UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

DEPARTAMENTO
DE INFORMÁTICA

PREDICTION OF MOLECULAR PARAMETERS FROM ASTRONOMICAL EMISSION LINES, USING NEURAL NETWORKS

Tesis entregada como requerimiento parcial
para optar al grado académico de

DOCTOR EN INGENIERÍA INFORMÁTICA

por

Alejandro Javier Barrientos Sessarego

Comisión Evaluadora:
Dr. Mauricio Solar Fuentes (Director de Tesis, UTFSM)
Dr. Marcelo Mendoza Rocha (Correferente, UTFSM)
Dr. Diego Mardones Pérez (Externo Nacional, UCHILE)
Dr. John M. Carpenter (Externo Internacional, NRAO)
Dr. Hernán Astudillo Rojas (Presidente Comisión, UTFSM)

AUGUST 2021

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Informática

TÍTULO DE LA TESIS:

PREDICTION OF MOLECULAR PARAMETERS FROM ASTRONOMICAL EMISSION LINES, USING NEURAL NETWORKS

AUTOR:

ALEJANDRO JAVIER BARRIENTOS SESSAREGO

Tesis presentada como requerimiento parcial para optar al grado académico de **Doctor en Ingeniería Informática** de la Universidad Técnica Federico Santa María.

Director de Tesis:

Dr. Mauricio Solar Fuentes
Departamento de Informática
Universidad Técnica Federico Santa María

Profesor Correferente:

Dr. Marcelo Mendoza Rocha
Departamento de Informática
Universidad Técnica Federico Santa María

Examinador Externo Nacional:

Dr. Diego Mardones Pérez
Departamento de Astronomía
Universidad de Chile

Examinador Externo Internacional:

Dr. John M. Carpenter
National Radio Astronomy Observatory (NRAO)

Presidente Comisión:

Dr. Hernán Astudillo Rojas
Departamento de Informática
Universidad Técnica Federico Santa María

To my family, my parents, my ancestors and my masters.

Agradecimientos

A mi esposa María José, que siempre me apoyó, alentó y levantó, en esos momentos más difíciles.

A mi hija Elisa, combustible inagotable de mi vida.

A mis padres, Paul y Sonia, quienes me dieron las herramientas y pavimentaron el camino para ayudarme a llegar hasta este hito.

A mis ancestros, cuyas decisiones y acciones permitieron a mis padres existir. Un beso al cielo para Luisa Ruiz y Eliana Ruiz quienes estarían muy felices en este momento.

A mis maestros, pues uno nunca logra nada sólo, siempre hay un maestro o maestra, un colega, un amigo o amiga, un compañero o compañera que se tomó el tiempo de enseñarte. Mi gratitud va para ellos también. Mención para los maestros Solar, Astudillo, Mendoza y todos aquellos que fueron mis profesores y compañeros en el programa de Doctorado. Agradezco especialmente a Pabla Valdebenito por toda su ayuda administrativa durante mi estadía en el programa.

Mi agradecimiento para los doctores Sergio Martín, Jon Holdship y Victor M. Rivilla, por su inestimable colaboración en el desarrollo de este trabajo.

Un agradecimiento especial para Sensei, Jorge Kishikawa, del Instituto Cultural Niten, por guiarme en el Camino y enseñarme los secretos del arte de la espada Samurai, en búsqueda de la felicidad, mediante la espada que da la vida. Un abrazo a mis camaradas y Senpais presentes y pasados del Dojo, que me apoyaron en esta batalla.

Finalmente, mi gratitud para Jorge Ibsen, José Parra y mis colegas del Archive/Pipeline Operations Group, del Joint ALMA Observatory, por darme el apoyo necesario para lograr esta meta. Un abrazo para mis colegas Diego Arredondo, Juan Cortés, Paulo Cortés y Antonio Hales.

Para todos ellos y a los que olvidé mencionar, “domo arigatou gozaimashita!”

Santiago, Chile — Alejandro Javier Barrientos Sessarego

Agosto 2021.

Financiamiento

Este trabajo no hubiese sido posible sin la valiosa beca de Arancel de la Dirección de Investigación y Posgrado (DGIP) de la UTFSM. Por favor reciban mis más profundos sentimientos de gratitud.

Resumen

La astronomía molecular es un campo que está floreciendo en la era de los grandes observatorios tales como el Atacama Large Millimeter/submillimeter Array (ALMA). Con radio telescopios modernos, sensibles y de alta resolución, tales como ALMA y el Square Kilometer Array, el tamaño de los cubos de datos está escalando rápidamente, generando una necesidad de poderosas herramientas automáticas de análisis. Este trabajo explora la habilidad de realizar predicciones de parámetros moleculares, tales como temperatura de excitación (T_{ex}) y densidad de columna ($log(N)$) desde líneas espectrales astronómicas, mediante el uso de redes neuronales. Se usaron como casos de prueba, los espectros de CO, HCO^+, SiO y CH_3CN entre 80 y 400 GHz. Los espectros de entrenamiento fueron generados con MADCUBA, una herramienta de análisis espectral, del estado-del-arte. El algoritmo presentado a continuación, fue diseñado para permitir la generación de predicciones para múltiples moléculas en paralelo, de una manera escalable y que presenta una aceleración lineal. Usando redes neuronales, es posible predecir la densidad de columna y la temperatura de excitación de estas moléculas con un error absoluto medio del 8.5% para CO, 4.1% para HCO^+, 1.5% para SiO y un 1.6% para CH_3CN. La precisión de la predicción depende del nivel de ruido, la saturación de la línea y el número de transiciones. Se realizaron predicciones sobre datos reales de ALMA. Los valores predichos por la red neuronal para estos datos reales difieren en sólo un 13% de los datos de MADCUBA en promedio. Las limitaciones actuales de la herramienta incluyen la no consideración del ancho de línea, tamaño de la fuente, múltiples componentes de velocidad y mezcla de líneas.

Palabras Claves: Astronomía molecular, Aprendizaje de Maquina, ALMA FITS, Astroquímica, MADCUBA, Astroinformática, MolPred.

Abstract

Molecular astronomy is a field that is blooming in the era of large observatories such as the Atacama Large Millimeter/-submillimeter Array (ALMA). With modern, sensitive, and high spectral resolution radio telescopes like ALMA and the Square Kilometer Array, the size of the data cubes is rapidly escalating, generating a need for powerful automatic analysis tools. This work explores the ability to perform predictions of molecular parameters, such as excitation temperature (T_{ex}) and column density ($log(N)$) from astronomical spectral lines by the use of neural networks. We used as test cases the spectra of CO, HCO^+, SiO and CH_3CN between 80 and 400 GHz. Training spectra were generated with MADCUBA, a state-of-the-art spectral analysis tool. Our algorithm was designed to allow the generation of predictions for multiple molecules in parallel, in a way that is scalable, and presents a linear speedup. Using neural networks, we can predict the column density and excitation temperature of these molecules with a mean absolute error of 8.5% for CO, 4.1% for HCO^+, 1.5% for SiO and 1.6% for CH_3CN. The prediction accuracy depends on the noise level, line saturation, and number of transitions. We performed predictions upon real ALMA data. The values predicted by our neural network for this real data differ by 13% from the MADCUBA values on average. Current limitations of our tool include not considering line width, source size, multiple velocity components, and line blending.

Keywords: Molecular astronomy, Machine Learning, ALMA FITS, Astrochemistry, MADCUBA, Astroinformatics, Mol-Pred.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Spectroscopy relies on the analysis of the interactions between matter and electromagnetic radiation. Matter in the universe emit or absorb radiation and we can observe such interactions by looking at the variations of the electromagnetic spectrum at different frequencies. The variations are continuous in nature, and the interactions are defined by radiative transfer equations. Such equations contain a large number of parameters, which include absorption, emission and scattering coefficients, for a variety of conditions surrounding the matter.

The study of the molecular composition of objects in space has been the primary driver of extensive research since the 1930s (Swings and Rosenfeld, 1937). Through the use of spectroscopy techniques over 200 molecules have been identified in interstellar or circumstellar medium (Woon, 2020), and the number keeps growing every year. The analysis process to detect these molecules is quite complex and requires effort from observers, astrochemists and laboratory spectroscopists (Cernicharo, 2012). The construction of astronomical observing facilities like the Atacama Large Millimeter / Submillimeter Array (ALMA) has created the possibility to observe the universe with an unprecedented combination of sensitivity and angular resolution. Such facilities allow astronomers to study the physical and chemical properties of the molecular gas in a variety of sources in the Universe (e.g. Nakajima et al., 2015).

The introduction of ever larger and more sensitive instruments in astronomy and the latest developments in computing performance has generated a need for tools to support the analysis of data sets (Berriman and Groom, 2011). This is illustrated by the number of new astronomical facilities that are implementing automated data reduction pipelines such as ALMA (Lightfoot et al., 2008), Vera C. Rubin Observatory (formerly LSST; Jurić et al., 2015) and E-ELT (Mach et al., 2016). Telescopes like the Square Kilometer Array (SKA), that will be deployed in the very near future, will even rely only on fully automatic reduction pipelines, due to the large amounts of data they will produce (Farnes et al., 2018).

Current facilities produce crowded spectra even in sources where previous facilities were only able to detect a limited number of bright transitions of the most abundant species. Line identification and extraction of physical parameters is generally still a manual, human-supervised and time consuming process even making use of state of the art tools briefly described below.

Having established that automated tools are necessary, we can contribute, by taking advantage of the continuous nature of the relationships between matter and energy, and use supervised machine learning models to decrease the level of required human interaction and predict some of the parameters that conform an electromagnetic spectrum. In this case, regression

techniques are suitable, due to their ability to perform predictions of target variables in continuous domain (Bishop, 2006). For the sake of probing feasibility and as a pilot study, in this work we constrain our efforts to the prediction of the parameters of excitation temperature (T_{ex}) and column density ($log(N)$) from molecular spectra.

The motivation for this thesis rises from the will to explore the opportunities that the Astroinformatics field offers. Our initial research involved different experiments such as using Artificial Neural Networks and Support Vector Machines (Barrientos and Solar, 2016), for the task of classifying astronomical spectra. Other experiments involved the use of mixed membership models (Mendoza et al., 2017), performing an analogy from the text world, where we would use molecular transitions as keywords. Unfortunately, such efforts were not fruitful as we experienced the lack of a suitable training examples source. Years later, a collaboration with colleagues at ALMA and other astronomical institutions, brought forward the question regarding predicting molecular parameters from a spectrum. This is how the research evolved, to find the question that we will explore in this work.

Chapter 2 describes the literature review, with a mention of the astronomical complexities involved in the process of line analysis, references to other projects that can be used in the molecular parameter estimation problem and a brief overview of Neural Network theory. Chapter 3 describes our proposal, in terms of objectives, hypothesis, design, workflows, training methods and a discussion of performance. Chapter 4 describes and discusses our test results with synthetic data, with a number of parameter combinations for the neural networks. We also present our results with astronomical data, performing predictions for $log(N)$ and T_{ex} for a spectrum coming from an ALMA Large Program project. Finally, we present a scalability test and performance analysis of our implemented solution.

Chapter 5 summarizes our conclusions and comments on future work.

Chapter 2

Literature Review

2.1 Astronomical Complexities

This section is intended for other researchers in the Computer Science field, that wish to venture into astrophysical work. It is not our objective, to go deep into these concepts, as they are mentioned for illustrative purposes.

There are a number of astronomical complexities that surround the problem of line analysis. It is possible to split the complexities into two levels, one at the astronomical source level, and another at the molecular level. This section is mostly based on the work of Tennyson (2005).

2.1.1 Source Complexities

In this section we describe some of the complexities inherent to the observed astronomical sources and their nearby regions in space.

Temperature

Molecules in the cosmos are in motion, thus contain energy that can be correlated to a measure of Temperature, most of the objects emit radiation in a relationship described by Planck's Law, which relates the radiation emitted by a "black body" as a function of its absolute temperature. There are several definitions for temperature, that can be used, depending on the problem at hand (Karttunen et al., 2017) . For our purposes, we use the excitation temperature definition by Mangum and Shirley (2015) in Equation 2.1 which defines the population between two energy levels.

$$T_{ex} = \frac{\frac{h\nu}{k}}{\ln \frac{n_l g_u}{n_u g_l}} \tag{2.1}$$

where n is the density (cm^{-3}) in the upper (u) or lower (l) energy level for a transition with frequency ν, g_u and g_l are the numbers of different physical states, h is Planck's constant, $6.626068 \times 10^{-34} Js$ and k is the Boltzmann's constant.

This work assumes that the excitation temperature is constant over all the transitions (Local Thermodynamical Equilibrium, LTE).

Optical Depth

Optical depth is a measurement of the transparency of the medium where the observed molecules exist, and is defined by the Equation 2.2:

$$\tau = \int \kappa dz \tag{2.2}$$

Where the integral runs over the path of light being absorbed, κ is called an Extinction Coefficient, and represents a product of the number density of the atoms and their opacity at a certain wavelength which is the ability to absorb or block photons.

If the optical depth is large ($\gg 1$), then it is not necessary for all the atoms to absorb the light to produce a transition. Alternatively, if the optical depth is small ($\ll 1$), the intensity of the observed spectrum is directly proportional to the number of absorbers, called column density. In this case, it is possible to estimate the column density which provides a number of molecules per unit of area along the line of sight (Mangum and Shirley, 2015).

Abundance

Abundance is a measurement that allows to compare the quantity of a given object in a certain environment, with respect to other objects. The comparison could be made in terms of the fraction of mass, volume or mole (fraction of atoms in the overall formula).

The abundance can be determined only if the intrinsic line strength being observed is known. The line strength of a transition is said to be directly related to the number of atoms that take part in the transition, under suitable conditions of optical depth.

Critical Density

Is an indicator that allows the observer to infer if the species that is emitting is in Local Thermodynamic Equilibrium (LTE) with its environment. For emissions below the critical density, it is said that the emissions are dominated by radiation. On the other hand, for emissions above the critical density, the emissions are said to be generated by collisions.

Source Kinematics

The source possesses some kinematic properties such as the velocity with respect to the local standard of reference (VLSR) and the distance to the observed objects in the sky. The distance affects the apparent frequency in which the emissions are detected, the frequency value "shifts" to either sides of the spectrum, depending on whether the objects in moving towards (blueshift) or away (redshift) from the observer's point of view, this phenomenon is described by the Doppler Effect.

The Doppler shift is modeled by the Doppler formula described in Equation 2.3

$$\frac{v}{c} = \frac{\Delta\lambda}{\lambda} \tag{2.3}$$

In the astronomical observations, the exact measurement of the distance to the observed object is not always available, which can cause problems trying to compare spectral features which have different Doppler effects, in principle it would

be necessary to perform corrections to have the observed object at the "rest frequency" so it is comparable.

A higher distance to the source would be reflected in a weaker signal.

Pressure

The density of the environment in which the transitions are taking place, can be inferred observing the line profile, although this requires very high resolution. Spectral line width is influenced by the collisions between species, the more collisions, the broader the line and therefore the environment is denser and has a higher pressure.

Tennyson (2005) also mentions that lines are also broadened by thermal motions, as hot species move faster than cold ones. Emphasis is placed that in order to accurately observe the effects of this phenomenon, a very high resolution is required.

2.1.2 Molecular complexities

In this section we describe the complexities inherent to the molecules that exist in the observed astronomical sources.

Abundance

This concept was already explored in the source complexities, but it is also applicable to the molecular complexity.

Internal Structure

Any molecule can be represented as a collection of the quantum numbers of its atoms, n, l, m and s among others, where n represents the quantum energy level, l represents the atom's angular momentum, m which represents the magnetic quantum number and s which represents the spin quantum number. With these numbers, it is possible to obtain information about the position of the last electron of the atom, the shape of the orbital path and the direction of the electron motion. A molecular quantum number k is related to the rotation of the molecule along its main rotation axis.

Electronic Transitions

All atoms and molecules have a number of discrete energy levels, an electron can either absorb energy from a source and go up one level, or it release energy in a process called emission and go down one level. A transition is a jump between energy states, that is expressed as a peak (for emission) or a valley (absorption), in the spectrum at a certain frequency. Each molecule species, can have a very high number of transitions, given enough energy is involved. Some transitions may coincide for different species, at the same frequency, however the complete series of transitions is unique for each species, the term "molecular fingerprint", can be coined to describe this effect. In order to increase the confidence of detecting certain species, it would be necessary to detect as many transitions as possible.

The difference between energy levels is described by Equation 2.4:

$$E = h\nu = \frac{hc}{\lambda} \tag{2.4}$$

This wavelength, λ, corresponds to the exact energy difference, E, between the energy levels via the Planck relationship, where ν is the frequency of the light and c is the speed of light. h is Planck's constant, and $h\nu$ gives the energy carried by each particle of light, known as a photon.

Not all transitions are equally likely, in addition to the transition frequency, there is also a probability associated with that transition. It requires light from another source at the exact wavelength in order to make the transition.

Consider that even though these conditions cannot be artificially arranged in nature as they can be in the lab, there are some astrophysical scenarios where this can be observed, for instance, on the atmospheres of stars, where the core of the star provides the necessary continuum light to perform absorption. The source of the light behaves as a black body curve with the temperature of the star.

Molecular Rotation

The structure of the molecule allows it to rotate along one or more rotation axes, these *rotors* can be used to determine line positions and transition energy levels, some types of rotors are: Lineal, Symmetrical Top and Asymmetrical Top.

Vibrational Rotation

The bonds between atoms of a molecule contain potential and kinetic energies, which make the molecule vibrate in certain levels called vibrational modes, each mode contains a number of transitions determined by a function called potential surface given by a bond Dissociation energy, bond length and potential energy.

The relationship between Electronic, Rotational and Vibrational energies is described by the Born-Oppenheimer approximation (Born and Oppenheimer, 1927).

Frequency

The radiation emitted by the objects in the sky can be detected in the form of an electromagnetic spectrum. The spectrum has two main components, Frequency and Radiation Intensity, both components can be affected by a number of astronomical conditions.

Line Shape

The shape of the spectral lines is influenced by the kinematics of the emission source, and the molecular rotovibrational configuration.

2.1.3 Other Factors

There are some other factors that could influence the incoming spectra such as Electromagnetic Fields, UV Ionization, Collisions and Recombinations. However we will not dive into these.

2.2 Existing applications

Several tools have been developed over time, to assist in line analysis. These tools have varying degrees of automation, we shall review them next.

2.2.1 Splatalogue

Splatalogue[1], is an online catalog of almost 6 million spectral lines, originated from several astronomical catalogs, it contains search tools and filters, which allow to query its database, and return collections of molecular transitions along with information useful to study the composition of astronomical objects. Splatalogue is not a line analysis tool by itself, but it should be included in this review, for completion purposes.

2.2.2 CASA

The Common Astronomy Software Applications (CASA, McMullin et al., 2007) package, developed by the National Radio Astronomy Observatory (NRAO), is the main data reduction software at the Joint ALMA Observatory. It contains several tasks that allow the user to reduce the raw data captured by the telescope. The CASA development team is constantly adding new functionalities, including a data reduction pipeline integration (Emonts et al., 2018). It allows the user to connect to Splatalogue catalog via casaviewer. This enables the user to identify lines by overlaying transitions from the catalog on their spectra. In this case, there is no automation. It does not seem to contain functions that allow for estimations of molecular parameters. Casaviewer is scheduled to be superseded by the Cube Analysis and Rendering Tool for Astronomy (CARTA, Comrie et al., 2020)

2.2.3 ADMIT

The ALMA Data Mining Toolkit (ADMIT, Teuben, 2015) is a code library, developed by the Universities of Maryland and Illinois, intended to enhance the capabilities of CASA, was also developed by NRAO. It includes a Spectral Line identification algorithm, which according to the documentation (Teuben, 2015) "first looks for segments spectral line emission. Spectral peaks are then searched for within these regions. Once spectral peaks are located, patterns are searched for (specifically patterns of rotation, expansion, and collapse). Each peak is then matched to the tier 1 list of molecules. If no match is found then the CASA task slsearch is used to generate a potential line list based on the peak's frequency and line width. This list is narrowed down to a single identification by comparing the transition energies, constituent atoms, and line strength". It does not contain Machine Learning elements to improve the accuracy of the identification.

2.2.4 MyXCLASS

The XCLASS interface is a package for data post-processing in the CASA environment. It includes the myXCLASS program (Möller and Schilke, 2015) which is a one dimensional radiative transfer code, which uses the Cologne Database for Molecular Spectroscopy (CDMS) and NASA's Jet Propulsion Lab (JPL) molecular data. It contains routines to fit models to observed spectra. The fitting model can be automated with multiple approaches as mentioned in Schilke et al. (2015). According to the interface manual, "The myXCLASS function models a spectrum by solving the radiative transfer equation for an isothermal object in one dimension, called detection equation".

2.2.5 CASSIS

The Centre d'Analyse Scientifique de Spectres Instrumentaux et Synthétiques (CASSIS, Vastel et al., 2015) is a standalone line analysis package, developed by the Institut de Recherche en Astrophysique et Planétologie (IRAP). The goal of this

[1] `https://splatalogue.online`

software is to integrate astronomical data, telescope parameters, spectroscopic databases (CDMS, JPL, NIST), collisional databases (BASECOL, LAMBDA) in order to model the physical attributes of the source and perform line identifications. The objective is to compute synthetic LTE models that can be compared with observations. CASSIS works with FITS files, coming from CLASS/GILDAS software. It uses the services from the International Virtual Observatory Alliance (IVOA) to retrieve any spectra.

2.2.6 RADEX

RADEX (van der Tak, F. F. S. et al., 2007) is a tool to perform statistical equilibrium calculations, developed by the University of Leiden. It allows to perform estimations of some physical conditions, maintaining certain parameters fixed. It does not use machine learning elements. The code is public, and also contains an online version for quick estimations.

2.2.7 MADEX

The MADrid EXcitation code (MADEX, Cernicharo, 2012), has been developed for the last 30 years by Jose Cernicharo. The code allows to predict the intensity of the molecular lines, given certain assumptions of LTE and using a Large Velocity Gradient approximation. It has all the catalog information embedded a FORTRAN code and in some measure, contains a higher degree of accuracy in some areas like rotational constants. The catalog grew with additions from IRAM and Nobeyama telescopes. Spectral models can be generated from a parameter file, and those models can be compared to the actual observations to identify transitions of a certain species. Although this comparison is not automated and requires the tuning of many input parameters. The code could be requested from the author, it was not free to distribute or modify.

2.2.8 MADCUBA

The MAdrid Data CUBe Analysis (MADCUBA, Martín et al., 2019) package, is a tool for line analysis and spectroscopic work. It was developed by the Center of Astrobiology of Madrid (CSIC-INTA), and provides support for data cubes from multiple astronomical facilities. It contains a feature called Spectral Line Identification and Modelling (SLIM-AUTOFIT) which allows the automatic fitting of molecular parameters returning "the best non-linear least-squared fit using the Levenberg-Marquardt algorithm". It allows the possibility to generate synthetic spectra programmatically, given the appropriate parameters, and also to merge several data cubes across a frequency axis. Although MADCUBA does not contain Machine Learning components, we found it to be very useful as a primary source for training examples. Another limitation of MADCUBA, is that in order to produce a fitting of parameters, it requires an initial estimation which is not too distant, otherwise, it will not converge. This limitation makes MADCUBA dependant on human interaction.

We can summarize the differences between the presented software packages in Table 2.1, where we can see that some of them allow to have automated estimations of physical parameters. However all of them share a common factor, which is that none of them use Machine Learning (ML) elements as a way to improve their estimations. This opens a window for exploration and contribution.

2.3 Neural Networks Overview

This section is intended to provide a brief overview of the concept of Neural Networks. Plenty of literature can be found on the matter, but we would like to include some basic definitions, for completion purposes.

Table 2.1: Comparison between existing tools

	CASA	ADMIT	RADEX/MADEX	myXCLASS	CASSIS	MADCUBA
Automated Predictions	×	×	×	✓	×	✓
Use of ML	×	×	×	×	×	×

The first references to the concept can be found in the work of Mcculloch and Pitts (1943), where an attempt to describe the workings of human neurons into logical terms is made. Neurons in the human brain, are a specific type of body cell, that react to stimuli from the surrounding environment, generating a corresponding reaction from the body. It can be inferred, that there is a relationship between the input stimulus, and the type of reaction that is generated. Other studies were done in the subsequent years, being the creation of the "Perceptron" (Rosenblatt, 1958) one of the most relevant, as it addresses the subject of how a stimulus that is remembered, can influence the future outcome of the reaction. This required the switch the perspective from a logical point of view, to a probabilistic theory approach. Another interesting conclusion is that if the Perceptron had a higher exposure to certain stimuli, its reactions would be easier to differentiate. This characteristic, made the Perceptron a widely used method to perform classification tasks.

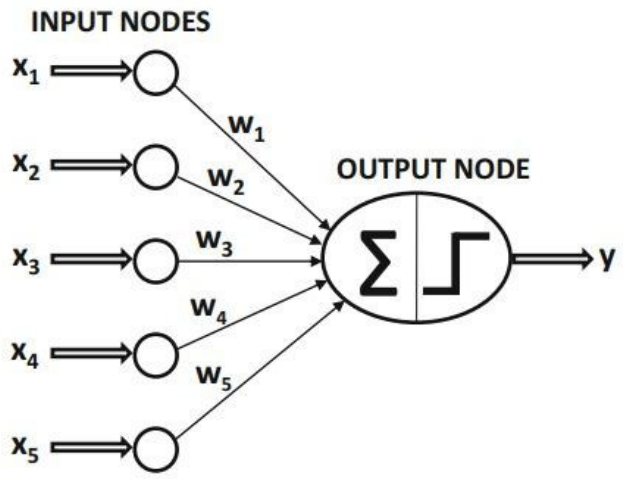


Figure 2.1: Perceptron Single Layer Neuron diagram

We can use the Perceptron illustration from Aggarwal (2018) in Figure 2.1 to study the flow of information. Initially, the Perceptron is designed to sense stimuli from the environment via d different sensors or *features*, in the diagram, $d = 5$. Let $\bar{X}$ be the vector of input features $\bar{X} = [x_1, ..., x_d]$. In the example, $\bar{X} = [x_1, x_2, x_3, x_4, x_5]$. The output node collects all the stimuli contributions via a Sum function. Each stimulus is multiplied by a weight factor w, which is used to represent the specific contribution of the stimulus, as not all the stimuli contribute in the same manner to the neuron. The weights can also be represented by a vector $\bar{W} = [w_1, ..., w_d]$, In the example, $\bar{W} = [w_1, w_2, w_3, w_4, w_5]$. The overall input can be defined as the multiplication of the each stimulus by its weight factor. The response of the neuron , defined as $\hat{y}$, will be determined by the computation of an *activation function*, defined as σ, which will be applied to the overall input.

Therefore, we can express the response of the neuron as shown in Equation 2.5:

$$\hat{y} = \sigma(\bar{X} \cdot \bar{W}) = \sigma(\sum_{j=1}^{j=d} w_j x_j) \tag{2.5}$$

There are several activation functions σ that can be chosen, some are simple as the *sign* function, which is useful for binary classification problems. There are other functions, such as rectified linear units (ReLU, Agarap, 2018), swish

(Ramachandran et al., 2017) and sigmoid (Cybenko, 1989), among others.

So far, we have reviewed that a neuron can react to its environment, and can provide certain level or response. We can use a process called *Supervised Learning*, to make the neuron adapt to an expected behaviour, this is done by using the ability of the neural network, to remember its prior experiences. We can determine a vector of expected responses y, also called a *label set*, and define a *loss*, as the difference between the expected responses y and the neuron's actual responses $\hat{y}$. The goal of minimizing the loss is defined as a *loss function*. One way to calculate the loss function is through the sum of the squared errors, also known as *least squares regression* as shown in Equation 2.6. Although there are other loss functions that can be chosen, such as the Mean Average Error, which is a choice suitable to address regression problems, as it is not sensitive towards outliers.

$$\text{Minimize}_{\hat{W}} L = \sum_{(\hat{X},y)\in\mathcal{D}} (y - \hat{y})^2 \tag{2.6}$$

To allow the neurons to learn, we define a *training set* $\mathcal{D}$ as a list of examples, where each example is comprised of a feature vector $\bar{X}$ which contains a series of inputs that will be fed to the neuron, accompanied by a vector of *labels* y, which describe the expected responses of the neuron. In the case of the Perceptron, the neuron only has one response, but more complex networks can be created, where there are multiple neurons in the output layer, causing the network to have multiple responses.

It should be noted, that the number of labels per example, is the same as the number of neuron responses. The neuron learns, by performing an update to the weights vector, based on the calculation of the loss. Therefore, each new weight is calculated by compensating the error as shown in Equation 2.7 where α is called a *learning rate*

$$\hat{W} = \hat{W} + \alpha(y - \hat{y})\hat{X} \tag{2.7}$$

A technique called *backpropagation* (Kelley, 1960), can be used to perform recursive updates of the weights, causing the loss function to change to a certain value, lower than the initial one. This process can be repeated multiple times, to improve the result until convergence, thus achieving the objective of minimizing the loss function. Finally, the neural network will display a preference, for its known outputs, when given an input stimulus that is similar to what the network has trained.

Chapter 3

Proposal

3.1 Objectives and Hypothesis

3.1.1 General Objective

- To develop an algorithm that is capable of predicting molecular parameters $log(N)$ and T_{ex} for an initial sample of four molecules, provided an input spectrum. This set should be able to be incrementally extended to more parameters and to the full set of species available in spectroscopic catalogs.

3.1.2 Specific Objectives

- To design an algorithm that can accept training examples from a valid source, and use Supervised Learning techniques to learn the relationships between the spectral features in a way that can be used later to obtain predictions of column density ($log(N)$) and excitation temperature (T_{ex}).
- To integrate parallelization techniques into the algorithm design, while taking advantage of parallelization technologies.
- To obtain prediction values for a minimum number of four molecules using astronomical data.
- To compare said predictions with another tool available in the field.
- To perform tests that can display the scalability potential of the algorithm.

3.1.3 Hypothesis

It is possible to formulate the following hypothesis, to fulfill the objectives previously stated:

"By using a newly designed algorithm, that uses parallelization techniques and neural networks, it is possible to recognize patterns from an astronomical input spectrum and predict the column density and excitation temperature of a group of trained molecules which emission contributes to generate said spectrum."

This hypothesis can be falsified, if an example is found, where a pattern cannot be recognized from the input spectrum, and the column density and excitation temperature cannot be predicted.

3.2 Methodologies

Given that astronomical spectra can be modelled from their underlying physical parameters, it should be possible to predict those parameters using a regression model. In this work, we use several neural networks which take information from the spectrum as an input.

Whilst even simple radiative transfer models require several parameters, we consider only $log(N)$ and T_{ex} in this work. The column density is an important quantity as it is a measure of how much of a species is present towards an object. The intensity of a line approximately scales with the column density when lines are optically thin. The excitation temperature reflects the relative population of energy levels of a molecule (e.g., following a Boltzmann distribution at local thermal equilibrium) and helps to characterize the conditions of the gas. It primarily determines the relative line intensity among the transitions of a given species.

This proposal suggests the use of a feed-forward Neural Network in a first stage, to process the input spectra, and train the learner, to recognize $log(N)$ and T_{ex}, for a series of molecules, which number of rotational transitions from the vibrational ground state in the frequency ranges considered in this work are described in Table 3.1

Table 3.1: List of molecular species and number of transitions.

Species	Name	Number of Transitions (80-400GHz)
CO	Carbon monoxide	3
HCO^+	Formylium	4
SiO	Silicon monoxide	8
CH_3CN	Methyl cyanide	437

In order to obtain the desired outcome, we propose the following algorithm, that we have named *MolPred*.

Algorithm 1: *MolPred* algorithm to perform molecular parameter predictions.

Result: Predictions for $log(N)$ and T_{ex} for all trained Molecules

Requirements: *Pre-trained neural networks*

Start by reading input spectrum;
Prepare individual components for prediction including scaling of the input spectrum;
Perform parallel predictions for all trained molecules by instancing multiple predictors that will feed upon the individual components at the same time;
Wait for all predictions to finish, to retrieve the results and scale back the predictions to the original dimensions;
Finish by displaying prediction results;

For this purpose, we have developed a tool homonimously called *MolPred*, which implements the above algorithm. To comply with the objectives, this set of parameters and molecules, can be incrementally extended to more parameters and to the full set of species available in spectroscopic catalogs. In this first prototype, we will not estimate uncertainties in the prediction values. However, they could be calculated later on as explained by Nair[1], by creating alternative datasets

[1] `https://medium.com/comet-ml/estimating-uncertainty-in-machine-learning-models-part-1-2bd1209c347c`

via Bootstrap Sampling (Efron, 1979), where a confidence interval can be defined.

We can establish a difference with other tools available, by introducing the use of ML techniques. We can see in Table 3.2, a comparison between *MolPred* and the tools mentioned in section 2.2.

Table 3.2: Comparison between existing tools and *MolPred*

	CASA	ADMIT	RADEX/MADEX	myXCLASS	CASSIS	MADCUBA	MolPred
Automated Predictions	×	×	×	✓	×	✓	✓
Use of ML	×	×	×	×	×	×	✓

3.2.1 Overall Design

An overview of the *MolPred* program is described in Figure 3.1. The flow starts by receiving an input spectrum for which *MolPred* will generate predictions of $log(N)$ and T_{ex}. The input spectrum is pre-processed as explained below, so the predictions for all the molecules can be done in parallel via individual neural networks. After all predictions are done, the results are collected and handled in a post-processing stage, where the resulting predictions are presented to the user.

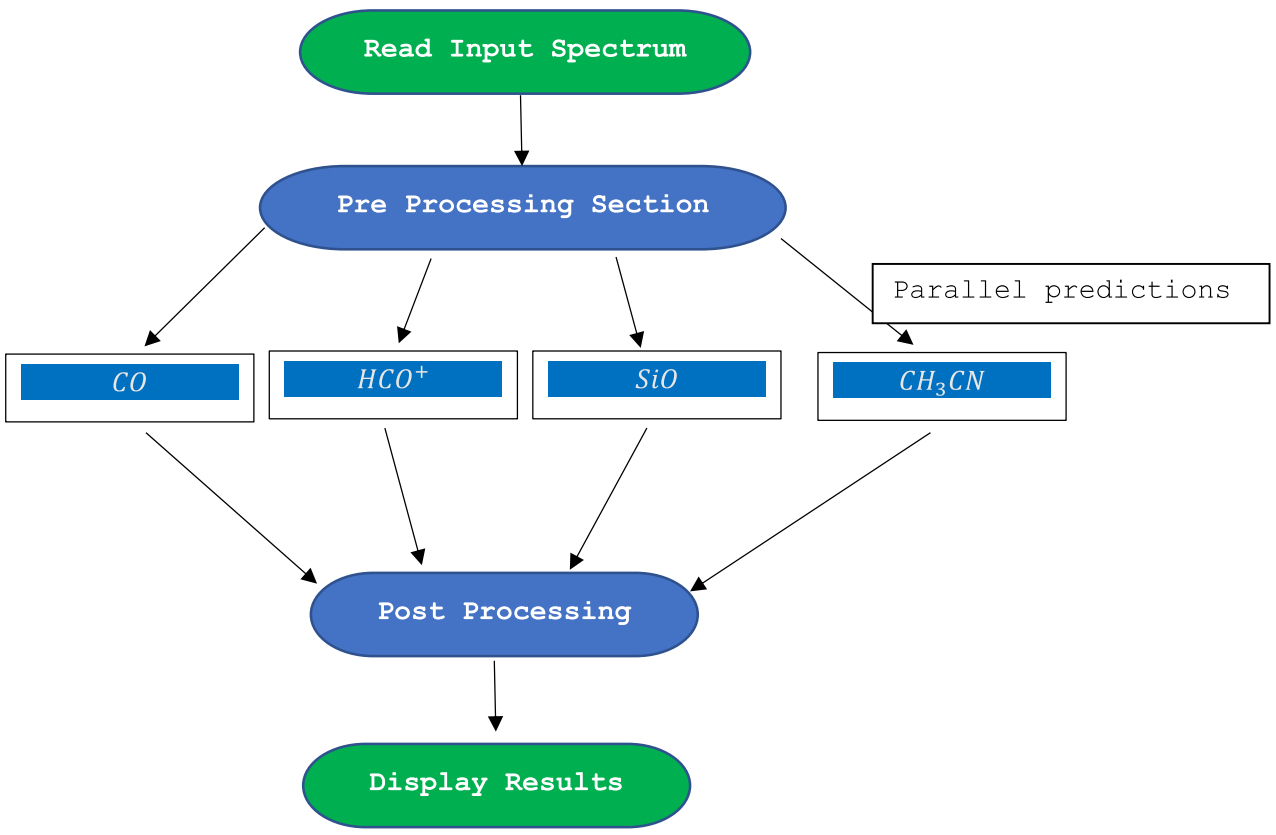


Figure 3.1: *MolPred* workflow diagram

In the pre-processing stage, the input arrays for the individual neural networks are generated. The neural networks require only the peak intensity of every transition of the molecule for which it predicts. To obtain this, the intensity of the input spectrum channel closest to the rest frequency of each transition of a molecule is extracted. These intensities are then scaled to take values from 0 to 1 using the minimum and maximum values that were in the training data.

In real astronomical data, it is possible that some transitions might not be observed. If there is no channel within 10 MHz of a transition, then *MolPred* assumes the transition is missing from the data and the pre-processing stage simply inserts a zero for the intensity of that transition.

In the prediction stage, the neural networks trained to predict the $log(N)$ and T_{ex} of each molecule take the input array and produce a prediction for $log(N)$ and T_{ex}, each scaled between 0 and 1. The neural network used in this stage is the best network from the grid discussed in section 3.2.4, the performance of each of these networks is discussed in Chapter 4. Finally, in the post-processing stage, the scaled predictions of $log(N)$ and T_{ex} are transformed to their actual values.

This solution relies on several python support scripts, that allow to create the necessary training examples, training models and provide preliminary testing results and finally use these trained models, to perform predictions upon an input spectrum coming from real observations. We've opted to perform offline training, as it is a time consuming process, that is only done once, which results can be saved and later retrieved for immediate use in predictions.

The relationship between the supporting scripts for training is described by Figure 3.2

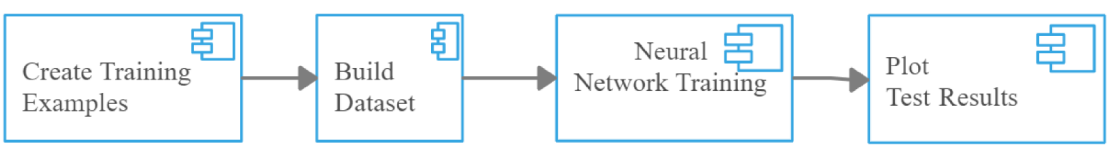


Figure 3.2: Training Workflow

In section 4.2, predictions for $log(N)$ and T_{ex}, for all molecules are presented. This is the final product, the users can use these predictions as a starting point for their work.

3.2.2 Neural Networks Design Process

To design the neural networks required for the implementation of the solution, first, we need to adopt a spectral line representation. We found that the representation selected by Miranda and Cabrera (2015) is useful for our purposes. A spectrum can be represented as a list of ordered pairs (frequency, intensity), where each frequency represents a feature, and the intensity represents the feature value. However, considering a 10 MHz channel width, the number of features needed to represent a single spectrum can become very large, depending on the overall frequency range defined for the experiment, in our case, this represented up to 32000 channels per training example. Holding these large spectra in memory, could make the training prone to out-of-memory errors, considering the dimensionality of the problem and our available hardware. Instead, a possibility would be to store one file on hard drive per spectrum, and load smaller batches in memory, which could be fed into potential learning algorithms. It should be noted that storing spectra representations in disk, would allow to process large size spectra without exhausting system memory, at the expense of adding a bottleneck into training algorithms due to be constantly loading them for training. However this approach would not be adequate when working with GPU, as they are better suited when loading the complete data set in memory. A better approach was to perform a dimensionality reduction on the inputs, as most of the intensities in the training example would be zero, for regions where no transition was present, which brought us to the conclusion that having a representation where we would use the same number of inputs as transitions in each molecule was a good compromise, although that presented other types of issues, as we describe in our results section. This is also the reason why we used two complementary scripts to create the training examples, we should merge those into just one script that reads the MADCUBA example, and immediately stores the large dataframe into disk, as it is done in the *build_dataset* script.

Having determined that the number of inputs of each neural network will depend on the transitions of the corresponding molecule, we can see an example in Figure 3.3, The input layer is conformed by three nodes, which receive the molecule

intensity at the specific transition frequencies. The layers of the network are densely connected and each network node uses an activation function that was passed as a parameter (blue nodes). The output layer has a fixed activation function of type RELU (green nodes), which is the default activation function for the Dense class in the keras library. The output of each node, represents our target Column Density and Excitation Temperature. This output layer should be modified in the future, to a layer of variable type, depending on the number of output parameters that are wished to predict. Doing this, would open the possibility to train multiple astronomical sources, by learning new parameters such as source size, line width and/or source velocity.

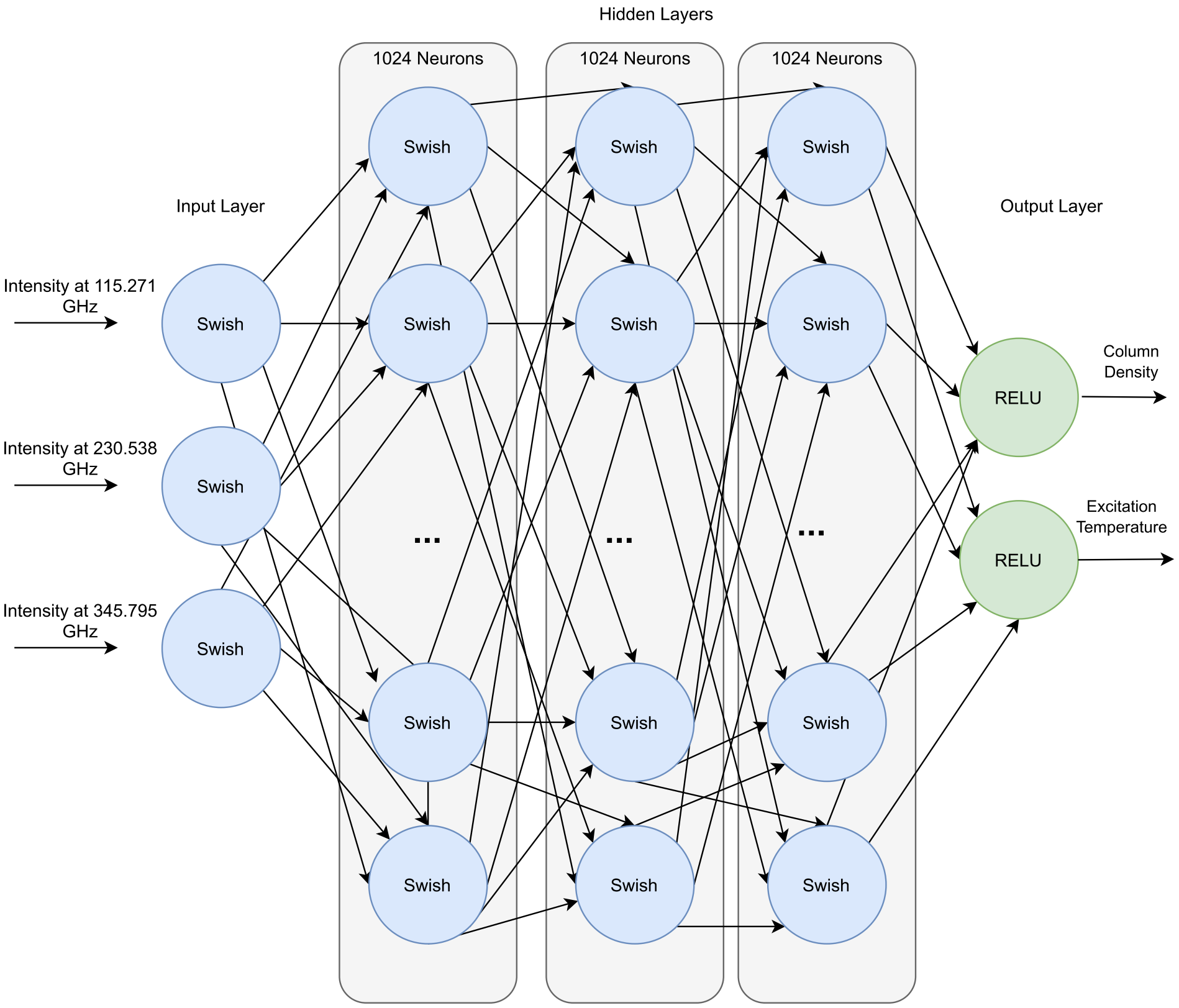


Figure 3.3: Diagram of the neural network structure, for CO molecule, 3 hidden layers of 1024 neurons each, with Swish activation function

With regard to the use of synthetic training data, in our initial experiments, we encountered the problem of training data availability. We have not been able to find a repository of standardized astronomical spectra examples, since they are very diverse, and there are several variables related to the spectrum generation, in the case of ALMA, these variations

can include the astronomical source, telescope observation modes, manual or automatic pipeline data reduction process, data reduction software versions, maturity of the ALMA Data Archive / Pipeline, etc. Years later, when MADCUBA was presented, this issue was in part solved, as the examples primary source would be handled by MADCUBA, we would generate as many variations of the primary examples as needed, by adding Gaussian noise, as will be explained later.

3.2.3 Training Data

The data used to train the neural networks which make up *MolPred* are a set of synthetic spectra generated by the SLIM LTE spectral model, which is part of the MADCUBA software package. It makes use of the spectroscopic parameters from JPL catalog (Pickett et al., 1998). We generate LTE spectra for each species, with 1 MHz resolution between 80 and 400 GHz using column densities in the range $log(N) = 12$ cm^{-2} to $log(N) = 19.9$ cm^{-2}, with steps of $log(N) = 0.1$ cm^{-2} and temperatures from 10 K, performing multiplicative increments of 30% up to 233 K, in this way we increase the coverage at lower temperatures.

In order to produce synthetic spectra, MADCUBA requires other parameters on top of $log(N)$ and T_{ex} considered in our pilot study. Thus these parameters were fixed as follows: output intensity units were set to Kelvin; line width and velocity were fixed to 150 and 250 km s^{-1} respectively; an emitting source size of $10''$ was assumed. These fixed parameters were selected to match those required to fit the actual astronomical spectra from the ALMA Comprehensive High-Resolution Extragalactic Molecular Inventory (ALCHEMI, Martin et. al, in preparation), as discussed below. Since our training datasets are created in rest frequency units, it is therefore agnostic to the velocity parameter. However, our predictor cannot be directly applied to astronomical data with different parameters of line width and source size. An extension of our neural network should be required to make our tool fully usable. All training and validation spectra used in this work were then created by combining these individual molecular spectra and adding Gaussian noise (from the Numpy.random.normal implementation) with a root mean square (rms) between 10 and 50 mK, which we consider to be a reasonable range of values for noise in an astronomical spectrum at these frequencies. For each molecule, excitation temperature and column density are defined as the output classes. The example generation process begins by picking a random data file from MADCUBA, and adding noise within the specified ranges. Then the generated spectrum is saved to disk.

The simplified overall workflow of the training example generation script can be seen in Figure 3.4. The detailed flow can be seen in Figure B.1. The process begins by determining the number of training and testing examples to generate, the script reads this value from its configuration section. We have determined that 32000 training examples is a suitable amount, the number of testing examples is an additional 20%, that is, 6400 testing examples. Details of the selection of a suitable number of training examples can be seen in section 4.1. Then, for each molecule, we select a noise level and pick a number of random examples from the MADCUBA files, associated to the current molecule. Next, we generate the labels of the examples, by extracting them from the headers of the selected MADCUBA files.

An example of a MADCUBA file is shown in Listing 3.1:

```
1 //molecules='CO|1' logn=17.5 tex=37.1293 velo=250.0 fwhm=150.0 sourcesize=10.0
2
3 2.3048184635971344E11  1.020e+01
4 2.304920473597134E11   1.143e+01
5 2.3050224835971344E11  1.228e+01
6 2.305124493597134E11   1.267e+01
7 2.3052265035971344E11  1.253e+01
8 2.305328513597134E11   1.190e+01
```

```
2.3054305235971344E11  1.084e+01
...
3.4573274435971344E11  3.742e+01
3.457429453597134E11   3.900e+01
3.4575314635971344E11  3.992e+01
3.4576334735971344E11  4.014e+01
3.457735483597135E11   3.965e+01
3.4578374935971344E11  3.847e+01
3.4579395035971344E11  3.665e+01
3.4580415135971344E11  3.430e+01
```

Listing 3.1: Example of the contents of a MADCUBA file

In the above example, we can extract the labels for CO_N=17.5 and CO_T=37.1293. Continuing with the workflow, the necessary directory structures are created to hold the newly created examples. Headers are created for the training and testing labels files. Finally, a parallel function called *generate_spectrum* is called, which receives a complete array of MADCUBA example file names. As each training example generation is independent, the process is executed, using the Python pool library.

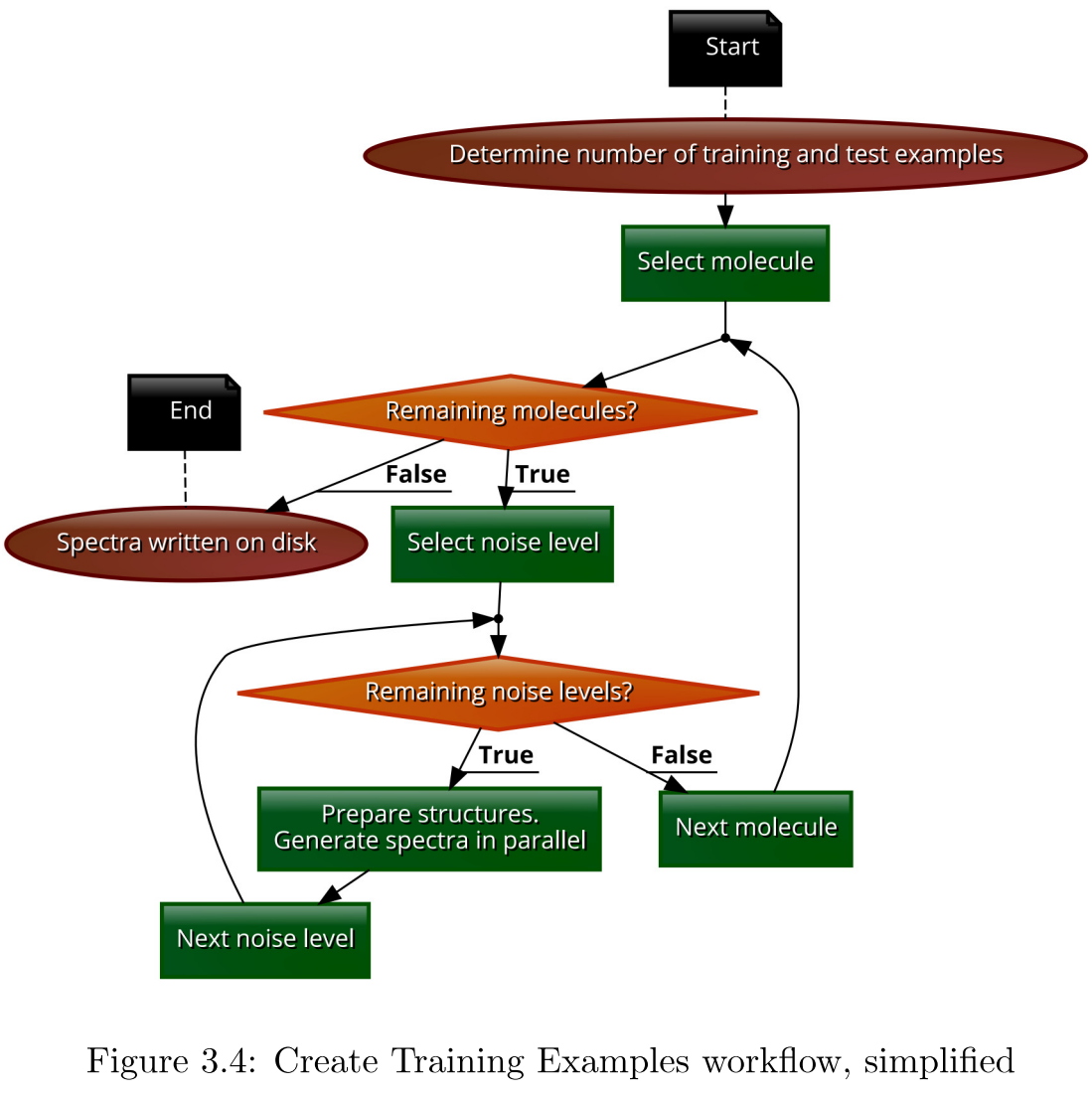


Figure 3.4: Create Training Examples workflow, simplified

The *generate_spectrum* function, is described in a simplified diagram in Figure 3.5. It can be reviewed with further detail in Figure B.2.

In this workflow, the current molecule, and the number of examples to generate is acquired. Then, the MADCUBA filename passed as a parameter, is used to open the file from disk, if the spectrum loaded contains data (it's length is larger than zero), we move forward. The next step is loading the transitions list for the molecule, we used Splatalogue to create a text file with the transition frequencies for the current molecule, as mentioned earlier, we only consider transitions

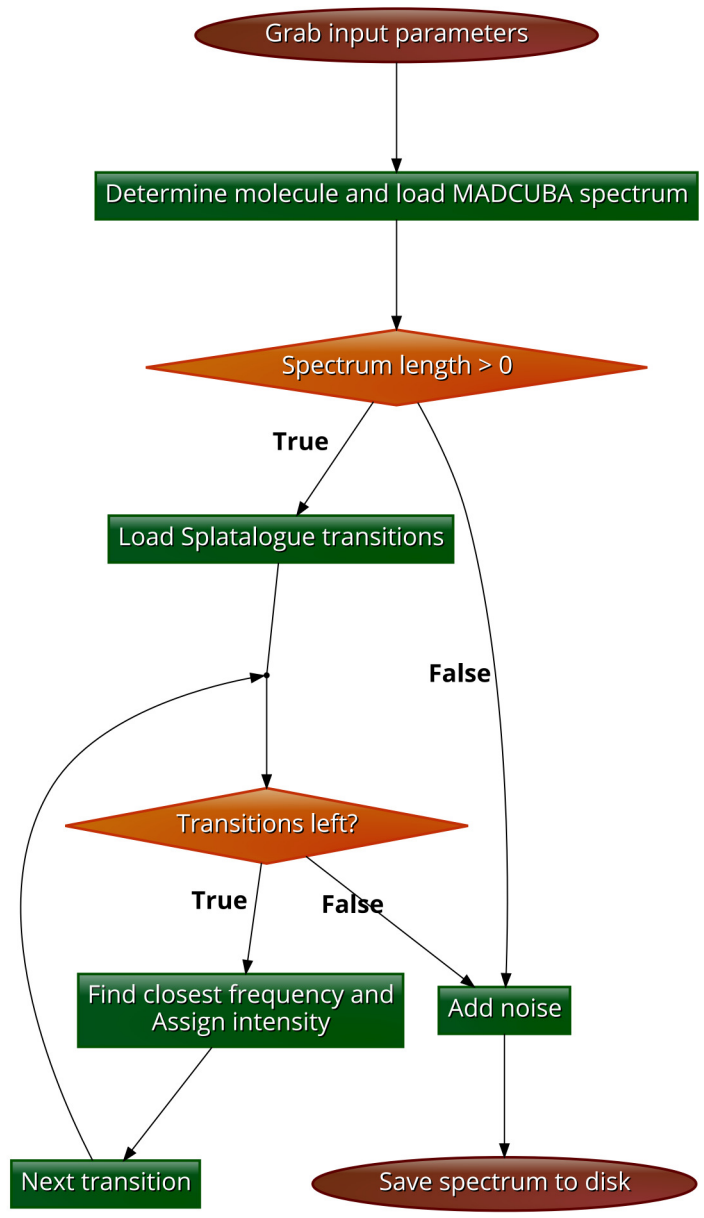


Figure 3.5: Generate individual training spectrum workflow, simplified

between 80 and 400 GHz. For each Splatalogue transition, we call a function to find the closest frequency in our current MADCUBA example file, to the current Splatalogue transition. With the returned frequency, we measure the difference to the current Splatalogue transition frequency, if the frequency is larger than 10 MHz, we consider that this transition is not present in the example file, so we will append a row with the Splatalogue transition frequency, accompanied by an intensity of zero. Otherwise, if the difference is less than 10 MHz, we consider that the transition exists within the MADCUBA example file, and we append a row with the Splatalogue transition frequency, accompanied by an intensity equal to the intensity of the MADCUBA file, in the closest frequency previously selected. When we have no more transitions to process, we add noise to the spectrum being built, this noise is in accordance to the current noise level of the parent function. We finish by saving the current built spectrum to disk, using the .dat file extension.

The example generation and neural network training is done once. If more molecules are added, the molecules parameter can be changed so only the examples for the new molecule are generated. A training example file contains as many features as transitions the molecule contains within the frequency coverage range. The intensities are taken from the closest frequencies from the MADCUBA data file. The transition frequencies are obtained from Splatalogue, from the JPL catalog, and are rest frequencies. We assume any input data to *MolPred* has been Doppler shifted such that the emission peak corresponds to the central frequency of the transition. Constructing example files in this way greatly reduces training times and memory footprints. In order to improve training times even further, we decided to pre build training and testing datasets, from the examples saved on disk, and pass the dataset as a pandas dataframe into the neural network model object. Training times are much faster when the complete dataset can be simplified and stored in RAM.

The workflow of the script that builds the complete dataset is presented in a simplified manner in Figure 3.6, further details can be seen in Figure B.3.

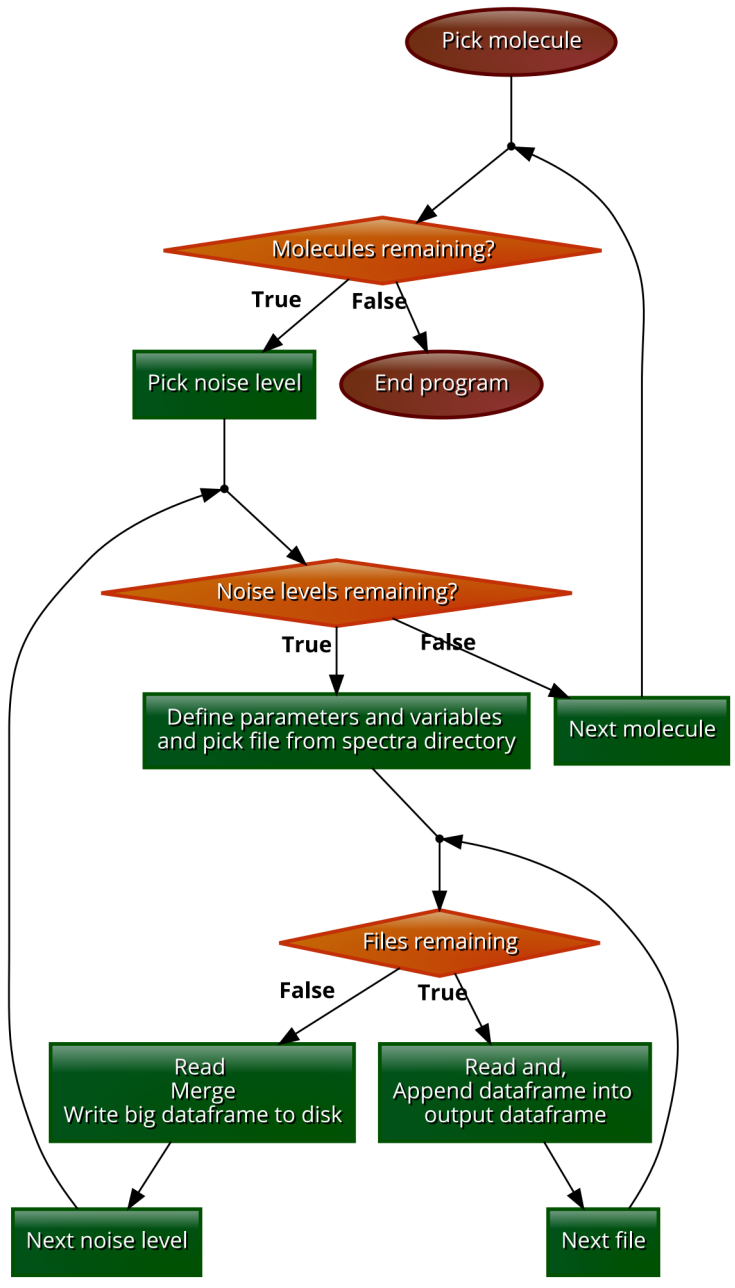


Figure 3.6: Build dataset workflow, simplified

The *build_dataset* flow starts by having the script pick a molecule, for each molecule, a noise level is chosen, then, a stored spectra directory is read, and both labels file and an output dataframe are defined. We read files from the spectra directory and append one row per spectrum to the dataframe. When all the training examples in the spectra directory have been read, the labels are read into another dataframe. Finally, both output and labels dataframe are merged and written as one large file into disk. Then, we move into the next noise level, and eventually, the next molecule.

3.2.4 Neural Network Training

We have used the *Keras* package from the *Tensorflow* library (Chollet et al., 2015) for the creation and training of the neural networks. Networks are created by adding Densely connected layers sequentially as seen in Listing 3.2:

```
19      def bm():
20          network_layers = []
21          i = 0
22          for layer in layers:
23              if len(network_layers) == 0:
24                  network_layers.append(Dense(layer, activation=activation, input_shape=[len(
    dataset_train.keys())], name="Layer_" + str(i)))
```

```
            else:
                network_layers.append(Dense(layer, activation=activation, name="Layer_" + str(i)))
            i = i + 1
        network_layers.append(Dense(n_molecules, name="Output_Layer"))
        model = keras.Sequential(network_layers, name=model_label)
        optimizer = 'adam'
        model.compile(loss='mean_absolute_error', optimizer=optimizer,  metrics=['mean_absolute_error'
    ])
        return model
```

Listing 3.2: Code block for the building of a neural network

Keras model files are saved after training, for later use in the prediction stages. Neural networks have a large number of hyperparameters. The hyperparameters that were varied are given in Table 3.3 along with the range of values trialled. For each possible combination of parameters, a model was created. Neural networks were trained for up to 1000 epochs, where an epoch refers to one forward pass and one backward pass of all the training examples. However, to keep training time low, we implemented an early stopping mechanism where the training would stop if the validation loss did not improve over 10 epochs.

The workflow of this stage is described in a simplified manner in Figure 3.7. Further details can be seen in Figure B.4.

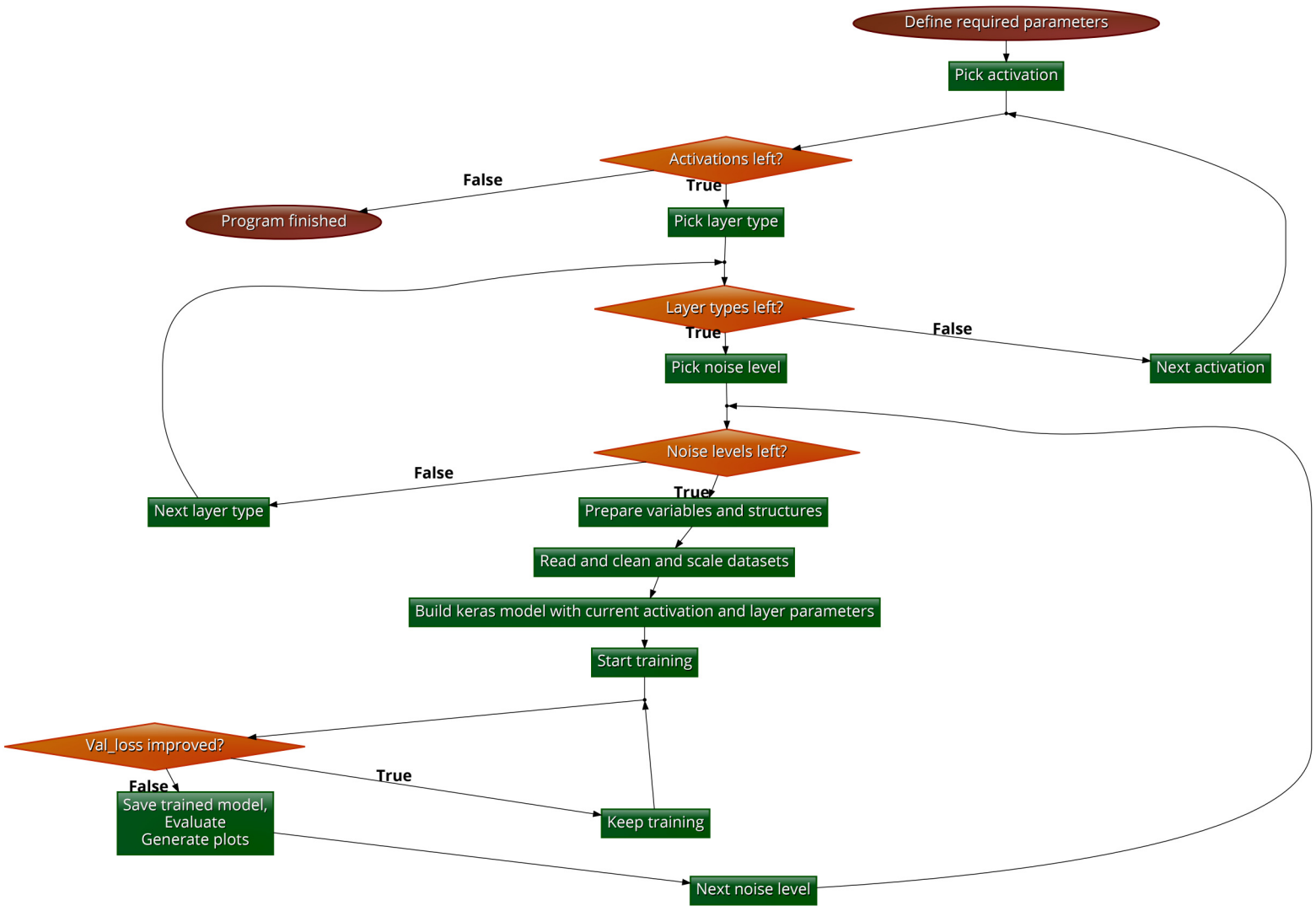


Figure 3.7: Neural Network Training workflow, simplified

The flow starts by having the training script read the number of training and testing examples, the molecule list, a dictionary that contains the types of layers and a list of activation functions. For each activation, layer type and noise level, we initialize the working variables, generate the required directories and read the large files created by the *build_dataset*

script into a dataframe which is later split into two dataframes, one with the training examples, and another with the training labels. A similar procedure is done to load the testing examples. The datasets are checked for unknown values, which rows are dropped. Also any duplicated rows are removed, if required. Some statistics are obtained from the training dataset, just for debugging purposes. MinMax scaler objects are generated for the datasets, which are then used to scale them to values within 0 to 1. Implementation details can be seen in Listing 3.3.

```
dataset_train_scaler = MinMaxScaler().fit(dataset_train)
train_labels_scaler = MinMaxScaler().fit(train_labels)
dataset_test_scaler = MinMaxScaler().fit(dataset_test)
test_labels_scaler = MinMaxScaler().fit(test_labels)

normed_train_data = dataset_train_scaler.transform(dataset_train)
normed_test_data = dataset_test_scaler.transform(dataset_test)
normed_train_labels = train_labels_scaler.transform(train_labels)
normed_test_labels = test_labels_scaler.transform(test_labels)
```

Listing 3.3: Code block for the scaling of the training and testing data and labels

In order to be used by the GPU, dataset objects, that contain both data and labels, are created as shown in Listings 3.4.

```
train_data = tf.data.Dataset.from_tensor_slices((normed_train_data, normed_train_labels))
val_data = tf.data.Dataset.from_tensor_slices((normed_test_data, normed_test_labels))
```

Listing 3.4: Code block for the creation of datasets from tensor slices to be used by the GPU in training.

With the data ready for training, we build a *Keras* model with the current activation function and layer parameters as shown previously. Then, the training starts, the network will continue training until a stop criteria is reached, as it was previously explained, details of the training speed can be seen later, in subsection 3.2.5. The call for the keras *fit* function is shown in Listing 3.5. It can be seen that the function takes use of the training and validation datasets, training by batches defined in the parameter *batch_size*. The callback for the early stopping function can be seen, as well as other callbacks, to save intermediate progress, while the network is training. Finally a callback to Tensorboard can be seen. This is to monitor the performance of the training, and evaluate the impact of tuning options. The resulting history object will contain the evolution of the loss function across each epoch.

```
history = model.fit(train_data, use_multiprocessing=False,
                        epochs=MaxEpochs, batch_size=batch_size,
                        validation_data=val_data,
                        verbose=1,
                    callbacks=[EarlyStopping(monitor='val_loss', patience=10, verbose=2),
                               ModelCheckpoint(model_save_name, monitor='val_loss',
                                               save_best_only=True,
                                               verbose=0), tboard_callback] )

```

Listing 3.5: Parameters used in model.fit call

Once the training has stopped, the trained model is saved to disk, and a history dataframe is created so it can be used to evaluate the model, by doing predictions with the scaled test dataset, and comparing those predictions with the scaled labels. An additional crossvalidation step is performed to obtain a final performance score, which is then plotted into several functions used for our analysis. Before moving to the next noise level and activation function and molecule, a garbage collector function is called, to reduce the likelihood of encountering an out of memory error.

The trained models were created using combinations the following potential parameters shown in Table 3.3.

Table 3.3: A list of neural network hyperparameters which were varied and the values that were tested.

Parameter type	**Parameter values**
Activation functions	Sigmoid, ReLU, Linear, Tanh, Swish.
Layers	Single, double, triple
Neurons	256, 1024
Molecules	CO, HCO^+, SiO, CH_3CN
Training examples	500, 1000, 2000, 4000, 8000, 16000, 32000
Testing examples	Additional 20% of training examples
Noise levels	0.01 to 0.05 K
Model type	Feed-Forward (implemented as keras.Sequential)
Number of inputs	Same as number of transitions in range
Output classes	2 ($log(N)$ and T_{ex}, for each molecule)
Optimizer	Adam
Loss function	MAE
Training autostop	Enabled
Training patience	10 epochs
Maximum amount of training	1000 epochs

We trained each neural network individually using spectra that contained only noise and emission from transitions of the molecule for which the network would predict. These neural networks could later be loaded together as part of the *MolPred* code to predict from full spectra. In this initial experiment we decided to use a simple feed-forward neural network with a maximum of 3 densely connected layers of up to 1024 neurons each. We wish to establish an initial set, as more molecules and output classes can be added later. The networks were trained to minimize the mean absolute error (MAE) between the scaled $log(N)$ and T_{ex} of an input spectrum and the predictions using the Adam optimizer (Kingma and Ba, 2017). We use the scaled predictions so that errors in N and T_{ex} are equally weighted as the unscaled variables differ by many orders of magnitude. To test these trained neural networks, the MAE on the scaled predictions across the entire validation set was calculated for each network. These MAE values were then compared between networks to select the best regressor to include in *MolPred*.

3.2.5 Discussion on Performance

With the proposed methods, the next step was to observe the training times differences based on the selection of CPU/GPU(s). We began by comparing training times for a single molecule (CH_3CN) and model (*relu_triple_1024*) with 32000 training examples and 10 mK of noise with a batch size of 32, and configuring *Tensorflow* for using either CPU, one GPU or two non-linked GPUs. We tried different data distribution strategies[2], to observe the impact on the MAE, the results, ordered by minimum MAE, can be seen in Table 3.4. For reference, these training times were obtained using a PC with an Intel I7-8700K CPU, 32 GB of RAM, two non linked Zotac GeForce GTX 1060 6GB video cards and a Kingston A2000 1 TB Solid State Drive - M.2 2280.

[2] `https://www.tensorflow.org/api_docs/python/tf/distribute/Strategy`

Table 3.4: Data distribution comparison, sorted by MAE

Strategy	Training Time	Epochs	Min MAE
Central storage 2 GPU	29 min 20 sec	100	0.01803
Collective all-reduce 1 GPU	10 min 8 sec	84	0.02096
Mirrored 1 GPU	9 min 43 sec	75	0.02203
Mirrored 2 GPU	21 min 57 sec	66	0.02238
Collective all-reduce 2GPU	20 min 49 sec	53	0.02384
Mirrored CPU	11 min 55 sec	54	0.02386
Collective all-reduce CPU	9 min 41 sec	44	0.02483
Central storage CPU	9 min 28 sec	43	0.0250
Central storage 1 GPU	5 min 3 sec	46	0.02568

We can observe that in terms of MAE, the error ranges between 1.8% and 2.5% and the training time ranges between 5 and 30 minutes, depending on the data distribution strategy selected. In order to compare the speed of the strategies, we should compare them based on the time they took to reach 43 epochs, which was the minimum number of epochs required to finish the training. The comparison can be observed in Table 3.5.

Table 3.5: Strategy performance comparison, sorted by training time

Strategy	Training Time	Epochs	Min MAE
Central storage 1 GPU	4 min 45 sec	43	0.02579
Collective all-reduce 1 GPU	5 min 8 sec	43	0.02553
Mirrored 1 GPU	5 min 35 sec	43	0.02536
Collective all-reduce CPU	9 min 28 sec	43	0.02462
Central storage CPU	9 min 28 sec	43	0.02500
Mirrored CPU	9 min 28 sec	43	0.02536
Central storage 2 GPU	12 min 46 sec	43	0.02601
Mirrored 2 GPU	14 min 15 sec	43	0.02434
Collective all-reduce 2GPU	16 min 55 sec	43	0.02564

It can be observed that the tests where just one GPU is used, performed faster. It is interesting to notice that the instances where the CPU was used for training, were faster than the instances where 2 GPUs were used, this seems reasonable, as both GPUs in the test setup are not linked, and in order to perform the training work, the data needs to be copied to both cards memory, consuming additional time. With this, we believe that our training should be done using just one GPU.

We are able to influence in the training speed, by adjusting the batch size. Table 3.6 shows the relation between batch size and minimum MAE. We selected the *central storage* distribution, with one GPU, and vary the batch size, from 32 examples per batch, to 2048 examples per batch, the noise remains at 10 mK.

To observe the specific relation between batch size, and MAE, we will stop at 30 epochs, the minimum required to complete the training and review the difference in Table 3.7.

Table 3.6: Batch size performance comparison, sorted by batch size

Batch Size	Training Time	Epochs	Min MAE
32	5 min 3 sec	46	0.02568
64	2 min 9 sec	38	0.02623
128	2 min 32 sec	75	0.2269
256	44 sec	30	0.03387
512	41 sec	37	0.03658
1024	1 min 11 sec	73	0.03532
2048	1 min 23 sec	99	0.03750

Table 3.7: Batch size vs training time vs minimum MAE comparison, sorted by batch size

Batch Size	Training Time	Epochs	Min MAE
32	3 min 18 sec	30	0.02836
64	1 min 42 sec	30	0.02746
128	1 min 2 sec	30	0.02971
256	44 sec	30	0.03387
512	34 sec	30	0.03995
1024	30 sec	30	0.04734
2048	26 sec	30	0.05072

We can observe the trade off between training time versus minimum MAE as given by the batch time. The larger the batch size, the faster the training finishes, at the expense of larger errors. In this test, by increasing the batch size to 2048 examples per batch, the model error increases about 3%, but this represents over 50% of the original error figure, however the training time has been reduced in almost 87%. Having lower training times, opens the possibility of using a larger number of training examples. In order to obtain the best possible predictions, we decided to aim for a lower MAE, rather than a fast training time, thus we will use a batch size of 32 examples per batch for the rest of the following experiments.

In section 4.15 we will calculate *MolPred*'s prediction speed acceleration or *speedup*, by using Equation 3.1

$$S = \frac{\text{Serial execution time}}{\text{Parallel execution time}} \tag{3.1}$$

Chapter 4

Experiments

As explained in sections above, we executed a grid search with the parameters described on Table 3.3, with a total of 600 combinations, the performance measurement selected was the mean average error.

4.1 Results on Test Data

Our first task was to determine an appropriate number of training examples for the neural networks. We present the evolution of the MAE as a function of the number of training examples in Figure 4.1. We found that the MAE initially improves by doubling the number of training examples but beyond 16000 examples, increasing the number of training examples gives a marginal improvement. We also included in Table 4.1 the training times, for evaluation, which we found to be roughly proportional to the number of examples. To balance low MAE values with reasonable training time, we reached a maximum of 32000 examples, which we defined as our baseline for all models shown in this section.

Table 4.1: Best mean absolute error and training time in hours as a function of number of training examples

Number of training examples	500	1000	2000	4000	8000	16000	32000
CO	0.14247	0.11857	0.11161	0.09755	0.09472	0.09576	0.08508
HCO^+	0.08056	0.06598	0.06801	0.05981	0.0561	0.05039	0.0417
SiO	0.0848	0.0578	0.05209	0.05229	0.0439	0.04295	0.03589
CH_3CN	0.08508	0.05788	0.04170	0.0352	0.03589	0.02133	0.01581
Training time (Hours)	0.86	1.38	2.38	3.83	10.83	20.08	25.76

Once we have established the number of training examples, we wished to observe the behaviour of the MAE in the training stage of each network. Figure 4.2 presents the evolution of the MAE for the neural network that performed best on each molecule (this is why HCO^+ shows a noise of 0.02 K). Each plot contains the molecule name, the best neural network model, the minimum MAE and its respective noise level. We allowed the networks to be trained for a maximum of 1000 epochs but stopped training early if the model did not improve for 10 epochs (training patience). In most models, the training stops close to 100 epochs, the lowest validation loss can be seen in the plots approximately ten epochs before the end of the plot.

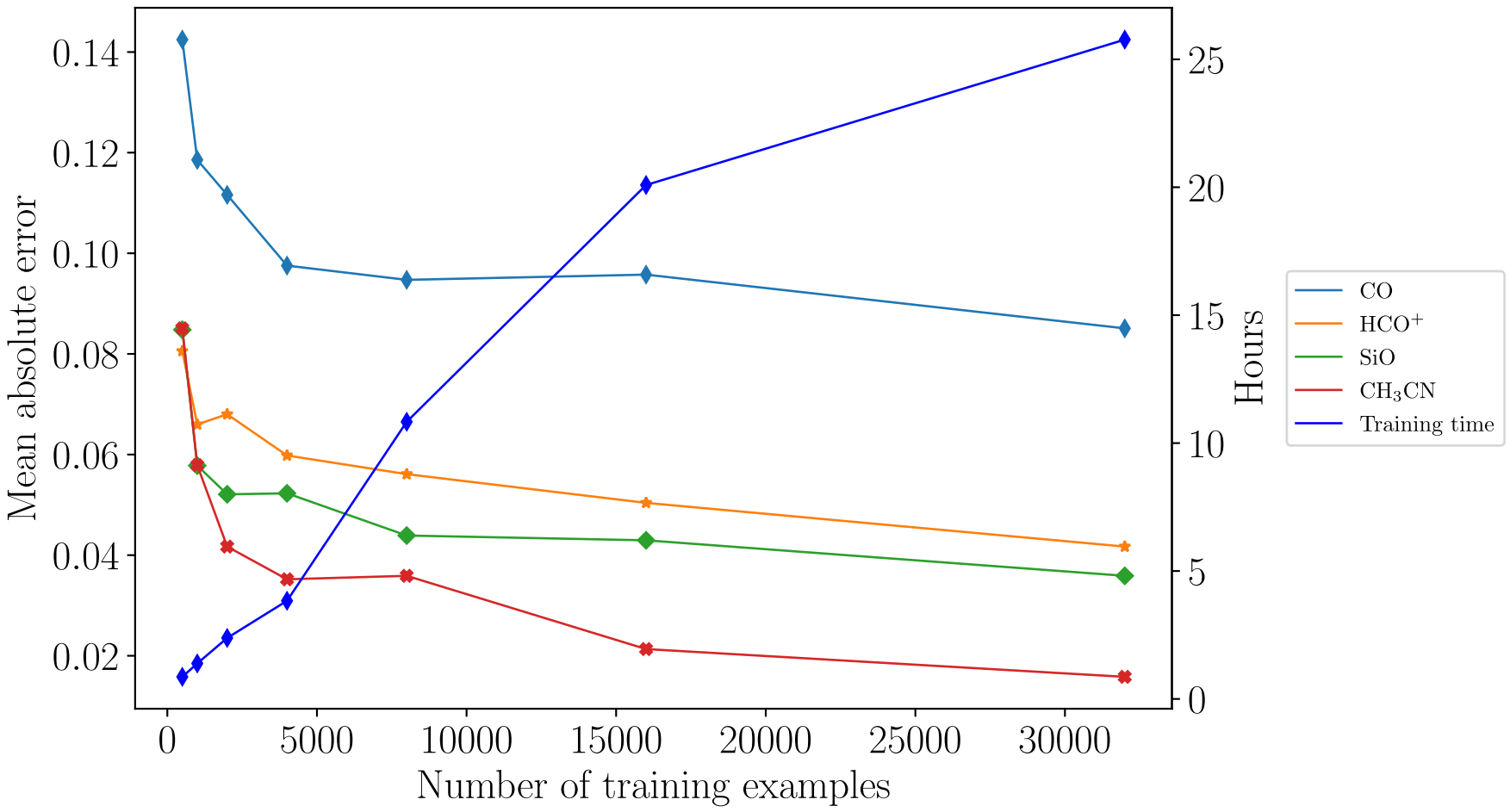


Figure 4.1: Mean absolute error and training time in hours as a function of number of training examples.

To obtain a deeper understanding of the behavior of the neural networks, we analyzed how the noise would influence the performance. In Figure 4.3 we can observe that the general trend is that the MAE increases with the noise. We assume the best network is the one with the lowest MAE at any noise level for that particular molecule, and use that network in *MolPred*. The MAE, noise level, number of epochs of the best neural network trained for each molecule, the name of the model which indicates the activation function, the number of layers (3), and number of neurons per layer can be observed in Table 4.3. However, since Table 4.2 shows the MAE is only weakly affected by the noise value, one could choose to train their networks with a large noise value to ensure the model is robust to the noise in real spectra.

Table 4.2: Noise vs mean absolute error.

Noise (K)	0.01	0.02	0.03	0.04	0.05
CO	0.08508	0.09054	0.08961	0.09206	0.09653
HCO^+	0.04491	0.0417	0.05042	0.05088	0.04927
SiO	0.03589	0.04027	0.04367	0.04328	0.04359
CH_3CN	0.01581	0.01786	0.0193	0.0212	0.02151

To verify the quality of the predictions on each neural network, we predicted the column density and excitation temperature for a test set of 6400 testing examples not used during training. We then checked the error distribution of the predictions, which is shown in Figures 4.4 and 4.5. The error distribution is centrally peaked which is an important result that our choice of loss function does not guarantee. The strongly peaked distributions mean that the networks typically give small errors and are unlikely to predict an extremely incorrect value.

Having obtained the best models for each molecule, we wanted to review the prediction behaviour of the networks, which

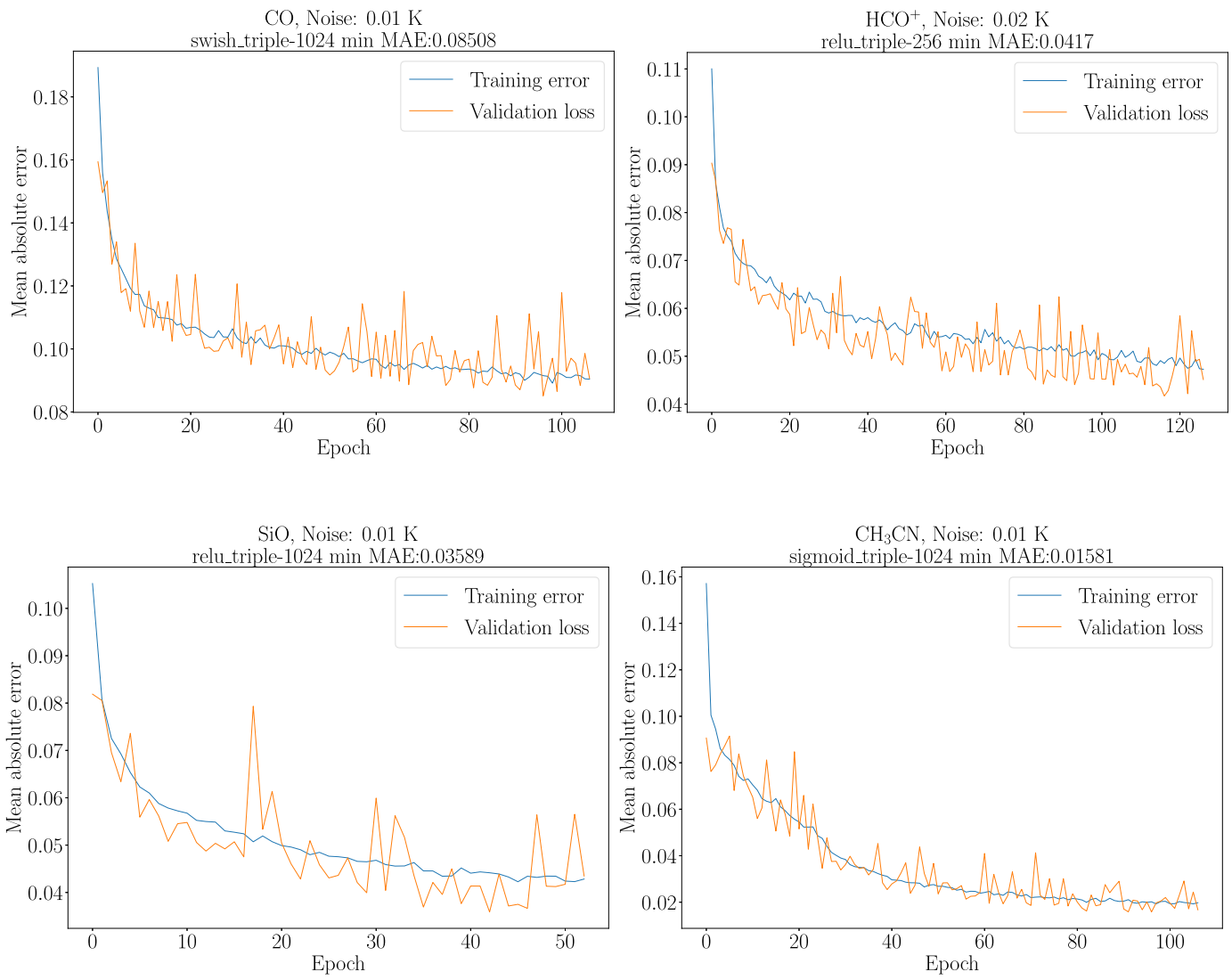


Figure 4.2: Training mean absolute error over Molecule set.

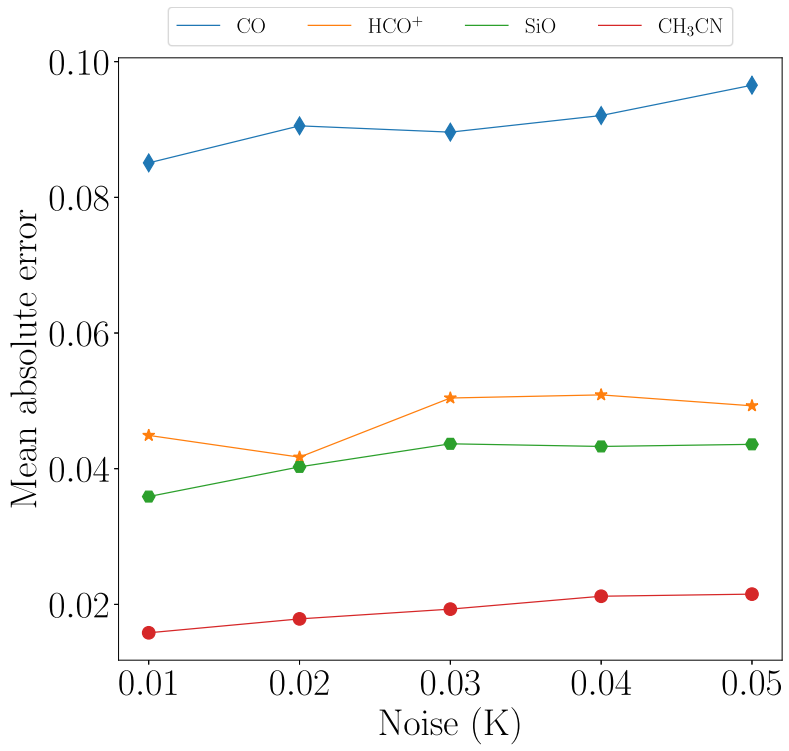


Figure 4.3: Noise vs mean absolute error.

was plotted in Figure 4.6. We see that for all the molecular species in our pilot study, the predictions follow a tight linear correlation with predicted values close to the true values, as also seen in the centrally peaked histograms in Figures 4.4 and 4.5. However, the prediction accuracy drops for both low and high values of $log(N)$. The $log(N)$ value at which this

Table 4.3: Molecule name, noise level, minimum MAE, number of epochs and best model name

Molecule	Noise (K)	MinMAE	Epochs	Model
CO	0.01	0.08508	96	swish_triple-1024
HCO^+	0.02	0.04170	116	relu_triple-256
SiO	0.01	0.03589	42	relu_triple-1024
CH_3CN	0.01	0.01581	96	sigmoid_triple-1024

deviation occurs is dependent on the molecule and has a physical explanation. For low values of $log(N)$ the line intensities become close to or below the noise level, which causes the neural network to predict values that are not dependent on the true value. Moreover, this also has a low dependency on the temperature as seen in the color coding in Figure 4.6, as it is purely due to the lack of signal in the input spectra. Essentially, the network cannot distinguish between input spectra below a $log(N)$ threshold. The colour scale indicates the excitation temperature of the spectrum. Each neural network was trained on spectra with a noise of 10mK for all molecules except for HCO^+ where the best network was trained using a noise of 20 mK.

On the other hand, for high values of $log(N)$, the spectral features from the input data are saturated. That is, all

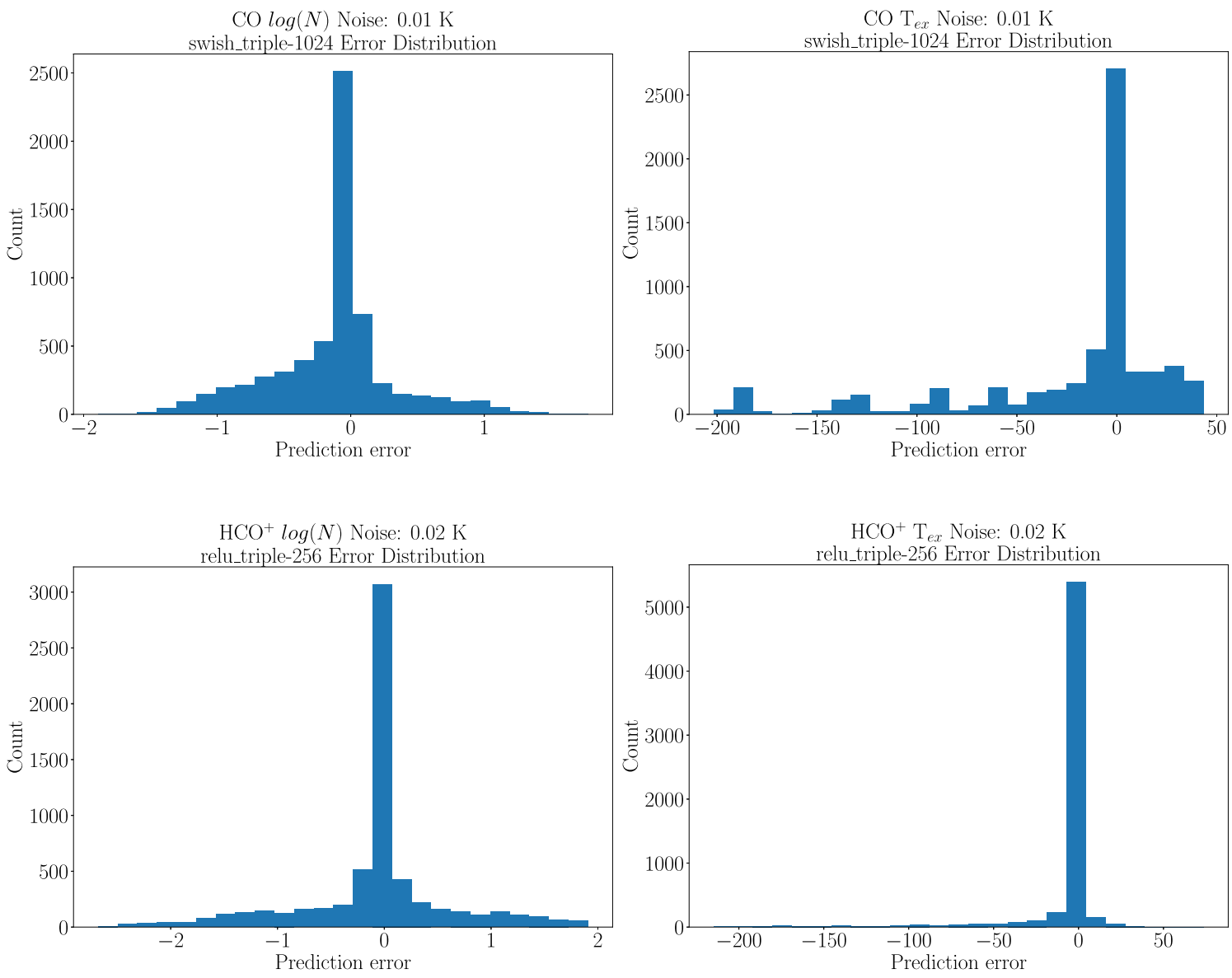


Figure 4.4: Distribution of errors for $log(N)$ (left column) and T_{ex} (right column) from predictions of CO and HCO^+.

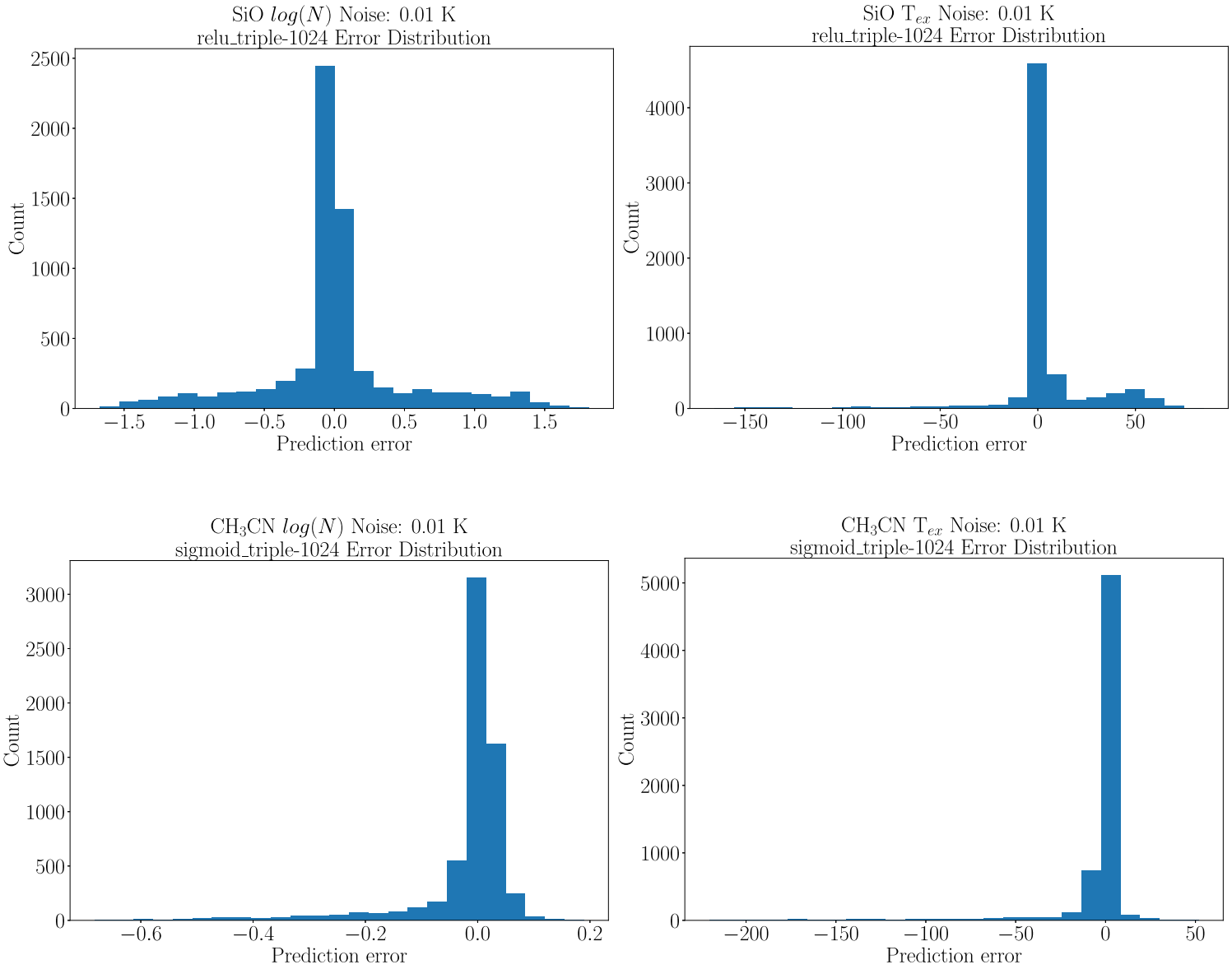


Figure 4.5: Distribution of errors for $log(N)$ (left column) and T_{ex} (right column) from predictions of SiO and CH_3CN.

transitions reach a saturation flux density once a value of $log(N)$ is surpassed (see Martín et al., 2019, for a description of line saturation). The $log(N)$ value at which saturation occurs has a strong dependency on the value of the temperature as seen in the color coding in Figure 4.6. The behaviour also depends on the number of transitions available for each molecule (Table 3.1). Thus, in the case of CH_3CN, even for high values of $log(N)$, there is still a significant number of low flux density unsaturated transitions and therefore we do not observe the effect of saturation in Figure 4.6.

The same physical explanation applies to the predictions of the temperature parameter. Figure 4.7 presents the test predictions of T_{ex} against their true values. In the same manner, the column density of the test spectra is given in the colour scale. The training noise was 10 mK for all molecules except HCO^+ which was 20mK. For all molecules, we can observe that for all values of T_{ex}, predictions appear dependent on the value of N. As can be seen in the figure, the predicted temperatures are closer to their real values for higher N. This has a similar explanation to the neural networks' poor performance on the column density prediction for low column densities. At low N, the intensities of the transition decrease and come closer to the noise. The noise then dominates the prediction, making the recognition of the transition impossible if the transition is below the noise level. We can compare the CO results with the results of other molecules, particularly with CH_3CN which has the largest number of transitions of the molecules in our set. These results suggest that the accuracy in the prediction of the temperature could be dependent on the number of transitions available to construct a better characterized model of the molecule, although his would require further testing.

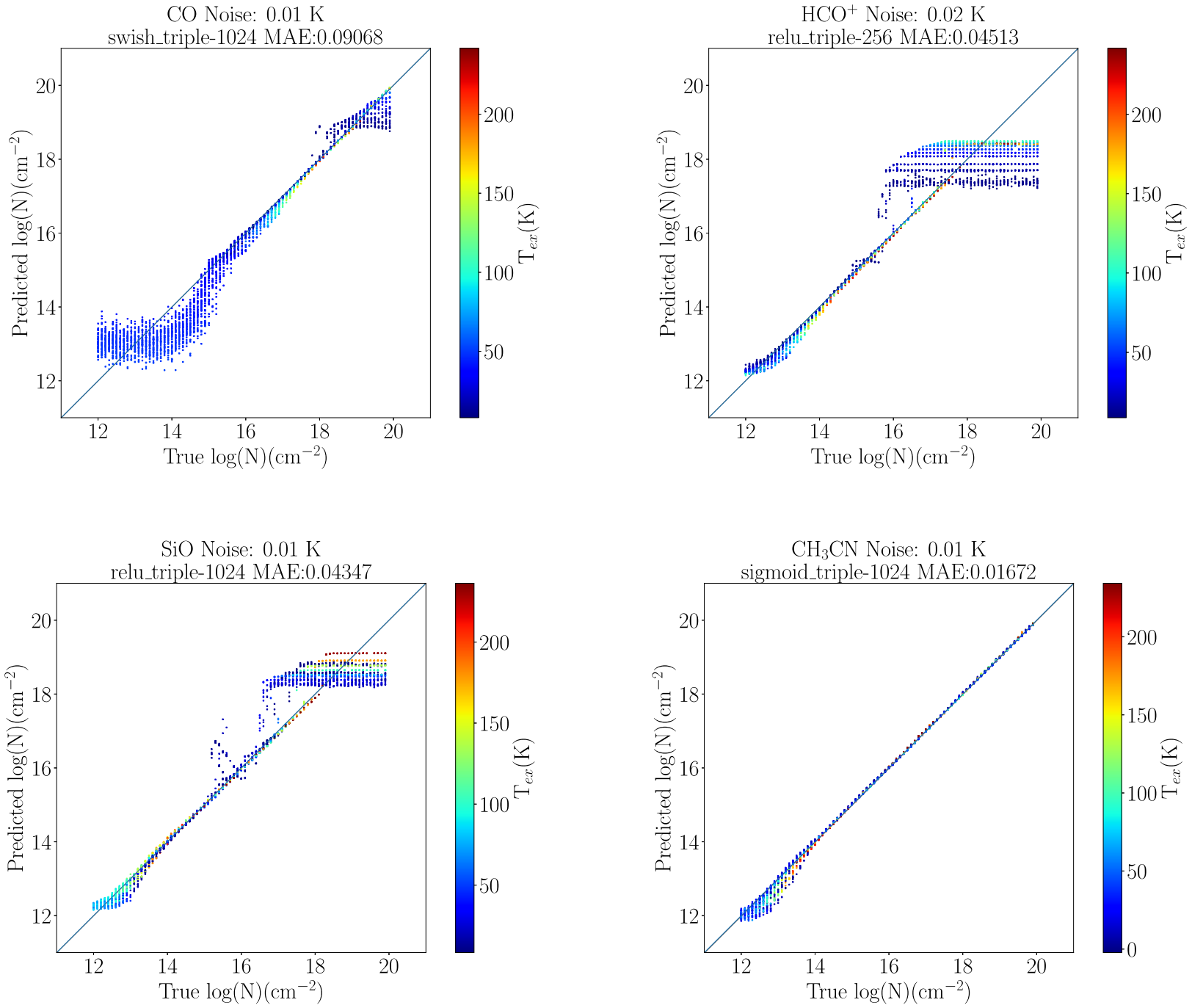


Figure 4.6: Scatter plots showing the predicted $log(N)$ against its true value for all spectra in the test data.

If we increase the noise to 0.05 (50mK) we can observe that the predictions of temperature spread over a wider range, rather than forming small groups, as in the case with lower noise. Still, the predictions remain accurate for the higher column density values (yellow/red values in Figure 4.8), where the spectral features are clearly identified above the noise level.

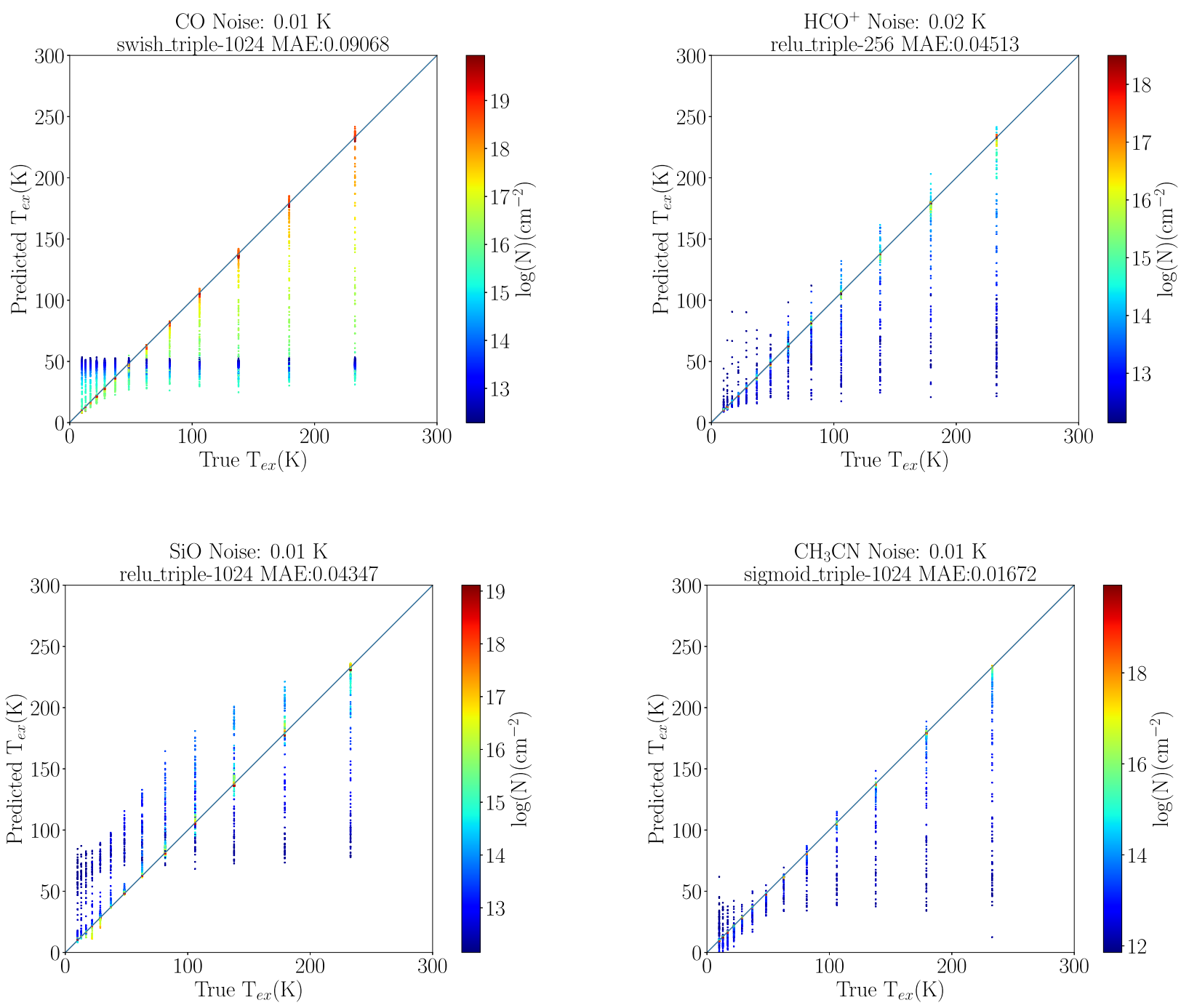


Figure 4.7: Predicted T_{ex} from the best neural networks against the true value for all spectra in the test set.

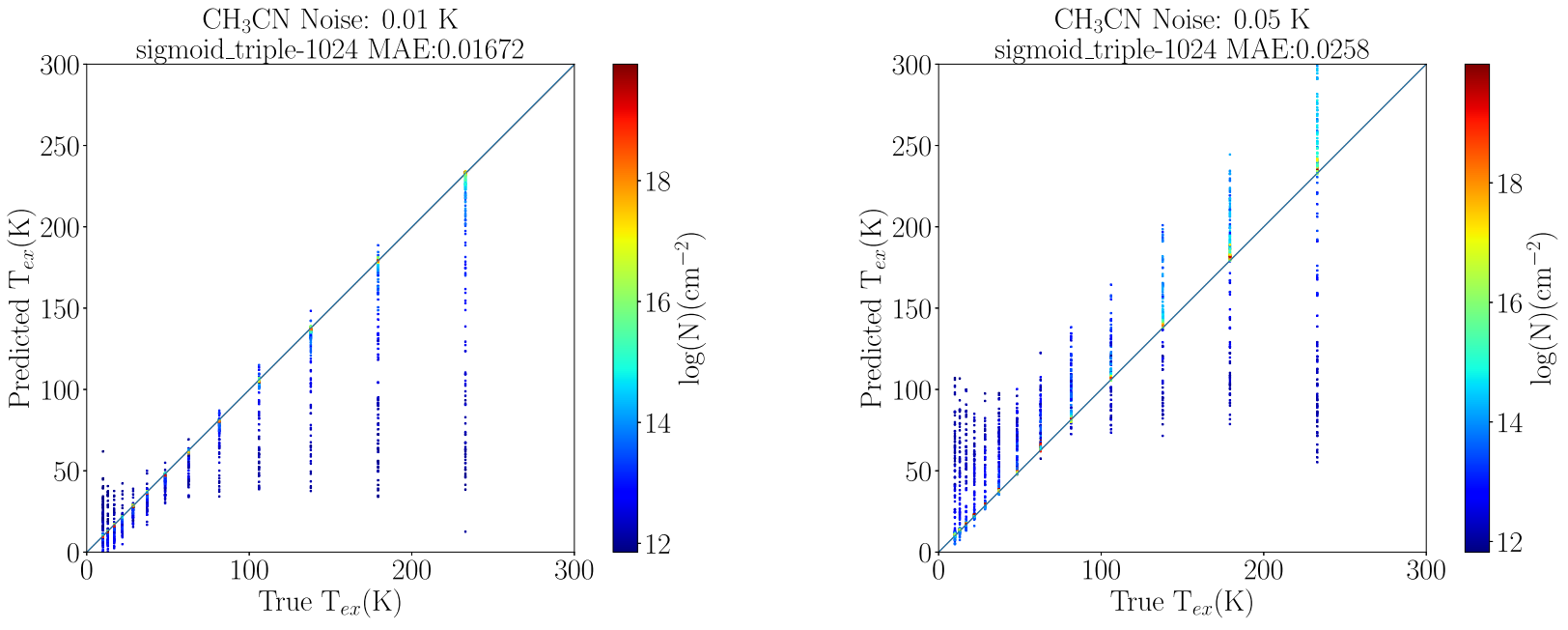


Figure 4.8: T_{ex} prediction comparison for CH_3CN with different noise levels, 10mK (left), 50mK (right).

4.2 Results on Astronomical Data

To further test *MolPred* we used data from ALCHEMI, which was one of the ALMA Large Programmes in Cycle 5. The astronomical target was the central molecular zone of the starburst galaxy NGC253. The project consists of a full spectral line survey continuously covering ALMA Bands 3 to 7. We used the low resolution spectrum from the ALMA Compact Array (Morita Array) which covers the frequency range of 125 GHz to 373.2 GHZ. It therefore contains an almost complete set of transitions from the sample molecules used in this study. The continuous coverage and uniform sensitivity makes this dataset an ideal test sample for our study. This wide band spectrum was shift to rest frequency assuming a Doppler shift of 250 km s^{-1} and used as an input to *MolPred*, which then generated $log(N)$ and T_{ex} predictions for each molecule following the flow of Figure 3.1.

The *MolPred* predictions are displayed in Table 4.4. For comparison, we also include the fitted values using the MADCUBA AUTOFIT package used for the actual spectroscopic analysis of the ALCHEMI data. The *MolPred* predictions for $log(N)$ are within 1.5% of the ones from MADCUBA, and for T_{ex} are within 25% from the ones by MADCUBA. Thus the predictions agree very well except in the case of the excitation temperature of CH_3CN. However, we note that MADCUBA fitting algorithm did not converge unless taking into account the contribution from other molecular species blended to the CH_3CN. Despite the fact *MolPred* did not include information on other molecular "contaminants", the predictions were still reasonably close to the value predicted by MADCUBA. We now examine these results by plotting our predictions together with the closest examples to the MADCUBA predictions from our training files.

Table 4.4: The $log(N)$ and T_{ex} values predicted by *MolPred* compared with the MADCUBA fit.

Molecule	*MolPred* $log(N)$ (cm^{-2})	*MolPred* $T_{ex}(K)$	MADCUBA $log(N)$ (cm^{-2})	MADCUBA $T_{ex}(K)$
CO	18.51	17.45	18.44	19.48
HCO^+	13.98	16.55	14.08	15.30
SiO	13.17	17.65	13.18	13.98
CH_3CN	13.39	58.95	13.27	38.95

We start the analysis with CO, the simplest molecule of our set. The *MolPred* prediction gives a very similar value of $log(N)$ to the MADCUBA fit but has a slightly lower excitation temperature. The result of this difference can be seen in Figure 4.9 where we plot the intensities of MADCUBA spectra generated using the *MolPred* and MADCUBA predictions alongside the ALCHEMI data. The plot shows CO line profiles from the ALCHEMI Data plotted in blue. Overplotted are the spectra generated from the *MolPred* predictions (orange) and MADCUBA predictions (green). For reference, we included a red vertical line to indicate the frequency of the closest transition according to the JPL Catalog. The ALCHEMI data set used in this test, only has data for 2 out of the 3 transitions in our training data. We can see the detail of the CO(2-1) in the left figure and CO(3-2) transitions in the right figure. Whilst both models underfit the data, the *MolPred* spectrum has a somewhat lower intensity than the others due to the low excitation temperature. Based on our analysis of the test data, we can expect the saturation effect seen in Figure 4.7 has affected the accuracy of predictions at these high column densities. Further, the value of the intensity passed to the *MolPred* predictor was taken at the rest frequency of the transition and does not match the peak value due to imprecise Doppler shifting. The peak intensity of the *MolPred* prediction is close to the value of the ALCHEMI data at the transition frequency for both CO transitions.

For HCO^+, the analysis was done following the same process as used for CO. For this molecule, the ALCHEMI dataset contains data for 3 out of 4 transitions in the working frequency range. It is interesting that despite the missing transition,

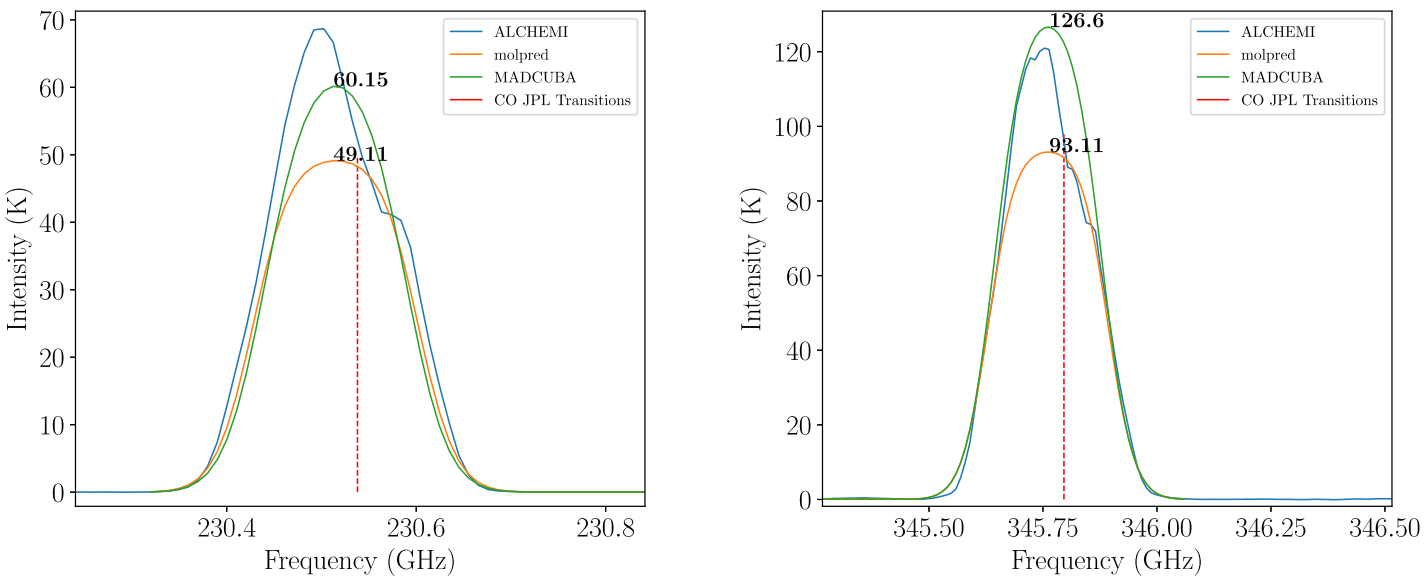


Figure 4.9: CO line profiles from the ALCHEMI Data compared with spectra generated from the *MolPred* and MADCUBA predictions.

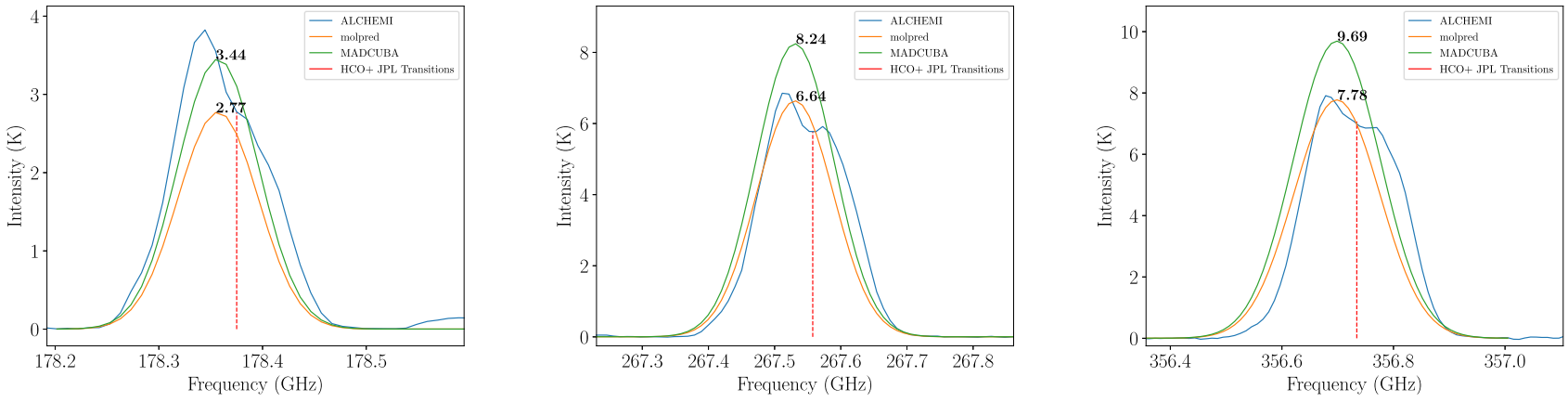


Figure 4.10: Similar to Figure 4.9 for HCO^+.

Table 4.5: Peak intensities from spectra generated using the *MolPred* and MADCUBA predictions for CO.

Molecule	J	Intensity *MolPred* (K)	Intensity MADCUBA (K)	Intensity ALCHEMI @ Transition Freq (K) @ (GHz)
CO	2-1	49.11	60.15	52.10 @ 230.5380
CO	3-2	93.11	126.60	96.14 @ 345.7959

the *MolPred* predictions are close to those obtained with MADCUBA. We can see the prediction differences per transition in Table 4.6 and visually observe them in Figure 4.10. In Figure 4.6, we can see the saturation limit beyond which the column density cannot be predicted is $log(N) = 14$ cm^{-2}. Therefore, we might expect the accuracy to be affected by this, since the column density predicted by *MolPred* is close to this value. Despite this, the spectra from the *MolPred* predictions are a good fit to the ALCHEMI dataset.

Table 4.6: Peak intensities from spectra generated using the *MolPred* and MADCUBA predictions for HCO^+.

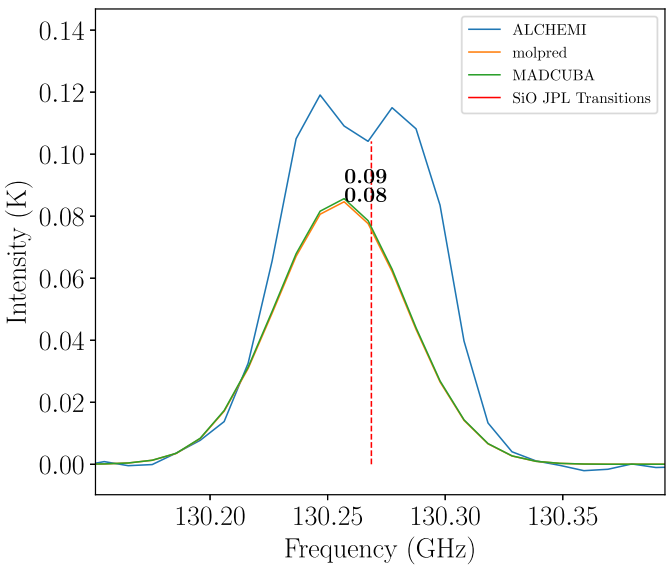
Molecule	J	Intensity *MolPred* (K)	Intensity MADCUBA (K)	Intensity ALCHEMI @ Transition Freq (K) @ (GHz)
HCO+	2-1	2.77	3.44	2.77 @ 178.3750
HCO+	3-2	6.64	8.24	5.77 @ 267.5576
HCO+	4-3	7.78	9.69	6.99 @ 356.7342

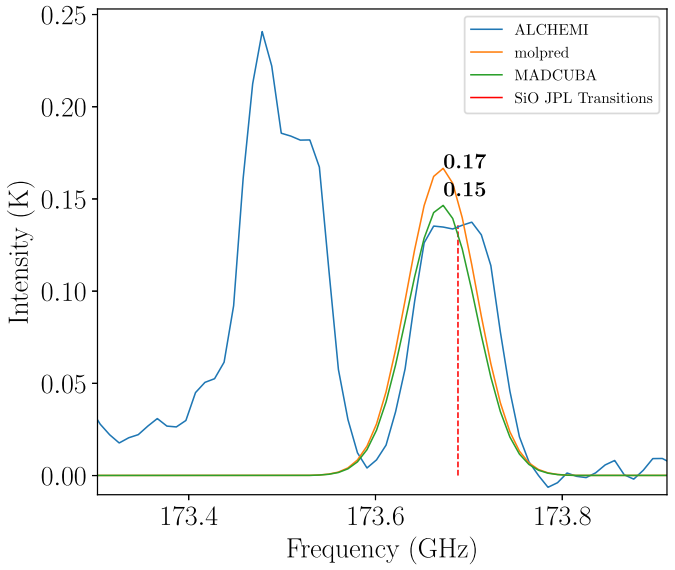
The ALCHEMI dataset contains 6 out of the 8 SiO transitions in the working frequency range. Similarly to HCO^+, we can see the prediction accuracy is good despite the missing transitions. Intensities predicted for SiO transitions can be examined on Table 4.7 and seen in Figure 4.11. The SiO lines are quite weak and the *MolPred* fits give a small $log(N)$. As a result, the temperature prediction suffers for SiO as discussed in section 4.1. When used to generate an LTE spectrum, the high temperature predicted by *MolPred* for SiO, results in line intensities that are often too large.

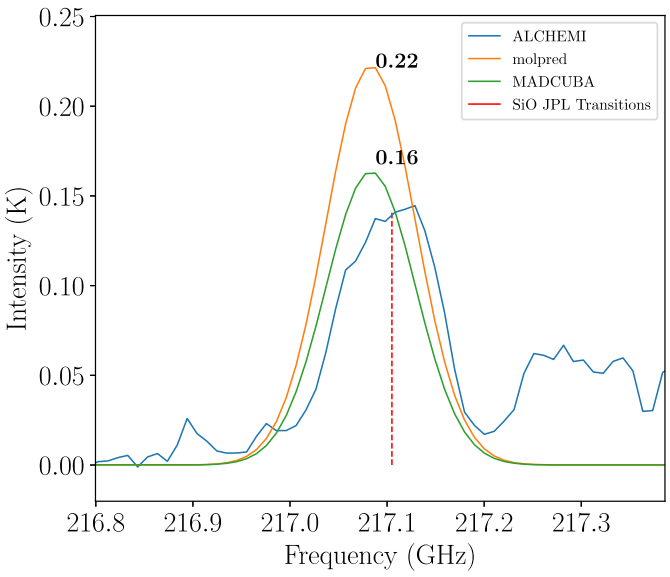
Table 4.7: Peak intensities from spectra generated using the *MolPred* and MADCUBA predictions for SiO

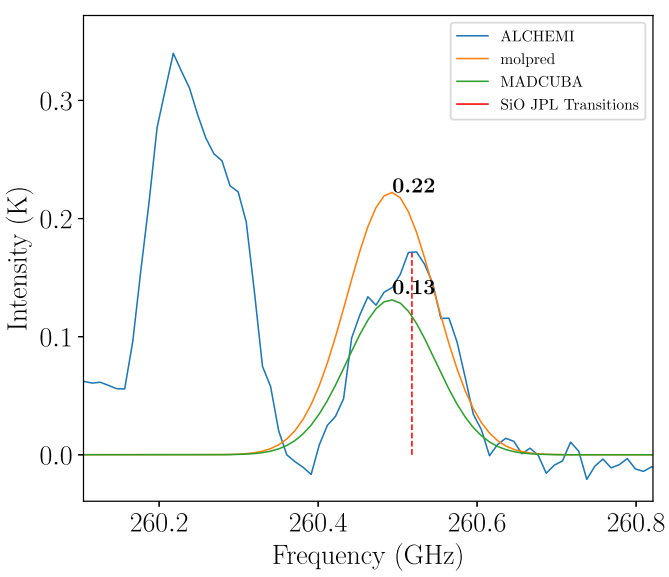
Molecule	J	Intensity *MolPred* (K)	Intensity MADCUBA (K)	Intensity ALCHEMI @ Transition Freq (K) @ (GHz)
SiO	3-2	0.08	0.09	0.10 @ 130.2686
SiO	4-3	0.17	0.15	0.13 @ 173.6884
SiO	5-4	0.22	0.16	0.14 @ 217.1050
SiO	6-5	0.22	0.13	0.17 @ 260.5180
SiO	7-6	0.17	0.08	0.15 @ 303.9270
SiO	8-7	0.11	0.04	0.08 @ 347.3306

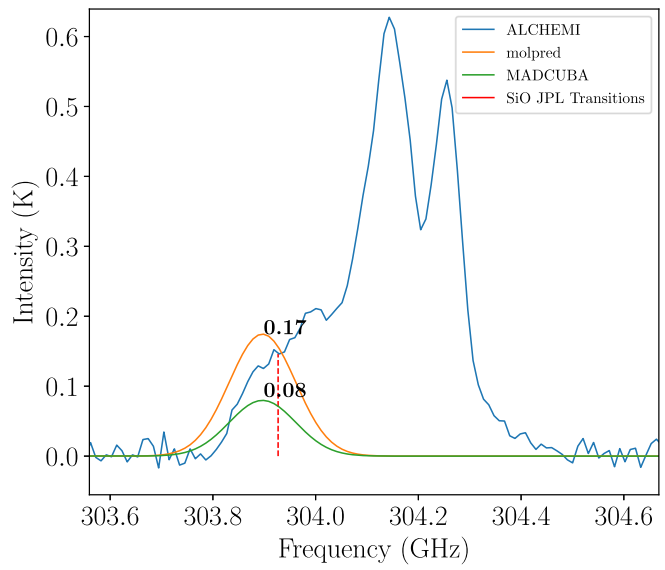
Finally we move to the molecule with the largest number of transitions in this exercise, CH_3CN. Since there are 14 groups of transitions with the same J quantum number, we will zoom in on each group to observe the behavior in Figures 4.12 and 4.13. The transitions are grouped by the J quantum numbers and the frequency of each transition are indicated by silver lines. Predicted intensities generated for each group are described in Table 4.8. Since the molecule contains many transitions per group, we decided to select a group representative frequency (GRF) as the frequency that is closer to the *MolPred* and MADCUBA's prediction peaks. We included in the table, the ALCHEMI intensity at that frequency. *MolPred* predictions for $log(N)$ are close to 13.4 cm^{-2}, at which point the noise starts to strongly affect the predictions (see Figure 4.6). This may explain why the temperature predicted by *MolPred* for CH_3CN differs so strongly from











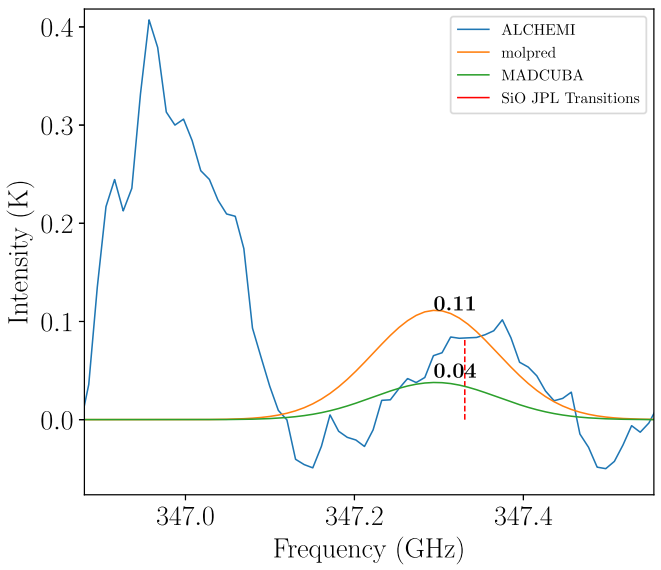


Figure 4.11: Similar to Figure 4.9 for the SiO transitions in the ALCHEMI data.

the MADCUBA prediction. We can see the effect of this inaccuracy on the intensities of several transitions, where the intensities derived from the *MolPred* prediction, are higher than the ALCHEMI intensity.

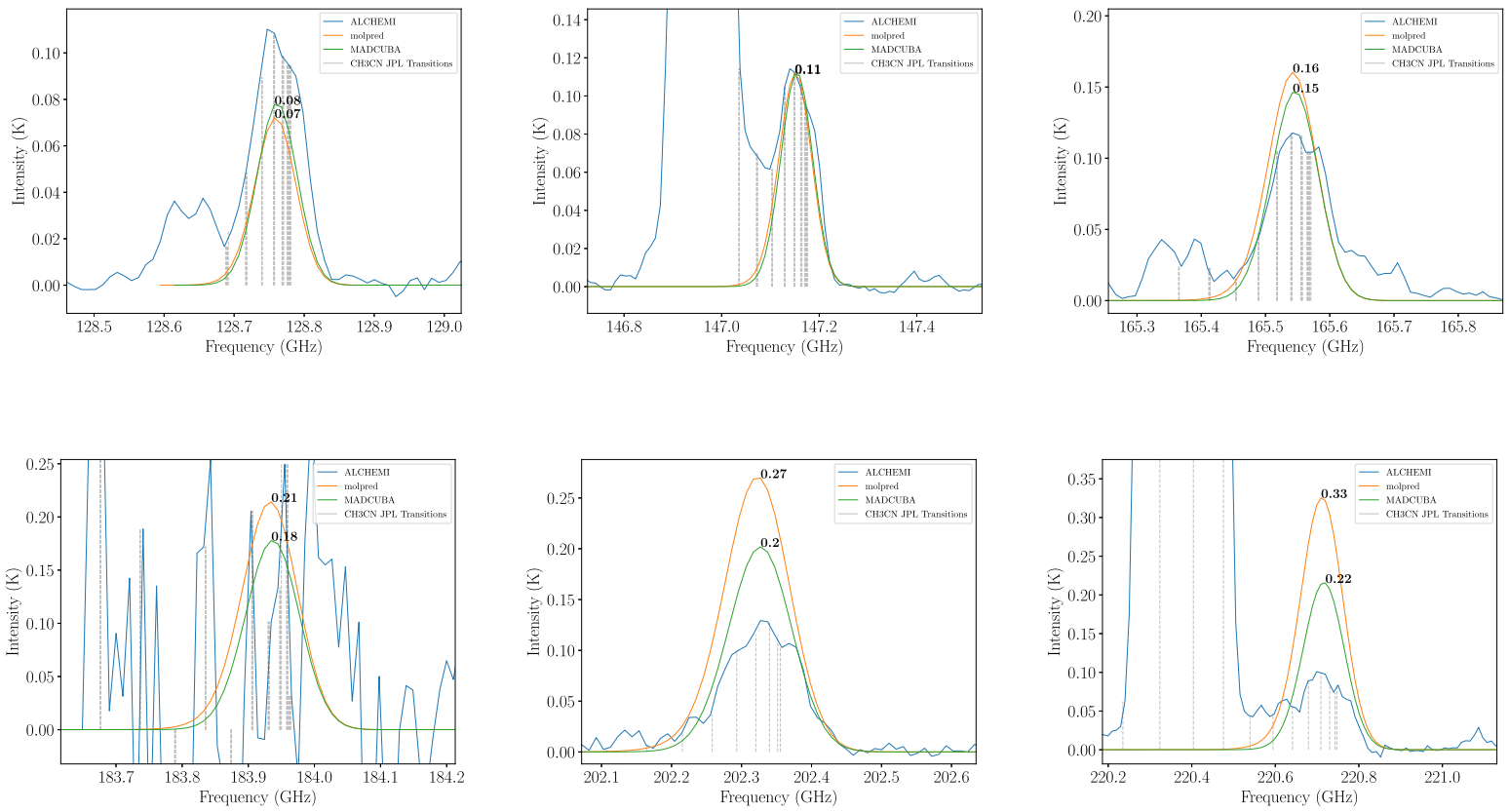


Figure 4.12: Similar to Figure 4.9 for the CH_3CN transitions in the ALCHEMI data. Groups 1 to 6

Table 4.8: Peak intensities from spectra generated using the *MolPred* and MADCUBA predictions for CH_3CN.

Molecule	J	Intensity *MolPred* (K)	Intensity MADCUBA (K)	Intensity ALCHEMI at GRF (K) @ (GHz)
CH_3CN	$7-6$	0.07	0.08	0.11 @ 128.7577
CH_3CN	$8-7$	0.11	0.11	0.11 @ 147.1499
CH_3CN	$9-8$	0.16	0.15	0.12 @ 165.5415
CH_3CN	$10-9$	0.21	0.18	0.07 @ 183.9320
CH_3CN	$11-10$	0.27	0.20	0.12 @ 202.3204
CH_3CN	$12-11$	0.33	0.22	0.10 @ 220.7090
CH_3CN	$13-12$	0.37	0.22	0.13 @ 239.0968
CH_3CN	$14-13$	0.42	0.21	0.08 @ 257.4830
CH_3CN	$15-14$	0.44	0.20	0.07 @ 275.8678
CH_3CN	$16-15$	0.46	0.17	0.07 @ 294.2514
CH_3CN	$17-16$	0.46	0.15	0.09 @ 312.6336
CH_3CN	$18-17$	0.45	0.12	0.08 @ 331.0143
CH_3CN	$19-18$	0.43	0.10	2.6 @ 349.3450
CH_3CN	$20-19$	0.40	0.07	0.04 @ 367.0777

Figure 4.14 summarizes the prediction differences seen in previous tables. In many cases, *MolPred* does very well, giving

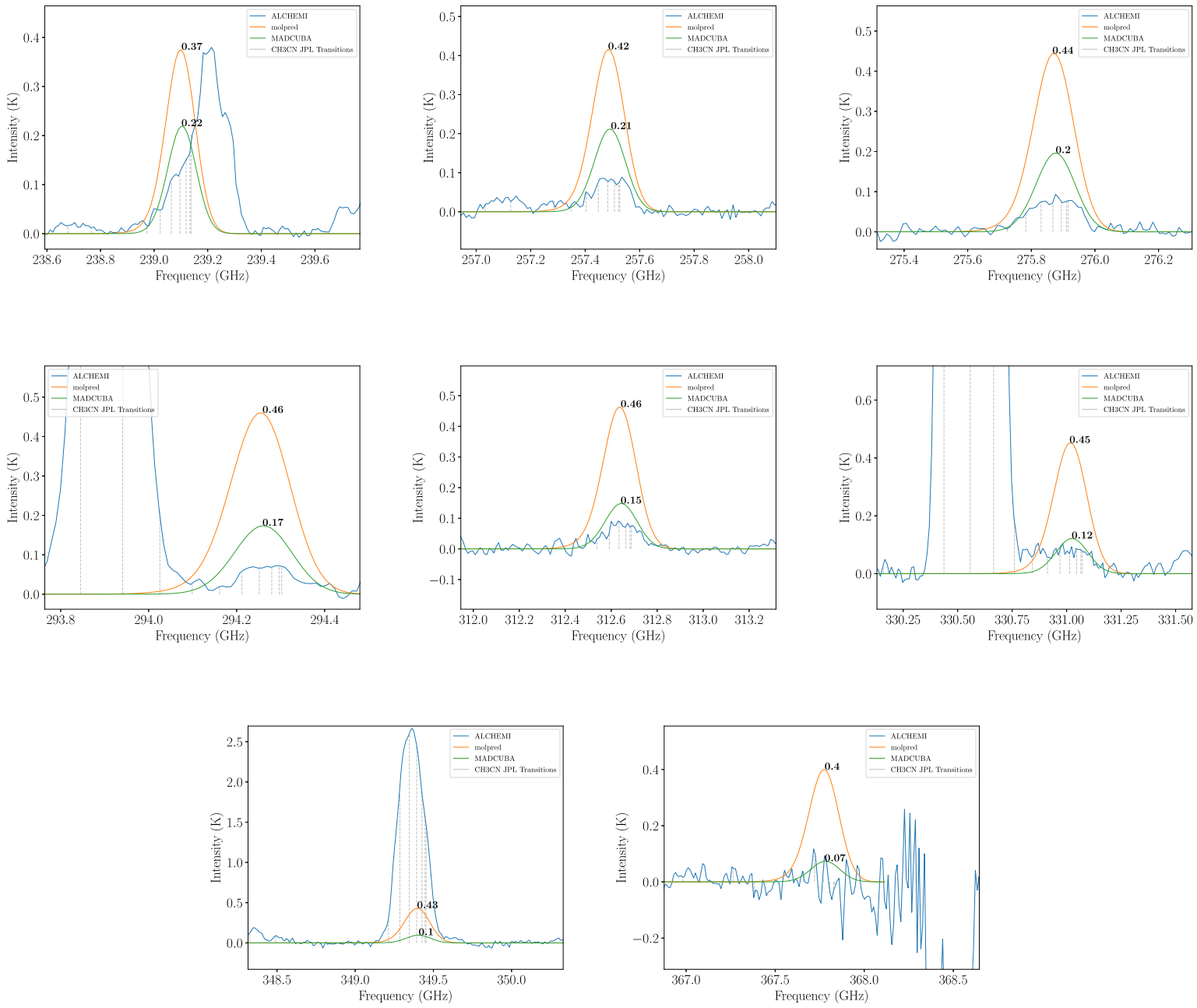


Figure 4.13: Similar to Figure 4.9 for the CH_3CN transitions in the ALCHEMI data. Groups 7 to 14.

predicted intensities that are closer to the data than MADCUBA. Where the results appear significantly different, these differences can be understood based on the way the parameters are obtained. In the case of MADCUBA, the entire spectrum is used to fit a comb of Gaussian profiles at the frequencies of the molecular transitions. This includes a simultaneous fit to the width of the line, which can help constrain the effect of opacity mentioned in Section 3.4 of Martín et al. (2019), and the fit may be more robust to individual channel variations due to line shape or noise. On the other hand, our neural networks are trained with single intensity values for each transition. We can see how in the particular cases of CO and HCO^+, the prediction from *MolPred* follows very closely the intensity at the reference frequencies.

It is important to indicate that the training examples were calculated for a velocity of 250 km s^{-1} which is slightly different from the source velocity and thus the channel closest to each transition's rest frequency is not the peak intensity. This misalignment affects our neural network predictions since it uses the intensity of a single channel instead of the integrated emission. The results from Figures 4.11, 4.12 and 4.13 suggest that a higher number of transitions may contribute to construct a better characterized model of the molecule. Figures 4.12 and 4.13 also show the strong impact of contamination from brighter transitions from other species on the fit results. This is severely affecting $19 - 18$ transition of CH_3CN. Despite its limitations, the *MolPred* predictions are close to the MADCUBA fits for our astronomical data test as seen in

Figure 4.14. The plot shows real and predicted intensities as a function of J from MolPred (orange), MADCUBA (green) and the ALCHEMI data (blue). Intensities are at the rest frequency of the transitions. We believe that its performance can be further tuned moving to a model which uses more information from the spectrum such as the full line profile or an integrated intensity.

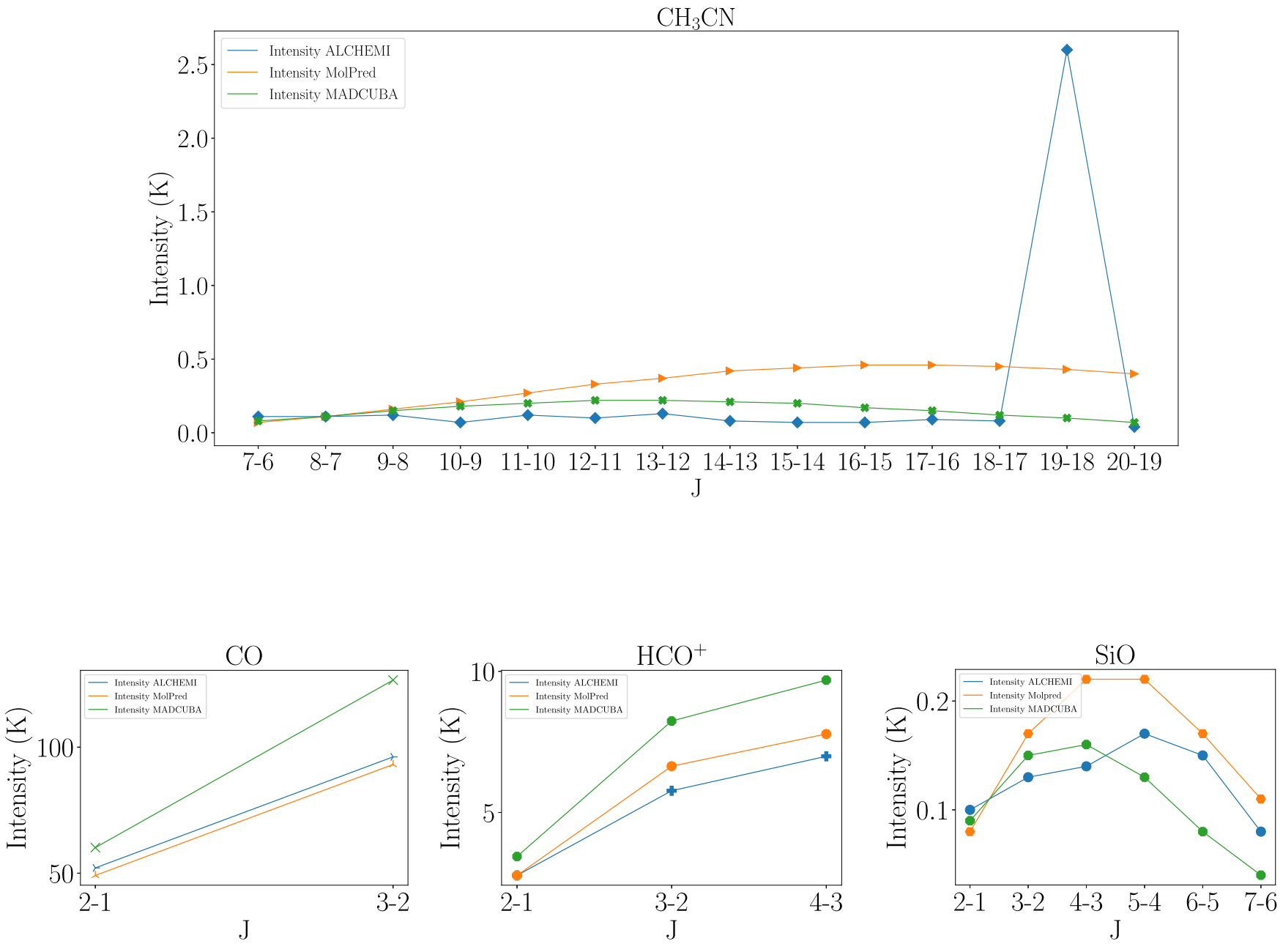


Figure 4.14: Real and predicted intensities as a function of J from MolPred, MADCUBA and ALCHEMI data.

4.3 Scalability and Linear Speedup

In order to observe the effects of the paralellization of the predictions, we devised a test where we would test four different scenarios based on the number of molecules passed to *MolPred*. Each value was obtained by executing *MolPred* thirty times, and averaging the execution times. Scenario number one (S1) was to run *MolPred*, passing one molecule each time. Scenario number two (S2) consisted in passing pairs of molecules on each run, CO and HCO^+ in the first run, and SiO and CH_3CN in the second run. Scenario number three (S3) consisted in passing three molecules CO, HCO^+ and SiO in one run. We kept the time for CH_3CN from its individual pass. Scenario number four (S4), consisted in passing all four molecules in a single run.

We can observe the times in Table 4.9 where Scenario number one shows a linear cumulative time of twenty seconds. Checking scenario number two, we can observe a 45% decrease in time, as expected from running in parallel. Furthermore,

we can observe that scenario number four displays a 68% time reduction, agreeing with the expected behaviour.

Table 4.9: Prediction speed test, average times in seconds after 30 executions

	CO	HCO+	SiO	CH3CN	Cumulative
S1	5.122	5.066	5.246	4.981	20.415
S2	5.671		5.604		11.275
S3	5.888			4.981	10.869
S4	6.573				6.573

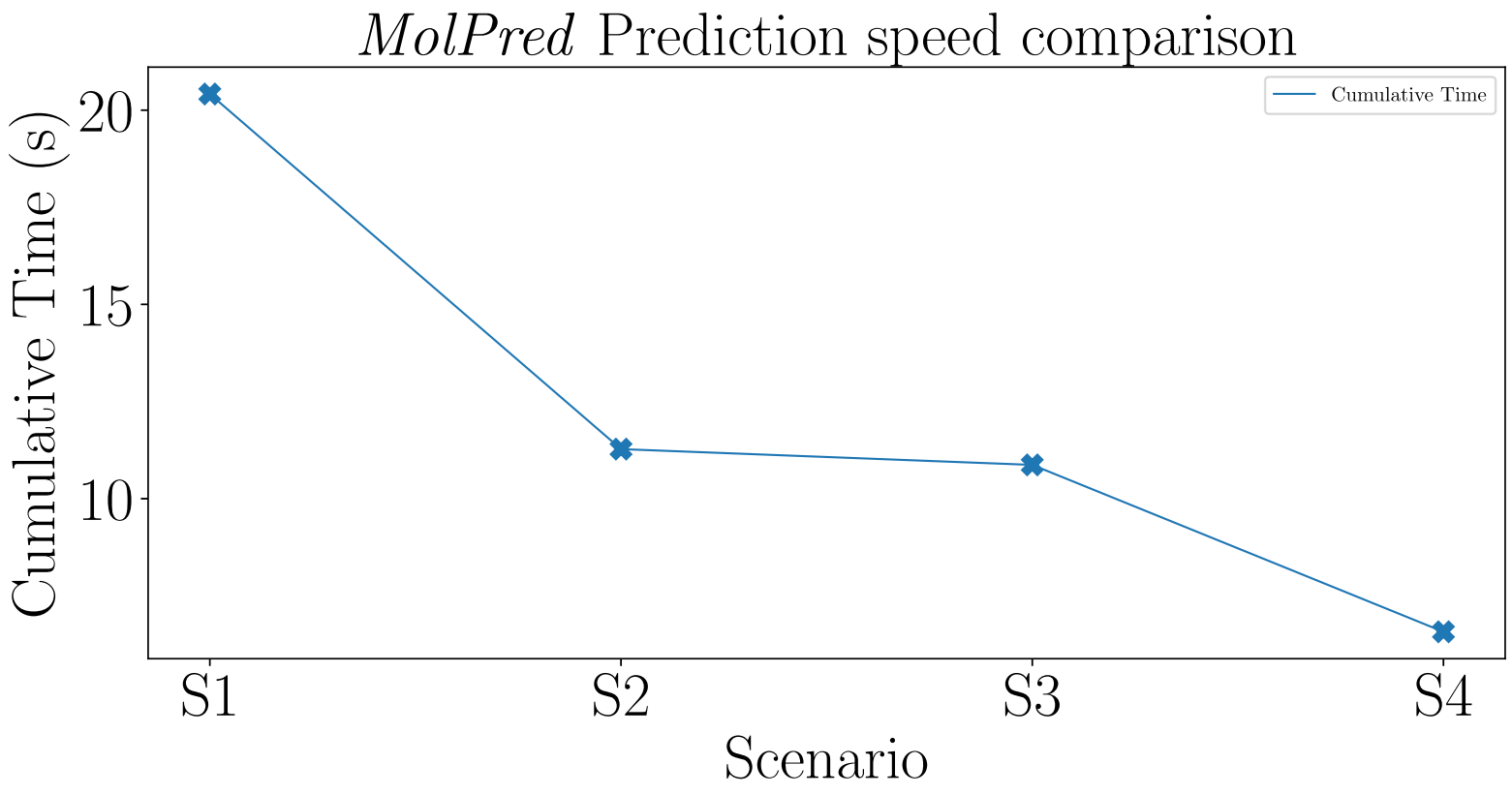


Figure 4.15: *MolPred* prediction speed comparison, average times in seconds after 30 executions

We can observe in Figure 4.15 that the *MolPred* design, allows to have faster predictions for a larger number of molecules than if they were calculated in a serial way. We can calculate the prediction *speedup* by applying Equation 3.1, to Scenario 2, as shown in Equation 4.1 and Scenario 4 in Equation 4.2.

$$S(S2) = \frac{20.415}{11.275} = 1.810x \tag{4.1}$$

$$S(S4) = \frac{20.415}{6.573} = 3.105x \tag{4.2}$$

It can be observed, that the speedup obtained for S2 and S4 are close to the theoretical optima of 2x and 4x, respectively.

4.4 Code Availability

Code is available for reproducibility in our GitHub repository: `https://github.com/MadScience01/molpred`

Chapter 5

Conclusions

We described a new algorithm called *MolPred* with a prototype implementation which is able to extract the peak intensities of molecular transitions from spectra and use them to estimate the column density and excitation temperature of the molecule. It is able to do this for CO, HCO^+, SiO and CH_3CN with a mean error of 1-9% on the predicted values when evaluated on synthetic data.

MolPred was also shown to perform well on real astronomical data. A spectrum from NGC 253 was processed by *MolPred* and the values obtained were similar to those found using the MADCUBA software package. This was despite the fact that some molecular transitions required for the networks were missing from the data. The predicted values of $log(N)$ and T_{ex} from *MolPred* were within 13% of those found using MADCUBA on average. The differences with MADCUBA are understood, and mostly related to the fact that *MolPred* is using single values per transition training and spectrum analysis rather than the whole spectral profile.

We concluded that the best way to train our networks, was by the use of a single GPU, and observed that increasing the batch size, allowed lower training times even achieving 87% faster training times, at the expense of a larger MAE, with an overall increase of 3%, however the relative increase was over 50% which motivated us to remain using lower batch sizes to achieve lower error levels. *MolPred*'s parallel prediction design allowed to train all the molecules in 68% less time (or 3x faster, closer to the theoretical optimum of 4x) than what it would have taken to train them one after another.

5.1 Contribution

By predicting molecular parameters, an astronomer could generate synthetic spectra based on those predictions, then subtract the emission from the input spectrum. A residual spectrum could then be produced where fainter lines can be seen in a clearer way. From the astronomical point of view, this is a contribution to the astrochemist line of work, assisting in the endeavour of identifying previously unidentified lines, this is aligned with the ALMA 2030 Development Roadmap (Carpenter et al., 2020), where upgrades are planned for the telescope capabilities, potentially increasing the bandwidth of the receivers, which can derive in the increment of the number of spectral surveys where algorithms like the one we proposed and tested, will be of use. This work is borderline, as it requires expert knowledge between two domains, it encourages synergy between Astronomy and Computer Science, adding to the knowledge base of the young discipline of Astroinformatics, and offers supporting evidence for our hypothesis.

5.2 Future Work

Further work should include the addition of uncertainty handling in the predictions, increasing the number of molecules and predicting a broader range of physical parameters such as the line width and source size, for which the whole line profile should be used. These additions would enable the possibility for *MolPred* to work with different astronomical sources. The number of molecules should be extended, and the capacity to predict highly blended spectrum should be explored. Another parameter to explore would be multiple velocity components. However, this will require further tuning of the networks as three of the four best neural networks in this work had three layers of 1024 nodes. This would be prohibitively large if many molecules were considered, each needing their own neural network. Therefore, either these networks must be greatly reduced or networks must be trained which can solve for more than one molecule. We believe that the door is open, to try other types of neural networks or algorithms to approach the problem addressed by this work.

Appendix A

Individual model results.

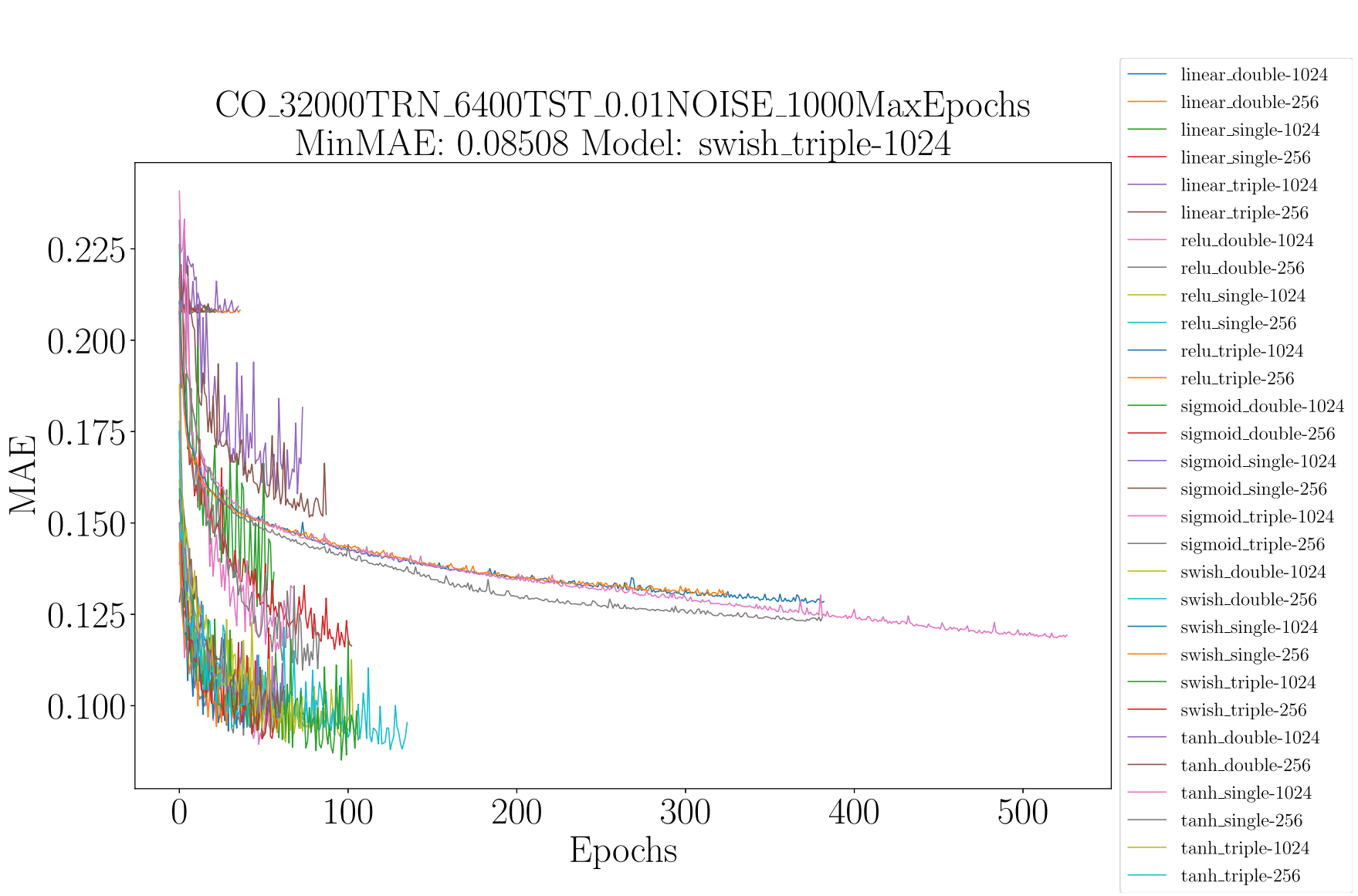


Figure A.1: Mean absolute error vs epoch for all combinations of activation functions and layer types for CO.

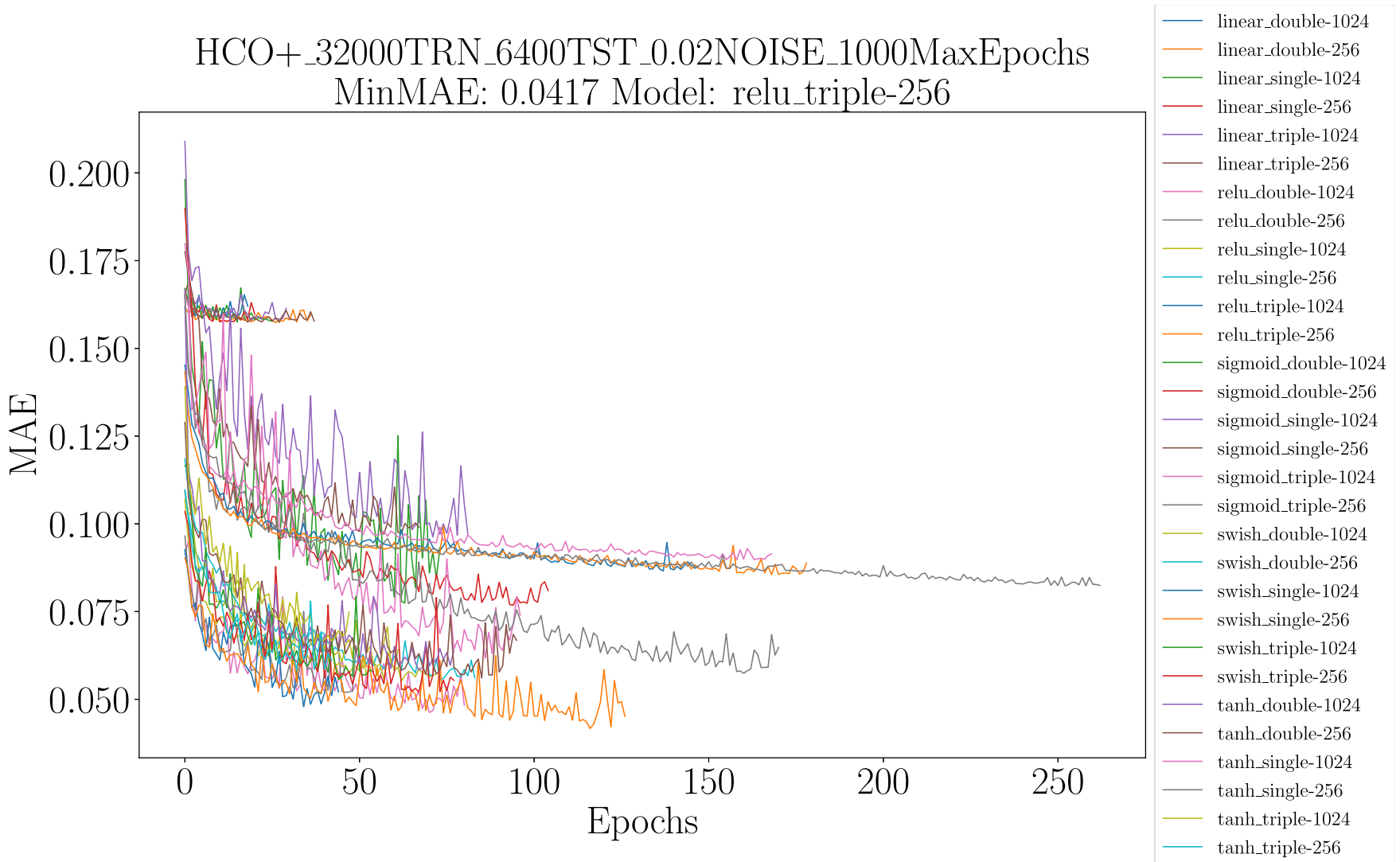


Figure A.2: Mean absolute error vs epoch for all combinations of activation functions and layer types for HCO^+.

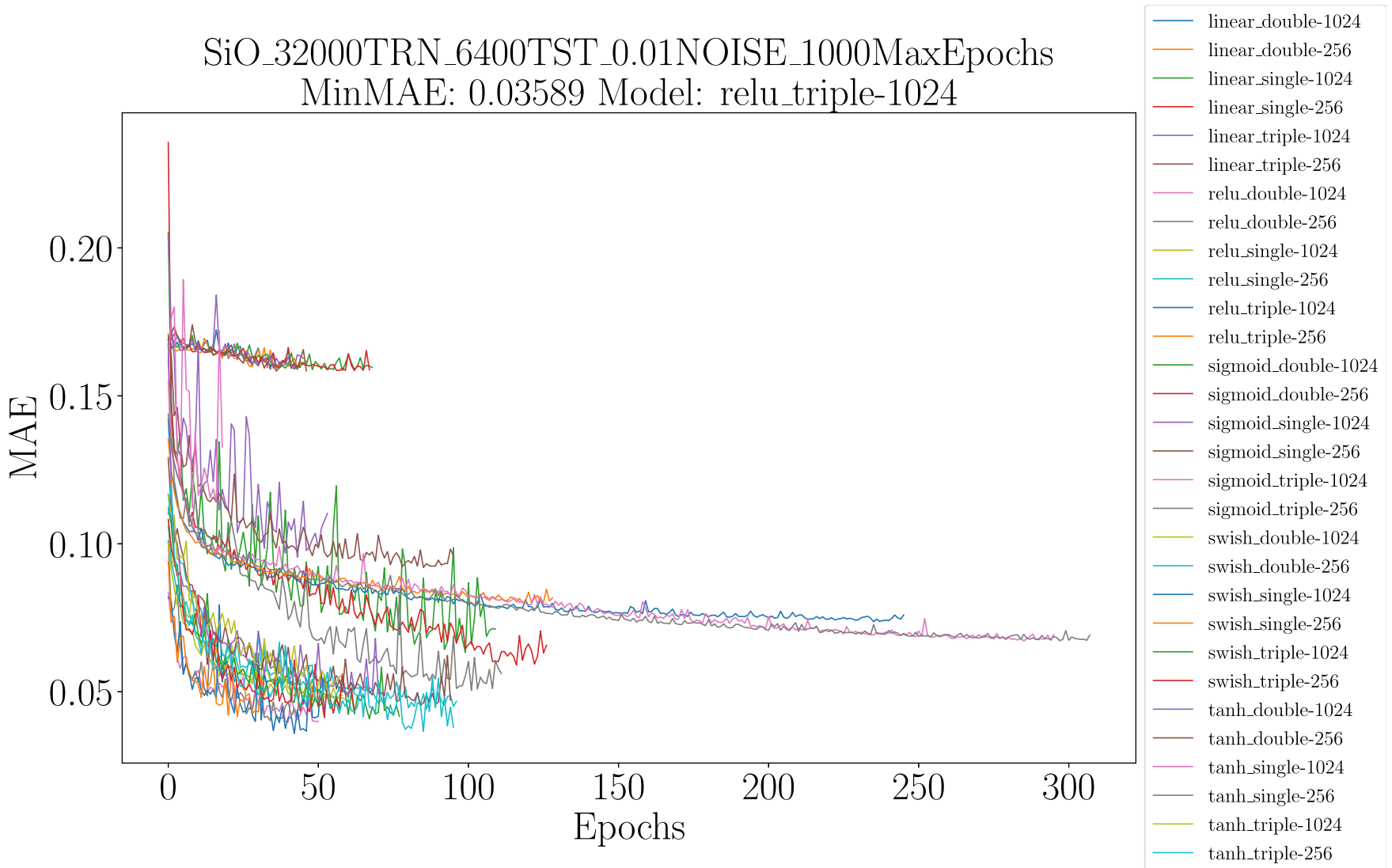


Figure A.3: Mean absolute error vs epoch for all combinations of activation functions and layer types for SiO.

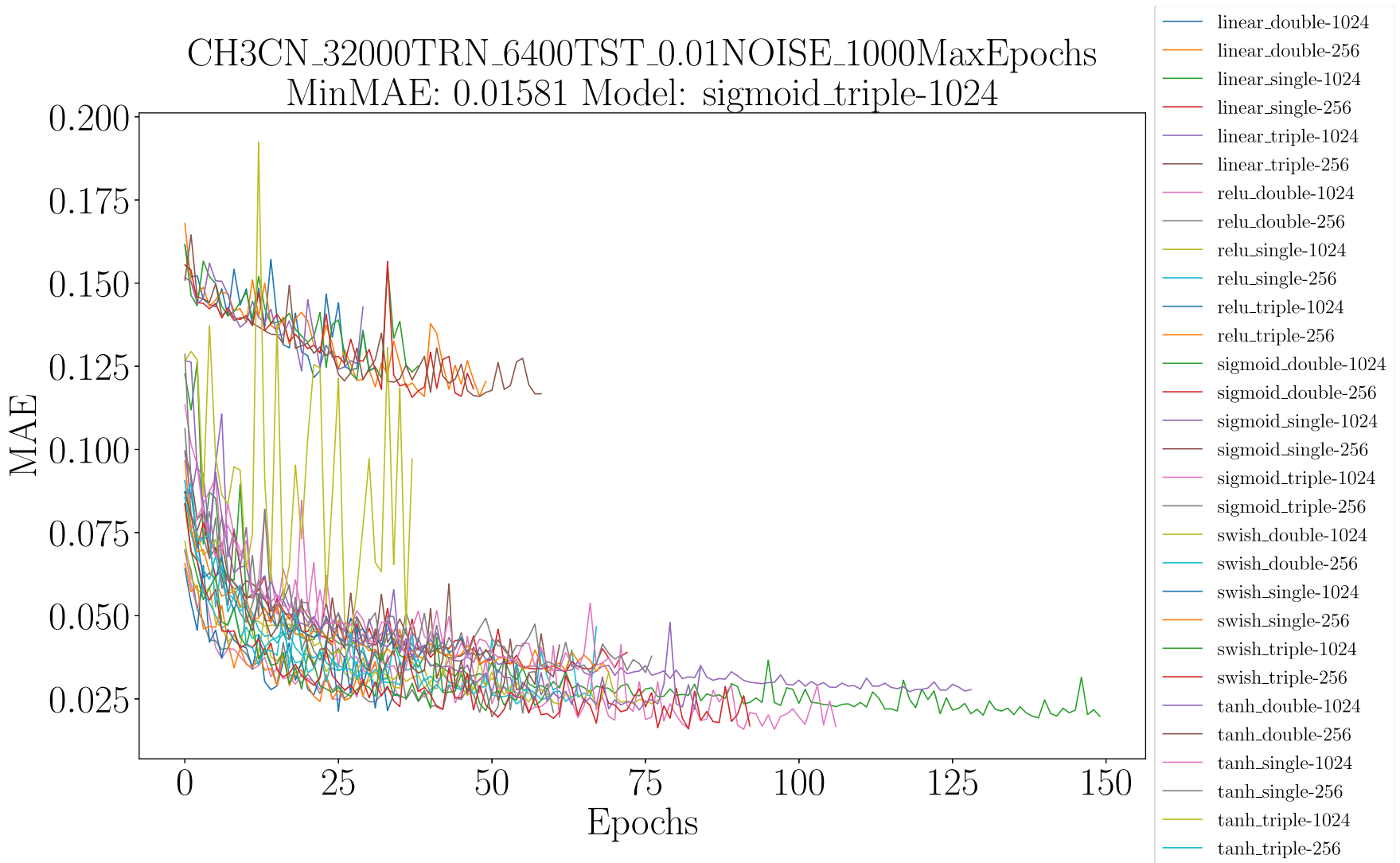


Figure A.4: Mean absolute error vs epoch for all combinations of activation functions and layer types for CH_3CN.

Appendix B

Detailed workflow models

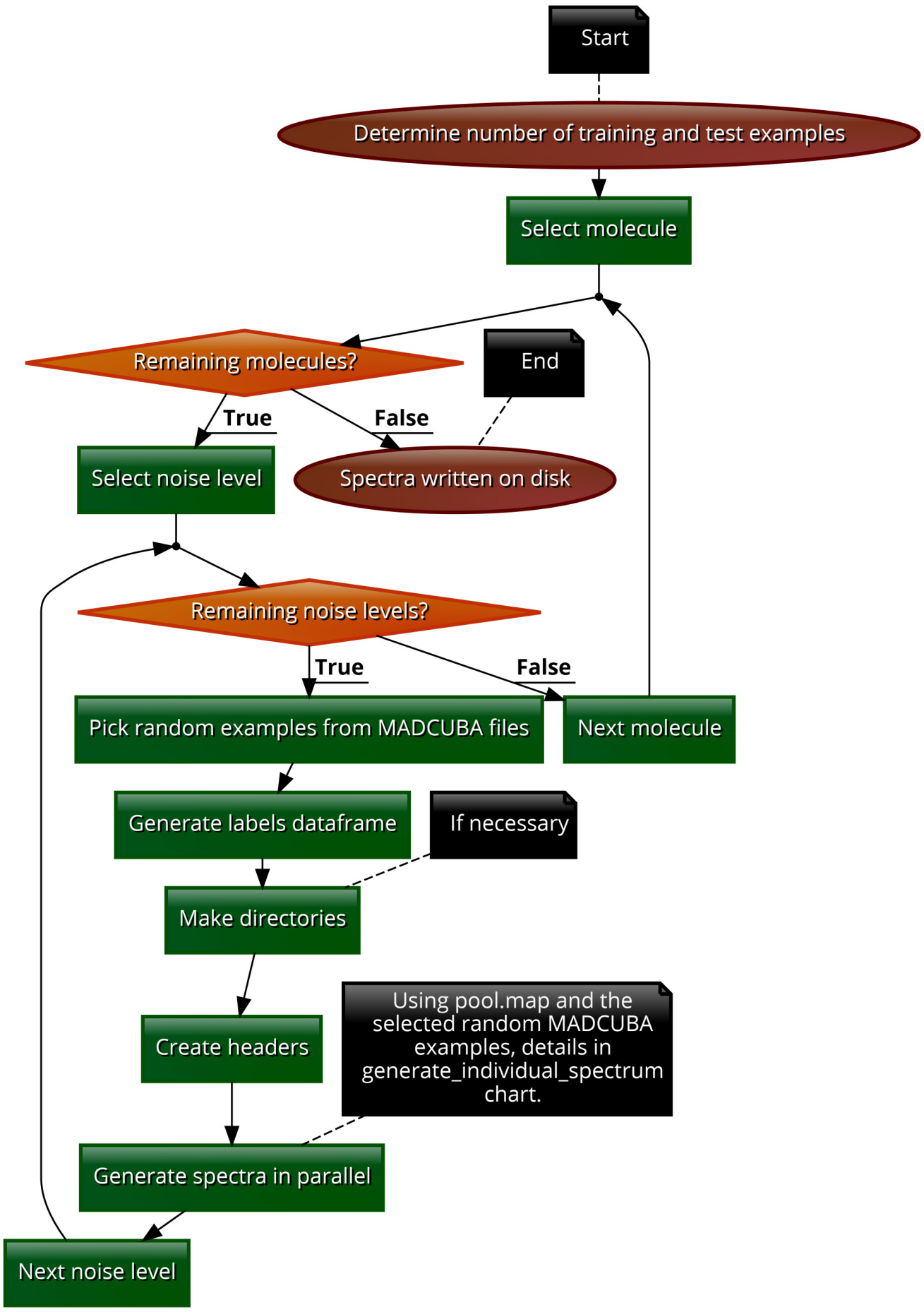


Figure B.1: Create Training Examples workflow

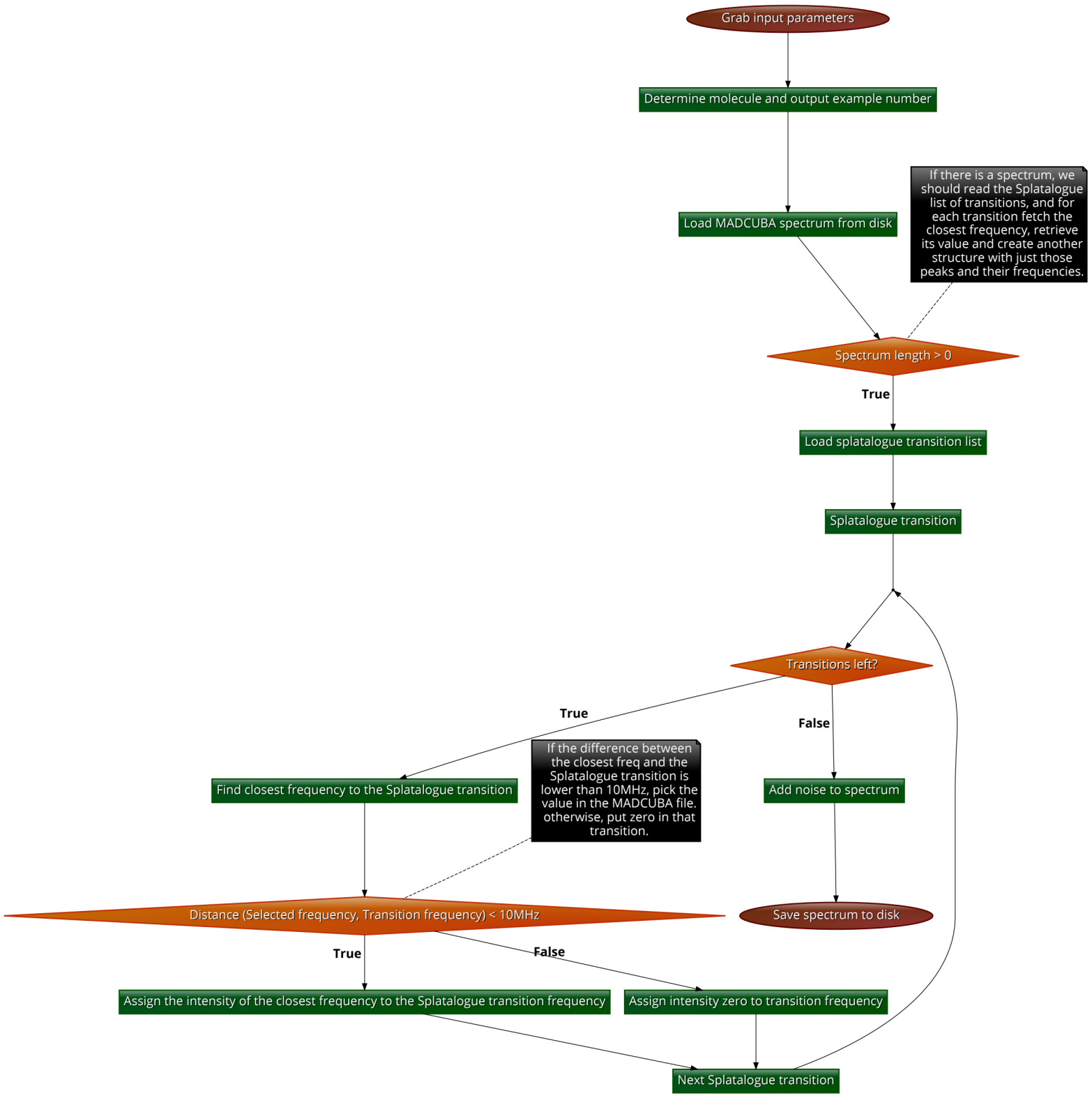


Figure B.2: Generate individual training spectrum workflow

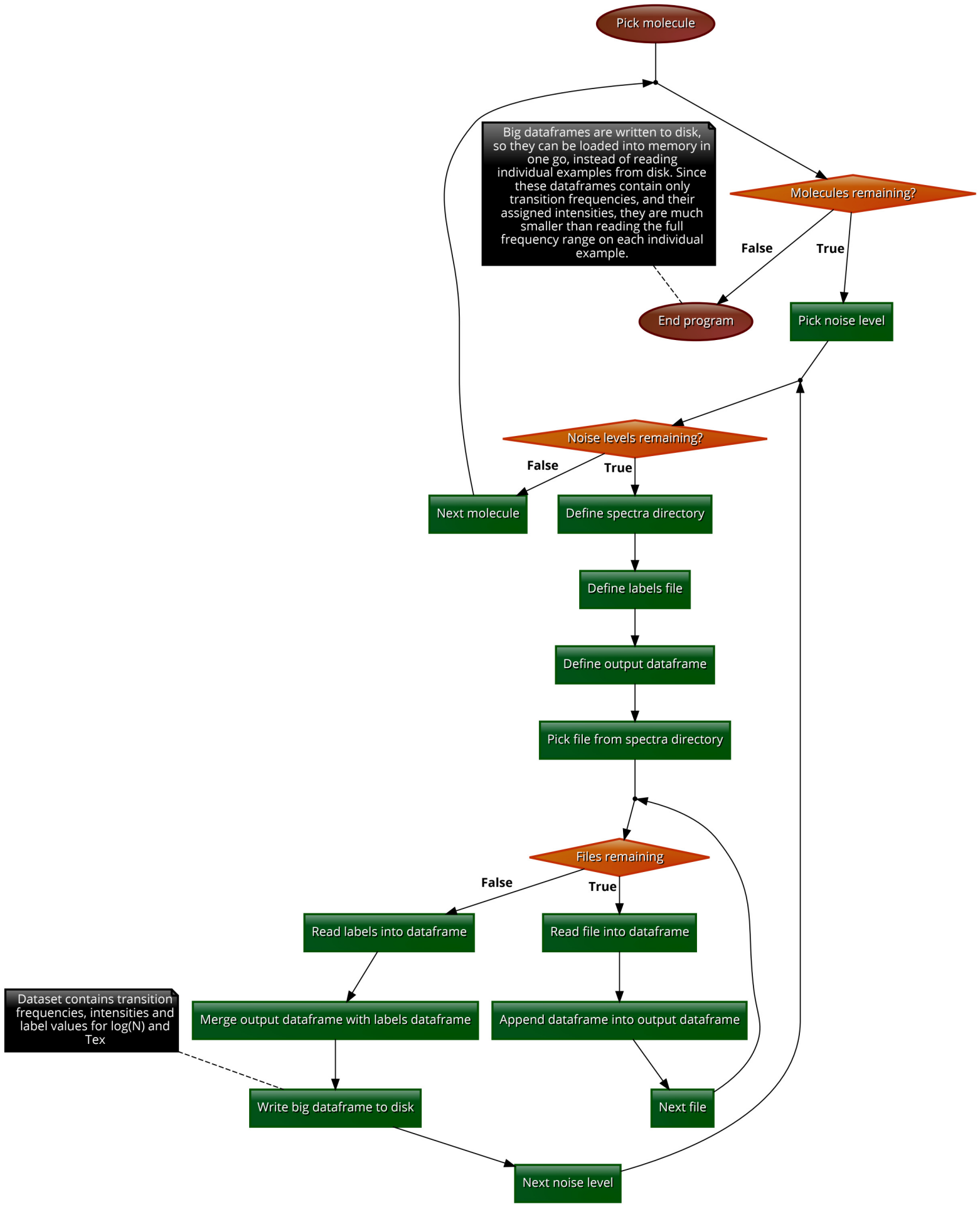


Figure B.3: Build Dataset workflow

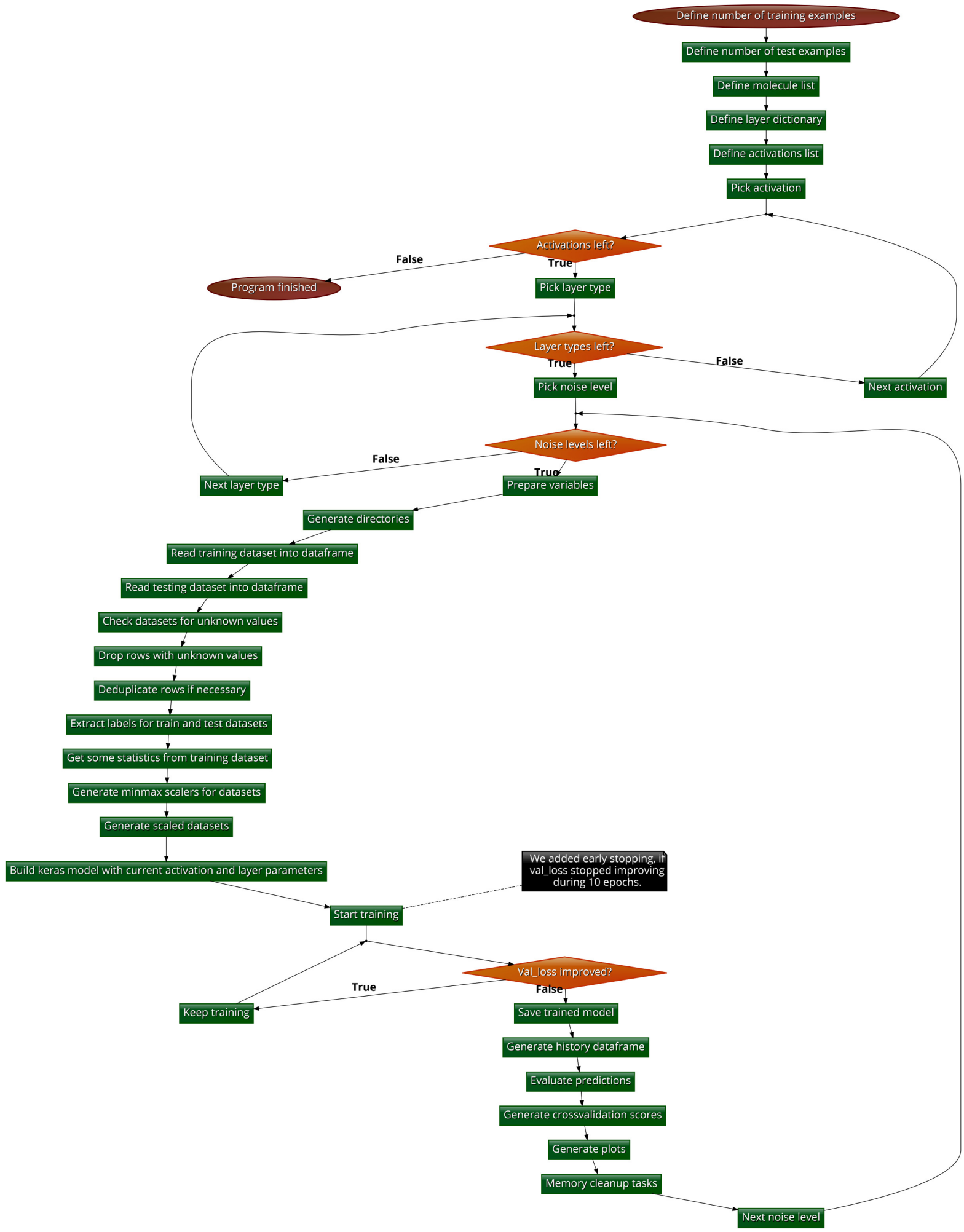


Figure B.4: Neural Network Training workflow

Bibliography

Agarap, A.F., 2018. Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375 .

Aggarwal, C.C., 2018. Neural Networks and Deep Learning. Springer, Cham. doi:10.1007/978-3-319-94463-0.

Barrientos, A., Solar, M., 2016. Machine learning approaches for detection and classification of astrochemical spectral lines. Astronomical Data Analysis Software and Systems XXVI (ADASS XXVI) 521, 189–192.

Berriman, G.B., Groom, S.L., 2011. How will astronomy archives survive the data tsunami? Commun. ACM 54, 52–56. URL: http://doi.acm.org/10.1145/2043174.2043190, doi:10.1145/2043174.2043190.

Bishop, C.M., 2006. Pattern Recognition and Machine Learning. Springer.

Born, M., Oppenheimer, R., 1927. Zur quantentheorie der molekeln. j-ANN-PHYS-1900 389, 457–484. doi:http://dx.doi.org/10.1002/andp.19273892002.

Carpenter, J., Iono, D., Kemper, F., Wootten, A., 2020. The alma development program: Roadmap to 2030. arXiv:2001.11076.

Cernicharo, J., 2012. Laboratory astrophysics and astrochemistry in the herschel/alma era. http://tinyurl.com/ogf562e.

Chollet, F., et al., 2015. Keras. URL: https://github.com/fchollet/keras.

Comrie, A., Pińska, A., Simmonds, R., Taylor, A., 2020. Development and application of an hdf5 schema for ska-scale image cube visualization. Astronomy and Computing 32, 100389. URL: https://www.sciencedirect.com/science/article/pii/S2213133720300433, doi:https://doi.org/10.1016/j.ascom.2020.100389.

Cybenko, G., 1989. Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals, and Systems (MCSS) 2, 303–314. URL: http://dx.doi.org/10.1007/BF02551274, doi:10.1007/BF02551274.

Efron, B., 1979. Bootstrap Methods: Another Look at the Jackknife. The Annals of Statistics 7, 1 – 26. URL: https://doi.org/10.1214/aos/1176344552, doi:10.1214/aos/1176344552.

Emonts, B., Raba, R.and Pouzols, F.M., Tsutsumi, T., Nakazato, T., Kepley, A., Schiebel, D., Castro, S., Comrie, A., Wang, .K., Bhatnagar, S., Brandt, P., Brogan, C., Meyer, J.D., Ford, P., Golap, K., García-Dabó, C., Garwood, B., Hale, A., Hunter, T., Kent, B., Kawasaki, W., Indebetouw, R., Mehringer, D., Miel, R., Moellenbrock, G., Nishie, S., Ott, J., Petry, D., Pokorny, M., Rau, U., Reynolds, C., Sugimoto, K., Suoranta, V., Schweighart, N., Tafoya, D., Wells, A., Yoon, I., 2018. The casa software for radio astronomy: Status update from adass 2018. Astronomical Data Analysis Software and Systems XXVIII) 523, 265–268.

Farnes, J., Mort, B., Dulwich, F., Salvini, S., Armour, W., 2018. Science Pipelines for the Square Kilometre Array. arXiv e-prints , arXiv:1811.08272arXiv:1811.08272.

Jurić, M., Kantor, J., Lim, K., Lupton, R.H., Dubois-Felsmann, G., Jenness, T., Axelrod, T.S., Aleksić, J., Allsman, R.A., AlSayyad, Y., Alt, J., Armstrong, R., Basney, J., Becker, A.C., Becla, J., Bickerton, S.J., Biswas, R., Bosch, J., Boutigny, D., Carrasco Kind, M., Ciardi, D.R., Connolly, A.J., Daniel, S.F., Daues, G.E., Economou, F., Chiang, H.F., Fausti, A., Fisher-Levine, M., Freemon, D.M., Gee, P., Gris, P., Hernandez, F., Hoblitt, J., Ivezić, Ž., Jammes, F., Jevremović, D., Jones, R.L., Bryce Kalmbach, J., Kasliwal, V.P., Krughoff, K.S., Lang, D., Lurie, J., Lust, N.B., Mullally, F., MacArthur, L.A., Melchior, P., Moeyens, J., Nidever, D.L., Owen, R., Parejko, J.K., Peterson, J.M., Petravick, D., Pietrowicz, S.R., Price, P.A., Reiss, D.J., Shaw, R.A., Sick, J., Slater, C.T., Strauss, M.A., Sullivan, I.S., Swinbank, J.D., Van Dyk, S., Vujčić, V., Withers, A., Yoachim, P., LSST Project, f.t., 2015. The LSST Data Management System. ArXiv e-prints arXiv:1512.07914.

Karttunen, H., Kröger, P., Oja, H., Poutanen, M., Donner, K.J., 2017. Fundamental Astronomy. 6th ed.

Kelley, H.J., 1960. Gradient theory of optimal flight paths. Ars Journal 30, 947–954.

Kingma, D.P., Ba, J., 2017. Adam: A method for stochastic optimization. arXiv:1412.6980.

Lightfoot, J., Kosugi, G., Wyrowski, F., Zapata, L., Muders, D., Boone, F., Tsutsumi, T., Davis, L., Wilson, C., Shepherd, D., 2008. ALMA Pipeline Heuristics, in: Argyle, R.W., Bunclark, P.S., Lewis, J.R. (Eds.), Astronomical Data Analysis Software and Systems XVII, p. 573.

Mach, M., Köhler, R., Czoske, O., Leschinski, K., Zeilinger, W.W., Kausch, W., Ratzka, T., Leitzinger, M., Greimel, R., Przybilla, N., Schaffenroth, V., Güdel, M., Brandl, B.R., 2016. Data reduction software for the Mid-Infrared E-ELT Imager and Spectrograph (METIS) for the European Extremely Large Telescope (E-ELT), in: Software and Cyberinfrastructure for Astronomy IV, p. 991327. doi:10.1117/12.2232436.

Mangum, J.G., Shirley, Y.L., 2015. How to calculate molecular column density. Publications of the Astronomical Society of the Pacific 127, 266–298. URL: https://doi.org/10.1086/680323, doi:10.1086/680323.

Martín, S., Martín-Pintado, J., Blanco-Sánchez, C., Rivilla, V.M., Rodríguez-Franco, A., Rico-Villas, F., 2019. Spectral line identification and modelling (slim) in the madrid data cube analysis (madcuba) package. Astronomy & Astrophysics 631, A159. URL: http://dx.doi.org/10.1051/0004-6361/201936144, doi:10.1051/0004-6361/201936144.

Mcculloch, W., Pitts, W., 1943. A logical calculus of ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5, 127–147.

McMullin, J.P., Waters, B., Schiebel, D.and Young, W., Golap, K., 2007. Casa architecture and applications. Astronomical Data Analysis Software and Systems XVI (ADASS XVI) 376, 127–130.

Mendoza, M., Barrientos, A., Solar, M., Araya, M., 2017. Mixed membership models for source separation of spectral lines, in: 8th International Conference of Pattern Recognition Systems (ICPRS 2017), pp. 1–6. doi:10.1049/cp.2017.0151.

Miranda, N., Cabrera, G., 2015. Association rules for spectral lines. URL: http://repositorio.uchile.cl/handle/2250/133022.

Möller, T., Schilke, P., 2015. Manual for xclass-interface. https://www.astro.uni-koeln.de/wd-schilke/myXCLASS/XCLASS-Interface__No-NR__Linux__version_1.1.6.zip.

Nakajima, T., Takano, S., Kohno, K., Harada, N., Herbst, E., Tamura, Y., Izumi, T., Taniguchi, A., Tosaki, T., 2015. A multi-transition study of molecules toward NGC 1068 based on high-resolution imaging observations with ALMA. Publications of the Astronomical Society of Japan 67, 8. doi:10.1093/pasj/psu136, arXiv:1410.5912.

Pickett, H., Poynter, R., Cohen, E., Delitsky, M., Pearson, J., Müller, H., 1998. Submillimeter, millimeter, and microwave spectral line catalog. Journal of Quantitative Spectroscopy and Radiative Transfer 60, 883–890. URL: https://www.sciencedirect.com/science/article/pii/S0022407398000910, doi:https://doi.org/10.1016/S0022-4073(98)00091-0.

Ramachandran, P., Zoph, B., Le, Q.V., 2017. Searching for activation functions. CoRR abs/1710.05941. URL: http://arxiv.org/abs/1710.05941, arXiv:1710.05941.

Rosenblatt, F., 1958. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review 65, 386–408.

Schilke, P., Möller, T., Comito, C., Sánchez-Monge, Á., Schmiedeke, A., Zernickel, A., 2015. Taming the Dragon: Automatic Line Fitting of ALMA data, in: Iono, D., Tatematsu, K., Wootten, A., Testi, L. (Eds.), Revolution in Astronomy with ALMA: The Third Year, p. 195.

Swings, P., Rosenfeld, L., 1937. Considerations regarding interstellar molecules. Astrophysical Journal 86, 483–486.

Tennyson, J., 2005. Astronomical Spectroscopy: An Introduction to the Atomic and Molecular Physics of Astronomical Spectra. Imperial College Press.

Teuben, P., 2015. Admit 0.5.2 documentation. http://tinyurl.com/osbu75s.

van der Tak, F. F. S., Black, J. H., Schöier, F. L., Jansen, D. J., van Dishoeck, E. F., 2007. A computer program for fast non-lte analysis of interstellar line spectra* - with diagnostic plots to interpret observed line intensity ratios. A&A 468, 627–635. URL: https://doi.org/10.1051/0004-6361:20066820, doi:10.1051/0004-6361:20066820.

Vastel, C., Bottinelli, S., Caux, E., Glorian, J.M., Boiziot, M., 2015. CASSIS: a tool to visualize and analyse instrumental and synthetic spectra., in: Martins, F., Boissier, S., Buat, V., Cambrésy, L., Petit, P. (Eds.), SF2A-2015: Proceedings of the Annual meeting of the French Society of Astronomy and Astrophysics. Eds.: F. Martins, S. Boissier, V. Buat, L. Cambrésy, P. Petit, pp.313-316, pp. 313–316.

Woon, D., 2020. Lists of interstellar and circumstellar molecules. http://tinyurl.com/m69l6qg.