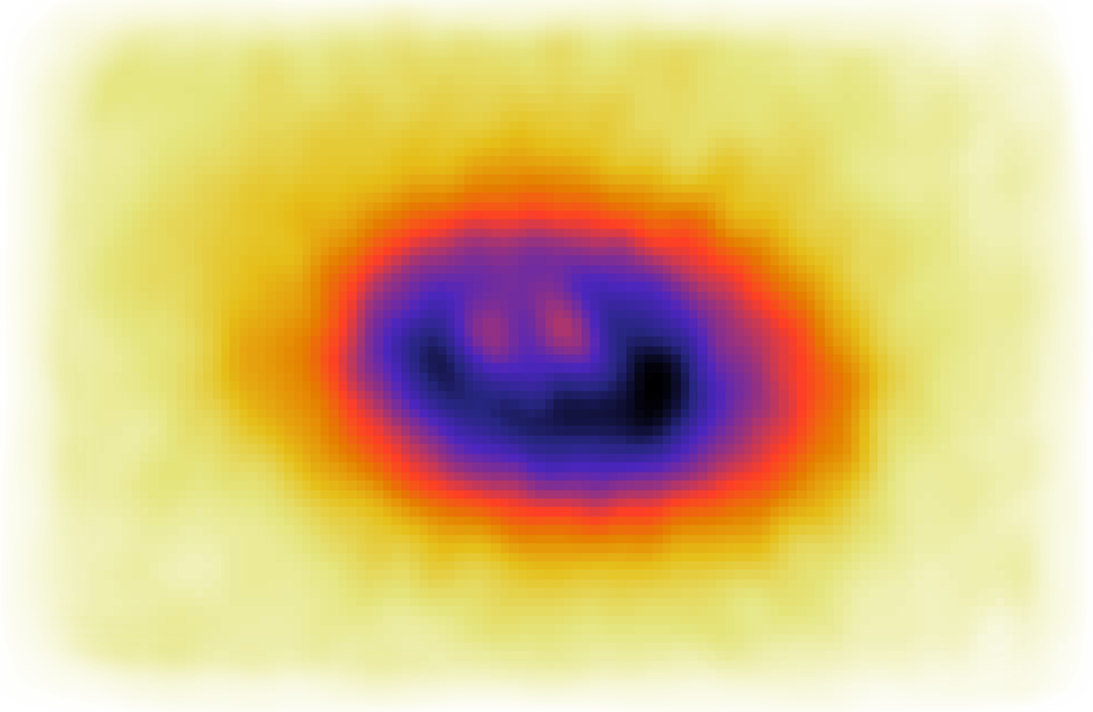
CHALMERS



Finding Potential Detections of Dust in Protoplanetary Disk Winds

Using Machine Learning to Filter, Classify and Present ALMA-data

Bachelor's thesis in Space, Earth and Environment

Peter Fagrell, Jens Kollberg, Elias Rasmussen,
Erik Redmo Axelsson, Oskar Svensson, Alexander Ybring

DEPARTMENT OF SPACE, EARTH AND ENVIRONMENT

CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2023

www.chalmers.se

Bachelor's thesis 2023

Finding Potential Detections of Dust in Protoplanetary Disk Winds

Using Machine Learning to Filter, Classify and Present ALMA-data

PETER FAGRELL
JENS KOLLBERG
ELIAS RASMUSSEN
ERIK REDMO AXELSSON
OSKAR SVENSSON
ALEXANDER YBRING

CHALMERS

Department of Space, Earth and Environment
SEEX16
Chalmers University of Technology
Gothenburg, Sweden 2023

Finding Potential Detections of Dust in Protoplanetary Disk Winds
Using Machine Learning to Filter, Classify and Present ALMA-data
PETER FAGRELL, JENS KOLLBERG, ELIAS RASMUSSEN,
ERIK REDMO AXELSSON, OSKAR SVENSSON, ALEXANDER YBRING

Supervisor: Per Bjerkeli, Department of Space, Earth and Environment
Supervisor: Maria Carmen Toribio, Department of Space, Earth and Environment
Examiner: Magnus Thomasson, Department of Space, Earth and Environment

Bachelor's Thesis 2023
Department of Space, Earth and Environment
SEEX16
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The protoplanetary disk of crAus_09 resembling a smiley extracted from the ALMA project 2019.1.01792.S. The image has been identified by a Convolutional Neural Network as candidate detection of dust in the wind.

Typeset in LaTeX
Gothenburg, Sweden 2023

Finding Potential Detections of Dust in Protoplanetary Disk Winds
Using Machine Learning to Filter, Classify and Present ALMA-data
Peter Fagrell, Jens Kollberg, Elias Rasmussen, Erik Redmo Axelsson,
Oskar Svensson, Alexander Ybring
Department of Space, Earth and Environment
Chalmers University of Technology

Abstract

Star formation begins when clouds of dust and gas starts to collapse under the force of their own gravity. Eventually, a protoplanetary disk is formed in the central region. Because of magnetic fields, gravity and rotation, material is ejected through jets and winds that sweep up gas from the surrounding envelope and form an outflow. These outflows remove angular momentum which permits a star to form in the center. In the disk, small grains eventually starts to clump together to become planets. Whether lifting of dust via winds affect the planet formation process is yet not understood, but it is a research topic (e.g. Pascucci et al., 2022) where this project could contribute.

The project aims to automatically extract images from the Atacama Large Millimeter/submillimeter Array (ALMA) Science Archive that indicates dust lifting from disks via the aforementioned winds, with the use of a Convolutional Neural Network (CNN). Additionally, the network should be flexible enough to be applied to other, similar problems. With the help of ALminer observations are fetched and then fed to the CNN to filter out objects of interest. Due to a low number of known observations that could possibly depict dust in the wind, images are augmented to produce sufficient training data. During training of the CNN, a portion of the training data is reserved for determining the accuracy of the network. Finally, it is applied to a part of the ALMA archive chosen through keywords and presents images it labels as positive.

The methods mentioned led to an extraction of several observations. Prediction of labels was performed by the CNN with high accuracy and it can, with modification, act as a general model for other astronomical phenomena. In conclusion, a publicly available and free tool was created that can support researchers in their collection of data from the ALMA archive, minimizing manual labor and advancing the studies of the universe.

Keywords: Protoplanetary Disk, Outflows, Star, Winds, ALMA, CNN, ALminer

Finna Potentiella Detektioner av Stoft i Protoplanetära Diskvindar
Använda Maskininlärning för att Filtrera, Klassifiera och Presentera ALMA-data
Peter Fagrell, Jens Kollberg, Elias Rasmussen, Erik Redmo Axelsson,
Oskar Svensson, Alexander Ybring
Institutionen för Rymd-, Geo- och Miljövetenskap
Chalmers Tekniska Högskola

Sammandrag

Stjärnbildning börjar när moln av stoft och gas kollapsar under sin egna gravitation. Till slut formas en protoplanetär skiva i den centrala regionen. På grund av magnetiska fält, gravitation och rotation accelereras material iväg via jetstrålar och vindar som sveper upp omgivande gas vilket bildar ett utflöde. Dessa utflöden avlägsnar rörelsemängdsmoment vilket tillåter en stjärna att bildas i centrum. I disken finns stoft som klumpar ihop sig för att till slut bli planeter. Ifall vindarna bär med sig stoft ur de protoplanetära diskarna och hur det påverkar planetbildning är för tillfället ett forskningsämne (t.ex. Pascucci et al., 2022), vilket detta projekt ämnar att bidra till.

Projektets syfte är att automatiskt finna bilder tagna av Atacama Large Millimeter/submillimeter Array (ALMA) som potentiellt visar på stoft som lämnar protoplanetära diskar genom de tidigare nämnda vindarna. Dessa ska hittas med hjälp av ett Convolutional Neural Network (CNN). Dessutom ska programkoden bakom vara flexibel nog för att kunna appliceras på andra, liknande problem. Observationerna hämtas från ALMA Science Archive genom verktyget ALminer, som sedan matas till ett CNN som filtrerar ut objekt av intresse. På grund av ett lågt antal kända observationer som möjligtvis avbildar stoft i vinden augmenteras bilder för att producera tillräcklig träningsdata. Under exikvering tränas CNN:et under ett givet antal epoker (gånger) på träningsdatan, testas på en del av den, för att sedan rapportera dess träffsäkerhet. Till slut appliceras den på en del av ALMA-arkivet som valts ut genom nyckelord och presenterar bilder den märkt som positiva.

Med de tidigare nämnda metoderna kunde ett CNN tränas till att uppnå syftet med projektet. Med en hög träffsäkerhet kunde ett flertal dittills okända bilder tagna av ALMA lokaliseras och tillges forskarsamfundet. Kungörandet av koden och dess dokumentation möjliggör en generalisering till andra, liknande forskningsyften. Projektet som helhet kan hjälpa forskare i deras analys av ALMA-arkivet, samtidigt som det kan minska mängden manuellt arbete och avancera studierna av rymden.

Nyckelord: Protoplanetär Skiva, Utflöden, Vindar, Stjärna, ALMA, CNN, ALminer

Acknowledgements

A big thank you goes out to Per Bjerkeli for his outstanding work as a supervisor. With great input, humour and spirit, he has guided us throughout the course of this project whilst maintaining our motivation. On top of this, his astronomic cunning and passion for knowledge made for a great cooperation, which allowed the success of this project. Hopefully, the tool we provide will help further his research henceforth.

Another massive effort was made by Maria Carmen Toribio. She deserves a thank you for also contributing to the lovely work environment we have had. Acting as a connection to the ALMA observatory, she immediately set us on the right path and made sure we had all the equipment and tools necessary. Great weight has been put on her feedback and expert opinion throughout the entirety of this project.

We would also like to thank Jon, Adele and Henrik for inspiration, understanding and support.

Peter Fagrell, Jens Kollberg, Elias Rasmussen,
Erik Redmo Axelsson, Oskar Svensson, Alexander Ybring,
Gothenburg, 2023

Plenty of ALMA data was collected during the process of training, testing and presenting images of potential dust in the wind. Its corresponding projects appear in respective appendix, see appendix A and B.

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

ALMA	Atacama Large Millimeter/submillimeter Array
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
FN	False Negative
FP	False Positive
GUI	Graphical User Interface
NN	Neural Network
PBCOR	Primary Beam CORrection
RELu	Rectified Linear Unit
TN	True Negative
TP	True Positive

Contents

1

Introduction

The process of star formation begins when clouds of dust and gas starts to collapse under the force of gravity (Bolles, 2023). Eventually due to conservation of angular momentum, a disk is formed in the central region. Here, magnetic fields, gravity and rotation conspire, to eject an outflow of material that moves away from the disk and that removes angular momentum from the system (Pudritz and Norman, 1983; Sheikhnezami et al., 2012; Gressel et al., 2015). The removal of angular momentum in turn permits a star to form in the central region. There is still much uncertainty regarding a variety of aspects about these outflows, in particular how the launching takes place. If dust can be carried out of protoplanetary disks in this manner and how this affects the system (Shu, 1997; Blandford and Payne, 1982).

Only a few known observations of what is potentially dust being carried out of protoplanetary disks by wind have been found. All of these are speculations built on similarities between these and the proposed ideas from Shoemaker et al. ([n. d.]), studying the source HH212. These extensions of dust, differentiable by irregularities around the disks, are fundamental to this project. Whether dust can be lifted away from disks via winds is not yet known.

The Atacama Large Millimeter/submillimeter Array (ALMA) (ALMA Observatory, 2023) is an observatory located at approximately 5000 meters above the sea level in northeastern Chile. Its 66 antennas work together to capture high resolution images of everything from our own solar system, to objects within the Milky Way Galaxy, to distant galaxies (NRAO, 2023). It does so by capturing light in the bandwidth between radio and infrared. These images are then stored in the public ALMA Science Archive.

The archive holds almost 60 000 unique observations to date (Atacama Large Millimeter/submillimeter Array, 2023). To manually review each and every one of these would be a time consuming and tedious effort. Once a specific observation is found however, it is often of interest to find more of a similar fashion. An example of this

is dust in protoplanetary disk winds. to automate the process of finding objects in space with correlating indicators is crucial for further analysis of the phenomena mentioned. One way of doing this is to utilize a machine learning technique called Convolutional Neural Network (CNN), which can be trained to recognize objects in images in similar ways as humans.

1.1 Aim

The aim of this project is to be able to filter through an extensive archive to separate images that are particularly interesting from an observational point of view. In our case, we want to search for images of disks with potentially winds that contain dust. Finding more potential sources of dust in the wind would enhance our understanding of to what extent winds can carry away dust from protoplanetary disks. If it is a common phenomena, recent results from HH212 could be put into a broader context. To achieve such a goal, a CNN is to be put in place to automatically separate these observations. The capabilities of this CNN should be to analyze observations from the ALMA Science Archive, single out certain traits of the given event and find similar objects.

The project is intended to simultaneously act as a blueprint for finding other types of objects in the archive, for example spiral galaxies. Therefore, the project's end result will be publicly available and free to use with documentation explaining the details of modification, which hopefully promotes a more effective research process with less manual labour. Because of the project's flexible applicability and the conception that users of this project's result might use it to find rare and exotic phenomena, methods intended to help extend data sets through augmentations is included in the program code.

2

Theory

This chapter presents all relevant facts and information that was used throughout this project. It starts with the foundations, explaining the relevant astronomical events that occur, including star formation and outflows. Following, ALMA and its archive is described and last comes neural networks, which are the backbone of the project.

2.1 Star Formation

Young stars originate from clouds of dust and gas that occur scattered throughout many galaxies, as described in (Bolles, 2023). It is the variation in molecular density in these clouds that starts the formation. The process is a complex phenomenon which still has many uncertainties regarding how factors like size of the clouds, and surrounding environment affect the formation (Shu, 1997; Blandford and Payne, 1982). The process will therefore be described one of the more studied cases which is for isolated young low-mass stars. The formation can be divided into four main stages, which are shown in figure 2.1.

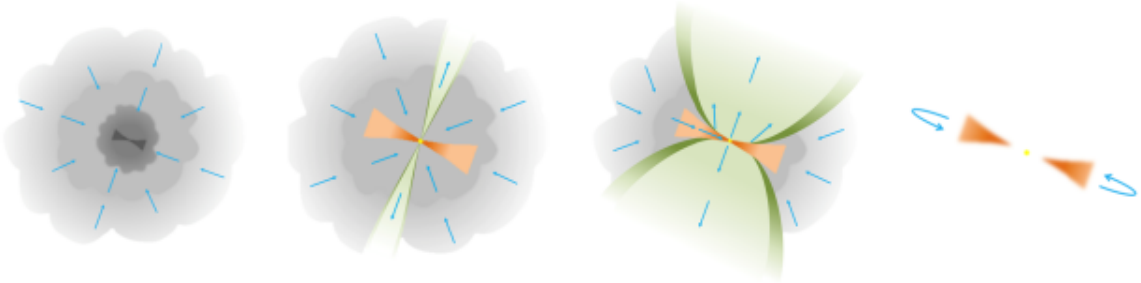


Figure 2.1: Stages of the formation of an isolated young low-mass star by Bjerkeli (2022).

The first stage begins when dense regions of dust and gas starts to form in the clouds, which are of such critical mass that the gravitational pull makes them start to collapse (Bolles, 2023). These pockets are created because of turbulence and ambipolar diffusion that occur in the clouds' magnetic field (Shu, 1997). The second stage is categorized by the creation of a star-core with a surrounding centrifugally supported disk, called a protostar. The gravitational collapse makes the disk accrete matter which then falls into the core at a velocity that increases as the radii gets smaller. This process is referred to as an inside-out collapse, meaning that as matter falls into the core, the front of the accretion expands outwards. The third stage starts when outflows is emitted from the system. These carry angular momentum, energy and mass from the system (Shu, 1997; Sheikhnezami et al., 2012; Pudritz and Norman, 1983; Gressel et al., 2015). Outflows are further elaborated in section 2.1.1. The fourth stage ends the formation and occur when the inflow cease, leaving only the star without a surrounding disk, marking the creation of a new star.

2.1.1 Winds and Jets

The outflows described in the third stage of star formation in section 2.1 consists of winds and jets that are emitted from the system (Shu, 1997; Sheikhnezami et al., 2012; Pudritz and Norman, 1983; Gressel et al., 2015). These remove angular momentum and material that are transported to the disk through inflow of material and are in many cases bipolar-streams that appear on both sides of the protostar. The jets are beams of high velocity that are directed out from the protostar perpendicular to the disk, while the winds spread at a wider angle. An example is shown in figure 2.2,

Figure 2.2: Illustrated examples of some possible outflows by Bjerkeli (2023).

and are illustrations of images of protoplanetary disks with possible evidence of dust in the wind as they would appear on images. These examples were provided to us by Bjerkeli (2023) whose scientific research instigated this project. The outflows are shown as extensions from a Gaussian disk[1]. There are many uncertainties regarding these extensions (Shu, 1997; Blandford and Payne, 1982) and the actual formation of the outflows. For example what it is that cause these winds to form, if they can launch dust from the protoplanetary disk and if so to which degree, and how these winds affect the formation of the star.

2.2 ALMA Observatory

The Atacama Large Millimeter/submillimeter Array (ALMA) is an astronomical observatory located in the Atacama Desert of northeastern Chile (NRAO, 2023). Together with the Republic of Chile, the site has been developed in collaboration with teams from North America, Europe and East Asia, and scientific data has been produced and extracted since 2011 (European Southern Observatory, 2011).

The observatory is made up of 66 antennas, with 54 of these measuring 12 meters in diameter and the other twelve measuring 7 meters. The constellation provides a tool to observe and study light in the frequency between radio and infrared (or millimeter to submillimeter). The dry and cold climate of the Atacama Desert has been carefully chosen since the millimeter to submillimeter light is easily absorbed by water vapor which comes in abundance in most habitable areas of the globe.

ALMA is undoubtedly one of the most advanced observatories in the world, and has provided scientists all over the world with data in their search for groundbreaking discoveries in astrophysics and cosmology. Currently, a group of scientists at Chalmers University of Technology are shining new light onto the research of solar

[1] A Gaussian disk is an ellipsoid with values spread as a normal distribution.

systems' formation by studying observations taken by ALMA (Bjerkeli et al., 2021). In order to continue their work, finding several, similar objects can be helpful for their process.

2.3 ALMA Science Archive

ALMA is currently producing roughly 500 TB worth of raw-data and reduced data products per year. These data are stored at the Joint ALMA Observatory in Santiago, Chile and are copied from there to the three ALMA Regional Centers in North America, Europe and East Asia. There are therefore complete copies of the ALMA Science Archive that are accessible to the users. The archive can be queried manually through its web interface at Atacama Large Millimeter/submillimeter Array (2023), as well as programmatically using Virtual Observatory protocols [2]. In 2023 the ALMA archive hosts more than 1 PB of data (ALMA, 2023c). There are many different kinds of products, such as calibration tables, or preliminary images (Wood et al., 2021). The focus of this project has been the public science-ready images available for download and further analysis.

2.3.1 ALminer

To extract data from the archive a specific Python toolkit has been developed; ALminer (Ahmadi and Hacar, 2021b). ALminer is the archive mining toolkit that enables users to access data from the ALMA Science Archive by querying, analysing, visualizing and downloading. The code for the toolkit is free to use and accessible via their GitHub (Ahmadi and Hacar, 2023). It comes well-documented which makes it both flexible and easily modifiable. The development has been a collaboration between Allegro, the ALMA Regional Centre in The Netherlands, and the University of Vienna as part of the EMERGE-StG project (Ahmadi and Hacar, 2020).

A fundamental feature of the ALMA Science Archive is the ability to filter its contents according to user-defined criteria (for example: filtering by scientific keyword, target name, observing set up, etc). Such filtering is performed thanks to the metadata attached to each observation. For example, each data set contains information about the object(s) that is/are in the image. This information has been added to the metadata of the observation before it has been incorporated into the archive and allows the user to filter based on target names. The main feature ALminer is that it allows performing user queries to the whole archive and direct data download in a programmatic manner, facilitating the systematic exploitation of all existing

[2]The Virtual Observatory protocol is a standard for astronomical data on a worldwide scale. It standardizes the quality of data for developed information exchange (202, 2023).

data. Through ALminer, one can carry out systematic filtering of the ALMA Science Archive data according to observing criteria. For example the user can filter the archive based on release date and image resolution. Besides the filtering of observations, there is the possibility of filtering by file type, or rather what the file name should include. There is also the option of choosing server connection spot (or archive mirror) to speed up the downloading of data depending on your location, but since the European archive mirror is currently down, most users are limited to its North American or Japanese counterparts.

2.3.2 Data Features

In order to find the appropriate data for our goal, that is, finding potential detections of dust in protoplanetary disk winds, a data feature filtering of the archive had to be done. The following sections describe the data features that were later used as conditions for the data in this project.

2.3.2.1 Primary Beam Correction

A possible post correction that can be added to an image is Primary Beam CORrection (PBCOR). "The primary beam describes the antenna response (sensitivity) as function of the angle away from the main axis. The primary beam can be approximated by a Gaussian function" (ALMA, 2023a). Overall, the sensitivity is therefore not uniform over the field of view, having a maximum at the center and tapering off towards the edges. To reflect this non-uniform sensitivity, output images are corrected by dividing them by the primary beam response pattern. This is what is known as primary beam correction. In the ALMA Science Archive, by default, the stored images have all been corrected by the primary beam response, and have the extension in their filename ".pbcor" accordingly.

2.3.2.2 Continuum

In the ALMA archive, the images with the file extension ".cont" indicate that they contain images of the continuum emission in the field, that is, images of the emission average in that frequency range, and after having removed areas of the spectrum that potentially contain spectral lines from molecules or atoms. Given the goals of this project, it was identified as a priority to work with continuum images in first instance [3].

[3]Radio continuum emission is the broadband radiation emitted in the radio part of the spectrum by celestial objects. Its intensity (brightness temperature) typically varies relatively slowly as a function of wavelength (or frequency). This is in contrast to the narrow emission lines produced at characteristic frequencies by atoms and molecules (NRF, [n. d.]).

2.3.2.3 Science Target Data

The "_sci" extension is an indicator of data produced during the reduction of ALMA data and calibrated for scientific analysis. Files having this extension are meant to have undergone a check for the quality of data processing (e.g, images of the flux, phase and bandpass calibrators). For this project, we are interested only in retrieving images that have the science target or calibrators, which is done by including the mentioned extension. The choice is largely dependent on the cycle[4] and type of data reduction that was performed as well as the data products that exist on the archive as a result. In most recent cycles, the science target can be filtered out with the flag "_sci" or its ALMA target name, see parameter "filename_must_include" in alminer.download_data function (Ahmadi and Hacar, 2021a).

2.3.2.4 Integration time

One feature of the telescopes taking the images is integration time. Here, it refers to the time the detector is collecting data on the particular source (Wallace, [n. d.]). In other words, the amount of time that light is let through. This configuration has a great impact on what the signal-to-noise ratio will be, meaning that with a longer integration time, the more light is let in and captured. Objects far away and/or faint objects are often observed as very faint or can be hidden behind noise. The longer the telescope collects light, the better signal from the source can be captured and distinguished from the surrounding noise, meaning the signal-to-noise ratio is intended to be higher. If the object is bright and close by regarding distance, shorter integration time will be producing clearer images. This balance is crucial for collecting images where the object can be distinguished and studied.

2.3.2.5 Spectral Window or Frequency Ranges

Another difference in settings when observing an object with ALMA is the spectral windows or frequency ranges that are covered, and their spectral resolution (i.e., the width of the frequency channels). The observing frequency ranges will constrain which spectral lines or continuum emissions are to be observed. In other words, it defines which frequency of light that will be detected. In astronomy, this is important due to the fact that different species (e.g atoms, molecules) radiates at different wave lengths, letting scientists target certain species (Fraknoi et al., 2022). Depending on the observing frequency covered by the telescope the same source may display different morphologies in the images or even appear undetected.

[4] A cycle indicates the schedule for the ALMA during a certain time period. See ALMA (2023b) for information regarding the current cycle in 2023.

2.3.3 Flexible Image Transport System

Flexible Image Transport System (FITS) is the standard file format of the data in the ALMA Science Archive and are therefore used in this project. The file format was first introduced in 1981 and had its latest release in 2016, designed with long term storage in mind which is reflected in the the maxim "once FITS, always FITS" (P. Smale, 2014). Each new iteration of FITS must always be backward compatible with every other iteration of the system. On top of that, FITS-files are predominantly used within astronomy and rarely seen elsewhere. This results in the format not living up to conventional modern standards while being well suited for what it was designed for.

When it comes to how the data is stored in a FITS-file one can think of it as a matrix where each cell of the matrix represents a pixel in the image. Each pixel value is represented by a floating point number that represents the amount electromagnetic radiation detected at that specific pixel when the observation was made. The FITS-images carry no information of how they are to be visualized. It is up to the viewer to interpret and color code the FITS-files in such a way that the features of interest are viewable. It is worth noting that the color coding used to visualize a FITS-file does not alter the data in any way and therefore does not change the way the FITS-file is interpreted by the CNN.

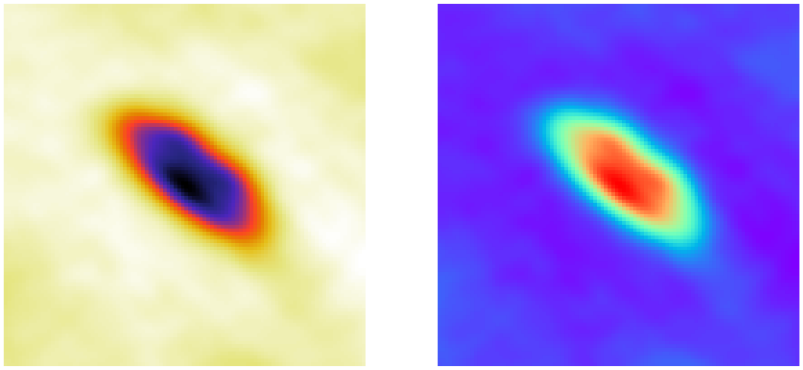


Figure 2.3: Two different color palette representations of the object HH212. Color palette on the left image: "CMRmap_R". On the right: "Rainbow". NB: The only difference in between the two figures are the color pallete.

The choice of color scale palettes is mostly dependant on personal preferences. The developers of this project has chosen the color scale palette "CMRmap_R" as seen in the left image in figure 2.3 [5].

[5]The image was one of the initial candidates mentioned in Dust In Wind (Shoemaker et al., [n. d.]) and extracted from the ALMA archive using ALminer, see section 2.3.1, while taking into account the data features detailed in 2.3.2.

2.4 Neural Networks

Neural networks, commonly referred to as artificial neural networks (ANNs), are computing systems inspired by the structure and function of the human brain (Stewart, 2019). They are capable of learning and can be used to recognize patterns, make predictions, and solve problems. ANNs consist of an interconnected group of nodes, also known as artificial neurons, which are organized into layers, producing the human-like behaviour.

For this project, a subspecies of neural networks have been used; Convolutional Neural Networks, for the classification of astronomical images. Therefore, this section aims to introduce the topic of neural networks, covering a brief history, different types, architecture, training, and evaluation.

2.4.1 History of Neural Networks

A brief history of neural networks can be traced back to the 1940s when Warren S. McCulloch and Walter Pitts published "A logical calculus of the ideas immanent in nervous activity," (IBM, [n. d.]) which aimed to understand how the human brain is able to create complex networks by connecting neurons. It was not until 1958 that Frank Rosenblatt developed the perceptron, which introduced weights to the equation and allowed a computer to differentiate between cards that are marked on the left side versus those that are marked on the right side.

Since then, neural networks have been widely studied and applied in various fields, from targeted marketing by social network filtering and behavioral data analysis to financial predictions by processing historical data of financial instruments (AWS Amazon, [n. d.]).

2.4.2 Convolutional Neural Networks

Neural networks come in various types, each with unique applications and architectures for various use cases and data types. Some of the commonly used networks include recurrent neural networks, feed-forward neural networks, modular neural networks, and convolutional neural networks (McGregor and freeCodeCamp, 2021). CNNs are an extension of artificial neural networks and are predominantly used for image recognition-based tasks (Madhava and IBM, 2021). They are inspired by the biological visual cortex and have bypassed standard computer vision, producing cutting-edge results. CNNs have had a great deal of success in real-life case studies and implementations, including image classification, object detection, segmentation, face recognition, and crystal structure classification (Sharma, 2017).

2.4.2.1 Architecture

CNNs are particularly well-suited for image recognition-based tasks due to their ability to automatically learn and extract relevant features from images. The process starts with the input image being passed through a series of convolutional layers, where each layer applies a set of filters (also known as kernels) to the input image or the output of the previous layer. These layers are visualized in figure 2.4.

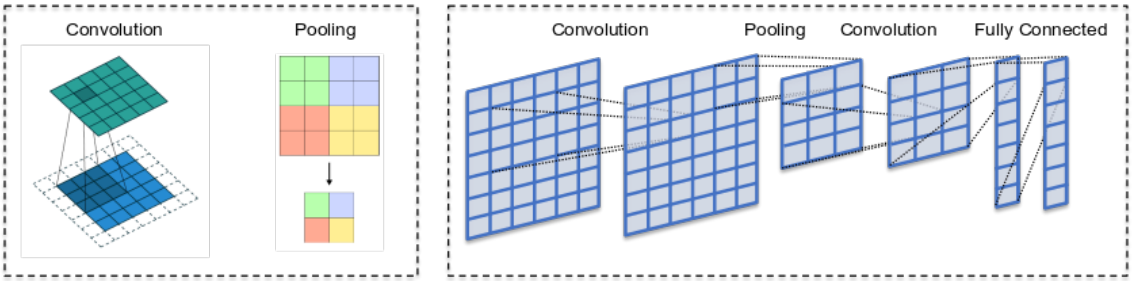


Figure 2.4: Structural behavior of convolutional and pooling layers provided by Maier ([n. d.]).

These filters are responsible for detecting different patterns or structures in the image, such as edges and textures (Izadkhah, 2022). The weights of these filters are learned during the training process, allowing the network to adapt and optimize its feature extraction capabilities based on the given data set. As the information flows through successive convolutional layers, the network learns to recognize increasingly complex and abstract patterns, effectively capturing relevant features for the classification task (Izadkhah, 2022), see figure 2.6 for a visualization of how the information flows. The weights of the network is visualised by w_1 and w_2 in figure 2.5 and consist of values in the real network. Thresholds to these values help produce a decision-like behaviour of the CNN.

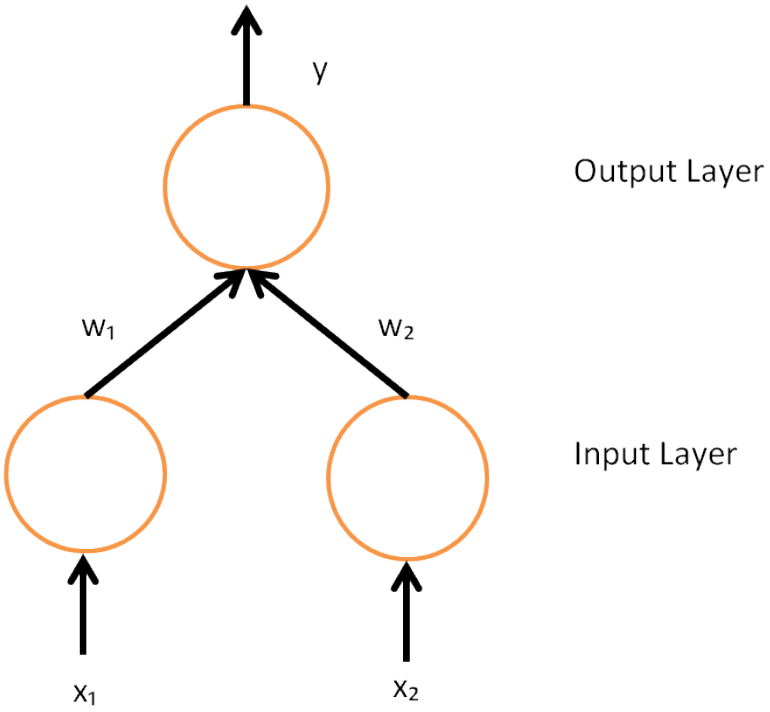


Figure 2.5: A simple neural network with two input units and one output unit provided by AI456 ([n. d.]).

Along with convolutional layers, CNNs also incorporate non-linearity layers, which are crucial for introducing non-linearity into the model. These layers, often called activation layers, are applied after the convolutional layers and help the network learn complex, non-linear relationships between the input data and the output. The way this is done is that the activation function perfoms a predefined mathematical expression on a single value and then selectively determines if the value should pass trough to the next layer. One commonly used activation function is the Rectified Linear Unit (RELu). The RELu function computes the maximum value between zero and the input, i.e. $f(x) = max(0, f)$, which results in an output equal to zero when the input value is negative and the input value when the value is positive (Stanford Visual and Learning Lab, [n. d.]).

Furthermore, in addition to convolutional layers, CNNs also utilize pooling layers which is visualised in figure 2.4, which reduce the spatial dimensions of the feature maps. Pooling layers work by aggregating neighboring pixel values into a single value, effectively compressing the representation and making the network more resilient to minor variations in the input data (Edge AI + Vision Alliance, 2015). This downsampling process helps the network focus on the most important features while

reducing computational complexity and the risk of overfitting [6]. One could think of it as a grid of numbers representing the pixel values in an image or a feature map. Pooling layers help to condense this grid by taking smaller sections of the grid and summarizing them into a single number. This summary can be done using different methods, such as taking the maximum value (max pooling) or the average value (average pooling) of the selected section. This process allows the CNN to focus on the most important features and patterns in the image while being more robust to variations in object appearance (Brownlee, 2019).

Together, the convolutional and pooling layers in a CNN form a hierarchical feature extraction network that automatically learns to detect and extract relevant features from the input images. This automatic feature extraction capability makes CNNs an ideal choice for image recognition tasks, as it eliminates the need for manual feature engineering and allows the network to adapt to different data sets and classification problems (Izadkhah, 2022).

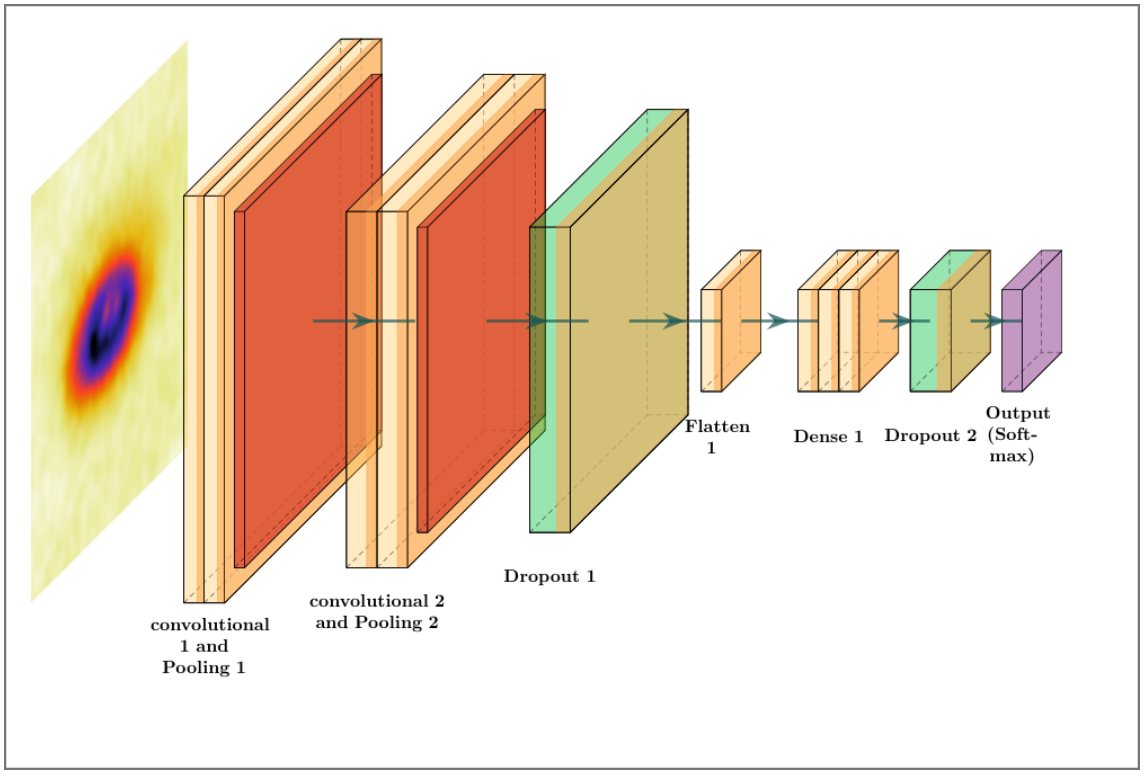


Figure 2.6: A convolutional neural network with different layers visualized. The red rectangles at the end of both of the convolutional layers indicates a pooling layer right after.

[6] Overfitting is a concept that occurs when a model fits the training data too closely; the model memorizes the training data too specifically. When overfitting occurs, the model will perform poorly on new, unseen data (IBM, [n. d.]).

As stated before, convolutional neural networks typically consist of a series of convolutional and pooling layers for feature extraction, followed by additional layers for further processing and classification. One such layer is the Dropout layer, which aims to reduce overfitting in the network by randomly deactivating a certain percentage of neurons during training (baeldung, 2023). This technique urges the network to learn more resiliently and gain emergent[7] features since it cannot heavily depend on any individual neuron.

Following the aforementioned layers, it is necessary to flatten the feature maps into a singular vector before proceeding to the fully connected layers, which are also referred to as Dense layers. The transformation of the feature maps into a singular vector is carried out by the Flatten layer (Mishra, 2020).

The flattened vector is then fed into the fully connected layer which functions as a crucial link between the previous layers and the output layer. Its main purpose is to merge and comprehend the advanced characteristics extracted by the preceding layers in the network. This is achieved by establishing connections between each neuron in the layer and every neuron in the preceding layer, thereby creating a fully connected network. This pattern of connectivity enables the network to grasp non-linear complex associations between the input features and the output (Unzueta, 2022).

The last component of the architecture in a CNN is typically a fully connected layer or a dense layer, followed by an output activation function. A visualization of the CNN architecture is shown in the figure 2.6 [8]. Usually one would use the sigmoid function when performing binary classifications, and the softmax function when dealing with multi-class classification. However, one could also use the softmax function when dealing with binary classification. The softmax classifier then simplifies to a binary softmax classifier. The sigmoid function maps the output to a probability score between 0 and 1. A single output neuron represents the probability of the input belonging to one of the classes. The probability of the other class can be calculated as 1 minus the output probability. The binary softmax classifier on the other hand, is used when there are two classes, but you want the output probabilities to be explicitly represented for both classes. The softmax function converts the output into probability scores for each class, ensuring that the sum of the probabilities equals 1. In summary, both options can be used for binary classification tasks, but they have different output representations. The choice between them depends

[7] "Emergent properties are properties that become apparent and result from various interacting components within a system but are properties that do not belong to the individual components themselves." (Comunale, [n. d.])

[8] The figure 2.6 is actually a representation of the CNN used in this project.

on the use case and how one wants to represent the output probabilities (Stanford Visual and Learning Lab, [n. d.]; Keras, [n. d.]).

2.4.3 Supervised learning

Supervised learning is where one supervises the training by having the model train on a labeled data set[9] (Delua and IBM Analytics, 2021). When doing binary image classification, supervised learning would be the case where each input image has an associated output label. By letting the model use labeled input images, accuracy and other metrics can be measured. Thus, the model can learn over time because the wrongful predictions are quantifiable. Analogously, one could think of supervised learning like a teacher who provides students with correct answers before the exam, and the students utilize these answers in order to learn how to solve similar types of problems.

2.4.4 Training a Convolutional Neural Network

To begin the process of training a CNN, it is necessary to define the network architecture. This involves determining the number of layers, the type of layers (such as convolutional, pooling, fully connected, and so on), and the connections between them. But in order to begin training the model, it is necessary to prepare the data. This is usually done by collecting a data set that includes images and their corresponding labels, i.e. the correct output values assigned to each input image, see subsection 2.4.3. The data set should be divided into training and test sets, with the former being used to train the model and the latter to evaluate its performance (McCullum, [n. d.]).

Once the data collection is done and the architecture is defined, the next step is to initialize the weights and biases of the network which are essential components of a CNN. Weights refer to the values that are assigned to the connections between the neurons in the network (Raitoharju, 2022). The weights are primarily found in the convolutional layers, where they define the convolutional filters or kernels that are applied to the input data. They dynamically change as the network learns in order to enhance the decision process. As the training progresses, the CNN acquires understanding of the characteristics of the input data and the network adapts the weights (Teuwen and Moriakov, 2019). In contrast, biases are values that are added to the output of every neuron prior to the activation function being applied. Their contribution is extremely important in modifying the activation function, which greatly improves the networks capability to adapt to data and enhances its overall

[9] In this case, labeled data set refers to the process of annotating images with specific labels that represent the category or class to which the object belongs, such as positive or negative.

performance (Raitoharju, 2022; Teuwen and Moriakov, 2019). Similar to weights, biases are also learned gradually throughout the training process and are updated in order to reduce the discrepancy between the anticipated output and true labels (D'Agostino, [n. d.]).

But how are parameters such as weights and biases learned by the network? This is done trough the use of loss functions and optimizers. Loss functions are the quantifiable measurement of the model's error. Early in the training of the model it is possible that the untrained network may not provide accurate predictions. In order to improve the performance, the aim is to minimize the loss function, which measures the difference between the predicted result and the desired correct labels. To determine the loss, the inputs from the data is compared to the correct data label value (PyTorch, [n. d.]). Thus, providing a way to quantify the error of the model. There are several different loss functions available with different use cases. Some of the common ones include Mean Squared Error (MSE) for regression tasks, Binary Cross Entropy for binary classification and Categorical Cross Entropy for multi-class single label classification (Raitoharju, 2022).

On the other hand, the loss function would not be minimized if there is no change in the network before the next iteration. This is where the optimizer function comes into play. The optimizer function in a CNN is critical in assisting the model to learn from data by iteratively increasing its weights based on a particular loss function. Specifically, the optimizer takes in the current weights and gradient values (i.e. how much the loss has changed as a result of tiny changes in the weights) throughout each training iteration and utilizes this information to update the weights toward a new value that aims to minimize the overall loss. Some common optimizers used in convolutional neural networks include Stochastic Gradient Descent (SGD), Adagrad, RMSprop and Adam (Raitoharju, 2022).

Additionally, it is necessary to assign values to certain hyperparameters. These hyperparameter include learning rate, which determines the rate of how the model adjusts its parameters with respect to the loss function. It is important to find the right balance, as setting it too high or too low can results in poor performance of the model. Another important hyperparameter is the number of epochs, which determines how many times the model will go through the training set. Setting this value too high might lead to overfitting, while a too low value might result in a too generalized model. Lastly, the batch size determines the number of samples processed at each iteration, and the optimal size depends on the specific task at hand (Zvornicanin and baeldung, 2023). Even though learning rate, number of epochs, and batch size may not be as central to the performance of a CNN as its architecture, they are still important factors to consider when training a model.

To summarize, the process of training a CNN involves defining the network architecture, initializing weights and biases, computing the loss function, and updating weights and biases using an optimization algorithm. This process requires careful monitoring of the model's performance to achieve the best possible results through iteration.

2.4.5 Display and Evaluation of Performance

The following section describes how to determine the performance of a CNN in the binary class case [10]. In figure 2.7 an example of a confusion matrix is displayed. It shows the true labels and the classified labels that the model has placed. It results in four different categories that the labeling can have. A true positive (TP) is when the label was positive and classified as positive. False positive (FP) corresponds to the situation where a negative label was classified as positive. False negative (FN) corresponds to a positive label with a negative classification and lastly, true negative (TN) represents a negative label being classified as negative. They display the probability of its labeling capabilities, and are therefore in the range of 0-1. FP is a type 1 error and FN is a type 2 error, and should be kept as low as possible whilst TP and TN should be high (Narkhede, 2018).

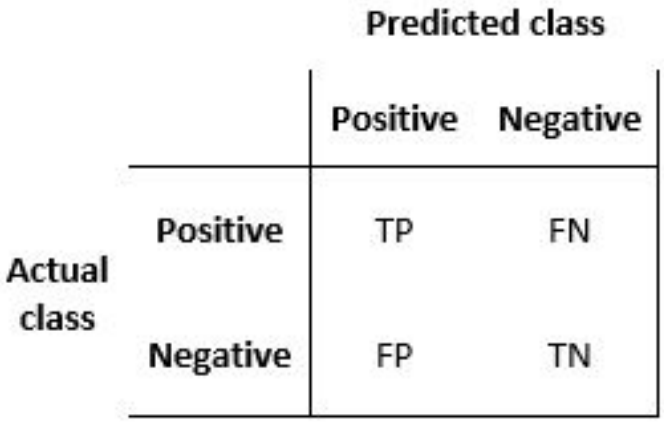


Figure 2.7: Structure of a confusion matrix provided by Errachete ([n. d.]).

From such a representation, three metrics can be extracted: recall, precision and accuracy. Recall indicates how many positive classed images were correctly classified. Precision shows how many of the classified positives that actually were positive. The last metric, accuracy, describes how many of both negative and positive class that were classified correctly. What they all have in common is that the higher the metric

[10] There are multi-class generalizations of these performance calculations, see Bex (2021) for more information on this topic.

value, the better the model performs (Narkhede, 2018). Calculation of the metrics is made with the help of three different formulas, see equations 2.1, 2.2 and 2.3.

$$Recall = \frac{TP}{TP + FN} \tag{2.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

$$Accuracy = \frac{TP + TN}{Total} \tag{2.3}$$

$$F - measure = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision} \tag{2.4}$$

Additionally, a comparison score can be constructed with the capabilities of measuring both recall and precision simultaneously, whilst punishing extreme values, see equation 2.4. It is called F-measure, and it achieves this by replacing the Arithmetic Mean with Harmonic Mean (Narkhede, 2018). Harmonic Mean is commonly used in information retrieval with ratios such as recall and precision. The difference is that it divides the number of data points by the reciprocal of each value. It is often called the true average when rates are involved, and is therefore preferred over Arithmetic Mean (DeepAI, 2020).

2.4.6 TensorFlow

TensorFlow is a machine learning system designed for building and training machine learning models (USENIX Association. et al., 2005). Google launched the system in 2015 and have been carefully developing it since. TensorFlow is free to download, install and use, and because of its extensive user base, it comes well documented on both their own website as well as on programmer forums such as StackOverflow (2023). The most prominent features of TensorFlow is the ability to develop and design neural networks, including deep learning models. It is available on a wide range of programming languages such as Python, C++, and Java (Abadi et al., 2015). In this project, it acts as the underlying service for the entire CNN. This project would not have been achievable without this tool given timeline, expertise and resources[11].

[11]There are other machine learning systems that have similar applicabilities, such as PyTorch (2023), but TensorFlow was the one that was used in this project.

2.5 Image Augmentation

FITS-files retrieved from the ALMA archive have widely different shapes, sizes and contents. These differences need to be accounted for, since a CNN will only work on images of a fixed size and shape. Therefore, being able to transform a wide range of different sized images into a standard form is critical for the projects success.

In order to avoid overfiting on the training data, mentioned in subsection 2.4.2, CNNs need large and diverse training data sets (IBM, [n. d.]). Image augmentation is commonly used to increase the size of the training data set to improve accuracy of classification on new data. It is a technique where existing images are copied and augmented by some transformation that changes the data of the image (Saxena, 2021). The transformations can be divided into two main categories, linear transformation and non-linear transformation. The figure 2.8 displays some examples of linear transformations.

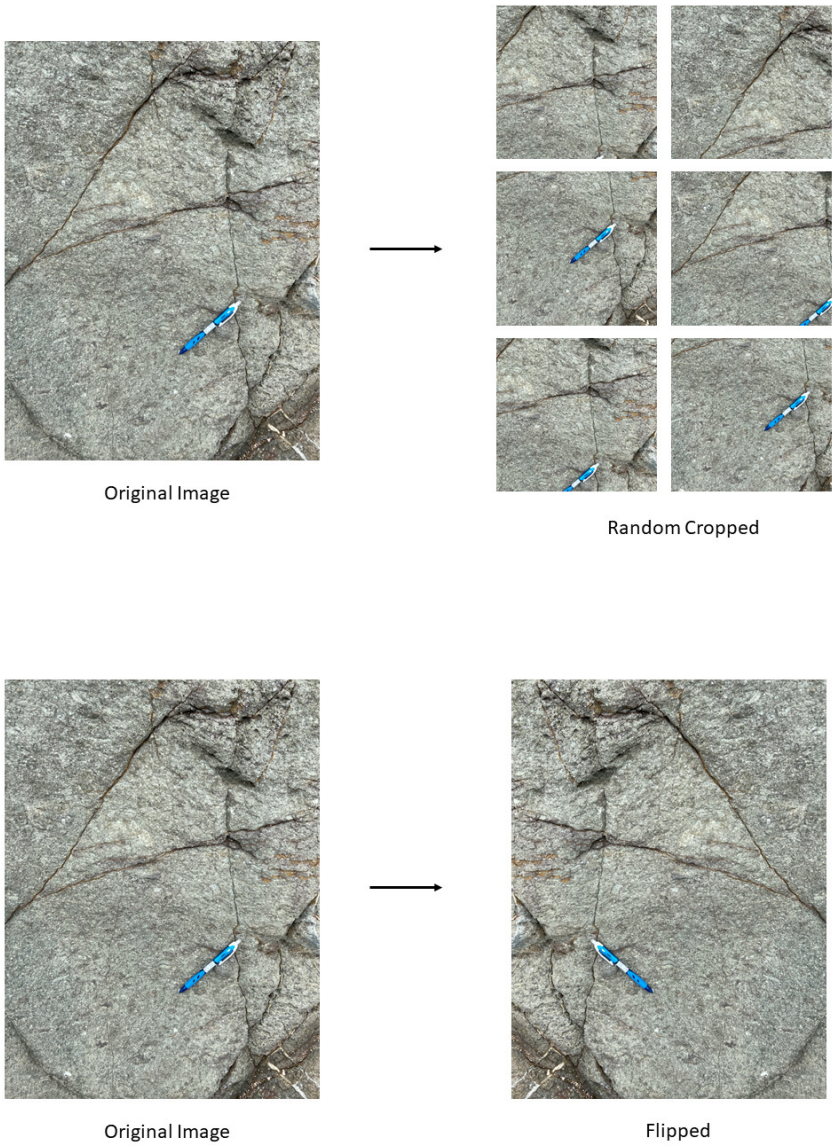


Figure 2.8: Example images of linearly augmented images provided by TseiKiChun under the license CC 4.0 TseKiChun ([n. d.]).

2.5.1 Linear Transformation

Formally, a linear transformation is a function $F : R^n \to R^m$ that satisfies additive and scalar multiplicative properties as shown in equation 2.5, where α, β are scalars, and x,y are vectors (Adams and Essex, 2017).

$$F(\alpha x + \beta y) = \alpha F(x) + \beta F(y) \tag{2.5}$$

Linear transformations are generally safe to be applied automatically since they do not tend to change the class of the image; the features of the image keep their linear relationships. This is true as long as the features are still contained in the image.

Examples of some linear transformations are: cropping, rotating, resizing and inverting, which can be seen in figure 2.8. Cropping is essential to isolate objects and to make data uniform for the CNN. Rotating an image allows alignment for consistency between images regarding orientation, or simply for augmentation. Resizing changes an image so that the contents are enlarged or made smaller while the aspect ratio of the image remains constant. Finally there are two ways of inverting an image, left-right or up-down. Left-right inverts the pixels along the Y-axis and up-down inverts along the X-axis.

2.5.2 Non-Linear Transformation

When performing a non-linear transformation on a positive class FITS-file, there is a higher risk that the transformation changes the class of the image. Essentially, non-linear transformation changes the features of the image and therefore one cannot assume that it retains its class.

With these risks in mind, the motivation for non-linear augmentations is that the data trained on is non-linear; the data cannot be classified by linear relationships. The linear transformations still add value as the model learns that rotations, mirroring etc of the same data does not matter, but it does not add new information to train on. Non-linear augmented images can however add new data with new inductive biases (Gavneet Singh Chadha, 2019) that generalizes a CNN, improving its classification ability of new, unseen data.

2.5.2.1 Geometric Mean Square

Geometric Mean Square is one technique to combine two images into a new unique image. It works by taking the square root of the product for each corresponding pixel value of two images, as shown in 2.6, where A and B are matrices.

$$F(A, B) = \sqrt{AB} \tag{2.6}$$

3

Methods

This chapter describes the prominent steps in realizing the CNN briefed in section 1.1. The order of the information is in chronological order of the CNN's process and aims to guide the reader naturally along the finalization of the project. It starts by explaining the retrieval of data from the ALMA archive, to annotating the data as either positive or negative with explanations regarding the qualitative reasoning behind. The following sections clarify the preprocessing of the data before finally describing the CNN itself, how it is trained, evaluated, and applied.

Research that combine FITS-files with machine learning is scarce. The work that were found had objectives that differed from this project's, therefore the approach for this project has been to adapt general techniques in machine learning and data handling to a set of new circumstances.

3.1 Data Retrieval

The data was downloaded from the ALMA archive using a Python toolkit, ALminer 2.3.1, that accesses the ALMA archive by code. The utility interconnects the archive to the user and has been fundamental in carrying out this project, as it allowed filtering and downloading useful data for our project in a user-friendly manner. The coming sections are intended to give the reader an insight as to how the retrieval is done and on what parameters the choice of data is built upon.

3.1.1 Querying and Extracting Data

For this project, the following scientific keywords was used to filter out the ALMA data most likely to contain observations of protostellar outflows.

- Low-mass star formation
- Intermediate-mass star formation
- Outflows, jets, feedback
- Outflows, jets and ionized winds
- Inter-stellar medium (ISM)/molecular clouds

After the above filtering, the archive was filtered by the following extra conditions.

- Release date: 2016 and later.
- Angular resolution: 0.4 arcseconds.
- File name must include: ".pbcor", "_sci", ".cont".
- Archive mirror: "NAOJ".

The filtering of release date was set up to discard data from the very first cycles of ALMA, during which the ALMA pipeline was not yet deployed and the output products were heterogeneous in their content and file naming. Given the large number of projects with high-quality data that satisfy our data release criterion, exploring data obtained before 2016 has not been a priority. The conditions on the angular resolution was simply an approximation of how high of a quality the image had to be in order to be able to detect any extensions possibly indicating dust in the wind. The included strings in the file name was added with different intentions. With the ".pbcor" extension, the intention was to filter by the primary beam correction described in subsection 2.3.2.1. ".cont" is an extension intended to filter by the image aggrevation described in subsection 2.3.2.2. Lastly, "_sci" was added in order to get the scientifically calibrated data described in subsection 2.3.2.3.

3.2 Annotating data

In supervised learning, annotating the data before feeding it to the network is fundamental for the learning process, see section 2.4.3. The procedure includes manually labeling each image that is being used for training data as part of a certain class. The data from ALMA archive comes in many different variations and file modifications. These variations are dependent on, among other things, the configuration of the telescope at the time of the observation and that of the file handling done after an image is taken. In some cases, the same object might be targeted by the telescope with different settings. These settings can affect the outcome of the annotations. This section describes the annotation process done by the developers of this project.

3.2.1 Positive vs. Negative

A total of 825 images FITS-files were downloaded from the ALMA archive. All of these were validated via inspection from multiple people in the project. The images were annotated with the previously mentioned factors in mind and with help from illustrations provided by the supervisors, shown in figure 2.2. The images of whose class were uncertain were moved aside to be inspected by our more knowledgeable supervisors. In the process multiple images that had been downloaded from the archive were found to have the characteristics of the positive class and were used as positive training data. The images that did not show any signs of extensions were used as negative training data.

Each image is taken with a specific calibration to ALMA. This calibration strongly affects the outcome of the image, primarily the quality of detail. When the same object is targeted by the telescope using different settings, the images might differ considerably. With differences in spatial and spectral resolution, integration time of the detector, etc, the outcome can affect the quality of the image and the properties of the features in them. The figure 3.1 shows the object NGC_1052 where the observational settings affected the images enough for them to differ in clarity.

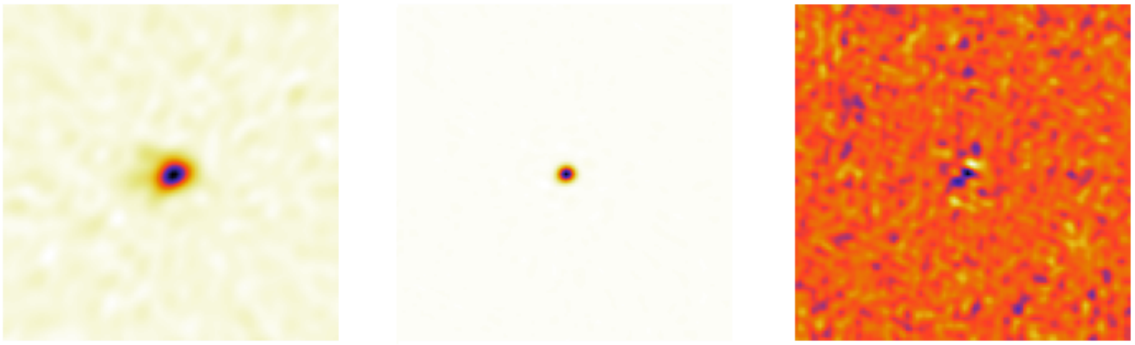


Figure 3.1: Three ALMA images of the same object NGC_1052 showing different quality and signs of extension. The images were obtained by different ALMA projects on the same source, but observing in different frequency bands, different angular resolution, different sensitivity, and polarization products. Left and middle panels show the total intensity image (Stokes I), but observed in different ALMA Bands (frequency, wavelength, etc), angular resolutions and sensitivity. An image from one full-polarization project displaying the Stokes Q (related to the linear polarization of the light) is also included (right panel). Left: ALMA image from observation with UID A001_X13f_X140. Middle: ALMA image from observation with UID A001_X133d_Xcf9 in Band 4 total intensity (Stokes I). Right: ALMA image from observation UID A001_X133d_Xcf9 in Band 4 Stokes Q.

Here, the difference in observational setup affects the quality of the signs of extensions in the images. In the right image, distinguishing an object and possible signs of extensions to this object is nearly impossible, leading the annotators to label it as negative class. Meanwhile, the figure in the middle shows no signs of extensions despite being quite clear. Making the annotators to annotate it as negative. The left image shows signs of extensions though, leading the annotators to label it as positive.

3.3 Data Set

The CNN in this project uses binary classification which implies that all FITS-files can be considered one of two classes; positive or negative. Positive class was set to be FITS-files that contain a protoplanetary disk with extensions in the shape of winds. The negative class is anything that is not positive class.

3.3.1 Positive Data Set

Initially, the project had only eight FITS-files, see appendix A, of positive class. This lack of positive FITS was the reason for one of the largest hurdles in this project. It is common for a CNN to be trained with thousands of both positive and negative data points. There are several techniques to create an artificial influx of training data, as previously stated in section 2.5. This project focuses on image transformation.

The main strategy was to combine the eight positive images in as many ways as possible. In the first step the positive FITS-files were centered, rotated and resized so that the Gaussian disk was in the center, aligned along the same axis and of the size 100x100 pixels. This ensures that the geometric mean square of any two images will have a high likelihood of being of positive class since the extensions line up in the correct orientation relative to the disk. The orientation of the extensions also remain aligned if each image is inverted along the X-, Y-, and XY-axis. Each of these images were combined by the geometric mean square and then manually sorted by hand to ensure that only positive FITS-files were present in the final positive data set.

3.3.2 Negative Data Set

In the ALMA archive, images of the negative class were more abundant than those of positive, therefore transformations and augmentation was not necessary. The algorithm works by finding the disk in a given FITS-image and taking a 100x100 pixel crop with the disk in the centre. The centre of the disk is found by found by taking

the brightest pixel in the middle 50% of a given FITS-file. This ensures that noise generated as a result of PBCOR, see subsection 2.3.2.1, will not be misidentified as a disk. Each image also went through manual inspection as described in section 3.2

3.4 Implementation of the Convolutional Neural Network

This section contains the implementation details of the Convolutional Neural Network (CNN). It covers the model's architecture, the selection of hyperparameters, and the reasoning behind some choices that were made. This aims to provide a comprehensive understanding of the approach taken to construct the CNN and its application in resolving the issue at hand.

3.4.1 Architecture

The CNN model was built and defined as a function 'model()' that takes an activation function as an input argument, with a default value of 'softmax'. The architecture of the model consists of several layers that are commonly used in Convolutional Neural Networks (CNNs). The first layer is a 2D convolutional layer with 32 filters, each with a kernel size of 3x3. This layer learns low-level features, such as edges and textures.

As described above, the activation function is set to the input argument, and the input shape is set to (100, 100, 1) for images of size 100x100. The next layer is a max-pooling layer with a pool size of 2x2, which reduces the spatial dimensions by a factor of 2. The model then includes a second 2D convolutional layer with 64 filters and a kernel size of 3x3, using the ReLU activation function. The aim of this layer is to learn more complex, high-level features that can distinguish between the two classes. Another max-pooling layer with a pool size of 2x2 follows. To reduce overfitting, a dropout layer with a dropout rate of 0.25 is added. The model then includes a flatten layer to convert the feature maps into a 1D array. A dense (fully connected) layer with 128 neurons, using the ReLU activation function, is added. Another dropout layer with a dropout rate of 0.5 is included. Finally, the model includes a dense layer with 2 neurons, using the softmax activation function, as specified in the input parameter 'activation' of the model.

In summary, the model includes two 2D convolutional layers, two max-pooling layers, two dropout layers, two dense layers, and a flatten layer. The ReLU (Rectified Linear Unit) activation function is used in the convolutional and dense layers, as it introduces non-linearity and helps the network learn complex patterns. The model

is designed to classify images of size 100x100 into two classes. The first convolutional layer extracts 32 features from the input image, and the second convolutional layer extracts 64 features from the output of the first convolutional layer.

3.4.2 Evaluation

The model evaluation is performed using the *evaluate_model*() function, which takes the training and testing data, the model, the loss function, learning rate, optimizer, and evaluation metric as input arguments. The default values for these parameters are *binary_crossentropy* for the loss function, 0.001 for the learning rate, *Adam* for the optimizer, and *accuracy* for the evaluation metric. The model is first compiled using the input parameters. Then, it is trained using the $fit()$ method with a batch size of 2, for 30 epochs. the model's performance is evaluated on the testing data using the *evaluate*() method.

3.4.3 Choices of Parameters and Functions

The filter sizes of 3x3 were chosen for both convolutional layers. Smaller filter sizes, like 3x3, are preferred because they can capture local patterns in the images, and they require fewer parameters compared to larger filter sizes (e.g., 5x5 or 7x7). A pool size of 2x2 was chosen for both max-pooling layers. This size is a common choice because it effectively reduces the spatial dimensions of the feature maps by a factor of 2, while still retaining important information. Two dropout layers with rates of 0.25 and 0.5 were added to the architecture. These rates were chosen as a balance between preventing overfitting and preserving the network's capacity to learn. A batch size of 2 and 30 epochs were chosen for training the model. Smaller batch sizes can provide better generalization and faster convergence, but at the cost of increased training time. The number of epochs was chosen to allow the model to learn the patterns in the data without overfitting. Increasing the number of epochs might improve the model's performance on the training set but could also lead to overfitting.

Multiple tests were conducted on the same data to select the best combination of optimizer function and hyperparameters, namely learning rate, batch size, and number of epochs, to achieve optimal results on the training data. Five different optimizers were tested, including Stochastic Gradient Descent (SGD), RMSprop, Adagrad, Adadelta, and Adam, with various learning rates evaluated for each optimizer. The loss function used when performing the tests were binary cross entropy. Additionally, adjustments were made to the batch size and number of epochs to ensure the best combination for the model's performance. The reason for testing different optimizers is that each optimizer has its unique approach to updating the

model's parameters during training, and it is crucial to test multiple options to find the best fit for the given task at hand.

4

Results

The CNN was supplied with augmented data to fulfill a necessary amount of training data, and examples of the end result from this process will be shown. After several epochs of training, the CNN returned a series of images that it classified as positive. In this chapter, the neural network's performance and accuracy along with the positive observations will be showcased.

4.1 Training Data

The results from the extraction of data is presented, with given numbers of observations. With the few known positive class sources, there was a lack of data for the CNN to be trained on. Due to this, both linear and non-linear augmentations were made, see section 2.5. Below, some of them are presented with descriptions of what specific alterations were made.

4.1.1 Extracted data

In table 4.1 the keywords have the corresponding amount of observations that contains the data retrieved when querying in ALminer described in the first conditions stated in section 3.1.1.

Table 4.1: Scientific keyword filtering in ALminer.

Keyword	No. of observations after filtering
Low-mass star formation	ERROR (Zero obs.)[1]
Intermediate-mass star formation	610
Outflows, jets, feedback	1094
Outflows, jets and ionized winds	1294
Inter-stellar medium (ISM)/molecular clouds	2922
Total	5920

The table 4.2 shows how many of the observations stated in table 4.1 passed the second conditions stated in section 3.1.1.

Table 4.2: Scientific keyword filtering with conditions in ALminer.

Keyword	No. of observations after filtering
Intermediate-mass star formation	Zero observations
Outflows, jets, feedback	84
Outflows, jets and ionized winds	389
Inter-stellar medium (ISM)/molecular clouds	755
Total	1228

Of these 1228 observations, 359 of these were duplicates of others and were removed, landing in a total of 869 filtered observations to consider. Of these observations, each contained a great variety in amount of files that were going to be used for the project. Some contained several, some contained only one, and others did not contain any files that could be used for the project. The interesting files from the observations were the FITS-files, see section 2.3.3. The image data contained in these FITS files was read and stored as numerical arrays in Python. From these 869 observations, a total of 825 files were downloaded for both training, evaluation and testing the CNN. Many of them contain data from the same observation and/or object. The different observing setups and observing conditions at ALMA affect the properties and quality of the images, sometimes enough to make an impact the annotations, see 3.2.

4.1.2 Linear Augmentation

Different linear transformations can be applied to augment the training data set, explained in 2.5 and 3.3. A sequence of these can be seen in figure 4.1.

[1]The error of filtering by "Low-mass star formation" was discovered by the developers of this project and issued at the ALminer GitHub Toribio ([n. d.])

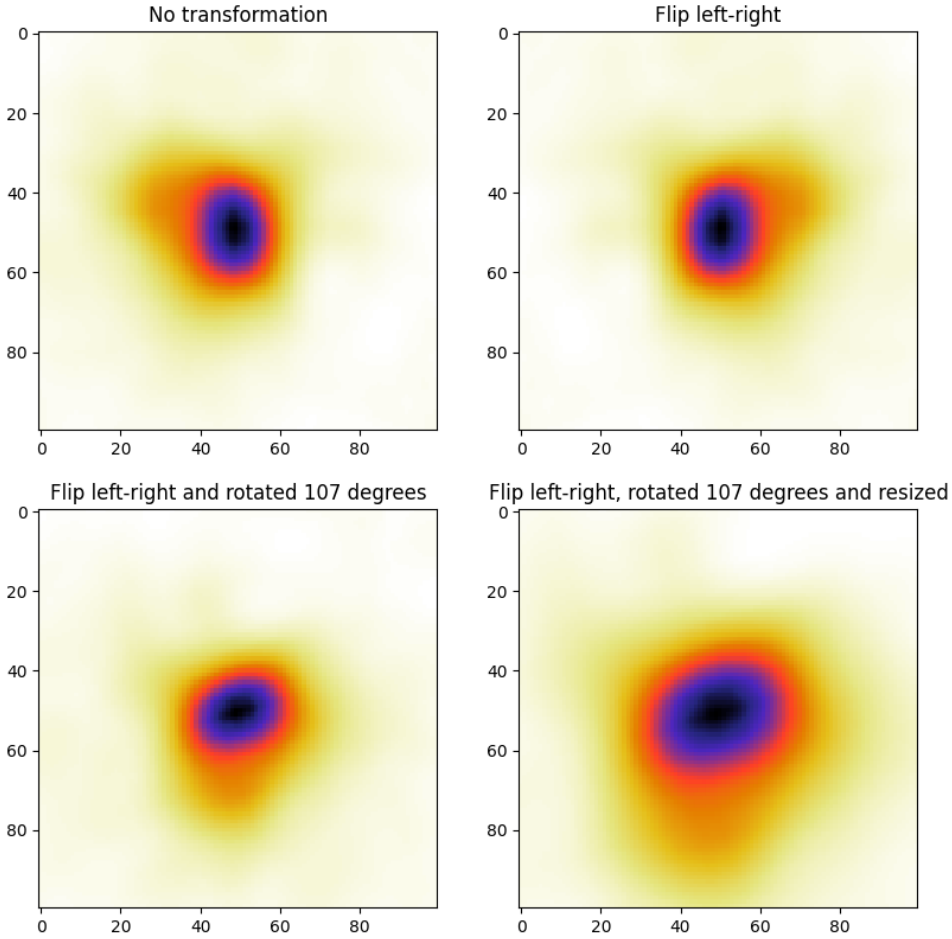


Figure 4.1: Linear transformations: flip, rotation and resizing. The upper left image is an artificially created disk with an extension.

4.1.3 Non-linear Augmentation

The only non-linear transformation used was the Geometric Mean, see subsection 2.5.2.1. Two examples of this augmentation can be seen in figure 4.2 and 4.3.

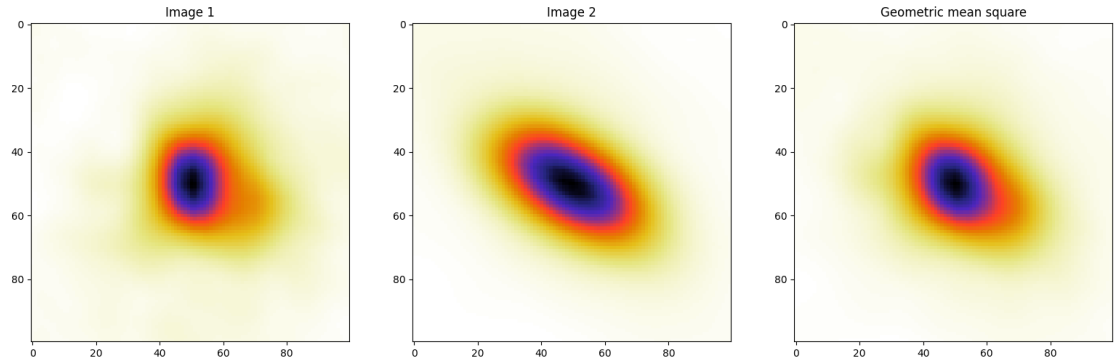


Figure 4.2: A Geometric Mean transformation with good result. Image 1 is from the source B335 and Image 2 is from HH212.

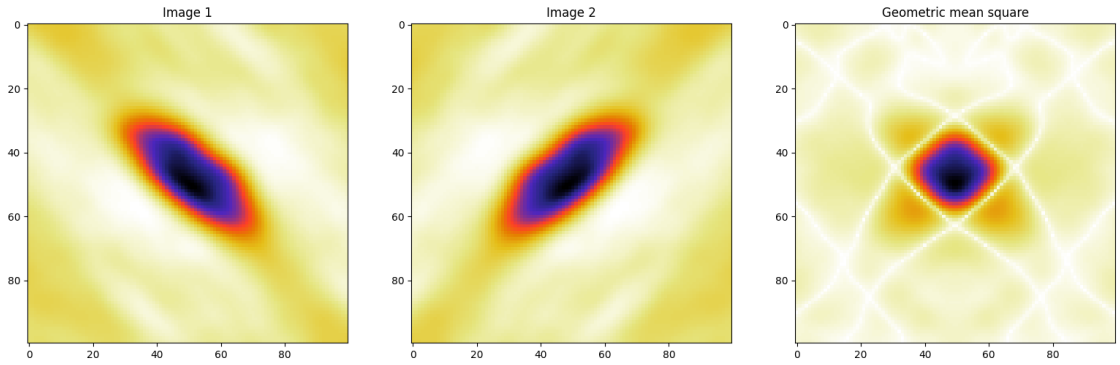


Figure 4.3: A Geometric Mean transformation with unnatural and therefore bad result. Image 1 and 2 are from the source HH212.

4.2 Convolutional Neural Network

Final execution of the CNN was run with the parameters shown in table 4.3 and 4.4, which gave the best results after several tests. These yielded a number of 37 positive classed images. The code for the results can be viewed at the following GitHub Repository https://github.com/nkatshiba/Alma-bachelor-project or at appendix C D E F

Table 4.3: Parameters for the CNN.

model.compile	
loss_function	binary_crossentropy
optimizer	Adam
learning_rate	0.001

Table 4.4: Parameters for the CNN.

model.fit	
batch_size	2
epochs	30

On the validation data, the performance of the network is described through a confusion matrix, see figure 4.4. As detailed in subsection 2.4.5, this brings three different metrics to calculate, as well as a comparison score.

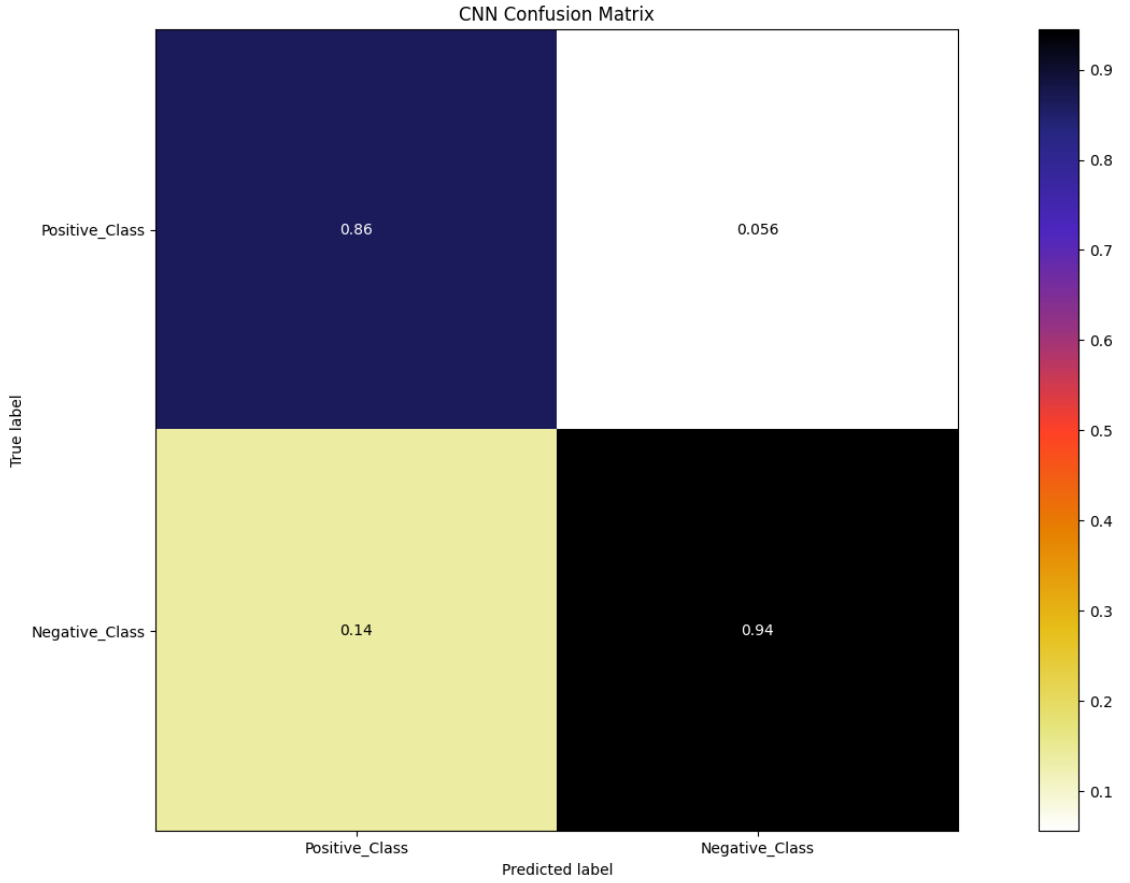


Figure 4.4: Confusion matrix for the model on validation data.

The confusion matrix gives us a TP of 0.86, a FP of 0.056, a FN of 0.14 and a TN of 0.94. A recall of 86% is found, see equation 2.1. What this means is that 86% of the positive class was predicted correctly. Looking solely at the positive class, 94% were actually positive, which is gathered from the precision's calculation, see equation 2.2. Lastly, an accuracy of 90% measures the amount of correctly predicted images from both classes, see equation 2.3. Additionally, the F-measure reached 90%, and can be used to compare this model to others, see equation 2.4.

4.3 Positive Classed Images

From the CNN, 37 images were positively classified. Seen in figure 4.5, 4.6 and 4.7 are the top three candidates for possibly depicting dust being carried out of a protoplanetary disk by wind. The rest of the results can be found in appendix B

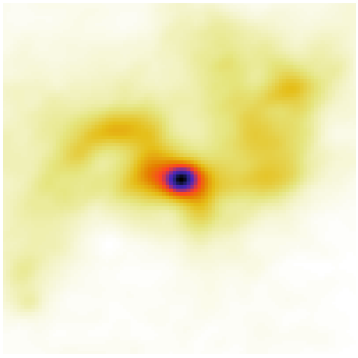


Figure 4.5: Observation G328.2551.

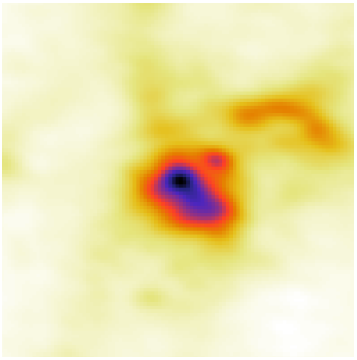


Figure 4.6: Observation G034.43.

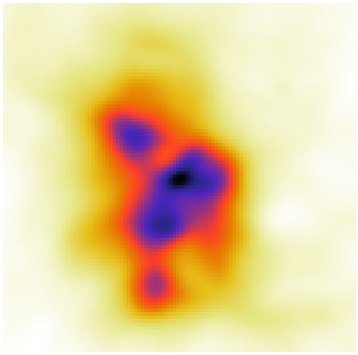


Figure 4.7: Observation NGC6334.tt1.

Sources of interest, that the CNN extracted observations of, are the following:

DG_Tau
M17-UC1
IC348
TMC1A
G10.3-0.1
hbc_494
AGAL337.916-00.477
AGAL332.826-00.549
IRAS_18566+0408
AGAL351.581-00.352
NGC6334I
NGC6334I_H2O
G034.43+0.24MM1_H2O
NGC_2264_CMM3
G328.2551-0.5321_A
GAL_005.88-00.41
BHR7
Serp_18
Serp_29
CrAus_01
CrAus_02
CrAus_05
CrAus_07
CrAus_08
CrAus_09
ChamII_01
Oph_34
Aql_13
HOPS_358

5

Discussion

In the following section, results, method and improvements in the different stages of the project will be discussed. This in terms of if the aim of the project has been reached and steps that can be taken to improve the results. Ways that the project can be taken further will also be discussed. These range from wider documentation to increased user friendliness.

5.1 Data Retrieval

The retrieval of data from the ALMA archive was successful within the framework of the project to that extent that data could be extracted for desired parameters. As shown in table 4.1 and 4.2 the keywords and filtration produced a list relevant observation, with the exception of the keyword "Low-mass star formation". The correlation between potential dust in the wind and low mass star formation is described in section 2.1, which makes it possible to believe that more relevant observation could be found using this keyword once the issue has been resolved.

5.2 Image Handling

The importing and processing of data to ensure that the CNN operated optimally met the requirements stated in the objective, see section 1.1. The algorithms always managed to find an object in the FITS-files and transform them into something the CNN could process. This is not to say that the methods used are without issues, in this section those improvement opportunities will be discussed.

5.2.1 Small FITS-files Compatibility

FITS-files come in different native sizes ranging from under 80x80 to over 8000x8000. The current implementation of the code will discard any images that are not over 800x800. The changes required to lower the compatibility to 200x200 images would

be simple and without any trade offs. After that, there would be trade offs but not impossible. One solution would be to pad the exceptionally small FITS-files with zeroes to artificially expand them. This would enable essentially all FITS-files to be used no matter the size.

5.2.2 Possible Mistakes in Annotation

When analysing which images have been used as positive or negative class to feed the network, some images were found to be labeled as the wrong class. The figure 5.1 shows two images of the same object but one of the images has been labeled as negative despite showing signs of extensions.

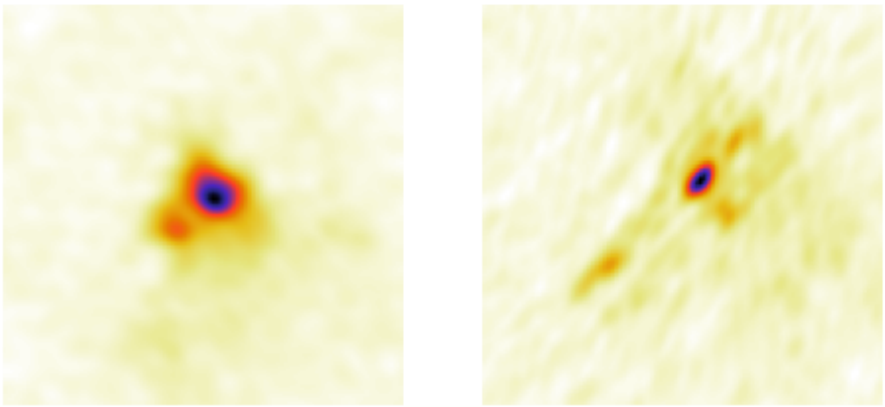


Figure 5.1: Left: The object S255_IRS3 labeled as positive due to signs of extensions. Right: The same object incorrectly labeled as negative.

The images depict the same object but they are taken on different occasions, i.e observations. Besides the fact that the observations are different, no further information explain any differences such as frequency ranges or integration time. Speculatively, the image on the right suffers from data reduction issues and could be improved by manual reprocessing by an expert to further remove instrumental effects if at all possible. Even though both images hints of extensions, only the left

was labeled and used as a positive class for the CNN, leading the authors to believe there might be other occasions where images were labeled incorrectly. This error is an affecting factor in how the network learns to distinguish interesting images from uninteresting ones, making it an improvement area where annotations could be done more thoroughly.

5.3 Bias

Because the classification of training data was manually done by mainly group members, who lack any astronomic background, unknown bias might be present that does not reflect real data. Bias could also have been induced by the geometric mean square algorithm as it changes the non-linear relationships between features. Potential bias could potentially have affected the accuracy and end result of the CNN. Unfortunately, biases are difficult to detect or measure.

5.4 Convolutional Neural Network

As figure 4.4 suggests, improving the CNN itself will probably not add any major benefits to the project as a whole. However, there are changes around the CNN that might allow it to classify more positive FITS-files, which are discussed in previous sections.

5.5 Images classified as positive by the Network

The CNN classified 37 images as positive, meaning that they are candidates to potentially contain images of dust in the winds, see section 4.3. These were displayed for the project's supervisors, who analyzed each image mentioned in the appendix B. Both supervisors expressed their interest in many of the images, increasing the candidate status for some of them. Hence, the aim of contributing to science explained in section 1.1 was accomplished to an extent. Some images were simultaneously discarded pretty quickly by the supervisors, indicating errors done by the network. The errors made showed new kinds of irregularities that were clearly not the winds we were looking for. We believe these misclassifications are a sign of the size of the training data being too small.

5.6 Documentation

For distribution purposes, an adequate documentation would have to be created. With one in place, it could enable new users to clearly understand the underlying

structure and use of the different features present. Another improvement this brings is better maneuverability and overview whilst improving upon the code itself. Since it acts as a guide, precise alterations could be applied by anyone. This might lead to easy customization on the user-side rather than specifically at the source code, which enables new applications of this project.

5.7 Further Work

In this section further work that can be done on the project will be discussed. They were not implemented for various reasons, ranging from time constraints to computational constraints.

5.7.1 User Friendliness

The plan was to present the project as a downloadable package with all dependencies included and an easy-to-use framework. Since it is supposed to be used by astronomers, who possibly lack sufficient experience in programming/optimizing CNNs, the user friendliness is an important factor. As it stands, different constants and parameters are separated throughout the code. It creates unnecessary strain on whoever wants to modify the details of the CNN. An alternative to the following solution could be to create a graphical user interface (GUI) for the preparation, execution and presentation of the CNN and its results.

Late in the process, an attempt of this was initiated, but remains unfinished. It included an interface to create training data, to create a CNN and to classify data on it. With links to local folders, it removes the struggle of locating the stored FITS-files and retrieving results from the CNN in the end. All it would take to review the obtained observations would be to open the specified folder and view the files.

Another added benefit of a GUI is the simplification of changing relevant parameters for the program by gathering all constants, parameters and options at the same place while keeping a clear overview. This would still require alteration of the code during set-up, but it is at least confined to a separate and collected space. As seen in figure 5.2, on the lower image, there are several options such as loss function and optimizer. To change these manually, you have to traverse the folder structure in the code and change each one by writing them out. Drop-down menus could replace that effort with all options listed and easily swapped.

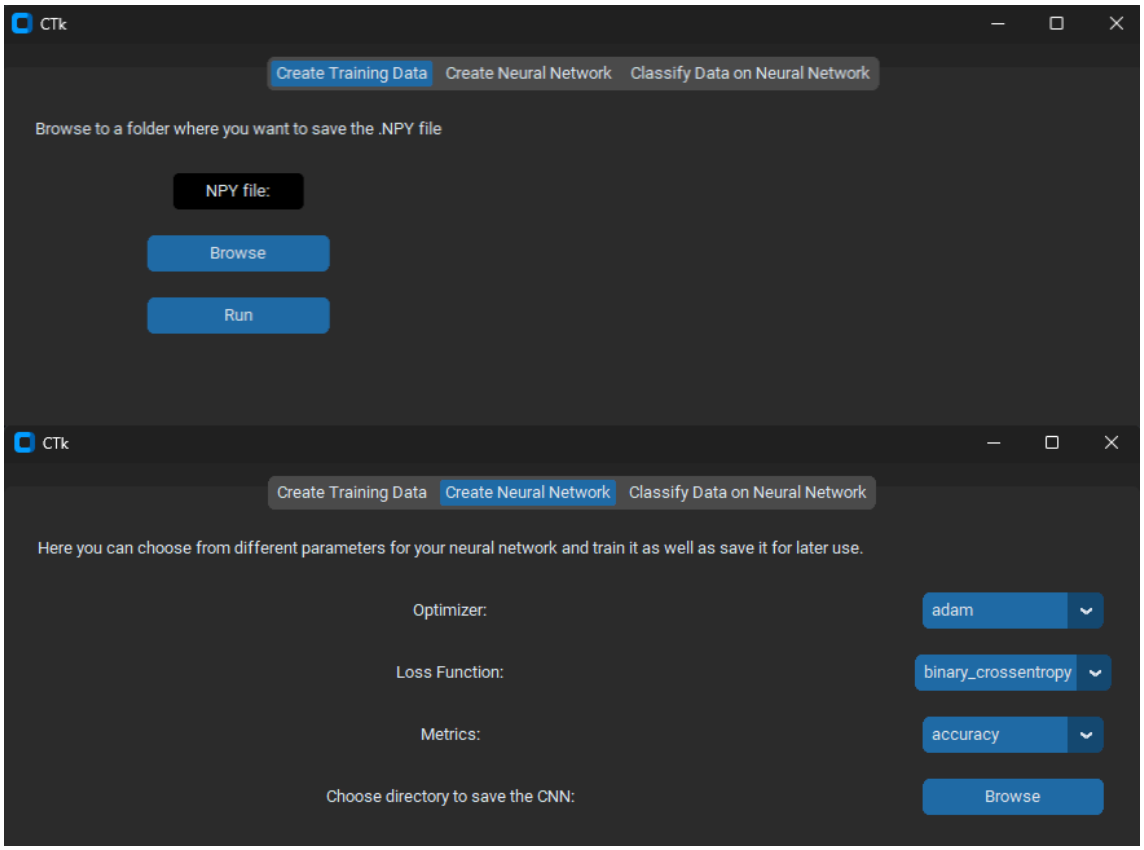


Figure 5.2: An example of a GUI to be used for the project.

5.7.2 Multiple Objects in Same FITS

The current algorithm for locating the centre of a Gaussian disk in a FITS-image takes the max value in the middle 50% of the image. This means that if there are several objects in one file only the brightest will be processed and classified. A variation of the flood fill algorithm (Sryheni, 2022) was implemented to find several objects in the file. The algorithm was ultimately not used in the final product because of constrictions in time and computational power.

5.7.3 More Non-linear Transformations

Geometric mean square was the only non-linear transformation used in the final product. A possible improvement would be to implement different non-linear methods to further improve the data.

One type of non-linear transformation that could be applied is adding artificial noise. Adding noise to data is commonly done for CNNs to improve accuracy on new data where some particular kind of noise occurs. Adding noise can also help the model to learn various aspects of each class by randomly occluding small portions

of features (Akbiyik, 2019); the overall shape of the image would stay the same but with small changes to numerical values the model would learn more general aspects of the class. Adding noise can also drastically increase the training data size. Two different algorithms of adding noise to the image were implemented but not used in the final product. We chose not to perform any noise augmentation as we did not have enough time to validate whether or not they would improve the accuracy. Adding the wrong kind of noise or noise with improper variance or level could harm the performance of the CNN (Akbiyik, 2019).

6

Conclusion

In this project, a CNN model was trained with the purpose to classify images in the ALMA archive in order to find observations that indicate a possible occurrence of dust present in a star formation wind.

With the results from the CNN in Appendix A, a final conclusion of this project can be made as overall successful. Since the aim was to produce a series of observations portraying potential dust in the wind carriage, the images shown back this up by at least being of interest for this thesis. Although some manual filtering was required from the end result, the images followed a pattern of being irregular in the sense of them not following the structure of a perfect two-dimensional Gaussian. However, there were instances of images with new kinds of undesired irregularities that were missclassified as positive. These missclassifications could be a sign that our training data was too small in size and variety.

Overall, several observations were recognized by the supervisor Per Bjerkeli from his research on the topic. The recognized observations had extensions that seemed to be in line with the boundaries of the outflow in star formation. These recognized observations acted as an affirmation of the model's ability to classify observations of potential dust in the wind and opens up possible future research and analysis on the objects.

Bibliography

2023. International Virtual Observatory Aliance. https://ivoa.net/index.html

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

Robert A. Adams and Christpher Essex. 2017. *Calculus: A Complete Course* (9 ed.). Pearson Education.

Aida Ahmadi and Alvaro Hacar. 2020. ASCL Code Record. https://ascl.net/code/v/2971

Aida Ahmadi and Alvaro Hacar. 2021a. ALminer API. https://alminer.readthedocs.io/en/latest/pages/api.html

Aida Ahmadi and Alvaro Hacar. 2021b. ALminer Documentation. https://alminer.readthedocs.io/en/latest/

Aida Ahmadi and Alvaro Hacar. 2023. ALminer Github. https://github.com/emerge-erc/ALminer

AI456. [n. d.]. A simple neural network with two input units and one output unit. https://openverse.org/image/2681ee1f-8a18-4553-9347-c01cd30bb4f9?q=neural%20network%20weights

Murtaza Eren Akbiyik. 2019. Data Augmentation in Training CNNs: Injecting Noise to Images. *ICLR 2020 Conference Blind Submission* (2019). https://openreview.net/forum?id=SkeKtyHYPS

ALMA. 2023a. ALMA Basics - Field of view. https://almascience.nrao.edu/about-alma/alma-basics

ALMA. 2023b. ALMA Cycle. https://almascience.eso.org/documents-and-tools/cycle-9-documents

ALMA. 2023c. The ALMA Science Archive has reached several milestones. almascience.org/news/the-alma-science-archive-has-reached-several-milestones

ALMA Observatory. 2023. Atacama Large Millimeter/submillimeter Array. https://www.almaobservatory.org/

Atacama Large Millimeter/submillimeter Array. 2023. ALMA Science Archive. https://almascience.nrao.edu/aq/?result_view=observations

AWS Amazon. [n. d.]. What is a Neural Network? https://aws.amazon.com/what-is/neural-network/

baeldung. 2023. How ReLu and Dropout Layers Work in CNNs. (4 2023). https://www.baeldung.com/cs/ml-relu-dropout-layers

T. Bex. 2021. Comprehensive Guide to Multiclass Classification Metrics. (6 2021). https://towardsdatascience.com/comprehensive-guide-on-multiclass-classification-metrics-af94cfb83fbd

Per Bjerkeli. 2022. Star formation.

Per Bjerkeli. 2023. Extention patterns.

P. Bjerkeli, D. Harsono, J. Ramsey, A. Plunkett, H. Calcutt, L. Kristensen, M. van der Wiel, J. Jørgensen, and Z. Li. 2021. The effects of winds on disk and planet formation. In *American Astronomical Society Meeting Abstracts (American Astronomical Society Meeting Abstracts, Vol. 53)*. Article 406.05, 406.05 pages.

R.D. Blandford and D.G. Payne. 1982. Hydromagnetic flow. *Monthly Notices of the Royal Astronomical Society* 199, 4 (8 1982), 883–903. https://doi.org/10.1093/mnras/199.4.883

Dana Bolles. 2023. Stars. https://science.nasa.gov/astrophysics/focus-areas/how-do-stars-form-and-evolve

Jason Brownlee. 2019. A Gentle Introduction to Pooling Layers for Convolutional Neural Networks. (4 2019). https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/

Joseph Comunale. [n. d.]. What are emergent properties? Interpretations and Examples.

Andrea D'Agostino. [n. d.]. Introduction to Neural Networks - Weights, biases and activation. ([n. d.]). https://medium.com/mlearning-ai/introduction-to-neural-networks-weights-biases-and-activation-270ebf2545aa

DeepAI. 2020. Harmonic Mean. https://deepai.org/machine-learning-glossary-and-terms/harmonic-mean

Julianna Delua and IBM Analytics. 2021. Supervised vs. Unsupervised Learning: What's the Difference? https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning

Edge AI + Vision Alliance. 2015. Using Convolutional Neural Networks for Image Recognition. (11 2015). https://www.edge-ai-vision.com/2015/11/using-convolutional-neural-networks-for-image-recognition/

Errachete. [n. d.]. Binary Confusion Matrix. https://commons.wikimedia.org/w/index.php?curid=85341147

European Southern Observatory. 2011. ALMA Opens Its Eyes. In *Eso 1137 - Organisation Release*, Tania Burchell (Ed.). European Southern Observatory. https://www.eso.org/public/news/eso1137/

Andrew Fraknoi, David Morrison, and Sidney Wolff. 2022. Spectroscopy in Astronomy. In *Astronomy* (first ed.). Vol. 2e. Rice University, Chapter 5.3. https://openstax.org/books/astronomy-2e/pages/5-3-spectroscopy-in-astronomy

Andreas Schwung Gavneet Singh Chadha. 2019. Learning the Non-linearity in Convolutional Neural Networks. *arXiv: 1905.12337* (2019). https://arxiv.org/pdf/1905.12337.pdf

Oliver Gressel, Neal J. Turner, Richard P. Nelson, and Colin P. McNally. 2015. Global simulations of protoplanetary disks with Ohmic resistivity and ambipolar diffusion. *Astrophysical Journal* 801, 2 (3 2015), 84. https://doi.org/10.1088/0004-637X/801/2/84

IBM. [n. d.]. What are Neural Networks? https://www.ibm.com/topics/neural-networks

IBM. [n. d.]. What is overfitting? ([n. d.]). https://www.ibm.com/topics/overfitting

Habib Izadkhah. 2022. Medical image processing: an insight to convolutional neural networks. In *Deep Learning in Bioinformatics*. Elsevier, 175–213. https://doi.org/10.1016/b978-0-12-823822-6.00015-9

Keras. [n. d.]. Layer Activation Functions. https://keras.io/api/layers/activations/#softmax-function

Samaya Madhava and IBM. 2021. Introduction to Convolutional Neural Networks. https://developer.ibm.com/articles/introduction-to-convolutional-neural-networks/

Andreas Maier. [n. d.]. Convolutionalandpooling. https://openverse.org/image/ef4cf2de-db98-47ab-8461-24daa9c6186f?q=convolutional%20layer%20network

Nick McCullum. [n. d.]. How to Build and Train a Convolutional Neural Network. ([n. d.]). https://www.nickmccullum.com/python-deep-learning/convolutional-neural-network-tutorial/

Milecia McGregor and freeCodeCamp. 2021. What is a Neural Network? A beginner's tutorial for Machine Learning and Deep Learning. https://www.freecodecamp.org/news/convolutional-neural-network-tutorial-for-beginners/

Mayank Mishra. 2020. Convolutional Neural Networks, Explained. (8 2020). https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

Sarang Narkhede. 2018. Understanding Confusion Matrix. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

NRAO. 2023. National Radio Astronomy Observatory. https://public.nrao.edu/telescopes/alma/

NRF. [n. d.]. Continuum Observing with the Hart 26m Telescope. http://www.hartrao.ac.za/continuum/

Alan P. Smale. 2014. A brief history to FITS. https://fits.gsfc.nasa.gov/fits_overview.html

Ilaria Pascucci, Sylvie Cabrit, Suzan Edwards, Uma Gorti, Oliver Gressel, and Takeru Suzuki. 2022. The Role of Disk Winds in the Evolution and Dispersal of Protoplanetary Disks. (3 2022). http://arxiv.org/abs/2203.10068

R E Pudritz and C A Norman. 1983. *CENTRIFU GALL Y DRIVEN WINDS FROM CONTRACTING MOLECULAR DISKS*. Technical Report. 677–697 pages.

PyTorch. [n. d.]. Optimizing Model Parameters - PyTorch. https://pytorch.org/tutorials/beginner/basics/optimization_tutorial.html

PyTorch. 2023. PyTorch. https://pytorch.org/

Jenni Raitoharju. 2022. Convolutional neural networks. In *Deep Learning for Robot Perception and Cognition*. Elsevier, 35–69. https://doi.org/10.1016/B978-0-32-385787-1.00008-7

Shipra Saxena. 2021. Image Augmentation Techniques for Training Deep Learning Models. https://www.analyticsvidhya.com/blog/2021/03/image-augmentation-techniques-for-training-deep-learning-models/

Aditya Sharma. 2017. Convolutional Neural Networks in Python with Keras. https://www.datacamp.com/tutorial/convolutional-neural-networks-python

Somayeh Sheikhnezami, Christian Fendt, Oliver Porth, Bhargav Vaidya, and Jamshid Ghanbari. 2012. Bipolar jets launched from magnetically diffusive accretion disks. I. Ejection efficiency versus field strength and diffusivity. *Astrophysical Journal* 757, 1 (9 2012). https://doi.org/10.1088/0004-637X/757/1/65

H. Shoemaker, P. Bjerkeli, J. Ramsey, and A Plunkett. [n. d.]. Dust In the wind: A search for dust in the outflow of the HH212 class 0 protostellar system.

F.H. Shu. 1997. Molecules in star formation. *Symposium - International Astronomical Union* 178 (1997), 19–30. https://doi.org/10.1017/s0074180900009219

Said Sryheni. 2022. Flood Fill Algorithm. (2022). https://www.baeldung.com/cs/flood-fill-algorithm

StackOverflow. 2023. Questions tagged [tensorflow]. https://stackoverflow.com/questions/tagged/tensorflow

Stanford Visual and Learning Lab. [n. d.]. *CS231n Convolutional Neural Network for Visual Recognition*. Technical Report. https://cs231n.github.io/neural-networks-1/

Matthew Stewart. 2019. Introduction to Neural Networks. (6 2019). https://towardsdatascience.com/simple-introduction-to-neural-networks-ac1d7c3d7a2c

Jonas Teuwen and Nikita Moriakov. 2019. Convolutional neural networks. In *Handbook of Medical Image Computing and Computer Assisted Intervention*. Elsevier, 481–501. https://doi.org/10.1016/B978-0-12-816176-0.00025-9

Carmen Toribio. [n. d.]. ALminer science keyword "Low-mass star formation" issue. https://github.com/emerge-erc/ALminer/issues/4

TseKiChun. [n. d.]. Data Augmentation of Rock Images Revised. https://commons.wikimedia.org/wiki/File:Data_Augmentation_of_rock_images_revised.jpg

Diego Unzueta. 2022. Fully Connected Layers vs. Convolutional Layer: Explained. (10 2022). https://builtin.com/machine-learning/fully-connected-layer

USENIX Association., ACM SIGMOBILE., ACM Special Interest Group in Operating Systems., and ACM Digital Library. 2005. *Papers presented at the Workshop on Wireless Traffic Measurements and Modeling : June 5, 2005, Seattle, WA, USA*. USENIX Association. 44 pages.

Jon Wallace. [n. d.]. Introduction to Radio Astronomy. https://www.radio-astronomy.org/pdf/sara-beginner-booklet.pdf

Sarah Wood, Erica Keller, and Catarina Ubach. 2021. ALMA Archive & Data Products - What to expect after your observations are made. https://science.nrao.edu/facilities/alma/community/alma-archive-data-products-cycle-8-2021

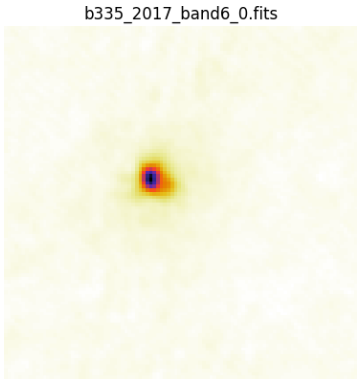
Enes Zvornicanin and baeldung. 2023. Relation Between Learning Rate and Batch Size . https://www.baeldung.com/cs/learning-rate-batch-size

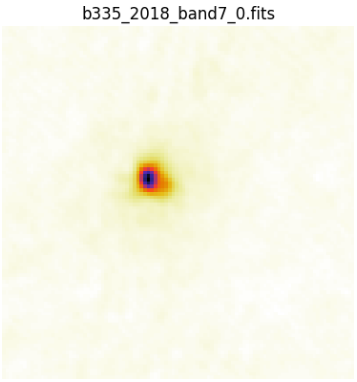
A

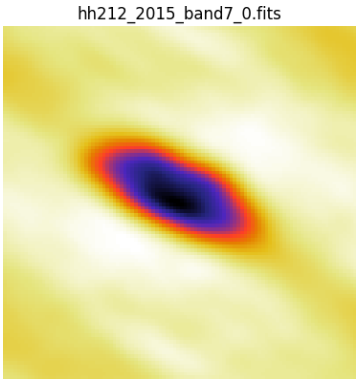
Initially Known Positive Images

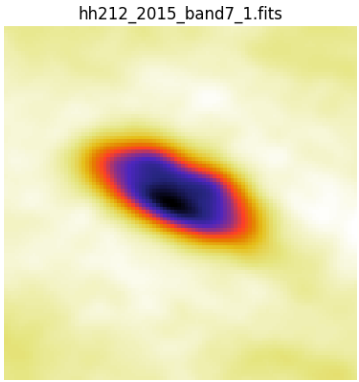
This appendix makes use of the following ALMA data:

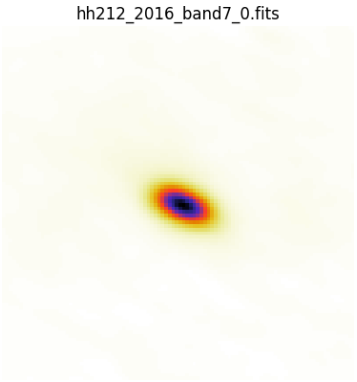
ADS/JAO.ALMA#2012.1.00122.S
ADS/JAO.ALMA#2015.1.00024.S
ADS/JAO.ALMA#2016.1.01475.S
ADS/JAO.ALMA#2017.1.00044.S
ADS/JAO.ALMA#2017.1.00288
ADS/JAO.ALMA#2018.1.01208
ADS/JAO.ALMA#2019.1.00912.S

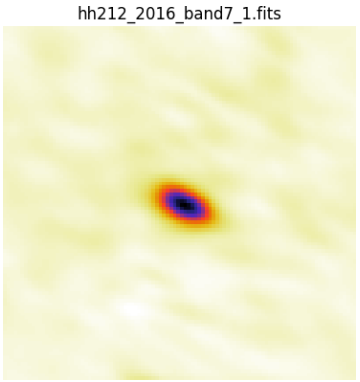


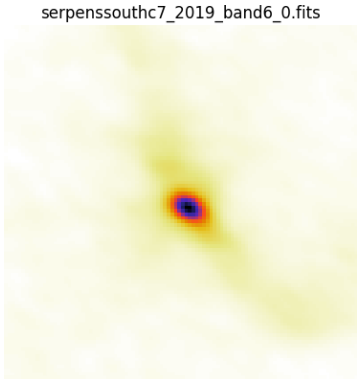


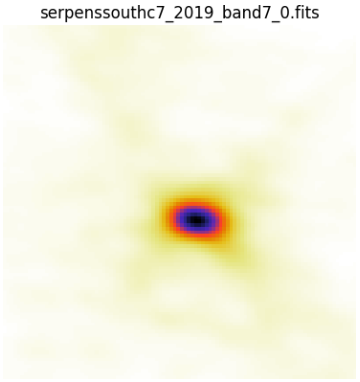









serpenssouthc7_2019_band6_0.fits

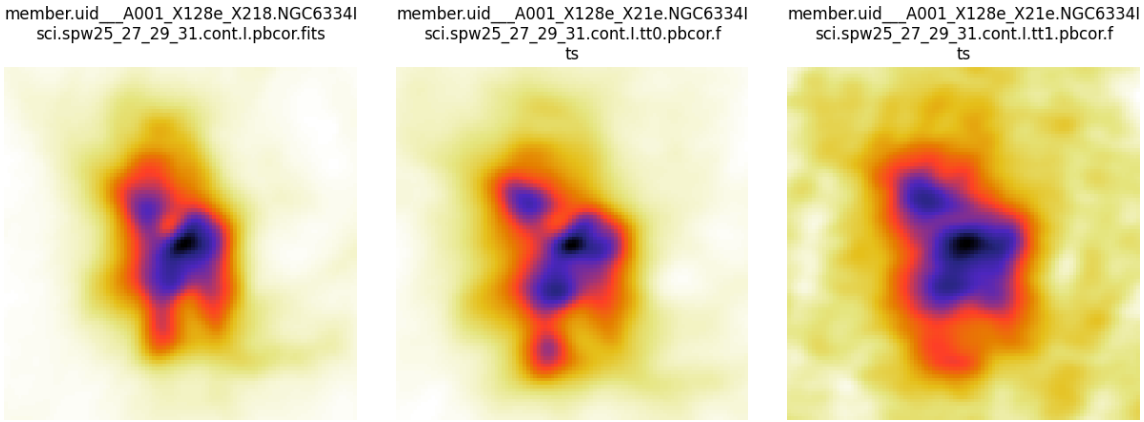

serpenssouthc7_2019_band7_0.fits

B

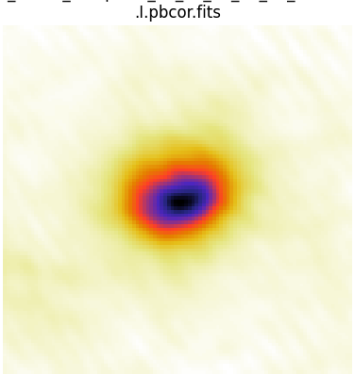
CNN Classified Positive Images

This appendix makes use of the following ALMA data:

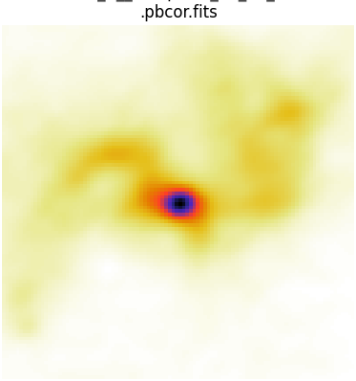
ADS/JAO.ALMA#2015.1.00722.S
ADS/JAO.ALMA#2015.1.01535.S
ADS/JAO.ALMA#2016.1.00711.S
ADS/JAO.ALMA#2016.1.01115.S
ADS/JAO.ALMA#2016.1.00846.S
ADS/JAO.ALMA#2016.1.00630.S
ADS/JAO.ALMA#2016.1.01347.S
ADS/JAO.ALMA#2016.1.00345.S
ADS/JAO.ALMA#2017.1.00661.S
ADS/JAO.ALMA#2018.1.00024.S
ADS/JAO.ALMA#2018.1.01647.S
ADS/JAO.ALMA#2018.1.01679.S
ADS/JAO.ALMA#2018.1.00513.S
ADS/JAO.ALMA#2019.1.00463.S
ADS/JAO.ALMA#2019.1.01792.S
ADS/JAO.ALMA#2019.1.00691.S
ADS/JAO.ALMA#2021.1.01578.S



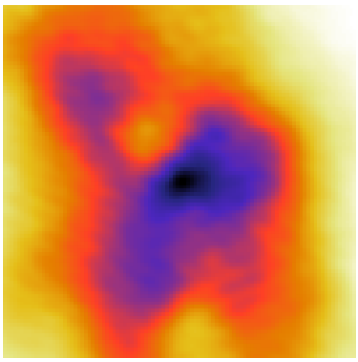
member.uid___A001_X133d_X3356.NGC_226_CMM3_sci.spw25_27_29_31_33_35_37.con.l.pbcor.fits



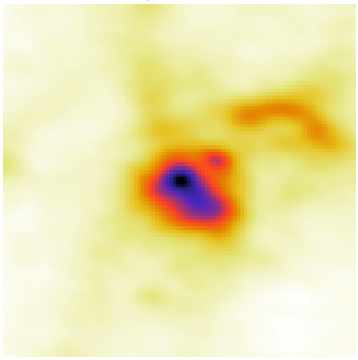
member.uid___A001_X133d_X3486._G328.251-0.5321_A__sci.spw25_27_29_31.cont..pbcor.fits



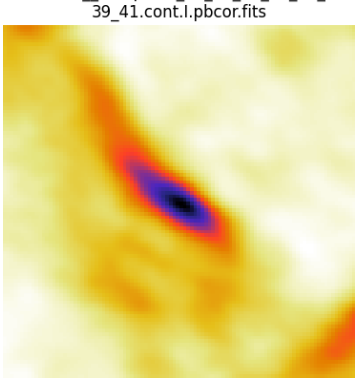
member.uid___A001_X133d_X4a5.NGC6334I H2O_sci.spw25_27_29_31.cont.l.pbcor.fits



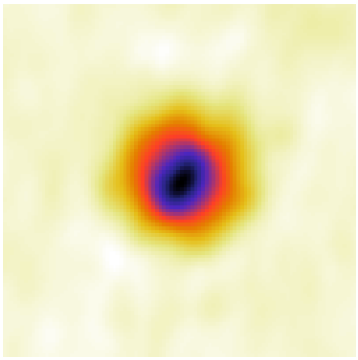
member.uid___A001_X133d_X4ad._G034.430.24MM1_H2O__sci.spw25_27_29_31.cont..pbcor.fits



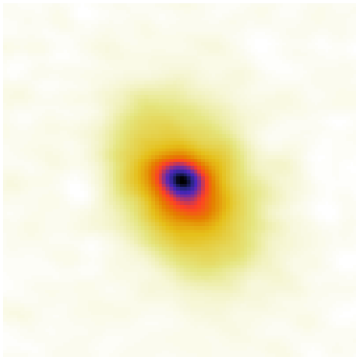
member.uid___A001_X133d_Xdb5._GAL_00588-00.41__sci.spw25_27_29_31_33_35_3739_41.cont.l.pbcor.fits



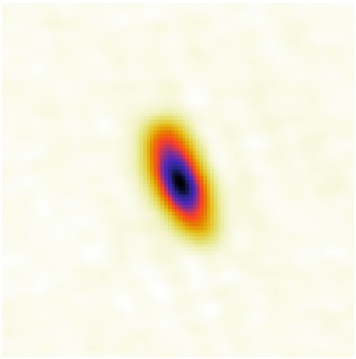
member.uid___A001_X1465_X132.ChamII_0_sci.spw17_19_21_23.cont.l.pbcor.fits



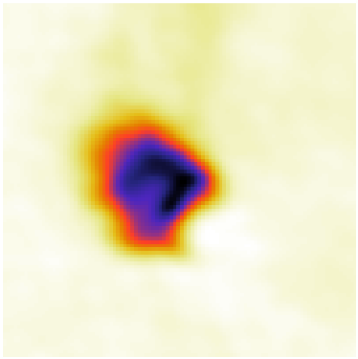
member.uid___A001_X1465_X146.Oph_34_si.spw17_19_21_23.cont.l.pbcor.fits



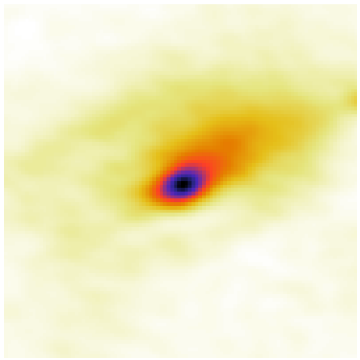
member.uid___A001_X1465_X157.Aql_13_si.spw17_19_21_23.cont.l.pbcor.fits



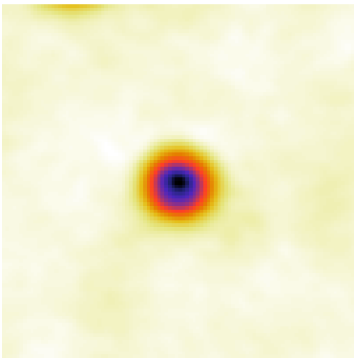
member.uid___A001_X1465_X15f.Serp_18_ci.spw17_19_21_23.cont.l.pbcor.fits



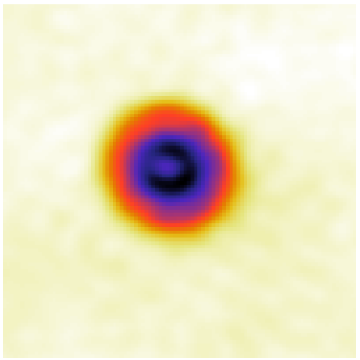
member.uid___A001_X1465_X15f.Serp_29_ci.spw17_19_21_23.cont.I.pbcor.fits



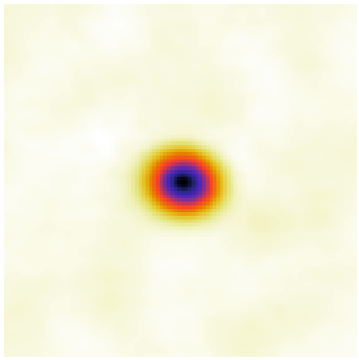
member.uid___A001_X1465_X16a.CrAus_01 sci.spw17_19_21_23.cont.I.pbcor.fits



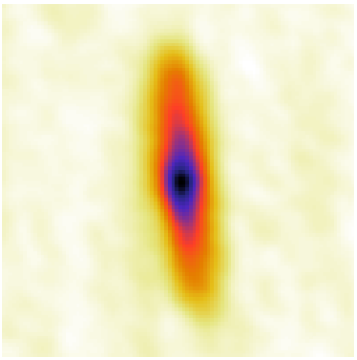
member.uid___A001_X1465_X16a.CrAus_02 sci.spw17_19_21_23.cont.I.pbcor.fits



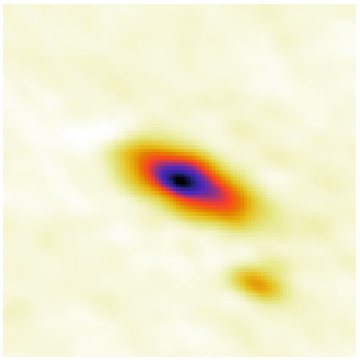
member.uid___A001_X1465_X16a.CrAus_05 sci.spw17_19_21_23.cont.I.pbcor.fits



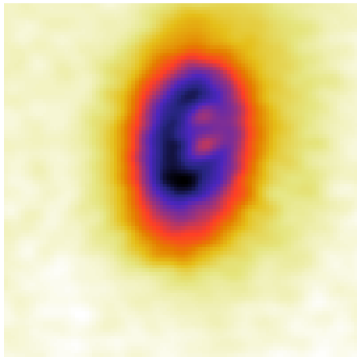
member.uid___A001_X1465_X16a.CrAus_07 sci.spw17_19_21_23.cont.I.pbcor.fits



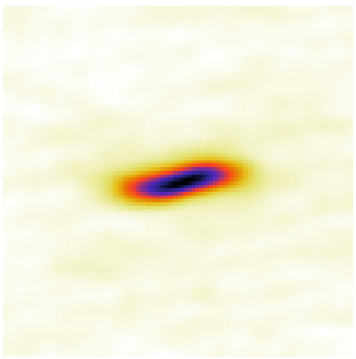
member.uid___A001_X1465_X16a.CrAus_08 sci.spw17_19_21_23.cont.I.pbcor.fits



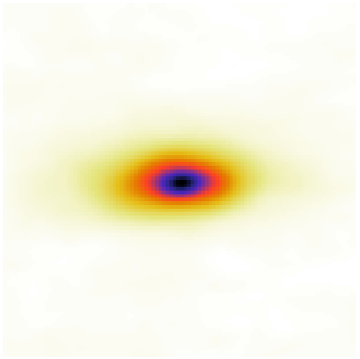
member.uid___A001_X1465_X16a.CrAus_09 sci.spw17_19_21_23.cont.I.pbcor.fits



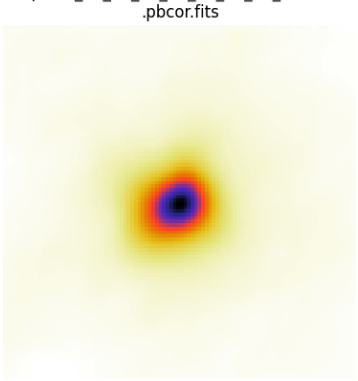
member.uid___A001_X1465_X2812.HOPS_35 _sci.spw25_27_29_31.cont.I.pbcor.fits



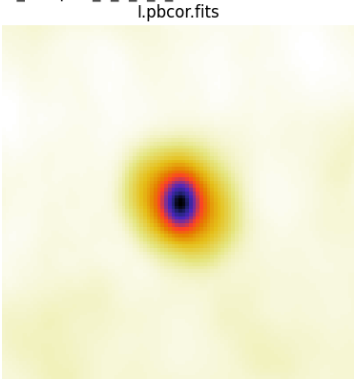
member.uid___A001_X146a_Xf.BHR7_sci.s w25_27_29_31_33_35_37_39_41.cont.I.pb or.fits



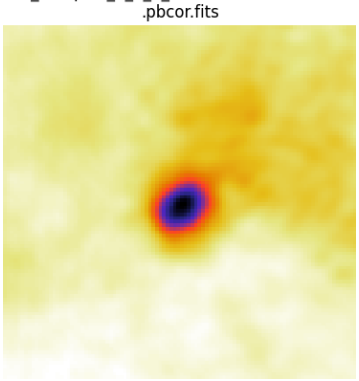
member.uid___A001_X15a1_X1a12.IC348_si.spw23_25_27_29_31_33_35_37_39.cont..pbcor.fits



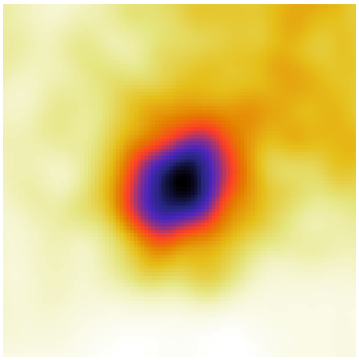
member.uid___A001_X5a3_X5c.ari_l.DG_Tau_sci.spw0_1_2_3_4_224920MHz.12m.contl.pbcor.fits



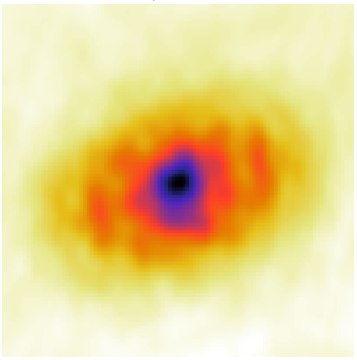
member.uid___A001_X5a3_X73.ari_l.M17-C1_sci.spw0_1_2_3_359905MHz.12m.cont..pbcor.fits



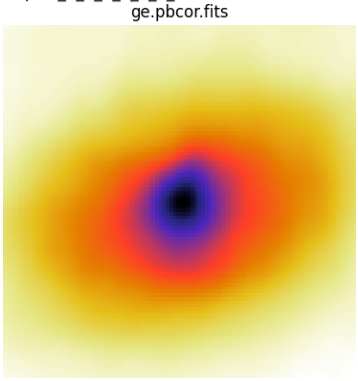
member.uid___A001_X5a3_X73.M17-UC1_sc.spw25_27_29_31.cont.l.pbcor.fits



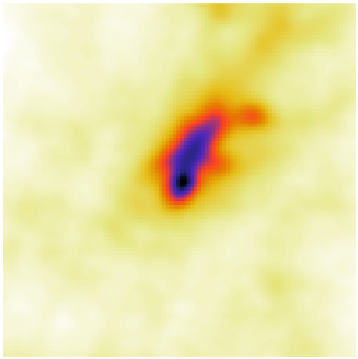
member.uid___A001_X87a_X816.ari_l.TMCA_sci.spw25_27_29_31_259583MHz.12m.cot.l.pbcor.fits



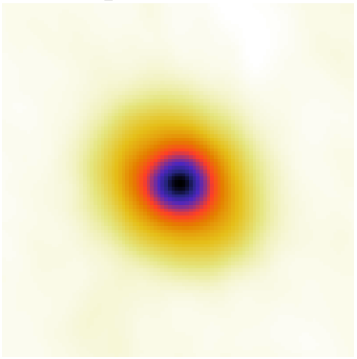
member.uid___A001_X87a_X816.TMC1A_scispw0_1_2_3_4_5_6_7.mfs.l.continuum.image.pbcor.fits



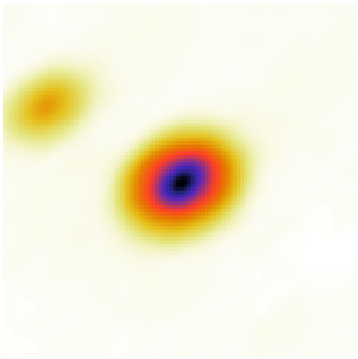
member.uid___A001_X87a_Xf1.s12_0._G103-0.1__sci.spw25_27_29_31.cont.l.manul.image.pbcor.fits



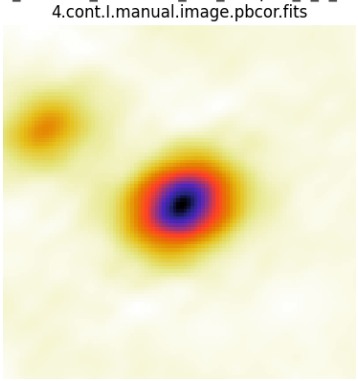
member.uid___A001_X87d_X5b9.DG_Tau_sc.spw25_27_29_31_33_35_37_39_41_43_45_7_49.cont.l.pbcor.fits



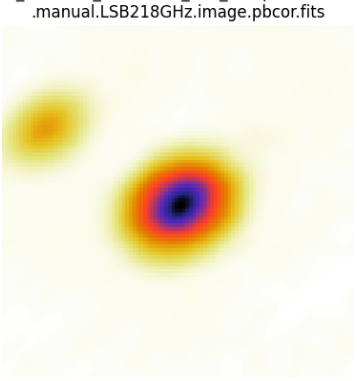
member.uid___A001_X87d_X678.uid___A00_Xc4d618_X2f92.hbc_494_sci.spw0.cont..manual.USB233GHz.image.pbcor.fits



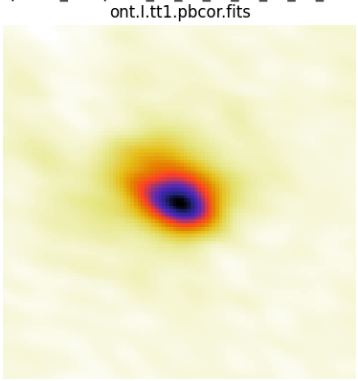
member.uid___A001_X87d_X678.uid___A00_Xc4d618_X2f92.hbc_494_sci.spw0_1_2_34.cont.I.manual.image.pbcor.fits



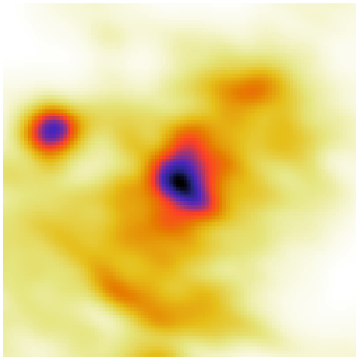
member.uid___A001_X87d_X678.uid___A00_Xc4d618_X2f92.hbc_494_sci.spw1.cont..manual.LSB218GHz.image.pbcor.fits



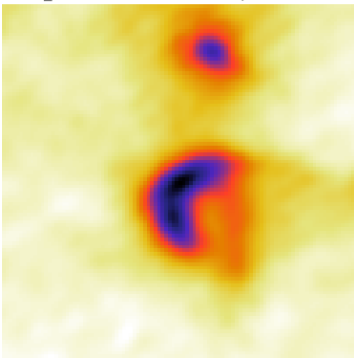
member.uid___A001_X88f_X19d.IRAS_1856p0408_sci.spw25_27_29_31_33_35_37_39.ont.I.tt1.pbcor.fits



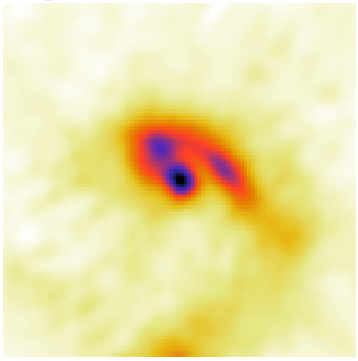
member.uid___A001_X88f_X25._AGAL351.51-00.352__sci.spw25_27_29_31_33_35_3739.cont.I.pbcor.fits



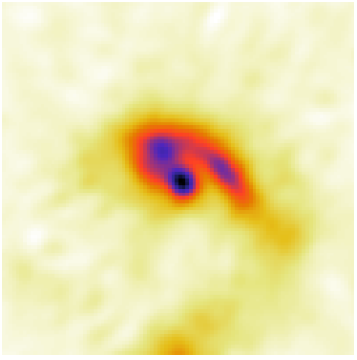
member.uid___A001_X88f_X2c.ari_l._AGA332.826-00.549__sci.spw0_1_2_3_4_5_6__224642MHz.12m.cont.I.pbcor.fits



member.uid___A001_X88f_X2c.ari_l._AGA337.916-00.477__sci.spw0_1_2_3_4_5_6__224632MHz.12m.cont.I.pbcor.fits



member.uid___A001_X88f_X2c._AGAL337.96-00.477__sci.spw25_27_29_31_33_35_3739.cont.I.pbcor.fits



C

Main and Neural Network Pipeline

```
from alma_classifier import pipeline_tensorflow
from alma_classifier.image_processing.manual_sorting import predict_fits
from keras.models import load_model
from alma_classifier.image_processing.image_augmentation import
    generate_pos_dataset
from alma_classifier.image_processing.manual_sorting import sort_manually,
    save_data_to_npy
from alma_classifier.image_processing.pre_processing import
    init_training_data_from_folder
import shutil

"""
*****//PATHS//******

    ** Before the first run, make sure that the paths are correct and exist at your
     local machine **

    ORIGINAL_POS_FITS: path to folder containing positive FITS files

    ORIGINAL_NEG_FITS: path to folder containing negative FITS files

    POS_TRAIN: path to .npy file containing positive training data

    NEG_TRAIN: path to .npy file containing negative training data

    CNN_MODEL: path to folder containing CNN model

    UNCLASSIFIED_FITS: path to folder containing FITS files to be classified

    CLASSIFIED_FITS: path to folder where classified FITS files will be saved

"""


ORIGINAL_POS_FITS = 'alma-classifier/data/fits/pos'
ORIGINAL_NEG_FITS = 'alma-classifier/data/fits/neg'

POS_TRAIN = 'C:/ChalmersWorkspaces/KandidatArbete/data/npy_train/pos_dataset.npy'
NEG_TRAIN = 'C:/ChalmersWorkspaces/KandidatArbete/data/npy_train/neg_dataset.npy'

CNN_MODEL = 'C:/ChalmersWorkspaces/KandidatArbete/data/CNN_model'

```

```
39 UNCLASSIFIED_FITS = 'C:/ChalmersWorkspaces/KandidatArbete/data/fits_to_classify'
40 CLASSIFIED_FITS = 'C:/ChalmersWorkspaces/KandidatArbete/data/classified_fits'
41
42
43 """
44
45 Note that the output of generate_pos_training_data() is saved to a .npy file and is
46 400x400 so you need to run either crop_middle_100x100 or linnear_transformation
    before
47 you can use the data in the CNN. When changing the input fits files you need to
    manually
48 change the degreees in the rotate array so that asll tha gaussian discs line up and
     is
49 compatible with being combined with eachother.
50
51 Example:
52
53 pos_data = [crop_middle_100x100(file) for file in pos_data if file.shape == (400,
    400)]
54 pos_data = np.array([linear_transformation(file) for file in pos_data for i in
    range(augment_factor) if file.shape == (400, 400)])
55
56 NOTE!: This is done in initinit_training_data_from_npy() in pre_processing.py which
     is called
57 in pippeline_tensorflow() in pipeline_tensorflow.py. Which is standard for the CNN.
58
59 """
60
61
62 def generate_pos_training_data(load_directory=ORIGINAL_POS_FITS, save_directory=
    POS_TRAIN):
63     pos_data = generate_pos_dataset(load_directory)
64     pos_data = sort_manually(pos_data)
65     save_data_to_npy(pos_data, save_directory)
66
67
68 """
69
70 Note that the aoutput of generate_neg_training_data() is saved to a .npy file and
    is ready to be
71 used in the CNN directly
72
73 """
74
75 def generate_neg_training_data(load_directory=ORIGINAL_NEG_FITS):
76     neg_data = init_training_data_from_folder(load_directory)
77     neg_data = sort_manually(neg_data)
78     save_data_to_npy(neg_data, NEG_TRAIN)
79
80
81 """
82
83 Creates and traines a CNN and saves the trained CNN to specified folder.
84
85 """
86
87 def train_CNN(pos_npy_path=POS_TRAIN,
88                 neg_npy_path=NEG_TRAIN,
89                 save_path=CNN_MODEL,
90                 lin_aug=False,
91                 aug_factor=1):
```

```
92      model = pipeline_tensorflow.pippeline_tensorflow(
93          pos_npy_path, neg_npy_path, lin_aug, aug_factor)
94      model.save(save_path)
95
96  """
97
98  Loads a trained CNN and uses it to classify the fits files in the specified folder.
99  The classified fits files are then saved to the specified folder.
100
101 """
102
103 def classify_data(read_path = UNCLASSIFIED_FITS, model = CNN_MODEL, save_path=
        CLASSIFIED_FITS):
104     pos_image = predict_fits(read_path, load_model(model))
105     [shutil.copy2(name, save_path) for (data, name) in pos_image]
106
107 """
108
109 Print call the functions in main() to run the program.
110
111 """
112
113
114 def main():
115     print('main')
116
117 if __name__ == '__main__': main()
```

```
1  from .models.tensorflow.model_01.data_handler import tensorflow_data_handler
2  from .models.tensorflow.model_01.evaluation import evaluate_model
3  from .image_processing.pre_processing import init_training_data_from_npy
4  from .models.tensorflow.model_01.model import model
5
6  """
7
8  Creates, trains and returns a CNN model.
9
10 """
11
12 def pippeline_tensorflow(pos_npy_path, neg_npy_path, linnear_aug=False,
       augmentation_factor=5):
13
14     #-------------------Importing data-------------------#
15
16     X, y = init_training_data_from_npy(pos_npy_path, neg_npy_path, linnear_aug,
       augmentation_factor)
17
18     #-------------------Reshape to tensor--------------------#
19
20     X_train, X_test, y_train, y_test = tensorflow_data_handler(X, y)
21
22     #-------------------Creat model-------------------#
23
24     nn_model = model()
25
26     #-------------------Compile model-------------------#
27
28     evaluate_model(X_train, X_test, y_train, y_test, nn_model)
29
30     return nn_model
31
```

```
32
33 __name__ == '__main__' and pippeline_tensorflow()
```

D

Image Processing

```
from astropy.io import fits
import numpy as np
import glob as glob
import numpy as np
import glob
import astropy.io.fits as fits
from .support_functions import *


"""

This function takes a path to a folder containing FITS files and returns the data in
an array of 2D-arrays of size 000x100. Ready to be used as input for a neural network.

Input:

    file_path: path to folder containing FITS files
    linnear_aug: if True, the data will be augmented by linear transformations
    augment_factor: how many times the data will be augmented

"""


def init_training_data_from_folder(file_path, linnear_aug=False, augment_factor=5):

    fits_files_data = [fits.getdata(file).squeeze() for file in glob.glob(file_path + '/*.fits')]
    fits_files_data = [crop_around_middle_50x50_percent(file) for file in fits_files_data if file.shape[0] > 500 and file.shape[1] > 500]
    fits_files_data = ([crop_around_max_value_400x400(file) for file in fits_files_data])
    if linnear_aug:
        return np.array([linear_transformation(file) for file in fits_files_data for i in range(augment_factor) if file.shape == (400, 400)])
    return np.array([crop_middle_100x100(file) for file in fits_files_data if file.shape == (400, 400)])


"""

Convert a folder of FITS files to a numpy array and save it to a npy file.

"""
```

```
39
40 def fits_to_npy(folder_path, npy_path, linnear_aug=False, augment_factor=5): np.
       save(npy_path, init_training_data_from_folder(folder_path, linnear_aug,
       augment_factor), allow_pickle=True)
41
42 """
43
44 Init X and y from numpy arrays of positive and negative data.
45
46 """
47
48
49 def init_training_data_from_npy(pos_path, neg_path, linnear_aug=False,
       augment_factor=5):
50
51     fits_pos = np.load(pos_path, allow_pickle=True)
52     fits_neg = np.load(neg_path, allow_pickle=True)
53
54     if linnear_aug:
55         fits_pos = np.array([linear_transformation(file) for file in fits_pos for i
        in range(augment_factor) if file.shape == (400, 400)])
56         fits_neg = np.array([linear_transformation(file) for file in fits_neg for i
        in range(augment_factor) if file.shape == (400, 400)])
57
58     else:
59         fits_pos = np.array([crop_middle_100x100(file) for file in fits_pos if file
       .shape == (400, 400)])
60         fits_neg = np.array([crop_middle_100x100(file) for file in fits_neg if file
       .shape == (400, 400)])
61
62
63     y = [0] * len(fits_neg) + [1] * len(fits_pos)
64
65     X = np.concatenate((fits_neg, fits_pos), axis=0)
66
67     return X, y
68
69
70
71 __name__ == '__main__' and print('pre_processing.py is working')
```

```
1 import numpy as np
2 import glob
3 import astropy.io.fits as fits
4 from scipy.ndimage import rotate
5 from skimage.transform import resize
6 from .support_functions import *
7
8
9 def generate_pos_dataset(folder_path):
10
11     print('Loading data from ' + folder_path)
12
13     pos_data = ([fits.getdata(file).squeeze() for file in glob.glob(folder_path + '
       /*.fits')])
14
15     pos_data = [crop_around_middle_50x50_percent(file) for file in pos_data]
16
17     # resize the data so that all the disks are roughly the same size exepct for
       the two largest ones
18     pos_data = [file if (file is pos_data[2] or file is pos_data[3]) else resize(
```

```
        file, (2000, 2000)) for file in pos_data]

    # # Rotate the data so that the stars are aligned according to predefined angles
    pos_data = [rotate(data, degrees, reshape=False) for (data, degrees) in list(
        zip(pos_data, [0, 0, -15, -15, -9, -9, -13, -13]))]

    pos_data = [crop_around_max_value_400x400(file) for file in pos_data]

    print('Augmenting data...')

    # Flip each image LR, UD and both
    f1, f2, f3, f4 = lambda file: file, lambda file: np.fliplr(
        file), lambda file: np.flipud(file), lambda file: np.flip(file)
    pos_data = [f(file) for file in pos_data for f in [f1, f2, f3, f4] if file.shape == (400, 400)]

    # # Make non-linnear combinations of all the fits files
    pos_data += [geometric_mean_square(pos_data[i], pos_data[j])
                 for i in range(0, len(pos_data)) for j in range(i+1, len(pos_data))]

    print('Done augmenting data. Total number of images: ' + str(len(pos_data)))

    return pos_data


__name__ == '__main__' and print('pre_processing.py is working')
```

```
import numpy as np
import random
from scipy.ndimage import rotate
from skimage.transform import resize

"""

Crop functions for the fits files. Self explanatory.

"""

def crop_around_middle_50x50_percent(fits): return fits[(int(fits.shape[0]*.25)):(int(fits.shape[0]*.75)),
                                                        (int(fits.shape[1]*.25)):(int(fits.shape[1]*.75))]


def crop_middle_100x100(fits): return fits[fits.shape[0]//2-50:fits.shape[0]//2+50,
                                           fits.shape[1]//2-50:fits.shape[1]//2+50]


def crop_around_max_value_400x400(fits):
    (x, y) = np.unravel_index(np.argmax(fits, axis=None), fits.shape)
    return fits[x-200:x+200, y-200:y+200]


"""

Geometric mean square function. Used to make non-linnear combinations of the fits files. To generate more data.

"""
```

```
30
31
32 def geometric_mean_square(a, b): return np.sqrt(np.multiply(abs(a), abs(b)))
33
34
35 """
36
37 Linnear transformation || Random  rezise, rotate, flip and return 100x100
38
39 """
40
41
42 def linear_transformation(fits):
43     ret_image = fits
44
45     random_resize = random.randint(150, 350)
46     ret_image = resize(ret_image, (random_resize, random_resize))
47
48     ret_image = rotate(ret_image, random.randint(0, 360), reshape=False)
49
50     if random.getrandbits(1):
51         ret_image = np.fliplr(ret_image)
52     if random.getrandbits(1):
53         ret_image = np.flipud(ret_image)
54
55     (lwr_bound, upr_bound) = int(random_resize/2) - 55, int(random_resize/2) - 45
56     x = random.randint(lwr_bound, upr_bound)
57     y = random.randint(lwr_bound, upr_bound)
58
59     return ret_image[x:x+100, y:y+100]
60
61
62 __name__ == '__main__' and print('helper_functions.py is working')
```

E

Neural Network Backend

```python
import numpy as np
import tensorflow as tf

from keras import utils as np_utils
from sklearn.model_selection import train_test_split


"""

    This function is used to reshape the data to a tensorflow format.

"""


def tensorflow_data_handler(X, y):

    X = np.array([tf.convert_to_tensor(fits) for fits in X])
    y = np.array([tf.convert_to_tensor(fits) for fits in y])

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)

    y_train = np_utils.to_categorical(y_train, 2)
    y_test = np_utils.to_categorical(y_test, 2)

    return X_train, X_test, y_train, y_test


__name__ == '__main__' and print('data_handler.py works!')
```

```python
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D

"""

    This function creates a CNN model.

"""

def model(activation='softmax'):

    model = Sequential()
    model.add(Conv2D(32, kernel_size=(3, 3),
                     activation=activation,
```

```
                        input_shape=(100, 100, 1)))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(2, activation=activation))

    return model


__name__ == '__main__' and print('model.py works!')
```

```
import keras


"""

    This function is used to evaluate a tensorflow CNN.

"""

def evaluate_model(X_train, X_test, y_train, y_test, model,
                   loss_function='binary_crossentropy' , learning_rate=0.0001,
    optimizer='Adam', metrics='accuracy'):

    model.summary()

    model.compile(loss = keras.losses.binary_crossentropy,
                  optimizer=keras.optimizers.Adam(learning_rate=0.001), metrics=['
    accuracy'])

    fit_info = model.fit(X_train, y_train,
                         batch_size=2,
                         epochs=30,
                         verbose=1,
                         validation_data=(X_test, y_test))

    print(model.evaluate(X_test, y_test, verbose=0))


__name__ == '__main__' and print('model.py works!')
```

F

ALminer

```
from os import system
import alminer
import pandas as pd
from astroquery.alma import Alma
from astropy.io import fits
import numpy as np
import os

# Below license is for ALminer since we have modified some code from there
"""
MIT License

Copyright (c) 2021 Aida Ahmadi , Alvaro Hacar

Permission is hereby granted , free of charge , to any person obtaining a copy
of this software and associated documentation files (the " Software ") , to
deal in the Software without restriction , including without limitation the rights
to use , copy , modify , merge , publish , distribute , sublicense , and/or sell
copies of the Software , and to permit persons to whom the Software is
furnished to do so , subject to the following conditions :
The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software .

THE SOFTWARE IS PROVIDED "AS IS" , WITHOUT WARRANTY OF ANY KIND , EXPRESS OR
IMPLIED , INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY ,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT . IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM , DAMAGES OR OTHER
LIABILITY , WHETHER IN AN ACTION OF CONTRACT , TORT OR OTHERWISE , ARISING FROM
, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE .
"""

"""Modified version of alminer.download_data. Here, the download randomly
   selects 500 files to download and download these. The code for this can
   be found on line 167."""


################################################
# Libraries
################################################
from constants_copy import band_names, band_color, band_min_freq, band_max_freq, \
    CO_line_names, CO_line_freq, CO_line_ha, CO_line_label, VALID_KEYWORDS_STR, \
    NEW_COLUMNS, COLUMN_TYPES
from pyvo.dal import tap
from astroquery.alma import Alma
```

```
from matplotlib.ticker import FormatStrFormatter, NullFormatter
import matplotlib.pyplot as plt
from astropy import constants as const
from astropy import units as u
from astropy.coordinates import SkyCoord
from astropy.coordinates import name_resolve
from astropy.coordinates import get_icrs_coordinates
from astropy.coordinates import Angle
import os
import re
import pandas as pd
import numpy as np

np.set_printoptions(threshold=np.inf)


def _format_bytes(size):
    """Convert the size of the dota to be downloaded in human-readable format."""
    power = 1000
    n = 0
    power_labels = {0: 'B', 1: 'KB', 2: 'MB',
                    3: 'GB', 4: 'TB', 5: 'PB', 6: 'EB'}
    while size > power:
        size /= power
        n += 1
    return size, power_labels[n]


def download_data_mod(observations, fitsonly=False, dryrun=False, print_urls=False,
     filename_must_include='',
                    location='./data', archive_mirror='ESO'):
    """
    Download ALMA data from the archive to a location on the local machine.

    Parameters
    ----------
    observations : pandas.DataFrame
         This is likely the output of e.g. 'conesearch', 'target', 'catalog', & '
    keysearch' functions.
    fitsonly : bool, optional
         (Default value = False)
         Download individual fits files only (fitsonly=True). This option will not
    download the raw data
         (e.g. 'asdm' files), weblogs, or README files.
    dryrun : bool, optional
         (Default value = False)
         Allow the user to do a test run to check the size and number of files to
    download without actually
         downloading the data (dryrun=True). To download the data, set dryrun=False
    .
    print_urls : bool, optional
         (Default value = False)
         Write the list of urls to be downloaded from the archive to the terminal.
    filename_must_include : list of str, optional
         (Default value = '')
         A list of strings the user wants to be contained in the url filename. This
     is useful to restrict the
         download further, for example, to data that have been primary beam
    corrected ('.pbcor') or that have
         the science target or calibrators (by including their names). The choice
    is largely dependent on the
```

```
99          cycle and type of reduction that was performed and data products that
    exist on the archive as a result.
100         In most recent cycles, the science target can be filtered out with the
    flag '_sci' or its ALMA target name.
101     location : str, optional
102         (Default value = ./data)
103         directory where the downloaded data should be placed.
104     archive_mirror : str, optional
105         (Default value = 'ESO')
106         The archive service to use. Options are:
107         'ESO' for Europe (https://almascience.eso.org),
108         'NRAO' for North America (https://almascience.nrao.edu), or
109         'NAOJ' for East Asia (https://almascience.nao.ac.jp)
110     """
111     print("================================")
112     # we use astroquery to download data
113     myAlma = Alma()
114     default_location = './data'
115     myAlma.cache_location = default_location
116     if archive_mirror == 'NRAO':
117         mirror = "https://almascience.nrao.edu"
118     elif archive_mirror == 'NAOJ':
119         mirror = "https://almascience.nao.ac.jp"
120     else:
121         mirror = "https://almascience.eso.org"
122     myAlma.archive_url = mirror
123     # catch the case where the DataFrame is empty.
124     try:
125         if any(observations['data_rights'] == 'Proprietary'):
126             print("Warning: some of the data you are trying to download are still
    in the proprietary period and are "
127                   "not publicly available yet.")
128             observations = observations[observations['data_rights'] == 'Public']
129         uids_list = observations['member_ous_uid'].unique()
130         # when len(uids_list) == 0, it's because the DataFrame included only
    proprietary data and we removed them in
131         # the above if statement, so the DataFrame is now empty
132         if len(uids_list) == 0:
133             print("len(uids_list)==0")
134             print("No data to download. Check the input DataFrame. It is likely
    that your query results include only "
135                   "proprietary data which cannot be freely downloaded.")
136             return
137     # this is the case where the query had no results to begin with.
138     except TypeError:
139         print("type error")
140         print("No data to download. Check the input DataFrame.")
141         return
142     # change download location if specified by user, else the location will be a
    folder called 'data'
143     # in the current working directory
144     if location != default_location:
145         if os.path.isdir(location):
146             myAlma.cache_location = location
147         else:
148             print("{} is not a directory. The download location will be set to {}".
    format(
149                 location, default_location))
150             myAlma.cache_location = default_location
151     elif (location == default_location) and not os.path.isdir(location):  # create
    the 'data' subdirectory
```

```
        os.makedirs(default_location)
    if fitsonly:
        data_table = myAlma.get_data_info(uids_list, expand_tarfiles=True)
        # filter the data_table and keep only rows with "fits" in 'access_url' and the strings provided by user
        # in 'filename_must_include' parameter
        dl_table = data_table[[i for i, v in enumerate(data_table['access_url']) if v.endswith(".fits") and
                               all(i in v for i in filename_must_include)]]
    else:
        data_table = myAlma.get_data_info(uids_list, expand_tarfiles=False)
        # filter the data_table and keep only rows with "fits" in 'access_url' and the strings provided by user
        # in 'filename_must_include' parameter
        dl_table = data_table[[i for i, v in enumerate(data_table['access_url']) if
                               all(i in v for i in filename_must_include)]]
    dl_df = dl_table.to_pandas()
    # Picking out 500 files of these
    dl_df = dl_df.sample(500)
    # remove empty elements in the access_url column
    dl_df = dl_df.loc[dl_df.access_url != '']
    dl_link_list = list(dl_df['access_url'].unique())
    # keep track of the download size and number of files to download
    dl_size = dl_df['content_length'].sum()
#    print(dl_size['content_length'])
    dl_files = len(dl_df['access_url'].unique())
    dl_uid_list = list(dl_df['ID'].unique())

    if dryrun:
        print("This is a dryrun. To begin download, set dryrun=False.")
        print("================================")
    else:
        print("Starting download. Please wait...")
        print("================================")
        try:
            myAlma.download_files(dl_link_list, cache=True)
        except ValueError as e:
            print(e)
    if dl_files > 0:
        print("Download location = {}".format(myAlma.cache_location))
        print("Total number of Member OUSs to download = {}".format(len(dl_uid_list)))
        print("Selected Member OUSs: {}".format(dl_uid_list))
        print("Number of files to download = {}".format(dl_files))
        dl_size_fmt, dl_format = _format_bytes(dl_size)
        print("Needed disk space = {:.1f} {}".format(dl_size_fmt, dl_format))
        if print_urls:
            print("File URLs to download = {}".format("\n".join(dl_link_list)))
    else:
        print("Nothing to download.")
        print("Note: often only a subset of the observations (e.g. the representative window) is ingested into "
              "the archive. In such cases, you may need to download the raw dataset, reproduce the calibrated "
              "measurement set, and image the observations of interest. It is also possible to request calibrated "
              "measurement sets through a Helpdesk ticket to the European ARC "
              "(see https://almascience.eso.org/local-news/requesting-calibrated-measurement-sets-in-europe).")
    print("--------------------------------")

```

```
205     # From here on there's download, we only want to download random amount of X
        declared when calling function
```

```
"""Global variables & constants to be used in alminer.py"""
"""This is a copy from the ALminer GitHub (https://github.com/emerge-erc/ALminer).
   The reason to have this copy is to provide for the modified download_data in
    alminer_mod."""

# Lists and dictionaries for analysis and plotting purposes
band_names = ["Band 3", "Band 4", "Band 5",
              "Band 6", "Band 7", "Band 8, 9, 10"]

band_color = {"Band 3": "#BDD9BF", "Band 4": "#929084", "Band 5": "#FFC857", "Band
    6": "#A997DF",
              "Band 7": "#E5323B", "Band 8, 9, 10": "#2E4052"}

band_min_freq = {'Band 3': 84., 'Band 4': 125., 'Band 5': 163.,
                 'Band 6': 211., 'Band 7': 275., 'Band 8, 9, 10': 373.}

band_max_freq = {'Band 3': 116., 'Band 4': 163., 'Band 5': 211.,
                 'Band 6': 275., 'Band 7': 373., 'Band 8, 9, 10': 950.}

# CO, 13CO, C18O lines covered in all ALMA bands
CO_line_names = ["CO (1-0)", "CO (2-1)", "CO (3-2)", "CO (4-3)", "CO (5-4)", "CO
    (6-5)",
                 "CO (7-6)", "CO (8-7)", "13CO (1-0)", "13CO (2-1)", "13CO (3-2)",
                 "13CO (4-3)",  "13CO (5-4)", "13CO (6-5)", "13CO (7-6)", "13CO
    (8-7)",
                 "C18O (1-0)", "C18O (2-1)", "C18O (3-2)", "C18O (4-3)", "C18O
    (5-4)", "C18O (6-5)",
                 "C18O (7-6)", "C18O (8-7)"]

CO_line_freq = {"CO (1-0)": 115.27120180, "CO (2-1)": 230.53800000, "CO (3-2)":
    345.79598990,
                "CO (4-3)": 461.0407682, "CO (5-4)": 576.2679305, "CO (6-5)":
    691.4730763,
                "CO (7-6)": 806.6518060, "CO (8-7)": 921.7997000,
                "13CO (1-0)": 110.20132180, "13CO (2-1)": 220.39861950, "13CO (3-2)
    ": 330.58786710,
                "13CO (4-3)": 440.7651735, "13CO (5-4)": 550.9262851, "13CO (6-5)":
     661.0672766,
                "13CO (7-6)": 771.1841250, "13CO (8-7)": 881.2728080,
                "C18O (1-0)": 109.78217340, "C18O (2-1)": 219.56035410, "C18O (3-2)
    ": 329.33055250,
                "C18O (4-3)": 439.0887658, "C18O (5-4)": 548.8310055, "C18O (6-5)":
     658.5532782,
                "C18O (7-6)": 768.2515933, "C18O (8-7)": 877.9219553}

CO_line_ha = {"CO (1-0)": "left", "CO (2-1)": "left", "CO (3-2)": "left", "CO (4-3)
    ": "left", "CO (5-4)": "left",
              "CO (6-5)": "left", "CO (7-6)": "left", "CO (8-7)": "left", "13CO
    (1-0)": "left", "13CO (2-1)": "left",
              "13CO (3-2)": "left", "13CO (4-3)": "left",  "13CO (5-4)": "left", "
    13CO (6-5)": "left",
              "13CO (7-6)": "left",  "13CO (8-7)": "left", "C18O (1-0)": "right", "
    C18O (2-1)": "right",
              "C18O (3-2)": "right", "C18O (4-3)": "right", "C18O (5-4)": "right",
    "C18O (6-5)": "right",
              "C18O (7-6)": "right", "C18O (8-7)": "right"}

CO_line_label = {"CO (1-0)": r'$\mathregular{\,CO\,(1-0)\,\,}$', "CO (2-1)": r'$\
```

```
    mathregular{\,CO\,(2-1)\,\,}$',
43                  "CO (3-2)": r'$\mathregular{\,CO\,(3-2)\,\,}$', "CO (4-3)": r'$\
    mathregular{\,CO\,(4-3)\,\,}$',
44                  "CO (5-4)": r'$\mathregular{\,CO\,(5-4)\,\,}$', "CO (6-5)": r'$\
    mathregular{\,CO\,(6-5)\,\,}$',
45                  "CO (7-6)": r'$\mathregular{\,CO\,(7-6)\,\,}$', "CO (8-7)": r'$\
    mathregular{\,CO\,(8-7)\,\,}$',
46                  "13CO (1-0)": r'$\mathregular{\,^{13}CO (1-0)}$',
47                  "13CO (2-1)": r'$\mathregular{\,^{13}CO\,(2-1)\,\,}$',
48                  "13CO (3-2)": r'$\mathregular{\,^{13}CO\,(3-2)\,\,}$',
49                  "13CO (4-3)": r'$\mathregular{\,^{13}CO\,(4-3)\,\,}$',
50                  "13CO (5-4)": r'$\mathregular{\,^{13}CO\,(5-4)\,\,}$',
51                  "13CO (6-5)": r'$\mathregular{\,^{13}CO\,(6-5)\,\,}$',
52                  "13CO (7-6)": r'$\mathregular{\,^{13}CO\,(7-6)\,\,}$',
53                  "13CO (8-7)": r'$\mathregular{\,^{13}CO\,(8-7)\,\,}$',
54                  "C18O (1-0)": r'$\mathregular{\,C^{18}O\,(1-0)\,\,}$',
55                  "C18O (2-1)": r'$\mathregular{\,C^{18}O\,(2-1)\,\,}$',
56                  "C18O (3-2)": r'$\mathregular{\,C^{18}O\,(3-2)\,\,}$',
57                  "C18O (4-3)": r'$\mathregular{\,C^{18}O\,(4-3)\,\,}$',
58                  "C18O (5-4)": r'$\mathregular{\,C^{18}O\,(5-4)\,\,}$',
59                  "C18O (6-5)": r'$\mathregular{\,C^{18}O\,(6-5)\,\,}$',
60                  "C18O (7-6)": r'$\mathregular{\,C^{18}O\,(7-6)\,\,}$',
61                  "C18O (8-7)": r'$\mathregular{\,C^{18}O\,(8-7)\,\,}$'}
62
63 # Define all possible keywords from TAP and their types
64 VALID_KEYWORDS_STR = ('obs_publisher_did', 'obs_collection', 'facility_name', '
    instrument_name', 'obs_id',
65                       'dataproduct_type', 'target_name', 's_region', 'pol_states',
    'o_ucd', 'band_list',
66                       'authors', 'pub_abstract', 'proposal_abstract', '
    schedblock_name', 'proposal_authors',
67                       'group_ous_uid', 'member_ous_uid', 'asdm_uid', 'obs_title', '
    type', 'scan_intent',
68                       'science_observation', 'antenna_arrays', 'is_mosaic', '
    obs_release_date', 'frequency_support',
69                       'obs_creator_name', 'pub_title', 'first_author', 'qa2_passed'
    , 'bib_reference',
70                       'science_keyword', 'scientific_category', 'lastModified', '
    access_url', 'access_format',
71                       'proposal_id', 'data_rights')
72 VALID_KEYWORDS_DOU = ('gal_longitude', 'gal_latitude', 's_ra', 's_dec', 's_fov', '
    s_resolution', 't_min', 't_max',
73                       't_exptime', 't_resolution', 'em_min', 'em_max', '
    em_res_power', 'em_resolution',
74                       'sensitivity_10kms', 'cont_sensitivity_bandwidth', '
    spatial_scale_max', 'bandwidth',
75                       'spatial_resolution', 'frequency', 'velocity_resolution', '
    pwv')
76 VALID_KEYWORDS_INT = ('calib_level', 'publication_year')
77
78 # Define new columns we manually add to the dataframe
79 NEW_COLUMNS = ['Obs', 'project_code', 'ALMA_source_name', 'RAJ2000', 'DEJ2000', '
    ang_res_arcsec', 'min_freq_GHz',
80                'max_freq_GHz', 'central_freq_GHz', 'bandwidth_GHz', 'freq_res_kHz',
     'vel_res_kms', 'LAS_arcsec',
81                'FoV_arcsec', 'cont_sens_bandwidth', 'line_sens_10kms', '
    line_sens_native', 'MOUS_id']
82
83 # setting the column types for the manually added columns
84 COLUMN_TYPES = {'Obs': 'Int64', 'project_code': str, 'ALMA_source_name': str, '
    RAJ2000': 'float64',
```

```
                'DEJ2000': 'float64', 'ang_res_arcsec': 'float64', 'min_freq_GHz': 'float64',
                'max_freq_GHz': 'float64',
                'central_freq_GHz': 'float64', 'bandwidth_GHz': 'float64', 'freq_res_kHz': 'float64',
                'vel_res_kms': 'float64', 'LAS_arcsec': 'float64', 'FoV_arcsec': 'float64',
                'cont_sens_bandwidth': 'float64', 'line_sens_10kms': 'float64', 'line_sens_native': 'float64',
                'MOUS_id': str}

# setting the column types for the TAP columns
COLUMN_TYPES.update({k: 'float64' for k in VALID_KEYWORDS_DOU})
COLUMN_TYPES.update({k: 'Int64' for k in VALID_KEYWORDS_INT})
COLUMN_TYPES.update({k: str for k in VALID_KEYWORDS_STR})
COLUMN_TYPES.update({'publication_year': object})
```

```
"""Download .fits files from the ALMA Archive based on keywords. Uses a modified version of
   ALminer's own function download_data in order to pick out 500 random images from the
   query. There is a cleanup script added at the end in order to delete "extra" .pickle files
   coming with the .fits files due to Python as well as larger files than 30 MB."""

import alminer
import pandas as pd
from alminer_mod import download_data_mod
import os, os.path

#'"disks around low-mass stars"' is not working and has been published as an issue at the code provider.

DIR = '<PATH TO DATA DIRECTORY>'  # Will be set to '/data' as default
KEYWORDS = ['"intermediate-mass star formation"', '"outflows, jets, feedback"', '"outflows, jets and ionized winds"', '"inter-stellar medium (ism)/molecular clouds"']

def download_routine(datadir, keywords, n_files, dryrun):
    obs_holder = pd.DataFrame()
    for i in range(len(keywords)):
        # Query and filtering
        print(" Querying with keyword : " + keywords[i])
        my_query = alminer.keysearch({'science_keyword':[keywords[i]]}, print_targets=False, tap_service='NAOJ')
        selected = my_query[my_query.obs_release_date > '2016']
        selected = selected[selected.ang_res_arcsec < 0.4]
        selected = selected.drop_duplicates(subset='obs_id').reset_index(drop=True)
        obs_holder = pd.concat([obs_holder, selected])


    print("Proceeding to download {files} fits files from the following dataframe.".format(files=n_files))
    print(obs_holder)
    print(alminer.summary(obs_holder, print_targets=False))
    download_data_mod(obs_holder, fitsonly=True, dryrun=dryrun, location=datadir, filename_must_include=[".pbcor", "_sci", ".cont"], archive_mirror='NAOJ', n_fits=n_files)

    return

```

```
35
36 def cleanup(dir):
37     for root, _, files in os.walk(dir):
38         for f in files:
39             fullpath = os.path.join(root, f)
40             if f.endswith(('.pickle')):
41                 os.remove(fullpath)
42             elif os.path.getsize(fullpath) > 30 * 1024 * 1000:
43                 os.remove(fullpath)
44
45 download_routine(DIR, KEYWORDS, n_files=500, dryrun=True)
46 cleanup(DIR)
```

DEPARTMENT OF SPACE, EARTH AND ENVIRONMENT
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS