Astronomy & Astrophysics manuscript no. output
August 22, 2025

Machine Learning Tools for Exoplanet Detection and Study

Camille Vallipuram Jaunsen, Anastasia Marchuk, Anika Vučićević

Norwegian University of Science and Technology, University of Padua

Received August 22, 2025; accepted -

ABSTRACT

In the following report, we present on the use of machine learning tools for detection of exoplanetary systems and atmospheric study of the planet K2-18b. For the detection of planetary systems, we use Support Vector Machine and Artificial Neural Network algorithms, while for the atmospheric retrieval of molecular abundances we use TauREx. In the detection of planetary system candidates, the Standard Vector Machine algorithm proves to be better at detecting stars with planets in their orbit than the Artificial Neural Network, however, both are proven to not be as effective for the detection of single star systems. For the atmospheric retrieval, the CO and CH_4 levels are within the margin of error for the literature abundances, however, other molecules were off by over an order of magnitude.

Key words. K2-18b, exoplanet, exoplanetary systems, SVM, ANN, transit curves, atmospheric retrievals.

1. Introduction

In 2015, the K2-18b super-Earth transit curve was discovered by the Kepler telescope mission, NASA's first planet-hunting mission. It was later confirmed with Spitzer Space Telescope observations (Sarkis et al. 2018) and has now been observed by JWST as well. In the paper Montet et al. (2015), the authors discuss how the mentioned system represents an interesting candidate target for atmospheric studies, as well as an ideal candidate for the (then future) space mission of JWST. The exoplanet K2-18b is located in the habitable zone of an M-dwarf star named K2-18, which makes it of bigger interest to astronomers due to its possibility to host life. With better spectra from JWST, because of its higher spectral resolution and and infrared detection, the paper Madhusudhan et al. (2023) gives a better insight into the exoplanet's atmosphere. In this report, we will present our own atmospheric analysis of the planet K2-18b with the spectra from JWST (Madhusudhan et al. 2023), as a part of a learning process to perform atmospheric retrievals.

As more space telescopes and missions with the ability to observe exoplanets are launched, they will be sending a constantly growing amount of data back to Earth. This presents exciting opportunities for new discoveries, but it also means that data analysis becomes constantly more difficult, time consuming and computationally inefficient. Because of that, it is important to develop the newly accessible tool of machine learning algorithms to make the continued discovery of exoplanets possible at a rate that can keep up and survey new systems even faster than we can today. In this paper we will look at how machine learning tools such as Support Vector Machines and Artificial Neural Networks can be used to identify planetary transits, as well as how TauREx can be used for atmospheric study, with our target planet being K2-18b.

2. Planetary system parameters

The star K2-18 is classified as an M2 star. The research of exoplanetary systems around M-dwarf stars is of special interest due to potential detection of low-mass planets orbiting in habitable zones.Stars with small radius and mass are ideal when looking for low-mass exoplanets with large radial velocities and transit depths. Since the probability of a transit is equal to the ratio between a radius of the host star and the orbiting planet, smaller stars with larger planets are ideal transit candidates. The low luminosity of these stars means that the habitable zone is closer to the star, and the period of the planet is shorter and easier to detect.

With the help of radial velocity measurements, astronomers were able to find more about the physical characteristics of the planet. These characteristics can be found in table 1 below (EXOplanet 2024, Sarkis et al. 2018).

Name	K2-18	K2-18 b
Mass	0.36 M_{Sun}	0.028 M_{Jup}
Orbital period	/	32.94 days
Radius	0.41 R_{Sun}	0.211 R_{Jup}
Temperature	3457 K	284 K
Spectral type	M2.5 V	/

Table 1: The physical parameters of K2-18 b and its host star.

3. Detection of transit light curves using Machine Learning

Exoplanet detection has considerably improved in the last 15 years, with thousands of new planets discovered (NASA 2024). The development in observational technology has hugely contributed to this, and with the launch of Kepler, the transit method of observation is at this moment responsible for 74.5% of all exoplanetary discoveries (NASA 2024). At the time of the submission of this report, astronomers have found 4310 exoplanets with this method (NASA 2024).

To help us recognize even more exoplanetary systems from our observations, we can train algorithms with machine learning to go through existing data, recognize transit curves, and flag them as exoplanetary systems for astronomers. This makes the

No injected data: Train Set				
Kernel Type	True Negatives	True Positives	False Negatives	False Positives
Linear	5050	37	0	0
Polynomial	5050	37	0	0
Gaussian	5050	36	1	0
No injected data: Dev Set				
Kernel Type	True Negatives	True Positives	False Negatives	False Positives
Linear	565	0	5	0
Polynomial	559	0	5	6
Gaussian	565	5	0	0
Injected data: Train Set				
Kernel Type	True Negatives	True Positives	False Negatives	False Positives
Linear	0	2861	0	2226
Polynomial	95	2811	50	2131
Gaussian	37	2847	14	2189
Injected data: Dev Set				
Kernel Type	True Negatives	True Positives	False Negatives	False Positives
Linear	0	316	0	254
Polynomial	55	253	63	199
Gaussian	0	316	0	254

Table 2: The confusion matrices for all of the datasets.

process much more efficient, as it filters out the systems that are not eligible candidates. Eligible systems in our datasets are labeled as "system with a planet" — labeled as 2 — and ineligible systems as "system without planet" — labeled as 1.

During a planetary transit, there is a dip in the flux of the observed star. While changes in a star's flux can be due to many factors (one being the existence of variable stars, sunspots, et.), astronomers are very familiar with the transit model and are able to identify these differences, and can train algorithms to do this as well.

For the class assignment, the given datasets — one with no injected light curves and one with injected synthetic curves created by Batman (Kreidberg 2015) — were normalized, Fourier transformed, Gaussian filtered, and then were converted to take the shape of matrices to make them readable to the Support Vector Machine algorithm and artificial neural network that we trained and used to identify exoplanetary systems in our datasets. For the SVM algorithm, the training dataset has a 5087×1 matrix, while the dev set is a 570×1 matrix that represent the flux values at different times.

3.1. Methodology

Machine learning is suitable for detecting planets mainly due to the large amount of data. Transits can be as often as every few hours, or very rare, and they are relatively short. However, it is difficult for astronomers to go through the data individually for every star that is a candidate to host an exoplanet and look for something that might not even be here. Training a machine learning algorithm to identify the light curve created when an exoplanet transits across a star can expedite this process by discarding observations that clearly do not contain light curves, and flagging possible candidates for verification. Astronomers encounter challenges regularly when analyzing data to look for exoplanet transits. Sunspots passing across the surface of the star facing Earth would create a transit curve similar to a planet passing in front, creating a false positive identification. False positives also commonly occur due to an improperly trained, overfit, or underfit machine learning model. Also, stellar noise can mimic a transit, which a subpar machine learning model may misidentify as a planet. Many projects are ongoing that are working on these common challenges, in order to help bring machine learning to the forefront of astrophysical research.

3.1.1. Support Vector Machine

A Support Vector Machine (SVM) is a type of machine learning algorithm that is used for classification problems (Goodfellow et al. 2016). An SVM works by drawing a decision boundary to separate the classes it's looking for, and optimizing for the maximum possible margin of separation for the data. The margin is the distance between the decision boundary, which is a hyperplane, and the closest data points by perpendicular distance. The data points closest to the decision boundary hyperplane are called support vectors. We want to optimize the parameters of the algorithm to maximize the distance.

When we run our SVM algorithm to find possible exoplanetary models, the output is a confusion matrix. A confusion matrix is a way of categorizing our results into true positives (TP) and negatives (TN), and false positives (FP) and negatives (FN). True negatives represent correctly identified systems that do not have planets orbiting them, while a true positive represent correctly identified exoplanetary systems. False negatives are systems that were predicted to be an exoplanet hosts, but the algorithm classifies them as non-planetary systems, while a false positive is a predicted non-planetary system that gets classified as a planetary one. The outputs of all of the confusion matrices can be found in table 2.

In this assignment, we used the scikit-learn module (Pedregosa et al. 2011), with Linear, Polynomial, and Gaussian kernels. The hyperparameters were taken to be the default, except for the degree of vetting, which was set to 4. The results of the SVM analysis can be found in tables A.1 to A.2. Different types of kernels allow the data points to be distributed in spaces of different dimensions, since the data points are not always linearly separable. This feature of projecting the data into a higher dimension space where it is linearly separable is one of the features of SVM, and allows analysis of more complex data.

The metrics of evaluations are the accuracy, precision, recall, and f1-score (Pedregosa et al. 2011). Accuracy as the name says tells us is how accurate our algorithm is and is equal to the ratio of the trace of the confusion matrix and the sum of all of its terms, that is, $(\mathtt{TP}+\mathtt{TN})/(\mathtt{TP}+\mathtt{TN}+\mathtt{FP}+\mathtt{FN})$. The recall of a negative (false) or positive (true) represents the ability of the algorithm to find those negatives or positives. For positives, it is equal to $\mathtt{TP}/(\mathtt{TP}+\mathtt{FN})$, while for negatives it is $\mathtt{TN}/(\mathtt{TN}+\mathtt{FP})$. The precision of a positive is $\mathtt{TP}/(\mathtt{TP}+\mathtt{FP})$, representing how precise our classifier is at finding the TPs, and will be $\mathtt{TN}/(\mathtt{TN}+\mathtt{FN})$ for the precision of a negative. The f1-score for a positive is equal to $2\mathtt{TP}/(2\mathtt{TP}+\mathtt{FP}+\mathtt{FN})$, while for a negative it is equal to $2\mathtt{TN}/(2\mathtt{TN}+\mathtt{FN}+\mathtt{FP})$. For all of these metrics, the best value will be equal to 1 while the worst value will be equal to 0.

Looking at the result tables, we can see that the linear and Gaussian kernels were the most effective at recognizing planetary systems for the injected systems datasets. However, at the same time, they were not able to confirm a single true negative for the dev set, while the polynomial kernel was. Both the linear and Gaussian kernel had an accuracy of 0.55, which could be improved in the future with hyperparameter tuning. For the non-injected data sets it can be said that the Gaussian kernel was the most effective one for flagging the planetary systems, as it was the only one to have true positives.

3.1.2. Artificial Neural Network

Structure: ANNs are constructed by layers consisting of individual nodes, called neurons, which mimic biological neurons (Goodfellow et al. 2016). In biological neurons, incoming signals can be excitatory or inhibitory. The sum of all the incoming impulses "decides" whether the neuron is activated or not. The artificial neuron, called a perceptron, works similarly. Algorithms "within" the perceptron instruct the supervised learning of binary classifiers, or instructions, that are used to decide whether an input vector of numbers fits into a specific class.

ANNs such as the one we used have 3 types of layers. In the input layer, the data is taken in and assigned to neurons. As the information enters the next layer, called the hidden layer, each neuron multiplies the data assigned to it by a weighting vector. Each neuron in the hidden layer sums those multiplied values, to create a weighted sum called the activation. A bias term can be added to the activation as well. Next, the neuron applies an activation function to the activation. The activation function can be any nonlinear function, but is usually a sigmoidal function. This is the final step for a hidden layer. The hidden layer outputs "output unit activations" for each neuron, that become the inputs for the next hidden layer. There can be any number of hidden layers. The final layer is called the output layer. This is where the task is completed - for example, for our purposes, where the planet is identified.

In our ANN, the default model was:

```
model = tf.keras.models.Sequential(
    [
        tf.keras.layers.Input(shape),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(1, activation
                              ="relu"),
        tf.keras.layers.Dense(1, activation
                              ="sigmoid"),
    ]
)
```

The default model contained an Input layer to begin the model and accept the data into the proper format. The Flatten layer flattens the input, but does not change the size. The Dense layers are the hidden layers as described above. The number after Dense correlates to the number of neurons used - as seen below, we used one neuron. We needed to try adding additional hidden layers, see that they use the "relu" activation function, and to add Dropout layers following the Tensorflow tutorial (Abadi et al. 2015). Dropout layers receive input, and randomly set some values to zero. They take a given rate at which to zero input values. This is important to avoid overfitting, which essentially is when the model becomes overcomplicated and/or overly focused on details of the training set, and will not perform properly on new sets of data. By dropping values, the model will, for example, learn not to depend on a small number of specific features and instead depend on a larger and broader set of features that can be relied upon when some are dropped.

After consulting the assignment instructions and the Tensorflow tutorial, we attempted several versions of this model, with up to five hidden layers and up to 128 neurons in some versions. The most complicated versions with more layers performed quite poorly, as the dataset was not large or complicated enough to warrant such a complex network.

For the dataset with no injections, we analyzed it using the setup below.

```
def build_network(shape):
    model = tf.keras.models.Sequential(
        [
            tf.keras.layers.Input(shape),
            tf.keras.layers.Flatten(),
            tf.keras.layers.Dense(1, activation
                                  ="relu"),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.Dense(1, activation
                                  ="sigmoid"),
        ]
    )
```

This contains 2 hidden layers, and one Dropout layer with a rate of 0.2. Each hidden layer contains 1 neuron. The first hidden layer uses the "relu" (rectified linear unit) activation function. The second uses a sigmoid activation function.

In order to try a different configuration and test to see how it would perform, we analyzed the injected dataset with a model that had 128 neurons in the first hidden layer.

```
def build_network(shape):
    model = tf.keras.models.Sequential(
        [
            tf.keras.layers.Input(shape),
            tf.keras.layers.Flatten(),
            tf.keras.layers.Dense(128, activation
                                  ="relu"),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.Dense(1, activation
                                  ="sigmoid"),
        ]
    )
```

Training: This model uses the adaptive moment estimation (adam) optimizer. Optimizers in neural networks are important,

No injected data					
Dataset	True Negatives	True Positives	False Negatives	False Positives	Precision
Train Set	5047	37	0	3	0.925
Dev Set	562	5	0	3	0.625

Injected data					
Dataset	True Negatives	True Positives	False Negatives	False Positives	Precision
Train Set	1343	1405	1456	883	0.540
Dev Set	23	295	21	231	0.558

Table 3: The results for the ANN algorithms.

and used to minimize the loss function. They do this by taking into account the value of the loss function and changing the weighting vectors and bias terms accordingly. Adam is a gradient descent optimization algorithm, so it tries to find the local minimum of a function. It finds the gradient of the loss function, finds the part of steepest ascent, and moves in the opposite direction. In adaptive moment estimation, the algorithm computes adapted learning rates for each parameter in each neuron. The learning rate is the size of the step used in each gradient descent. It also uses a method called momentum, which saves computing power by storing a decaying average of past gradients and squared gradients so that the optimizer can make an informed guess about the next direction.

Loss functions are used to find the difference between the outputs of the network and actual results. A loss function for a model searching for exoplanet transits compares the events that it thinks might be exoplanet transits to a training set with known transits, and quantifies the difference between them. The loss function in our model is binary cross entropy. This loss function is "calculated from the negative value of the summation of the logarithm value of the probabilities of the predictions made by the machine learning algorithm against the total number of data samples" (Richmond Alake 2024). We use it to classify the data as either a transit, or not a transit.

```
loss_fn = tf.keras.losses.BinaryCrossentropy(
          from_logits=True)
model.compile(optimizer="adam", loss=loss_fn,
              metrics=["accuracy"])
return model
```

Since the model does not utilize backpropagation, part of the training process is epochs. One epoch is one run of the dataset through the network. The model is updated each epoch and the parameters improved. We used 50 epochs.

As seen in 3, the simpler ANN with only one neuron per hidden layer performed much better on the non-injected dataset than the more complicated 128 neuron ANN did on the injected dataset. We tested both models on both datasets, but retain only one set of results for each dataset for simplicity as the models' respective performance on one dataset vs. the other was very similar. The simpler ANN identified 3 false positives on the non-injected train and dev sets, with 92.5% precision on train and 62.5% precision on dev. It had no false negatives. The more complicated ANN on the injected dataset identified 883 false positives and 1456 false negatives on the train set, and 231 false positives and 21 false negatives on the dev set. The precision on the train set was 54%, and was 55.8% on the dev set. It is very important to note, of course, that the size of the datasets are different across dev and train sets, and between injected and non-injected, and so exact numbers of false positives and negatives should not be compared. However, it is clear still that the simpler ANN was much more reliably able to identify transits than the more complex model, which clearly experienced underfitting issues. This means the model was not able to properly identify the patterns in the data and could not properly identify transits. We still refer to this model as the "more complex" model, even though it underfit the data, because of the large number of neurons. Having a lot of neurons in the first hidden layer and only one in the second most likely contributed to this issue.

3.2. Results

Looking at the results for the injected and non-injected datasets, the SVM algorithm using the Gaussian and Linear kernels overall end up recognizing more exoplanetary systems than the ANNs. In the no injection data set, the simple ANN and the SVM perform similarly, which is to say very well. In the injected dataset, the SVM performs overall in all kernels better than the complex 128 neuron ANN, but struggles to identify true negatives. It is important to emphasize here that the results for the injected dataset from the ANN come from the worse-performing neural network model. The complex 128 neuron model underfit the data and could not produce good results. Therefore this report, the SVM has proven to be more effective in recognizing systems with planets. However, looking at tables 3 and 2, we can see that the simple ANN with better construction somewhat outperformed the SVM in the non-injected dataset, more readily identifying true negatives, and it is reasonable to extrapolate that the simple ANN would perform equally as well or better on the injected dataset.

4. Atmospheric study on exoplanet K2-18b

The transmission spectrum was captured using the JWST NIRSpec (near-infrared spectrograph) instrument, with a wavelength range of about 2.7 - 5.2 μm (Madhusudhan et al 2023). Data was retrieved from two detectors (NRS1 and NRS2), spanning 2.73–3.72 μm and 3.82–5.17 μm, resulting in a small gap in our spectrum. The authors concluded that the retrieval shows strong contributions of CH_4 and CO_2 in the spectrum, which was prominent in all three offsets. There was no great contribution from H_2O, NH_3, or CO. The mixing ratios found from literature (in $\log_{10}$) can be seen in table 4, where the different offsets are the authors's approach in Madhusudhan et al. (2023) to fix instrumental errors for the two detectors.

4.1. Model

The authors used the AURA retrieval code, with the chemical and P-T parameters as free parameters. Their model included opacity contributions from H_2O, CH_4, NH_3, HCN, CO, and CO_2, collision induced absorption from H2–H2 and H2–He,

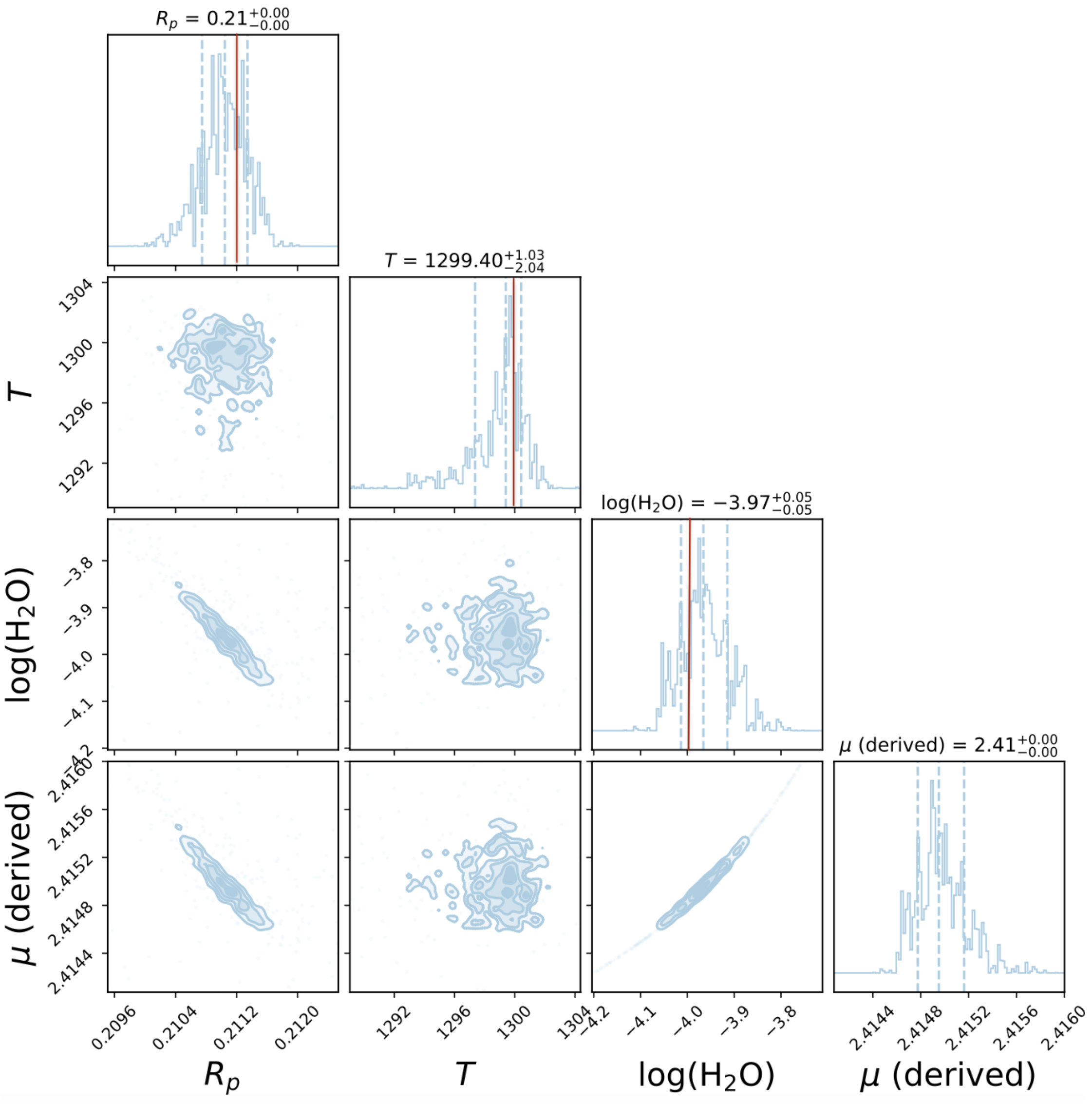

Fig. 1: Posterior from retrieval test

	CH_4	CO_2	H_2O	NH_3	CO
0 offset	$-2.04^{+0.61}_{-0.71}$	$-1.75^{+0.45}_{-1.03}$	<-3.21	<-4.46	<-3.00
1 offset	$-1.74^{+0.59}_{-0.69}$	$-2.09^{+0.51}_{-0.94}$	<-3.06	<-4.51	<-3.50
2 offset	$-1.89^{+0.63}_{-0.70}$	$-2.05^{+0.5}_{-0.84}$	<-3.49	<-4.93	<-3.19

Table 4: Mixing ratios found in literature, $\log_{10}$ values.

and pressure broadening for all molecules. Their model also includes inhomogeneous clouds/hazes with Rayleigh-like spectral contributions. They additionally considered five molecules that have been suggested to be promising biomarkers in habitable rocky exoplanets. These were $(CH_3)_2S$ (or DMS), CS_2, CH_3Cl, OCS, and N_2O.

4.2. Retrieval

Today there exists many programs for retrieval of exoplanet atmospheres, and they are constantly improving. In this assignment we used TauREx 3.1.14, which is a Bayesian inverse atmospheric retrieval framework. Overall TauREx consists of two main components, the forward model and the retrieval framework. The forward model allows you to easily create a model spectrum by mixing and matching parameters such as chemistry, pressure, temperature and so on. Then the retrieval framework fits the forward model to an observed spectrum. The only constraint on these spectra is that they must be of the same shape, which is easily fixed through binning. The interplay between the frameworks can be seen in the figure 2.

For the retrieval, TauREx has multiple optimizers that can be used. In this project we used Nestle, a Bayesian nested sampler Al-Refaie et al. (2021), as the other optimizers need additional Fortran libraries to be built/installed. To choose which parameters from the forward model you want to retrieve, you simply add them to the 'Fitting' class. This allows the optimizer to perform the nested sampling on these parameters. The more parameters you want to retrieve the more computationally heavy it will be as well.

As an initial test on our retrieval set up, we used the forward model to create a synthetic spectrum. Then we used this spec-

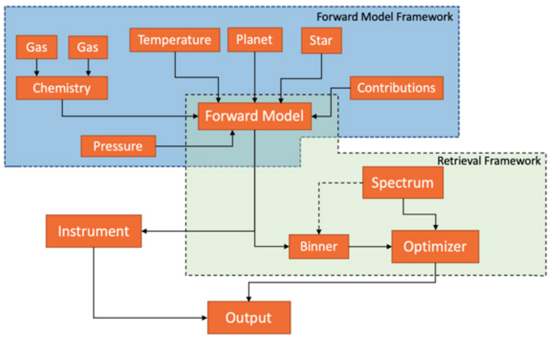

Fig. 2: TauREx setup

Parameter	Value
Temperature	300
H_2O opacity	1.118e-4
CH_4 opacity	1.18e-6
CO_2 opacity	5.30e-4
CO opacity	1.5e-4

Table 5: Fixed parameters used in forward retrieval

Fitted parameter	Bounds
Temperature	100, 500
Planet Radius	0.1, 0.5
H_2O	1e-6, 1e-1
CH_4	1e-6, 1e-1
CO_2	1e-6, 1e-1
CO	1e-6, 1e-1

Table 6: Fitting parameters used in retrieval

trum as our "observations" and ran a retrieval on it to see if we would retrieve the same parameters as we modeled it with. At this point we only retrieved H_2O, radius and temperature. In this test the retrieval did a good job for all parameters and are shown in the posterior plot in figure 1.

We used this same baseline of parameters, but adding retrieval for more molecules. Our model set up can be seen in the table 5, and we included transmission contributions from molecular absorption, collision induced absorption for H2-H2 and H2-He, and Rayleigh scattering.

For the retrieval of the observed spectrum we chose to bin down our observed spectrum to 471 data points, to make it easier for the model to fit to the curve. We did try for even larger numbers of data points, but it seemed harder for the model to fit and it became more flat. All the fitting parameters used for the retrieval can be seen in the table 6.

We were able to retrieve H_2O, CO_2, CO and CH_4 so that we can match it with the data from the paper. In the study we cite, they ran the retrieval for many more parameters. We can look at our retrieved parameters in the posterior distribution figure A.1, compared to the article's in the figure 4. Below is also a table comparing the exact amount of retrieved abundances (for no offset as an example).

Our posteriors are not as smooth, especially in the case of CH_4. Our best posterior distribution would be the one for CO_2. Looking at the mixing ratios retrieved, the amounts for CO_2 and CH_4 seem to coincide quite well with their values. Abundances in H_2O and CO are the biggest outliers for us, as they seem much more abundant in our model opposed to theirs. Basically all the

Molecule	Literature abundance	Our retrieved abundance
H_2O	<-3.21	$-1.61^{+0.41}_{-0.74}$
CO_2	$-1.75^{+0.45}_{-1.03}$	$-1.24^{+0.14}_{-0.68}$
CH_4	$-2.04^{+0.61}_{-0.72}$	$-1.74^{+0.48}_{-0.68}$
CO	<-3.00	$-1.52^{+0.32}_{-0.71}$

Table 7: Molecule abundances

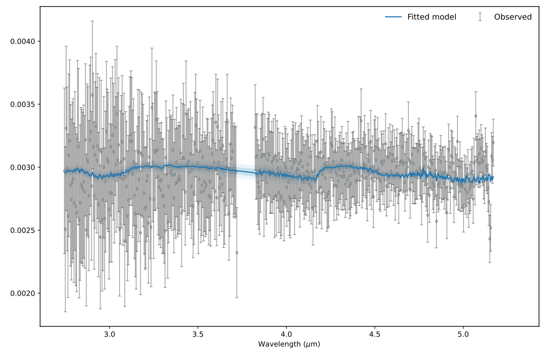

Fig. 3: TauREx fitted spectrum compared to observed spectrum

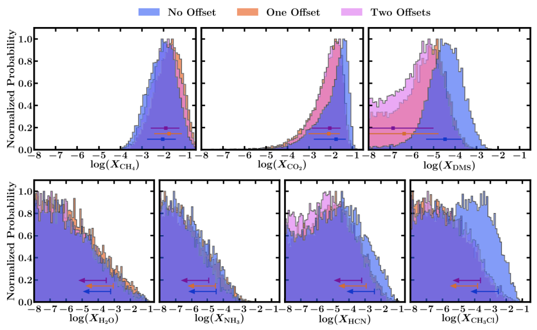

Fig. 4: Posteriors from literature

mixing ratios are of the same order. This might simply be a consequence of the model not really fitting the observations that well. If we look at our fitted spectrum in figure 3 compared to the observed spectrum it is clear it is too flat, not showing as much absorption as it should.

When we tried to run our retrieval before binning the observed spectrum (with around 3000 data points), this flatness was even more prominent. So it seems the signal to noise ratio is low, and we also see quite large errors in the observations which may be overfitted in our retrieval. The overestimation of H_2O might also be because our model does not take into account any clouds or hazes that there most likely is a lot of in its atmosphere. Therefore the atmospheric temperature-pressure profile also plays a big role in the observed spectrum. While we only used an isothermal profile for our model, we could have implemented, for example, the Guillot profile to better our model. The data quality is also a point we have not touched on. In the literature they have used algorithms to fix for possible star activity interfering with the observed transit curve. This is something we have not done for our data.

5. Conclusions

As we see from this project, the implementation of machine learning played a crucial part of both detecting exoplanets and analyzing the atmosphere them.

For the detection of exoplanetary systems, SVM proved to be more effective than ANN, however, both algorithms had problems detecting single star systems. For future analysis, we can improve on this by tuning our hyperparameters for SVMs, while for ANNs, we can use a simpler model on both sets of data, and trying different configurations to find the best model that works on a smaller dataset. For larger and more complex datasets, we could use models with more neurons, as well as more hidden layers.

Using the Bayesian Nested Sampling algorithm on our detected transit curve, we found that the mixing ratios of the fitted molecules are all in the same range of magnitude. Comparing with retrieved abundances found in Madhusudhan et al. (2023) the high abundances of CH_4 and CO_2 coincides well, while our amounts of CO and H_2O too high. Eventually this is caused by a bad fitting of the observed spectrum, causing absorptions of molecules to be missed. To better our chances at retrieving amounts closer to those of the literature we could firstly have improved the quality of our data, by accounting for star spots and not binning our spectrum as much as we did. In our model we could account for clouds/hazes and a more complex pressure parameters.

The K2-18b exoplanet is a very interesting subject of research, showing how important the implementation and improvements of machine learning tools are for the future of exoplanet studies.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Al-Refaie, A. F., Changeat, Q., Waldmann, I. P., and Tinetti, G. (2021). Taurex 3: A fast, dynamic, and extendable framework for retrievals. *The Astrophysical Journal*, 917(1):37.

EXOplanet (2024). Planet k2-18 b.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Kreidberg, L. (2015). batman: BAsic Transit Model cAlculatioN in Python. *PASP*, 127(957):1161.

Madhusudhan, N., Sarkar, S., Constantinou, S., Holmberg, M., Piette, A. A. A., and Moses, J. I. (2023). Carbon-bearing molecules in a possible hycean atmosphere. *The Astrophysical Journal Letters*, 956(1):L13.

Montet, B. T., Morton, T. D., Foreman-Mackey, D., Johnson, J. A., Hogg, D. W., Bowler, B. P., Latham, D. W., Bieryla, A., and Mann, A. W. (2015). Stellar and planetary properties of k2 campaign 1 candidates and validation of 17 planets, including a planet receiving earth-like insolation. *The Astrophysical Journal*, 809(1):25.

NASA (2024). Discoveries dashboard.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Richmond Alake (2024). Loss functions in machine learning explained.

Sarkis, P., Henning, T., Kürster, M., Trifonov, T., Zechmeister, M., Tal-Or, L., Anglada-Escudé, G., Hatzes, A. P., Lafarga, M., Dreizler, S., Ribas, I., Caballero, J. A., Reiners, A., Mallonn, M., Morales, J. C., Kaminski, A., Aceituno, J., Amado, P. J., Béjar, V. J. S., Hagen, H.-J., Jeffers, S., Quirrenbach, A., Launhardt, R., Marvin, C., and Montes, D. (2018). The carmenes search for exoplanets around m dwarfs: A low-mass planet in the temperate zone of the nearby k2-18. *The Astronomical Journal*, 155(6):257.

Appendix A: Tables and Figures

Linear Kernel: Train Set	Precision	Recall	F1-score	Support	Accuracy
False	1.00	1.00	1.00	5050	1.00
True	1.00	1.00	1.00	37	
Linear Kernel: Dev Set	Precision	Recall	F1-score	Support	Accuracy
False	0.99	1.00	1.00	565	0.99
True	0.00	0.00	0.00	5	
Polynomial Kernel: Train Set	Precession	Recall	F1-score	Support	Accuracy
False	1.00	1.00	1.00	5050	1.00
True	1.00	1.00	1.00	37	
Polynomial Kernel: Dev Set	Precision	Recall	F1-score	Support	Accuracy
False	0.99	0.99	0.99	565	0.98
True	0.00	0.00	0.00	5	
Gaussian Kernel: Train Set	Precision	Recall	F1-score	Support	Accuracy
False	1.00	1.00	1.00	5050	1.00
True	1.00	0.97	0.99	37	
Gaussian Kernel: Dev Set	Precision	Recall	F1-score	Support	Accuracy
False	0.99	1.00	1.00	565	0.99
True	0.00	0.00	0.00	5	

Table A.1: Classification report for the no injected data datasets.

Linear Kernel: Train Set	Precision	Recall	F1-score	Support	Accuracy
False	0.00	0.00	0.00	2226	0.56
True	0.56	1.00	0.72	2861	
Linear Kernel: Dev Set	Precision	Recall	F1-score	Support	Accuracy
False	0.00	0.00	0.00	254	0.55
True	0.55	1.00	0.71	316	
Polynomial Kernel: Train Set	Precision	Recall	F1-score	Support	Accuracy
False	0.66	0.04	0.08	2226	0.57
True	0.57	0.98	0.72	2861	
Polynomial Kernel: Dev Set	Precision	Recall	F1-score	Support	Accuracy
False	0.47	0.22	0.30	254	0.54
True	0.56	0.80	0.66	316	
Gaussian Kernel: Train Set	Precision	Recall	F1-score	Support	Accuracy
False	0.73	0.02	0.03	2226	0.57
True	0.57	1.00	0.72	2861	
Gaussian Kernel: Dev Set	Precision	Recall	F1-score	Support	Accuracy
False	0.00	0.00	0.00	254	0.55
True	0.55	1.00	0.71	316	

Table A.2: Classification report for the injected data.

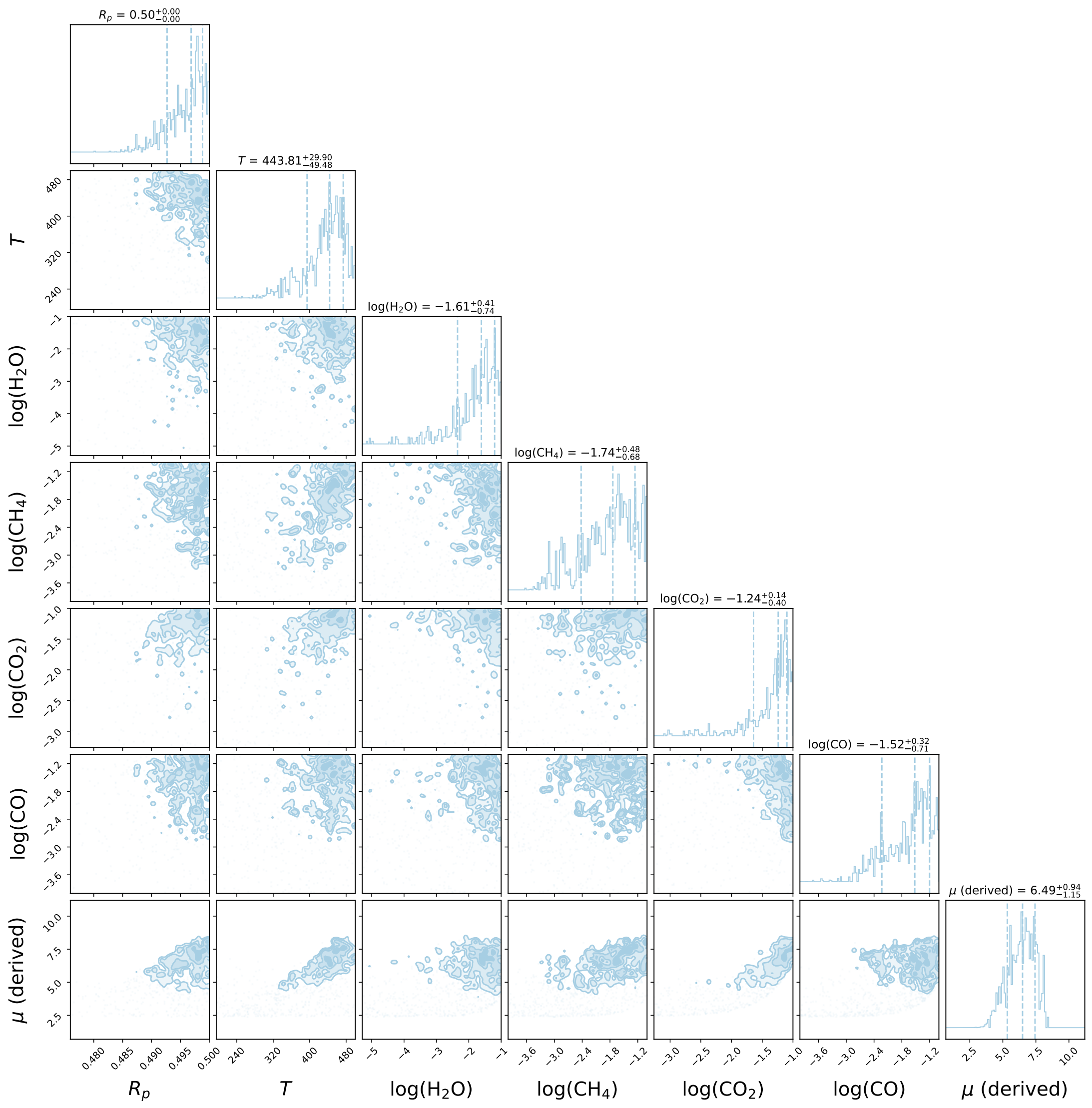

Fig. A.1: Posterior distribution of the retrieved abundances.