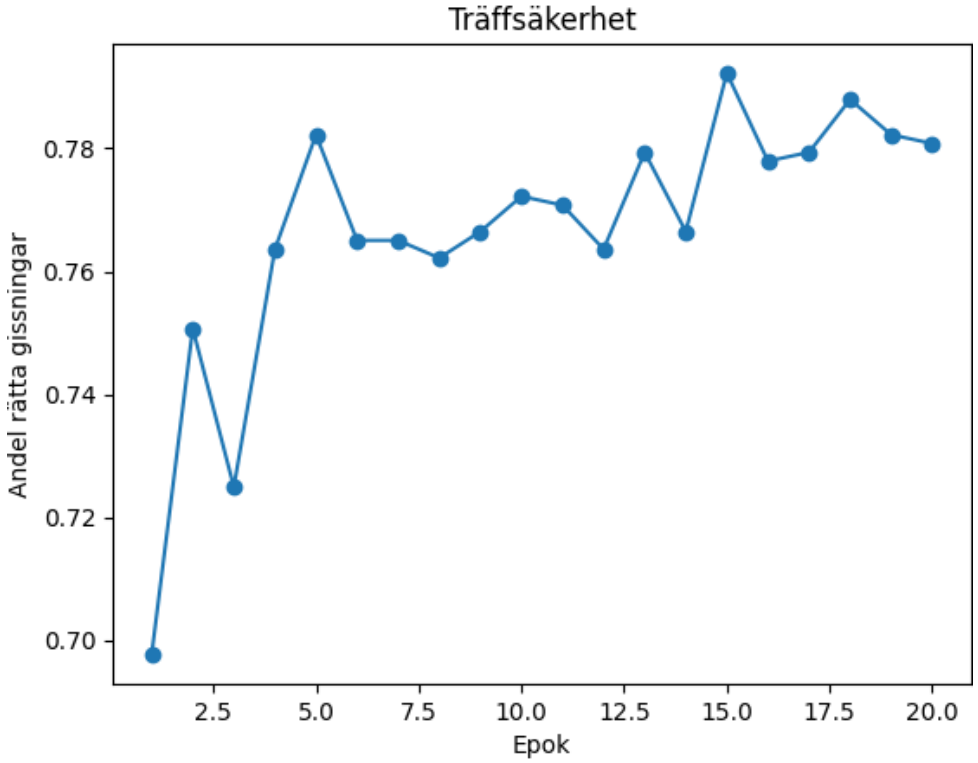
Med hjälp av den tidigare förvirringsmatrisen i figur 4.4 kan modellens 'precision' samt 'recall' beräknas enligt ekvationer 3.3.1a och 3.3.1b. I valideringsdatan var det totala antalet moln 222, 320, och 156 för *Kluster*, *Filament*, respektive *Samlingar*. Detta gav en 'precision' på 71,8%, 84,3%, och 79,3% samt 'recall' på 75,7%, 84,1%, och 73,7% för respektive klass vilket visas nedan i tabell 4.1. Modellens 'precision' ger att säkerheten då den gissar på antingen *Filament* eller *Samlingar* är högre än för *Kluster*, om den gissar på någon av dessa klasser är det alltså mycket troligt att

den har rätt. För 'recall' gäller att modellen lyckas identifiera en stor mängd av alla moln inom klassen *Filament* men lyckas sämre för de andra klasserna. Detta ger att modellen överlag fungerar väldigt bra för *Filament* men tappar något gentemot *Kluster* och *Samlingar*.

Tabell 4.1: Värden på maskininlärningsmodellens 'precision' samt 'recall' för samtliga underklasser.

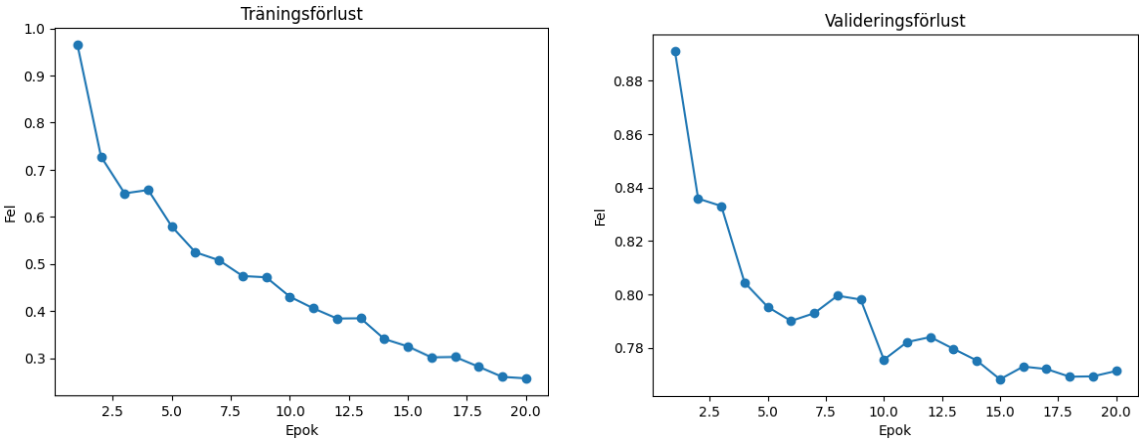
Klass	Precision	Recall
Kluster	0,718	0,757
Filament	0,843	0,841
Samlingar	0,793	0,737

För att evaluera en maskininlärningsmodell som denna krävs även att data redovisas under träningsprocessen. I figur 4.5 visas modellens gissningssäkerhet för varje epok. Denna data visar ett fluktuerande men ändå ökande värde på modellens träffsäkerhet, där den femtonde epoken motsvarar modellens maximum. Träffsäkerheten verkar öka som mest i början och sedan plana ut något för de senare epokerna.



Figur 4.5: Graf över modellens träffsäkerhet för varje epok som den tränades. Träffsäkerheten fluktuerar mycket under de första epokerna, men ökar överlag tills den når sitt högsta värde vid den femtonde epoken med ungefär 80,4%.

Även tränings- och valideringsförlusterna som visas i figur 4.6 är av intresse att undersöka. I figur 4.6a visas en tydlig minskning i träningsfelet för varje epok. För valideringsförlusterna i figur 4.6b är detta något otydligare då felet avviker något vid mitten av träningen och den verkar dessutom plana av mot slutet av träningen men överlag sjunker den. Skillnaderna mellan förlusterna är inte alltför stor och båda två har avtagit fram tills den slutliga epoken vilket tyder på att modellen inte är alltför överanpassad. Detta innebär däremot att modellen inte är underanpassad.

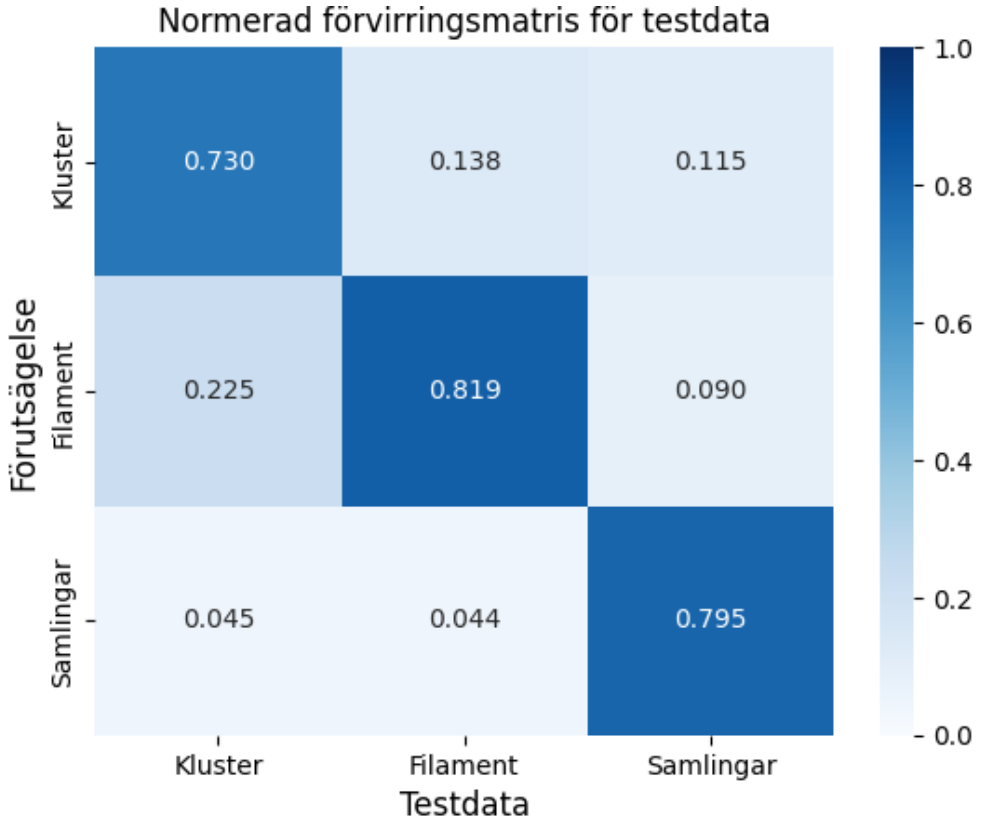


(a) Stadigt sjunkande träningsförluster under inlärningsprocessen.

(b) Överlag sjunkande valideringsförluster men fluktuerande värde under mitten av träningen.

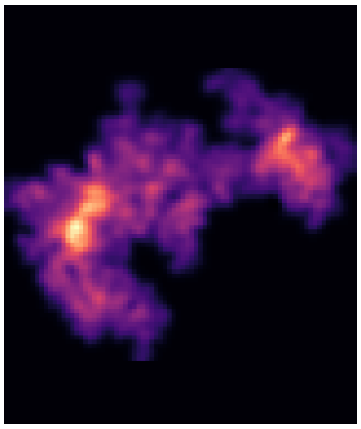
Figur 4.6: Grafer över modellens tränings- samt valideringsförluster under träningen.

Modellen testades sedan på den okända testdatan som ej varit inblandad i inlärningsprocessen. Detta resulterar i en till förvirringsmatris enligt figur 4.7. Testdatan innehöll 698 moln vilket var samma antal som i valideringsdatan. För *Kluster*, *Filament*, och *Samlingar* gissade modellen rätt cirka 73,0%, 81,9%, respektive 79,5% av gångerna. Dessa resultat liknar de tidigare för valideringsdatan, där modellen är lite sämre på att separera vissa klasser från varandra. Till exempel gissar modellen på *Filament* ungefär 22,5% av gångerna då molnet faktiskt tillhör *Kluster*. Överlag uppvisar däremot modellen lovande tecken på att den generaliserar bra, då träffsäkerheten på den nya datan är relativt hög och liknar den för valideringsdatan. Modellen är som tidigare mest säker på klassen *Filament*.

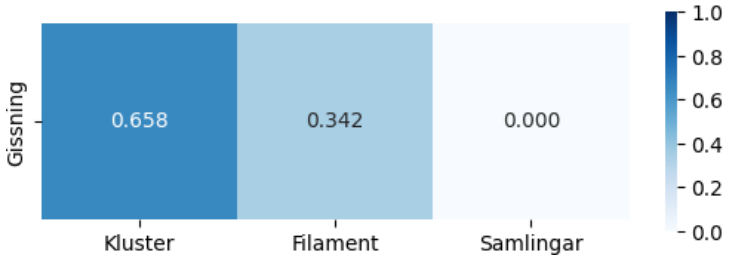


Figur 4.7: Förvirringsmatris som visar den slutgiltiga modellens förutsägelse på ny testdata med klasserna *Kluster*, *Filament*, och *Samlingar*. Detta visar att modellen gissar rätt med ungefär 73,0%, 81,9%, respektive 79,5% av gångerna för dessa klasser på den nya datan, med liknande mönster som i figur 4.4 med avseende på felaktiga gissningar.

Som nämnt tidigare finns det nackdelar att endast undersöka resultaten med hjälp av förvirringsmatriser. Det kan alltså vara av intresse att undersöka enskilda moln för att kontrollera hur säker modellen är på sina egna gissningar. I figur 4.8 nedan visas ett exempel på när modellen är osäker på vilken klass ett moln borde tillhöra.



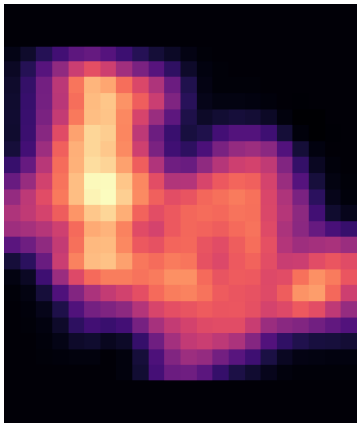
(a) Inmatad bild av moln för modellens förutsägelse.



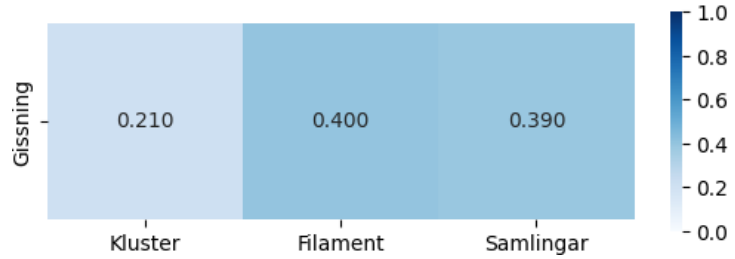
(b) Modellens gissningar på vilken klass molnet bör tillhöra.

Figur 4.8: Resultat av förutsägelse av ett moln som var del av testdatan för maskininlärningsmodellen. Modellen ger sannolikheterna 65,8% och 34,2% att molnet tillhör *Kluster* respektive *Filament*, samt 0,0% för *Samlingar*.

Figuren visar att modellen gör gissningen att molnet tillhör *Filament* respektive *Kluster* med sannolikheterna 65,8% och 34,2% samt en sannolikhet 0% att det tillhör *Samlingar*. Modellen har alltså lite svårt att avgöra huruvida molnet bör tillhöra *Kluster* eller *Filament*. Om annotering skulle göras för detta moln skulle det förmodligen hamna under klassen *Kluster* eftersom det är en för bred struktur för *Filament*. Däremot kan en mer avlång form hittas i molnet vilket kan ge denna osäkerhet även i annoteringen. Ytterligare ett exempel på ett moln som modellen visade sig osäker på visas nedan i figur 4.9.



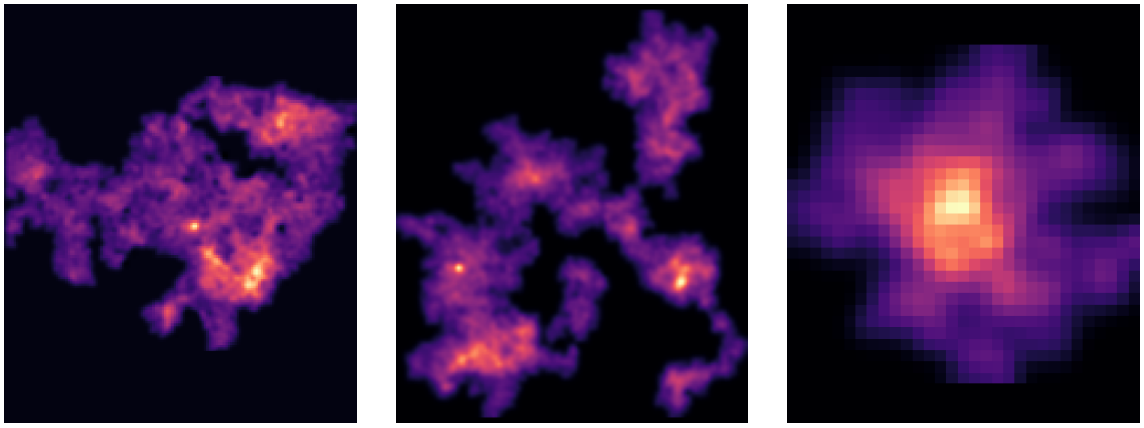
(a) Inmatad bild av moln för modellens förutsägelse.



(b) Modellens gissningar på vilken klass molnet bör tillhöra.

Figur 4.9: Resultat av förutsägelse av ett moln som var del av testdatan för maskininlärningsmodellen. Modellen ger sannolikheterna 21,0% respektive 40,0% att molnet tillhör *Kluster* respektive *Filament*, men 39,0% för *Samlingar*.

I denna figur visas igen modellens svårighet att bedöma vilken klass ett moln bör tillhöra men den här gången är gissningarna på alla klasser relativt stora. Modellen gissar att molnet bör tillhöra *Kluster*, *Filament*, och *Samlingar* med sannolikheterna 21,0%, 40,0%, respektive 39,0%. Förmodligen skulle detta moln klassificeras som *Samlingar* vid annotering. Detta visar på bristerna i förvirringsmatriserna där i detta fall detta moln skulle klassas som *Filament* och modellens tydliga osäkerhet blir ignorerad. Moln som dessa visar återigen osäkerheter i modellen och kan innebära att det finns brister i själva annoteringsprocessen för molnets form och sammansättning. I annoteringen behövde molnen delas upp manuellt till de olika klasserna och under denna process förekom osäkerhet i uppdelning som verkar speglas i modellen. Mängden moln i varje klass kan även vara problematiska för inlärningen. I detta fall innehåller en klass, *Filament*, betydligt fler moln än resterande klasser vilket kan orsaka problem i hur säker modellens träffsäkerhet är. Med denna imbalans i datamängden kan modellen helt enkelt sakna den datagrund den behöver för att klassificera dessa moln med tillräckligt säkert.



(a) Moln som klassats med 96,5% sannolikhet som *Kluster* enligt modellen.

(b) Moln som klassats med 100,0% sannolikhet som *Filament* enligt modellen.

(c) Moln som klassats med 99,2% sannolikhet som *Samlingar* enligt modellen.

Figur 4.10: Exempel på tre moln som placerats inom de tre klasserna med hög sannolikhet.

Denna modell verkar däremot vara väldigt säker på vissa typer av moln. I figur 4.10 nedan visas tre exempel på moln där modellen anser sig väldigt säker på vilken klass respektive moln bör tillhöra. I figurer 4.10a, 4.10b, och 4.10c visas moln som modellen har klassat som *Kluster*, *Samlingar*, och *Filament* med 96,5%, 100,0%, respektive 99,2% säkerhet. Dessa moln är alltså typexempel på vad modellen anser tillhöra de olika klasserna inom denna kategori.

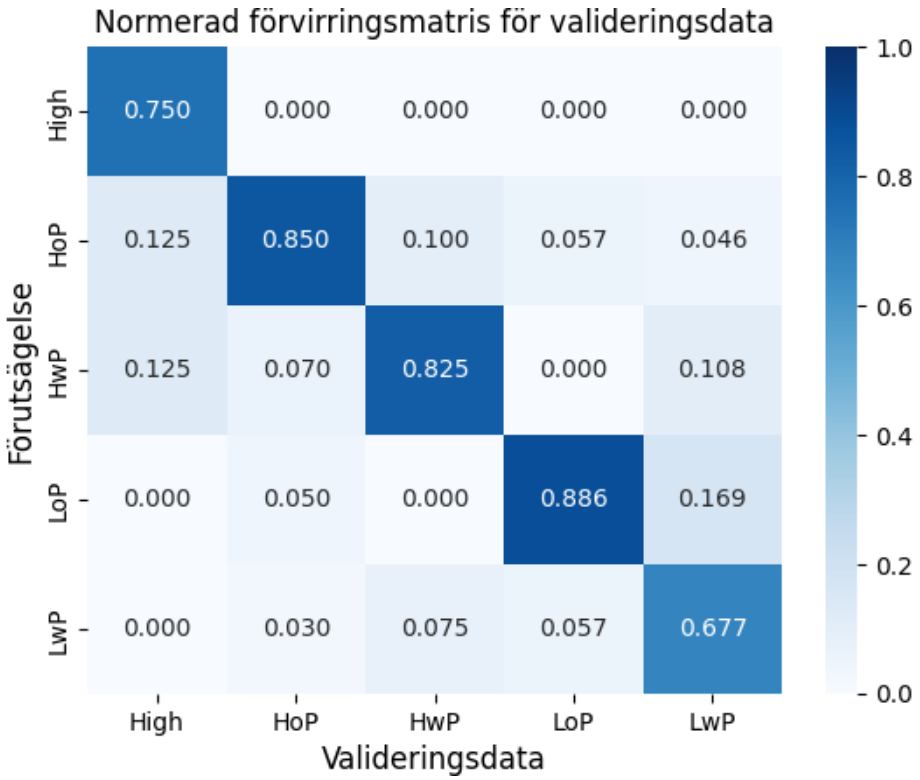
4.2.2 Molnens inre strukturer

Maskininlärningsmodellen för klassificering av molnens inre struktur tränades på Ultralytics *YOLOv8x-cls* modell. Modellen tränades över 30 epoker med ungefär 2800 bilder. Detta resulterade i att modellen kunde klassificera molnen med bäst träffsäkerhet på ungefär 82% för epok 29 (se tabell 4.2).

Tabell 4.2: Bästa resultatet uppnått med *YOLOv8x-cls*, vilket erhölls under epok 29 med en säkerhet på 81,967%. Träningsförlust och valideringsförlust för modellen gav värdena 0,15491 respektive 1,0974.

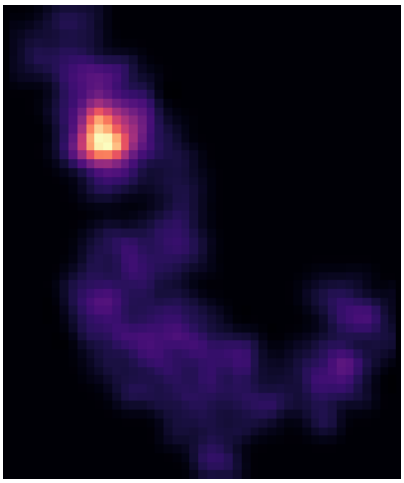
Epok	Träffsäkerhet klassificering	Träningsförlust	Valideringsförlust
29	0,81967	0,15491	1,0974

För slutgiltigt resultat för den tränade modellen, se förvirringsmatrisen i figur 4.11. Förvirringsmatrisen visar på hur modellen klassificerat bilder av en viss klass, och därmed sannolikheten att modellen gissat rätt på molnets klass. Modellen gissade rätt på moln i klassen *High* med en träffsäkerhet på 75%. För klasserna *High_one_peak* och *High_with_peaks* gav modellen en träffsäkerhet på 85% respektive 82%. Sist så gissade modellen rätt på moln i klassen *Low_one_peak* med en träffsäkerhet på 89%, samt med en träffsäkerhet på 68% för klassen *Low_with_peaks*. Prestandamåtten 'precision' och 'recall' räknas ut med hjälp av resultatet från figur 4.15, och presenteras i tabell 4.3.



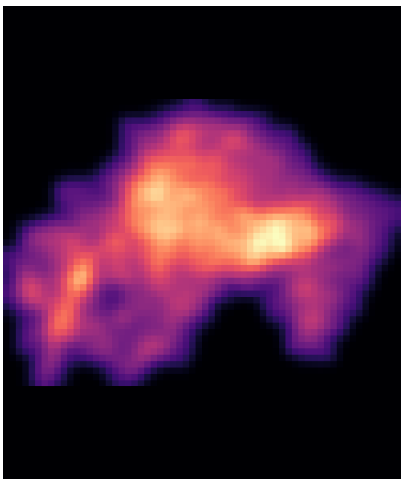
Figur 4.11: Normaliserad förvirringsmatris som visar resultatet av klassificeringsmodellen för molnens inre struktur. Förkortningarna HoP, HwP, LoP och LwP står för *High_one_peak*, *High_with_peaks*, *Low_one_peak* respektive *Low_with_peaks*.

Modellen testades på nya bilder för att se hur den hanterar okänd data. Den testades på molnet i figur 4.12, ett moln som ska klassas som *Low_one_peak*. Modellen gav att det var 100% sannolikhet att molnet var av klassen *Low_with_peaks*.



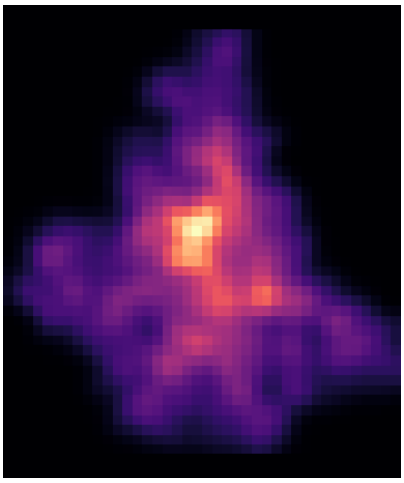
Figur 4.12: Molnet i figuren ska klassas som *Low_one_peak*.

Modellen testades också på molnet i figur 4.13. Modellen gav sannolikheten att molnet klassades som *High_one_peak* var 59%, *High_with_peaks* var 40% och *High* var 1%. Molnet klassades som *High_with_peaks* vid manuell annotering.



Figur 4.13: Molnet i figuren ska klassas som *High_with_peaks*.

Ett sista moln som testade finns i figur 4.14. Modellen gav att sannolikheten för att molnet klassas som *Low_one_peak* var 79% och *High_one_peak* var 21%. Molnet ska klassas som *High_one_peak*.



Figur 4.14: Molnet i figuren ska klassas som *High_one_peak*.

Resultatet visar något viktigt: modellen har svårt att urskilja klasser som liknar varandra. Anledningen till detta kan bero på flera olika saker. Det första man bör undersöka gällande resultatet från en tränad klassificeringsmodell är hur klasserna för träningsdatan definierades, samt hur varje enskilt moln klassificerades. I fallet för denna modell klassificerades alla moln för hand av en person. Detta kan leda till problem såsom felklassificering och subjektivitet, och kan därmed påverka resultatet.

Något annat man bör undersöka är hur träningsdatan som användes ser ut. I fallet för denna modell kunde skillnader mellan olika molnbilder vara minimala men ändå placeras i två olika klasser, ett direkt resultat av subjektivitet från personen som klassificerade bilderna. Tyvärr kan detta vara svårt att hantera om man inte väljer att använda sig av och jämföra flera olika personers klassificering av samma data, något som kan vara både svårt, tidskrävande och kostsamt.

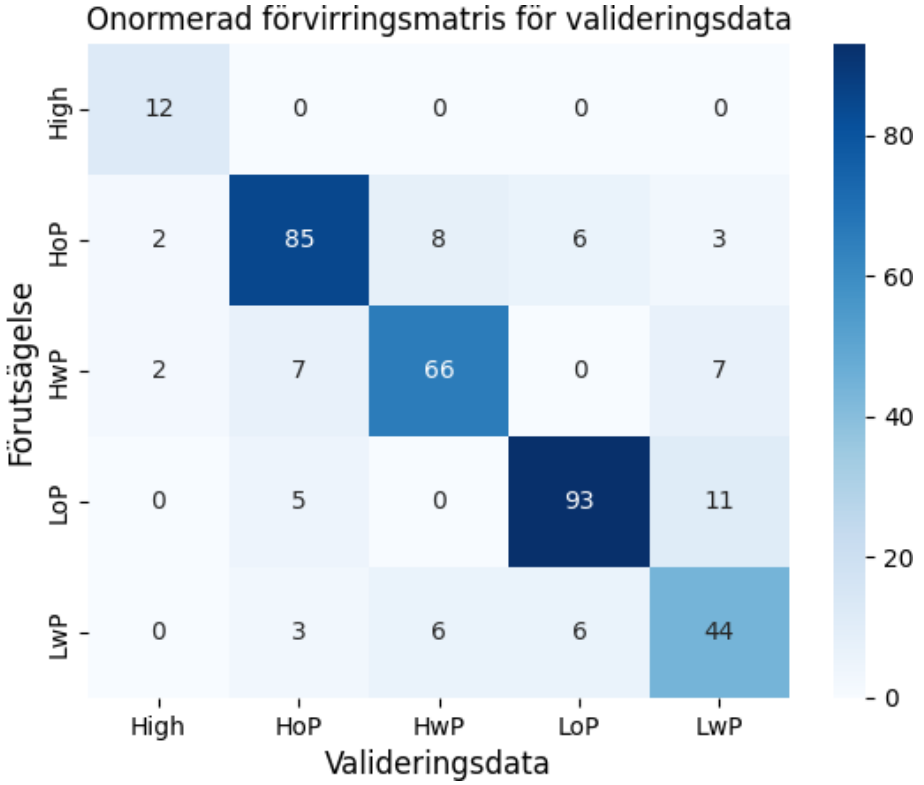
Gällande datan så påverkas modellen också av hur mycket träningsdata, alltså antal molnbilder, som fanns för varje klass. För modellen som klassificerade molnens inre struktur så fanns det en skillnad på flera hundra bilder mellan olika klasser i träningsdatan. Den obalanserade uppdelningen av datan för varje klass är lik den som nämns i resultatet för molnens form och sammansättning, och resulterar också i liknade problem. Träningsdatan för klassen *High* innehåller 151 bilder jämfört med klassen *Low_one_peak* som innehåller 1188 bilder. Obalanserad data kan resultera i 'Accuracy Paradox', att en hög träffsäkerheten inte betyder att modellen är effektiv och bra [19]. Därför bör man även titta på andra prestandamått och resultatredovisningar, som exempelvis en förvirringsmatris och dess 'precision' samt 'recall' värden.

Resultatet från tabell 4.3, som visar att alla klasser har en relativt hög 'precision', tyder på att modellen ofta gissar på rätt klass utav alla bilder som den anser ingå i samma klass. Resultatet för 'recall' tyder på modellen har svårare att hitta alla moln som ska tillhöra en specifik klass. Det är inte orimligt att 'precision' ger bättre resultat än 'recall', då en ökning av den ena ofta görs på bekostnad av den andra.

Tabell 4.3: 'Precision' och 'Recall' för varje klass i molnens inre struktur.

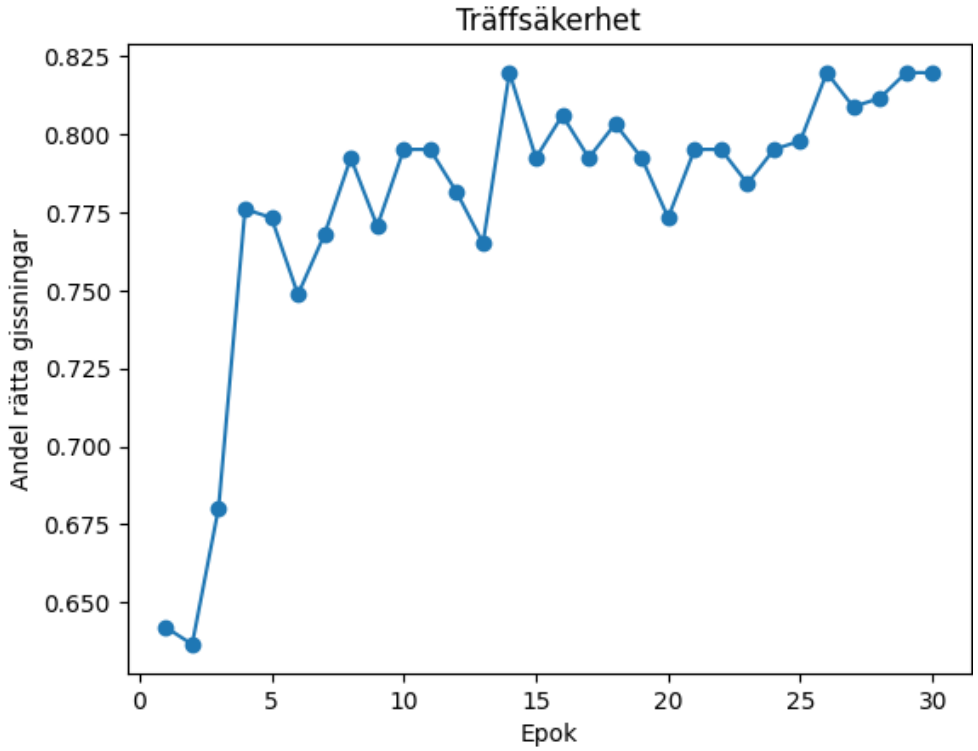
Klass	Precision	Recall
High	1	0,75
High_one_peak	0,817	0,85
High_with_peaks	0,805	0,825
Low_one_peak	0,853	0,886
Low_with_peaks	0,746	0,677

Att modellen har svårt att urskilja närliggande klasser behöver dock inte tyda på ett dåligt resultat. För framtida arbete kan det vara rimligt att se över klasserna igen och eventuellt skapa nya. Det kan vara kombinationer av nuvarande klasser, eller bygga på gemensamma egenskaper mellan de nuvarande klasserna. Detta kan tydliggöra skillnader mellan klasser för modellen, och kan kanske också skapa mer jämlikhet mellan klasser om träningsdatan blir mer jämt fördelad.

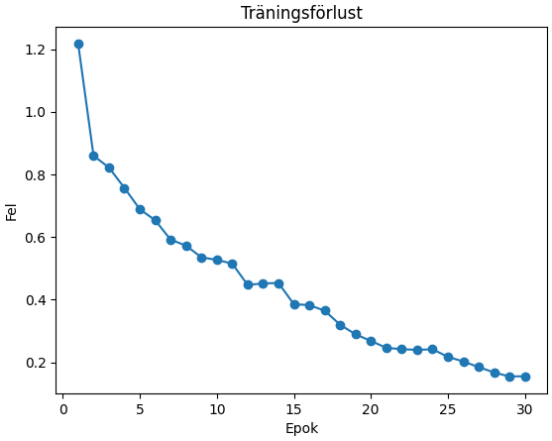


Figur 4.15: Normaliserad förvirringsmatris som visar resultatet av klassificeringsmodellen för molnens inre struktur. Förkortningarna HoP, HwP, LoP och LwP står för *High_one_peak*, *High_with_peaks*, *Low_one_peak* respektive *Low_with_peaks*.

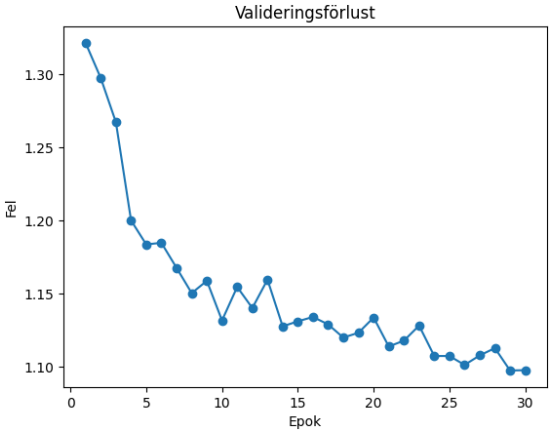
I figur 4.16 visas hur modellens säkerhet gällande klassificering ändras under varje epok. Grafen visar på att modellen utvecklades snabbt under tidigare epoker, och börjar sedan plana ut något runt epok 5. Grafen visar att modellen var instabil gällande träffsäkerhet efter ungefär epok 5, och tränades därför tills den visade en mer stabil skillnad mellan epoker. En anledning till att modellen inte tränades i mer än 30 epoker var för att undvika överanpassning. Överanpassning av en modell är när den har tränat för mycket på specifik data, alltså att den är för anpassad på specifika egenskaper för varje klass och har därmed svårt att hantera tidigare okända egenskaper från ny data [20]. I figur 4.17 visas tränings- respektive valideringsförluster för alla epoker. Figur 4.17a visar på att modellen fortsätter utvecklas för varje epok, men figur 4.17b visar på att modellen växlar mellan förbättring och försämring gällande valideringsförluster mot slutet. Ett ökande värde för valideringsförlusterna när träningsförlusterna går neråt kan vara ett tecken på att modellen är överanpassad och därmed inte lika lämplig att använda på ny data.



Figur 4.16: Graf som visar modellens säkerhet för varje tränad epok.



(a) Träningsförluster sjunker stadigt för alla epoker.



(b) Valideringsförluster sjunker till en början, men börjar svikta mot slutet.

Figur 4.17: Grafer över modellens tränings- samt valideringsförluster under träningen.

För att säkerhetsställa att *YOLOv8x-cls* var värd att använda testades även modellen *YOLOv8n-cls*, Ultralytics snabbaste men minst säkra klassificeringsmodell. Modellen tränades över 20 epoker och träningstiden var snabbare än för *YOLOv8x-cls*, dock som tidigare nämnts försumbart. Den gav bäst resultat för epok 17 med en säkerhet på 71% (se tabell 4.4). Resultatet visar på en betydligt sänkning av säkerhet för klassificering och tyder därför på att *YOLOv8x-cls* var en bättre träningsmodell för vår datamängd och arbetets omfattning.

Tabell 4.4: Bästa resultatet uppnåt med *YOLOv8n-cls* uppstod under epok 17 med en säkerhet på 71%. Träningsförlusten för modellen är 0,74314. Valideringsförlusten är 1,2514.

Epok	Träffsäkerhet klassificering	Träningsförlust	Valideringsförlust
17	0,71038	0,74314	1,2514

4.2.3 Lämplighet som klassificeringsverktyg

Överlag verkar övervakad maskininlärningen fungera som ett klassificeringsverktyg men mycket beror på valet av klasser som görs samt hur väl och noggrant annoteringen av den använda datan är. De två undersökta modellerna uppnådde träffsäkerheter kring 80% vilket innebär att en stor majoritet av den data som testas skulle klassificeras enligt annoteringen av datan. Detta anses vara en rimlig träffsäkerhet att sikta på med tanke på den begränsade mängden data och svårigheterna som dyker upp i att skapa och dela upp moln i klasser. Om det uppstår situationer i annoteringen där det är osäkert vilken klass ett moln bör tillhöra kan inte modellen förväntas göra samma val utan osäkerhet.

Datamängderna inom respektive klasser bör även tas i åtanke. Om någon av de valda klasserna innehåller betydligt fler bilder än en annan sker obalans i datan. Detta var även fallet för båda huvudindelningarna för maskininlärningen där minst en av klasserna innehöll en betydligt större mängd bilder. Sådana imbalanser i data kan leda till att modellen får en inbyggd bias för en sådan klass vilket inte är optimalt för ett rättvist klassificeringsverktyg. Till exempel kan detta leda till att en hög träffsäkerhet i modellen återspeglar en bra förmåga att gissa på specifikt denna klass då en motsvarande andel av denna klass är mycket större än för resten.

Som modellen ser ut i nuläget är de valda klassindelningarna begränsande. Det finns många egenskaper hos dessa moln som inte valdes att eller ens kunde undersökas. Om en mer övergripande klassificering ska kunna utföras behöver molnens alla egenskaper tas i åtanke för att skapa en bättre helhetsbild.

5

Möjliga förbättringar och andra tillvägagångssätt

Arbetet var en lärande process. Stora delar av arbetet handlade om att undersöka och pröva olika metoder för att slutligen bygga upp ett program som kunde hantera så många moln som möjligt på ett acceptabelt sätt. För fortsatt arbete kring klassificering av molekylmoln kan förslagen nedan vara lämpliga att undersöka och eventuellt implementera. Dessa förslag bygger på ändringar som vi tror hade påverkat arbetet positivt eller kunnat användas för att undvika problem som uppstod under arbetsprocessen.

5.1 Bildhantering och klassiska metoder

För att komma igång med övriga delar av projektet behövde en tidig version av processeringskoden skapas. Detta ledde dock till att radikala förändringar av dess struktur och därmed resultat blev problematiska, framför allt i senare skeden och vad gällde maskininlärningen. Därför valde vi i slutändan att inte implementera sådana omfattande möjliga förbättringar, utan bara förfina den etablerade metoden. Det hade dock varit önskvärt att i mer detalj undersöka andra tillvägagångssätt. Då parametrar beräknats direkt från den behandlade datan hade bearbetningen en stor inverkan på deras resultat. Därför hade det bästa sättet att öka resultatets tillförlitlighet för de klassiska metoderna varit att förbättra denna.

Något som då borde undersökas är tidig Gaussisk utjämning av molndatan innan vidare hantering, då detta sannolikt hade förenklat urskiljning av övergripande struktur. Detta gjordes ursprungligen ej då det förfarades att det skulle förvränga strukturen och försvåra separation av områden som enbart var löst sammankopplade. I efterhand anses det att detta ändå till viss grad skett och att det därför är sannolikt att det hade varit bättre att använda utjämningen tidigt. Det skulle möjligtvis ha vinklat resultaten för framförallt de klassiska metoderna mindre.

Utöver Gaussisk utjämning hade programbiblioteket Astrodendro [21] som tar fram astronomiska dendrogram kunnat användas för att ta fram molntoppar och dela upp den underliggande strukturen mellan dem. Liknande algoritmer hade tidigare undersökts, men kunde inte implementeras på ett fördelaktigt sätt. Programmet upptäcktes dock sent i arbetet, och då det testades för den dåvarande koden gjordes bedömningen att den i det stadiet ej var användbar. Anledningen

var att den inte hade resulterat i förbättrad behandling av molnen utan fler stora modifikationer av metoden, vilket ville undvikas. Detta berodde till stor del på bruset i datan, vilket hade kunnat förbättras genom tidig Gaussisk utjämning. Vidare hade Astrodendro kunnat användas för att tidigt gruppera molnen utifrån exempelvis areaförhållanden, vilket hade öppnat möjligheten för olika behandling i efterkommande bildhantering. Det hade varit en intressant punkt för framtida studier då många av de samband som upptäckts är storleksberoende, även om det är osäkert till vilken grad detta beror på generaliseringen i bildbehandlingen. En jämförelse av resultaten skulle i så fall kunna verifiera detta samband.

En annan möjlig förändring är just att undersöka hur olika behandling av olika grupper av moln påverkar resultatet. En av de begränsningar som stöttes på under arbetet var att koden behövde fungera så bra som möjligt för så många moln som möjligt. Det hade varit intressant att se hur exempelvis anpassning av standardavvikelse för den Gaussiska utjämningen eller hur man väljer angränsande pixlar för erosionen efter filstorlek hade påverkat den upplevda överrepresentationen av runda, små moln.

5.2 Maskininlärningsmetoder

Det finns många olika tillvägagångssätt vad gäller maskininlärning, och därför också många förbättringsmöjligheter. Förbättringar kan göras inom flera moment, vilka beskrivs nedan.

5.2.1 YOLO-modeller

I början av projektet valdes YOLOv8 ut som den ledande kandidaten för inlärningsmodellen men det existerar självklart många andra modeller med liknande syften som hade kunnat användas istället. Dessa modeller hade även kunnat fungerat som jämförelse för att komma fram till hur bra modellen faktiskt fungerar och hur mycket rum det finns för utveckling med nuvarande metod. Den använda YOLOv8-modellen använder 8 bitars färgdjup för att representera färg. Eftersom datan har istället har 32 bitars färgdjup kommer denna behöva komprimeras. Detta innebär att bilderna inte representerar datan fullständigt utan är en något förenklad version. Ett annat alternativ hade varit att skapa en maskininlärningsmodell själva för att kunna säkert veta hur allting fungerade samt kontrollera varje liten detalj i samtliga processer. Det här förslaget har däremot nackdelen att den kräver mer avancerad kunskap kring maskininlärning samt att tidsåtgången för ett sådant projekt hade blivit mycket större.

I YOLOv8 finns ett antal hyperparametrar som används för att instruera modellens inlärning och som även går att ändra på inför inlärningen. Till dessa parametrar hör till exempel tidigare nämnda *batch*, antal epoker samt *imgsz* men även parametrar som påverkar modellens inlärningshastighet som *lr0*, *lrf*, eller *cos-lr*. Standardinställningarna för YOLOv8 innebär att modellen börjar på en angiven inlärningshastighet ($lr0 = 0{,}01$) och slutar på en annan ($lrf = 0{,}01 \cdot lr0$). Parametern *cos-lr* bestämmer

om denna minskning ska ske linjärt eller likt en cosinuskurva [6]. Experimentering med dessa parametrar hade kunnat innebära att de framtagna modellerna hade kunnat finjusteras för att göras mer effektiva och träffsäkra för klassificeringen.

5.2.2 Data för modellen

Som tidigare nämnt skiljde sig storleken på träningsdatan markant för respektive klass. Detta kan ha uppstått som resultat av hur varje klass definierats, eller bara den mängd data som fanns tillgänglig för projektet. Om man utesluter möjligheten att skapa nya klasser så hade detta eventuellt kunnat lösas med hjälp av dataökning (användning av syntetisk data). Denna dataökning kan ske på flera olika sätt, till exempel genom att rotera eller spegla bilderna men man kan även introducera brus eller förändra bildernas färg eller kontrast. Att göra detta innebär att modellen kan få in nya bilder utan att upprepa identiska uppgifter. För en rättvis och välfungerande modell bör mängden data för de annoterade klasserna ligga inom rimliga avstånd från varandra.

Med den valda datahanteringen uppstår även problem att göra jämförelse mellan olika moln. Till en början är den valda färgskalan endast satt relativt det enskilda molnets höjd. Att två moln ser ut att ha liknande hög intensitet innebär alltså inte att de faktiskt har det, utan skalan visar endast den högsta intensitet hos varje moln. Sedan saknas även en konsekvent skalning mellan molnen vilket kan observeras i den varierande upplösningen på molnbilderna. Detta innebär att inga kopplingar om molnets storlek relativt varandra är lätta att göra då pixeldensiteten är svår att konkretisera endast med hjälp av synen i bilderna. Dessa faktorer ger direkt begränsningar i hur klassificeringen av molnen behöver gå till eftersom inga klasser baserade på relativ storlek eller intensitet mellan olika moln är möjliga.

5.2.3 Oövervakad maskininlärning

En viktig uppgift för att utveckla en övervakad maskininlärningsmodell är att förutbestämma klasser och genomföra annotering på dessa. I denna undersökning visade sig dessa klasser svåra att tänka ut och sedan separera och definiera ordentligt. För att minska behovet av dessa kunde en oövervakad maskininlärningsmodell, en modell som skapar klasser själv, använts i samband med den övervakade. Målet med denna modell hade då inte behövt vara att klassificera molnen utan snarare hjälpa till att hitta gemensamma egenskaper mellan molnen, alltså hitta mönster som annars hade förblivit oupptäckta. Det hade underlättat för den manuella klassificeringen för övervakad inlärning, främst då klasserna är mer relevanta för hela datamängden.

6
Slutsatser

Resultaten från projektet visar på att arbetet kring att definiera klasser för interstellära molekylmoln är svårt. Det är komplicerat att med grundläggande formbeskrivning och analys ta fram parametrar som beskriver de subjektiva upplevelser som kan vilja utnyttjas för dessa uppdelningar. Interstellära molekylmoln har ofta komplexa former som med mänskligt öga skulle kunna delas in i olika klasser, men som saknar tydlig grund i parametrar hämtade från rådatan. För de egenskaper som undersöktes hittades inga tydliga grupperingar; för samtliga parametrar observerades istället kontinuerliga fördelningar, varför det med denna metod var mer intressant att titta på relationen mellan parametrarna. För direkt klassificering med metoden gjordes subjektiva bedömningar, precis som vid annotering av datan för maskininlärning. För en sådan subjektiv indelning kan dock en maskininlärningsmodell vara ett mer kraftfullt verktyg, eftersom man inte behöver begränsa sig till explicit definierade parametrar för annoteringen.

Vi ser dock samma problem angående gränsdragning vid annoteringen för maskininlärningsmodellen som för den parameterbaserade metoden: det finns en stor variation vad gäller form och struktur, vilket innebär att det förekommer en hel del moln som kan klassas som både det ena eller det andra både av individen som annoterar datan, och av modellen. Hur väl modellerna fungerar är alltså mycket beroende av den subjektiva annoteringen och valet av klasser, men även mängden indata, både överlag och inom respektive klass.

I projektet har vi utgått från ett större dataset och utvecklat en process som genom applikation av olika bildhanteringsmetoder tar ut enskilda moln och framhäver deras form. Vi har undersökt egenskaper och morfologi hos de interstellära molnen som kan vara relevanta för att definiera klasser, studerat relationen mellan parametrar, och gjort försök till klassifikation både baserat på dem och med maskininlärningsmodeller. Vid eventuella vidare studier hade det varit intressant att undersöka andra parametrar samt metoder såsom oövervakad maskininlärning.

Litteratur

[1] J. Kainulainen och M. Zhang, "PRobing the Origins of MassIve molecular cloud StructurEs (PROMISE): high-fidelity column density maps for molecular clouds in the Galactic plane," opublicerad.

[2] D. Ward-Thompson och A. P. Whitworth, *An Introduction to Star Formation*. Cambridge, UK: Cambridge University Press, 2011.

[3] C.-J. Hsu, *Chemodynamical Simulations of Star-Forming Molecular Clouds*. Chalmers University of Technology, 2023.

[4] N. S. Schulz, *From dust to stars: Studies of the Formation and Early Evolution of Stars*. Praxis publishing, 2005.

[5] K. Scherer. "Taurus Molecular Cloud 1." (dec. 2017), URL: https://www.flickr.com/photos/kees-scherer/39234929921/in/photostream/.

[6] Ultralytics. "Ultralytics YOLOv8 Docs." (2024), URL: https://docs.ultralytics.com/models/yolov8/.

[7] J. Kainulainen och J. C. Tan, "High-dynamic-range extinction mapping of infrared dark clouds," jan. 2012. DOI: https://doi.org/10.1051/0004-6361/201219526. URL: https://www.aanda.org/articles/aa/full_html/2013/01/aa19526-12/aa19526-12.html.

[8] T. Jaffe, *FITS Primer*, 2014. URL: https://fits.gsfc.nasa.gov/fits_primer.html#:~:text=A%20FITS%20file%20is%20comprised,point%20numbers%20using%20IEEE%20representations.

[9] R. C. Gonzalez och R. E. Woods, *Digital Image Processing*, 4. utg. Pearson, 2018.

[10] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias m. fl., "scikit-image: image processing in Python," *PeerJ*, årg. 2, e453, juni 2014, ISSN: 2167-8359. DOI: 10.7717/peerj.453. URL: https://doi.org/10.7717/peerj.453.

[11] scikit-image, "skimage.measure". URL: https://scikit-image.org/docs/stable/api/skimage.measure.html.

[12] K. Benkrid, D. Crookes och A. Benkrid, "Design and FPGA implementation of a perimeter estimator," *Proceedings of the Irish Machine Vision and Image Processing Conference*, s. 51–57, jan. 2000.

[13] C. Forssén, *Bayesian inference and machine learning*. Kurslitteratur Chalmers, 2022. URL: https://cforssen.gitlab.io/tif385-book/content/Intro/preface.html.

[14] A. Lindholm, N. Wahlström, F. Lindsten och T. B. Schön, *Machine Learning - A First Course for Engineers and Scientists*. Cambridge University Press, 2022. URL: https://smlbook.org.

[15] Azure bidragsgivare. ”Evaluation metrics.” (2023), URL: https://learn.microsoft.com/en-us/azure/ai-services/language-service/custom-text-classification/concepts/evaluation-metrics.

[16] J. Redmon, S. Divvala, R. Girshick och A. Farhadi, *You Only Look Once: Unified, Real-Time Object Detection*, 2016. arXiv: 1506.02640 [cs.CV].

[17] J. Terven och D. Cordova-Esparza, ”A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS,” april 2023. DOI: 10.3390/make5040083.

[18] Ultralytics. ”Ultralytics YOLOv8 Docs.” (2024), URL: https://docs.ultralytics.com/usage/cfg/.

[19] Wikipedia contributors. ”Accuracy paradox — Wikipedia, The Free Encyclopedia.” (2023), URL: https://en.wikipedia.org/w/index.php?title=Accuracy_paradox&oldid=1176847835.

[20] Azure bidragsgivare. ”Prevent overfitting and imbalanced data with Automated ML.” (2023), URL: https://learn.microsoft.com/en-us/azure/machine-learning/concept-manage-ml-pitfalls?view=azureml-api-2.

[21] T. Robitaille, C. Beaumont, A. Ginsburg, B. MacDonald och E. Rosolowsky. ”Astronomical Dendrograms in Python.” (Inget datum), URL: https://dendrograms.readthedocs.io/en/stable/.

[22] Astropy Collaboration, A. M. Price-Whelan, P. L. Lim m. fl., ”The Astropy Project: Sustaining and Growing a Community-oriented Open-source Project and the Latest Major Release (v5.0) of the Core Package,”, årg. 935, nr 2, 167, s. 167, aug. 2022. DOI: 10.3847/1538-4357/ac7c74. arXiv: 2206.14220 [astro-ph.IM].

[23] J. D. Hunter, ”Matplotlib: A 2D graphics environment,” *Computing in Science & Engineering*, årg. 9, nr 3, s. 90–95, 2007. DOI: 10.1109/MCSE.2007.55.

[24] C. R. Harris, K. J. Millman, S. J. van der Walt m. fl., ”Array programming with NumPy,” *Nature*, årg. 585, nr 7825, s. 357–362, sept. 2020. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[25] The pandas development team, *pandas-dev/pandas: Pandas*, version 2.2.2, febr. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[26] P. Virtanen, R. Gommers, T. E. Oliphant m. fl., ”SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, årg. 17, s. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.

[27] M. L. Waskom, ”seaborn: statistical data visualization,” *Journal of Open Source Software*, årg. 6, nr 60, s. 3021, 2021. DOI: 10.21105/joss.03021. URL: https://doi.org/10.21105/joss.03021.

[28] G. Jocher, A. Chaurasia och J. Qiu, *Ultralytics YOLOv8*, version 8.0.0, 2023. URL: https://github.com/ultralytics/ultralytics.

A
Programbibliotek

Utöver Pythons standardbibliotek användes följande programbibliotek i projektet:

Astropy [22] är ett programbibliotek som låter användaren enkelt arbeta med astronomi och astrofysik i kod. Astropy används framförallt för hantering av FITS-filer i detta projektet.

Matplotlib [23] är till för att skapa figurer och tabeller i Python för att visualisera data. I detta projekt används det för just detta, samt omvandling från FITS-bilder till JPG-filer.

Numpy [24] används för att underlätta arbete kring vetenskapliga projekt som hanterar flerdimensionella vektorer eller arrayer i kod. För detta projekt används Numpy för diverse matrishantering och matematiska funktioner.

Pandas [25] är ett programbibliotek som används för hantering av dataset.

Scipy [26] är ett programbibliotek som bygger på Numpy. Det används för att utföra matematiska algoritmer och innehåller underbibliotek för olika specifika forskningsområden och används i projektet för bildhantering och parameterberäkning.

Seaborn [27] användes likt matplotlib för visualisering av data och resultat.

Skimage eller **Scikit-image** [10] är ett programbibliotek bestående av en samling algoritmer för datorseende, det vill säga datorers förmåga att analysera visuella medium, och bildbehandling. Det används i projektet för bildbehandling och beräkning av omkrets.

Ultralytics är till för maskininlärning och datorseende, och används i detta projektet för **YOLOv8** [28].

INSTITUTIONEN FÖR Rymd-, geo, och miljövetenskap
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige
www.chalmers.se

CHALMERS