universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„From Code Optimization to AI-Driven Modeling: Improving the KOMPOT Code for Accelerated Upper Atmosphere Simulations“

verfasst von / submitted by

Axinya Tokareva, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2024 / Vienna, 2024

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 910
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Computational Science
Betreut von / Supervisor:	Univ.-Prof. Dipl.-Phys. Dr. Manuel Güdel

Acknowledgements

I would like to express my deep gratitude to my supervisor, Professor Manuel Güdel, for the opportunity to be part of this great project, for his understanding and for finding solutions to support me as much as possible in my difficult times, and for giving me so much time to write this thesis.
Special thanks also go to Ph.D. student Gwenaël Van Looveren for his time and co-supervision, including moral support, technical assistance and constant willingness to help, for the advice and ideas that have been incorporated into this project.
Thanks also to Assistant Professor Sudeshna Boro Saikia for taking the time to attend our many meetings and for her helpful comments.
I owe a huge debt of gratitude to Rosa Zimmermann, my best friend and soulmate, for her unwavering emotional support and proofreading of this thesis, even during her own stressful periods of life.
I would also like to thank my friend Maximilian Apel for his additional proofreading.
I would definitely not be here without my wonderful parents, Lidiya Tokareva and Igor Tokarev, who love me unconditionally and have always believed in me and supported me no matter what.
Last but not least, I would like to thank myself, once a little girl born in a small town in Kazakhstan, for having the courage to move here alone, knowing neither English nor German, for all my efforts, perseverance, and for never turning away from my dreams and goals.

Abstract

Introduction: The study and simulation of the upper layers of exoplanetary atmospheres are crucial for understanding planetary evolution and assessing their habitability potential, which is becoming expandingly relevant with the rapidly increasing number of exoplanets being discovered. Recent studies based on JWST exoplanet samples show that some of the most important processes in determining the dynamics of upper atmospheres are photochemical reactions. In contrast to many traditional tools, the 1D first-principles code KOMPOT is self-consistent because it is not limited in its ability to incorporate such complex interactions. Along with dynamic atmospheric escape phenomenon calculations, this code covers about 500 chemical reactions, including photoreactions, which has an expected negative impact on its computational efficiency. The goal of this master's thesis was **to optimize the KOMPOT code** without losing the accuracy of its simulations.

Methodology: In this thesis, **high performance computing** techniques were used, followed by the integration of **AI modeling** into the simulation process. **(1)** First, the code was optimized through code profiling (gprof), compiler optimizations, parallel computing (OpenMP), and the use of a fast scientific library (OpenBLAS) to replace manually programmed algorithms for solving linear equations. **(2)** Then, an innovative scalable and updatable approach was developed by using altitude-temperature atmospheric profiles predicted using dual cascade ensemble deep learning-based neural networks (NNs) as machine learning (ML) informed initial estimates for the KOMPOT code to find a whole physically consistent model with fewer iterations needed. A **data set** of 136 atmospheric models was generated using KOMPOT, covering almost all TRAPPIST-1 planets in terms of size, with initial standardizing values based on Earth-like parameters: the base of the atmosphere was 258.2 K, the corresponding number density value was $1.73077 \cdot 10^{-16}$ cm^{-3}, and the zenith angle at which the simulated segment lay was 66 degrees; the percentages of N_2 and CO_2 ranged from 1% to 99%. **(2.1)** The first **Attention-based NN** predicts the temperature and altitude of the Exobase, the conditional boundary where the probability that the atmospheric particles will fly away without further collisions becomes significant, bases on known initial values: *planet mass, orbital distance*, and CO_2 *mixing ratio.* **(2.2)** The second **Convolutional NN**, using the Exobase values predicted by the first NN in addition to the same initial values, predicts complete temperature profiles starting at 50 km.

Results: While maintaining the same computational accuracy, this approach **accelerated the code by a factor of 4.3** for the low irradiation case (similar to Earth) **and by a factor of 6.5** for the high irradiation case (similar to Mercury) compared to the baseline simulation on 4 threads; or by a factor of 12.1 and 18.3 compared to the baseline code on a 1 thread. Separately, the code optimization part made the code 3.63 times faster, or 10.21 times faster when compared to non-parallelized code. Both NNs achieved

high prediction accuracy ($\boldsymbol{R^2 \simeq 0.99}$, $RMSE \simeq 175$ for Exobase Temperatures and $RMSE \simeq 213$ for Exobase Altitudes, relative errors were within acceptable $\pm 10\%$) and computational efficiency ($1e$-3 minutes for either 1 model or 1000 models) and reduced the computation time when integrated into KOMPOT by a factor of 1.2 and 1.8 for low and high irradiation cases, respectively.

Conclusion: This thesis has shown how high performance computing and AI modeling can be optimally used, and has also confirmed the robustness of the targitized dual NN approach when working with a limited data set, which has significant implications for astroinformatics and exoplanetary atmospheric science. ML models can be used separately to predict the thermal profiles of Earth-like planets, which serve as an atmospheric baseline for subsequent predictions of chemical profiles. Attention-based NN for exobasic values predictions can also be used to identify patterns, including thresholds for the interaction between predictors and resultant exobase values. Accelerated and accurate KOMPOT simulations will be able to more rapidly reconstruct the planetary evolutionary history of the future space mission PLATO's exoplanet sample in 2026, allowing the calculation of optimal orbital distances of exoplanets that could contain, for example, liquid surface water and, importantly, be sufficiently capable of sustaining their atmospheres, which may help in the selection of targets for space missions such as Ariel in 2029. Further experiments and adjustments with more diverse and larger data sets are needed to bring us closer to being able to identify potentially habitable worlds.

Keywords: exoplanetary atmospheres, exoplanets, atmospheric simulation, atmospheric loss, ensemble learning, machine learning, deep learning, dual neural network, CNN, attention-based NN, neural networks, cascade learning, high performance computing, code optimization, code acceleration, AI modeling, KOMPOT, altitude-temperature profiles, thermal profiles, multiple regression, predictions, parallel computing, data science, data analysis, LU decomposition with partial pivoting, Gaussian elimination, TRAPPIST-1 planets, terrestrial planets, habitability.

Kurzfassung

Einleitung: Die Erforschung und Simulation der oberen Schichten exoplanetarer Atmosphären sind von entscheidender Bedeutung für das Verständnis der Planetenentwicklung und die Einschätzung ihres Habitatspotenzials, was mit der schnell wachsenden Zahl der entdeckten Exoplaneten immer aktueller wird. Jüngste Studien an Exoplanetenproben von JWST zeigen, dass einige der wichtigsten Prozesse, die die Dynamik der oberen Atmosphären bestimmen, photochemische Reaktionen sind. Herkömmliche Tools sind oft nur begrenzt in der Lage, diese komplexen Wechselwirkungen zu berücksichtigen, nicht aber der 1D-Erstprinzipien-Code KOMPOT, der daher auch selbstkonsistent ist. Zusammen mit dynamischen Berechnungen atmosphärischer Entweichungsphänomene deckt dieser Code etwa 500 chemische Reaktionen ab, einschließlich Photoreaktionen, was sich jedoch negativ auf seine Laufzeit auswirkt. Das Ziel dieser Masterarbeit war es, den **KOMPOT-Code zu optimieren**, ohne die Genauigkeit der Simulationen zu beeinträchtigen.

Methodik: In dieser Arbeit wurden **Techniken des Hochleistungsrechnens** verwendet, gefolgt von der Integration von **KI-Modellierung** in den Simulationsprozess. **(1)** Zunächst wurde der Code durch Code-Profiling (gprof), Compiler-Optimierungen, paralleles Rechnen (OpenMP) und die Verwendung einer schnellen wissenschaftlichen Bibliothek (OpenBLAS) optimiert, um manuell programmierte Algorithmen zur Lösung linearer Gleichungen zu ersetzen. **(2)** Anschließend wurde ein innovativer, skalierbarer und aktualisierbarer Ansatz entwickelt, indem Atmosphärenprofile mit Hilfe von dual cascade ensemble deep learning neuronalen Netzen (NNs) als über maschinelles Lernen (ML) informierte anfängliche Schätzungen für den KOMPOT-Code verwendet wurden, um ein vollständiges, physikalisch konsistentes Modell mit weniger erforderlichen Iterationen zu finden. Ein **Datensatz** von 136 Atmosphärenmodellen wurde mit KOMPOT generiert, die fast alle TRAPPIST-1 Planeten in Bezug auf die Größe abdecken, mit anfänglichen Standardisierungswerten, die auf erdähnlichen Parametern basieren: die Basis der Atmosphäre war $258.2\,K$, der entsprechende Wert der Teilchendichte war $1.73077 \cdot 10^{-16}\ cm^{-3}$, und der Zenitwinkel, unter dem das simulierte Segment lag, betrug 66 Grad; die Anteile von N_2 und CO_2 reichten von 1% bis 99%. **(2.1)** Das erste **Aufmerksamkeitsbasierte NN** sagt die Temperatur und Höhe der Exobase voraus, die bedingte Grenze, wo die Wahrscheinlichkeit, dass die atmosphärischen Teilchen ohne weitere Kollisionen wegfliegen, signifikant wird, basierend auf bekannten Anfangswerten: *Planetenmasse, Orbitalabstand*, und CO_2 *Mischungsverhältnis*. **(2.2)** Das zweite **Faltende NN (CNN)**, das die vom ersten NN vorhergesagten Exobase-Werte zusätzlich zu denselben Anfangswerten verwendet, sagt vollständige Temperaturprofile voraus, die bei $50\,km$ beginnen.

Ergebnisse: Unter Beibehaltung der gleichen Rechengenauigkeit **beschleunigte dieser Ansatz den Code um einen Faktor von 4.3** für den Fall geringer Bestrahlung

(ähnlich wie bei Erde) **und um einen Faktor von 6.5** für den Fall hoher Bestrahlung (ähnlich wie bei Merkur) im Vergleich zur Basissimulation auf 4 Threads; oder um einen Faktor von 12.1 und 18.3 im Vergleich zum Basiscode auf einem Thread. Die Code-Optimierung separat beschleunigte den Code um den Faktor 3.63 beziehungsweise 10.21 verglichen mit dem nicht parallelisierten Code. Beide NNs erreichten eine hohe Vorhersagegenauigkeit ($\boldsymbol{R^2 \simeq 0.99}$, $RMSE \simeq 175$ für Exobase-Temperaturen und $RMSE \simeq 213$ für Exobase-Höhen, die relativen Fehler lagen innerhalb akzeptabler Grenzen ($\pm 10\%$)) und eine hohe Recheneffizienz ($1e$-3 Minuten sowohl für ein Modell als auch für 1000 Modelle) und reduzierten die Berechnungszeit, wenn sie in KOMPOT integriert wurden, um einen Faktor von 1.2 und 1.8 für Fälle mit niedriger beziehungsweise hoher Bestrahlung.

Fazit: Diese Arbeit hat gezeigt, wie das Hochleistungsrechnen und die KI-Modellierung optimal genutzt werden können, und hat auch die Robustheit des zielgerichteten dualen NN-Ansatzes bei der Arbeit mit einem begrenzten Datensatz bestätigt, was erhebliche Auswirkungen auf die Astroinformatik und die exoplanetare Atmosphärenforschung hat. ML-Modelle können separat zur Vorhersage der thermischen Profile von erdähnlichen Planeten verwendet werden, die als atmosphärische Basis für nachfolgende Vorhersagen von chemischen Profilen dienen. Aufmerksamkeitsbasierte NN für Vorhersagen von Exobasiswerten können auch verwendet werden, um Muster zu erkennen, einschließlich Schwellenwerte für die Interaktion zwischen Prädiktoren und resultierenden Exobasiswerten. Beschleunigte und genaue KOMPOT-Simulationen werden in der Lage sein, die planetare Entwicklungsgeschichte der Exoplanetenprobe der zukünftigen Weltraummission PLATO im Jahr 2026 schneller zu rekonstruieren, was die Berechnung optimaler Orbitalabstände von Exoplaneten ermöglichen kann, die zum Beispiel flüssiges Oberflächenwasser enthalten könnten und — was auch wichtig ist — ausreichend in der Lage sind, ihre Atmosphären aufrechtzuerhalten, was bei der Auswahl von Zielen für Weltraummissionen wie Ariel im Jahr 2029 hilfreich sein kann. Weitere Experimente und Anpassungen mit vielfältigeren und größeren Datensätzen sind erforderlich, um uns der Identifizierung potenziell bewohnbarer Welten näher zu bringen.

Schlüsselwörter: Exoplanetare Atmosphären, Exoplaneten, atmosphärische Simulation, atmosphärischer Verlust, Ensemble-Lernen, maschinelles Lernen, Tiefes Lernen, duales neuronales Netz, CNN, aufmerksamkeitsbasiertes NN, neuronale Netze, Hochleistungsrechnen, Codeoptimierung, Codebeschleunigung, KI-Modellierung, KOMPOT, Höhen-Temperatur-Profile, thermische Profile, multiple Regression, Vorhersagen, paralleles Rechnen, Datenwissenschaft, Datenanalyse, LU-Zerlegung mit partiellem Pivoting, Gauß-Elimination, TRAPPIST-1-Planeten, terrestrische Planeten, Habitabilität.

Structure and Layout

This master's thesis describes the work done to accelerate the 1D first-principles code KOMPOT for the simulation of exoplanet upper atmospheres. The work can be divided into *two separate parts*: code optimization using high performance computing techniques and AI modeling of exoplanet atmospheric altitude-temperature profiles. These parts will be further connected by the integration of machine learning models into the simulation process of the optimized code.

The chapter *«Introduction»* gives a first overview of the study and simulation of exoplanet atmospheres and describes the main motivation and importance of this project. It also describes the features and workflow of the KOMPOT code as the research object in this thesis. The introduction also states the research question and hypothesis, while describing the scope of the project and the contribution proposed by the author of this thesis.

The next chapter, *«Related Work»*, provides an overview of the previous KOMPOT project on which this thesis is based, as well as other studies that used a similar methodology.

For the *«Methodology»* itself, together with the description of the rationale, methods, and data used in the AI modeling part, there is a separate chapter with the corresponding title.

The description of all optimization manipulations with the code takes place in the *«Code Optimization»* chapter, including profiling, the scientific library used, and parallelization.

The next chapter, *«AI-Driven Modeling and Its Integration into the KOMPOT Code»*, describes the creation of machine learning models for temperature profile predictions, including the selection of features, hyperparameters, training results, and a restart file capable of integrating the predictions into the simulation.

All results regarding computational and predictive performances, as well as related constraints, are presented in the chapter *«Findings and Discussion»*: first the results of the code optimizations, then the results of the predictive abilities of the machine learning models, and finally the results of the time reduction through the symbiosis of optimized code and machine learning predictions.

The final chapter *«Conclusion»* summarizes the results of this work, its limitations, and also gives recommendations for future work.

Additional figures can be found in the last chapter *«Appendix»*.

Contents

List of Tables

List of Figures

List of Algorithms

Listings

1. Introduction

1.1. Research Interest

Exoplanetary science is one of the fastest-growing fields in modern astronomy. With around 130 exoplanets discovered in the not-so-distant 2005 (Peplow, 2005), the number of exoplanets now stands at a staggering 7283[1], paving the way for the exploration of such remote, sometimes similar, but mostly very different to our own (Ballmer and Noack, 2021), worlds. Indeed, many of the confirmed detected planets, as shown in Figure 1.1, have parent stars' temperatures comparable to the Sun, but are mostly closer to their stars; even fewer exoplanets are similar in mass to our Earth.

Since one of the main goals of extrasolar planet research is to find habitable planets, the study of their atmospheres is justified as the only way to infer whether a given planet is likely to be habitable (Seager and Deming, 2010). Although the contribution of atmospheres to planetary mass is largely minimal, they play an important role in determining many planetary properties. These include energy balance, thermal structure and chemical composition. As such, atmospheres therefore act as windows into a planet's formative history and evolution; and questions about their temperature structure, chemical abundance or escape processes are just the tip of the iceberg of our interest. (Fortney, 2018; Deming and Seager, 2017)

The study and modeling of the intriguing physical and chemical atmospheric properties, encoded in its spectrum — itself represented by molecular emission and absorption lines and bands (Gargaud et al., 2023) — provides insight not only into the various atmospherical processes, as Madhusudhan (2019) described, but also into its accretion origins, evolutionary trajectory, interactions with solid materials (Min et al., 2020), and, as mentioned earlier, a planet's habitability potential (Johnstone et al., 2021; Zhang, 2020; Cockell et al., 2016).

Various methods have been developed to infer atmospheric properties, including (1) *direct imaging*, in which the light emitted or reflected by the planets is studied directly and distinguished in advance from the light of their host stars using a coronograph; (2) *transmission spectroscopy*, which analyzes the depth of absorption of stellar light by gases in a planet's atmosphere as it passes in front of its parent star (during primary transit); (3) *secondary eclipse observations*, which measure changes in the planet's light as it passes behind its parent star, temporarily blocking the planet's thermal emission and reflection; (4) *phase curve observations*, where such observations are made at each phase and throughout the entire orbital period; or (5) *high-resolution Doppler spectroscopy* (radial velocity), which measures shifts in the spectral lines of the host star due to the

[1] `https://exoplanet.eu/` as at 26.08.2024

orbital motion of a planet. Transit methods are still the most widely used methods for detecting atmospheres and planets themselves (see Figure 1.1 below). (Gressier, 2022; Madhusudhan, 2019; Matthews et al., 2024)

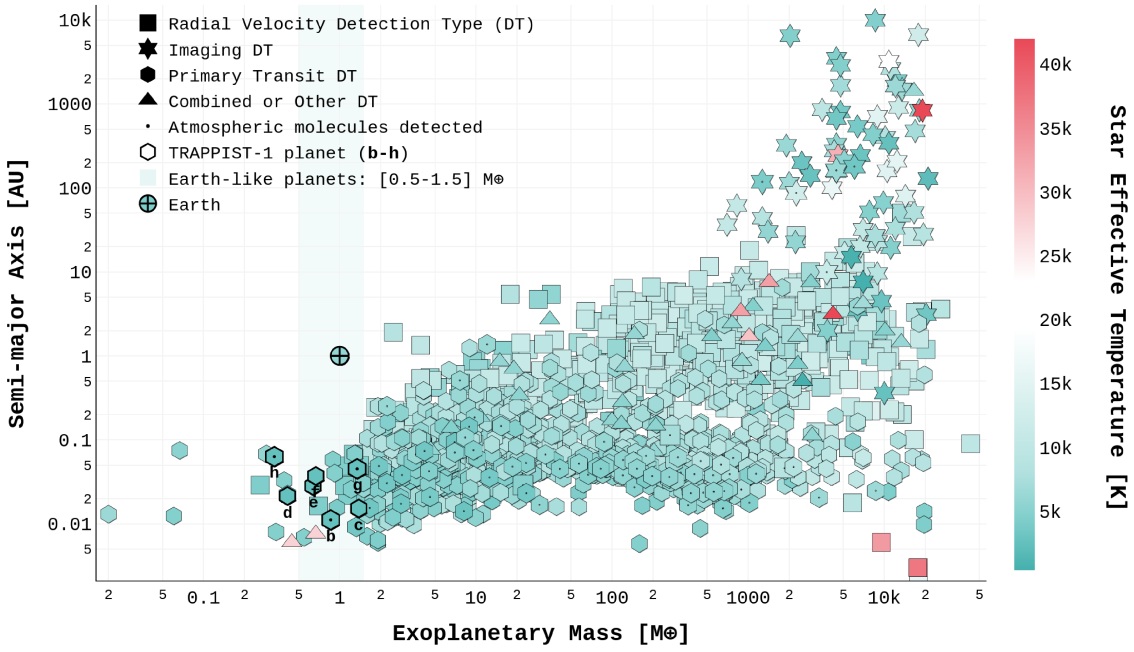

Figure 1.1.: **Stellar temperature, orbital characteristics and detection methods of diverse confirmed exoplanets with known effective temperatures of their host stars**: scatter plot of confirmed 4845 exoplanets, showing the relationship between an exoplanet's mass in Earth masses and its semi-major axis in AU. The main planet detection methods are marked with unique symbols. Planets with atmospheric molecules detected in practice are marked with a black dot in the center (except for upper limit detections). The highlighted area indicates Earth-like planets in the Earth mass range $[0.5 - 1.5]$. The color indicates the temperature of the planet's star in K. The planets of the TRAPPIST-1 system are highlighted separately by the last letters of their names. Data was obtained from Exoplanet Archives as at 05.05.2024: `https://exoplanet.eu/` and `https://research.iac.es/proyecto/exoatmospheres/`.

So far, scientists have managed to observe a considerable number of exoplanetary atmospheres (Figure 1.1), but it remains a challenging task: the problems of observing atmospheres are closely related to the limited observations of the exoplanets themselves. In order to make accurate inferences about atmospheric parameters, many of the above methods are combined, but even this does not guarantee the detection of all the signals we need. What can make it difficult to detect the light of a faint planet is the overwhelming brightness of its host star (Leconte et al., 2010), which can mask subtle spectral signatures that would be desirable to detect. Stellar activity can even mimic atmospheric features (Murgas et al., 2019), thereby falsifying molecular detections. The accuracy

of measurements is also affected by instrumental noise (Morello et al., 2022) and the interstellar medium, so observations require high accuracy and stability of the telescope. Even excessive cloudiness (aerosols) can partially or completely block the absorption spectrum of the planet (Deming and Seager, 2017). (Gressier, 2022)

The nascent field of exoplanetary atmospheric spectral characterization has recently undergone transformative developments with the influx of high-resolution data we have been getting from the James Webb Space Telescope (JWST) since 2022 due to its increased sensitivity over wider wavelength range (Gardner et al., 2023; Charnay and Drossart, 2023). Additionally, we expect to receive high-quality data from future missions such as PLATO (PLAnetary Transits and Oscillations of stars), focused on the study of Earth-like planets in the habitable zone of Sun-like stars (Rauer et al., 2021), and ARIEL (Atmospheric Remote-sensing Infrared Exoplanet Large-survey) (Pascale et al., 2018), focused on the study of the atmospheres of a wide variety of exoplanets, in 2026 and 2029, respectively.

However, even the modern JWST operates within the context of the faintness of many exoplanet signals due to the vast distances involved, the instrumental and astrophysical systematics, and its own detection limits (Mullin et al., 2024). This underscores the importance of indispensable numerical solutions that can both effectively complement and interpret sparse observational data and simulate the atmospheric conditions of various celestial bodies (Giobergia et al., 2023; Komacek et al., 2021; Madhusudhan, 2018).

For instance, theoretical solutions make it possible to trace the transport of trace gases back to their sources, setting a period or upper limit during which the observations in question can still be associated with their source, which is important for understanding random and scattered measurements, since direct observations may be limited, for example, due to certain orbital constraints (as is the case with the ExoMars Trace Gas Orbiter by Rajendran et al. (2019)).

Through theoretical frameworks, we can study different atmospheric conditions (temperature profiles, chemical compositions) (Min et al., 2020) and changes in their dynamics (Biasiotti et al., 2022), as well as the stability of planetary systems, including orbital stability[2] (Hinse et al., 2008), even for planets that cannot yet be fully studied. These simulations help to identify the most promising planets and to determine other important parameters, such as the time it would take for telescopes to detect, for example, water vapor on a planet by calculating the depth of its spectral features, as Suissa et al. (2020) demonstrates — all of this information is crucial in preparing a list of targets in which each planet deserves closer attention from space telescopes. (Gandhi and Madhusudhan, 2017; Wolf et al., 2019; Fortenbach and Dressing, 2020)

These models play also an important role in determining the so-called *habitable zones* around the stars in question because through modeling we can obtain information about the likelihood of atmospheric retention and the possibility of water persistence by answering the question: «How do different atmospheric compositions respond to different levels of stellar radiation?». (Kodama et al., 2019)

Undoubtedly, the synergy of rapidly improving observational capabilities and accurate

[2]The stable planetary systems are more likely to have long-lived terrestrial planets of interest for further study.

numerical models will profoundly refine our understanding of planetary atmospheres, their formation and evolution, and hopefully open the possibility of discovering signatures of habitable or even inhabited worlds. (Charnay and Drossart, 2023; Rengel and Adamczewski, 2023)

1.2. State of The Art

Historically, the tools used to model exoplanetary atmospheres have their roots in those used for solar system planets, cool stellar atmospheres, or brown dwarfs (Fortney, 2018). A continuum of models, ranging from simplified 1D models representing the mean vertical structure of a planet (Gargaud et al., 2023) to complex 3D Global Climate Models (GCMs), as the most comprehensive representation modeling global circulation patterns and cloud formations (Forget et al., 2013), capture the core physical and chemical processes that govern planetary atmospheres (Madhusudhan et al., 2016). In between are 2D models of intermediate complexity that include some horizontal variability (for example, zonal wind models) (Langton and Laughlin, 2008). The choice of model dimension depends on the research question and the computational resources available.

Beyond dimensionality, the models can be categorized by their target purpose. *Self-consistent models*, also known as forward models, predict spectra of exoplanetary atmospheres based on assumed parameters (such as irradiation, and elemental abundances) to explore the potential diversity of atmospheres and to interpret observations within theoretical expectations. *Retrieval*, or inverse models, are used to model atmospheric properties based on the observed spectrum. Typically, these are forward models with parameter estimation algorithms (Bayesian statistical methods as an example) to quantify uncertainties and explore the range of possible properties given spectral data. The remaining models include various disequilibrium processes (photochemistry, atmospheric escape, clouds and others), so-called *disequilibrium chemistry models*. These models allow each individual process to be studied in detail, simplifying interactions with other processes, since a model that includes all possible atmospheric processes is currently impractical. (Madhusudhan, 2019, 2018; Fortney, 2018)

Although many tools focus on the lower layers of exoplanet atmospheres (troposphere and stratosphere), modeling the *upper atmosphere* (mesosphere, thermosphere and exosphere (see Figure 1.2)) is critical for understanding atmospheric composition and dynamics (Koskinen et al., 2011), magnetospheric interactions, effects of stellar

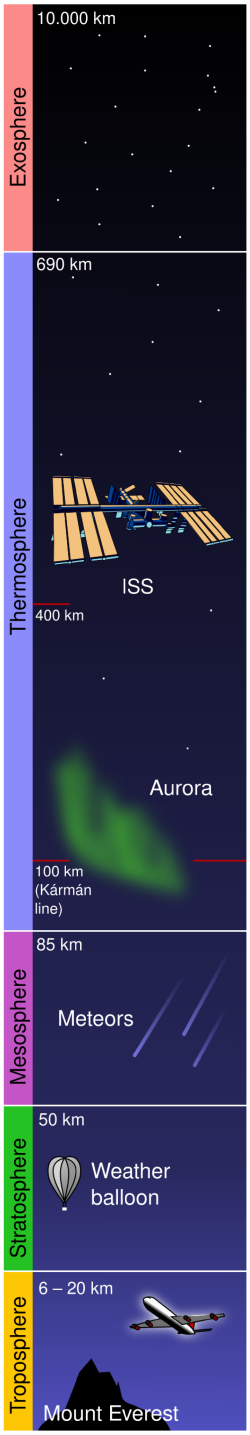

Figure 1.2.: **Layers of Earth's atmosphere** (NOAA & Mysid, 2014)

wind and extreme ultraviolet radiation (XUV) (Kislyakova et al., 2020), escape processes and their impact on planetary evolution (Johnstone, 2018; Johnstone et al., 2018; Owen and Wu, 2016; Owen, 2019; Kalinicheva et al., 2021). Each of these processes contributes to determining the potential for having and sustaining life as we know it. (Cockell et al., 2016; Buccino et al., 2006; Lammer et al., 2008)

The following examples illustrate the aforementioned models:

THOR: 3D Global Circulation Model THOR is used to study the dynamics of lower as well as upper atmospheres without assuming hydrostatic equilibrium, including wave interactions, cloud formation processes and detailed atmospheric chemistry. (Mendonça et al., 2016)

WACCM6: The Whole Atmosphere Community Climate Model Version 6 studies the Earth's atmosphere up to the exosphere in order to expand our understanding of its chemical structure, the effects of solar radiation and long-term climate variability. (Gettelman et al., 2019)

3D fully self-consistent model: The 3D Global Multi-species Hydrodynamic Aeronomy model of the hot Jupiter HD189733b, which can be adapted to studies of other hot Jupiters, takes into account the effects of high-energy stellar flux, stellar wind and cooling processes. (Rumenskikh et al., 2022)

VULCAN: A more simplified 2D VULCAN model is used to model photochemical transport in the upper atmospheres of hot Jupiters, which takes into account vertical and horizontal dynamics to depict chemical heterogeneity (it can be day-night transitions), making the model particularly relevant for tidally locked exoplanets. (Tsai et al., 2024)

PICASO 3.0: It is a 1D radiative-convective equilibrium (climate) model with a focus on giant planets and brown dwarfs. It includes radiative transfer, equilibrium and disequilibrium chemistry and convective mixing processes occurring in upper atmospheres. [(Mukherjee et al., 2023)

KOMPOT: The next tool is the KOMPOT code, which is used for 1D first-principles modeling of the various exoplanetary upper atmospheres. Unlike many other atmospheric modeling codes, which tend to focus more narrowly on radiative transfer or chemical equilibrium, KOMPOT offers a holistic approach by including the dynamic atmospheric escape phenomena along with thermal, chemical dynamics and hydrodynamic processes (for more details, see the following Section 1.3). (Johnstone, 2018; Johnstone et al., 2018)

Despite the advantages of existing numerical tools, each year it becomes more difficult to keep up with the ever-increasing volume of data from telescopes and space missions, even for 1D models — the field of astronomical data science, which integrates astronomical observations, statistical methods and data science techniques, is growing rapidly. The

increasing complexity of the data requires advanced methodologies to deal with it: more sophisticated and faster methods for analysis / interpretation and simulations (Siemiginowska et al., 2019). The authors of the following studies agree that machine learning is a promising candidate for solving these problems: Baron (2019); Meher and Panda (2021); Ivezić et al. (2020).

1.3. Research Object — The KOMPOT Code[3]

The *research object* of this thesis is *the KOMPOT code*. The KOMPOT framework has been developed by Johnstone et al. (2018) to better understand the evolution of planetary atmospheres and their escape into space based on modeling and, subsequently, on understanding the physical structure of these atmospheres under different conditions. In particular, the code can provide insight into how different planetary parameters, chemical compositions, and stellar inputs affect the thermal and chemical structure of planetary upper atmospheres.

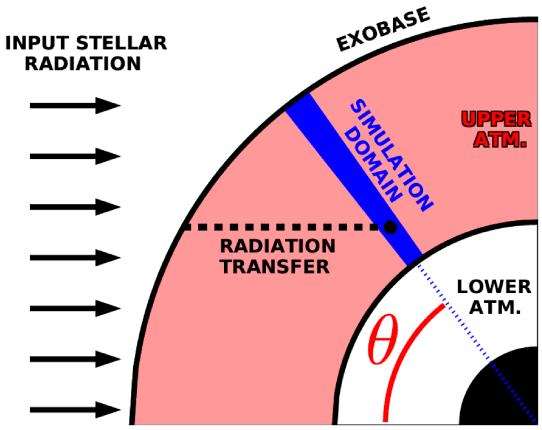

Figure 1.3.: **Visualization of the KOMPOT simulation domain** (Johnstone et al., 2018)

The code produces simulations (or models) — simulation refers to the calculation of atmospheric properties — in 1D, focusing on producing vertical profiles of these properties to analyze their variation with altitude. As illustrated in Figure 1.3, the simulation domain, directed radially outwards from the center of the planet under study, can be located anywhere from the middle atmosphere to the exobase itself, and at any angle relative to the incident radiation rays.

The main features of KOMPOT

- It is based on *first-principles physics*, providing an accurate representation of atmospheric processes.
- It is *flexible and adaptable*, allowing it to be applied to a wide range of systems with different planetary parameters, chemical compositions, and stellar inputs, and allowing easy modification of the underlying physics directly in the code.
- It includes a *wide range of physical processes*.

[3]This entire chapter is based on the following works: Johnstone (2018); Johnstone et al. (2018).

The key components of the physical processes used (Johnstone et al., 2018)

- For *thermal structure* calculations:
  - Heating mechanisms: Heating from absorption of stellar X-ray, ultraviolet (UV), and infrared radiation (IR), as well as heating from exothermic chemical reactions, collisions of electrons with non-thermal photoelectrons, and Joule heating.
  - Cooling processes: Infrared radiation from various species in the atmosphere such as CO_2, NO, O and H_2O.
  - Energy exchange: Heat conduction and energy exchange between neutral, ionic, and electron gases.
- For *chemical structure* calculations:
  - Chemical reactions: 64 species with 30 ionic species among them and 503 chemical reactions (including 56 photoreactions and 7 photoelectron reactions).
  - Eddy and molecular diffusion.
  - Advection.
- And also *hydrodynamic equations*, describing the dynamics of the atmosphere in response to various heating, cooling and energy exchange processes.

Code setup and functionality overview

First, the user must specify the *main initial parameters* of the planet (mass, radius, its orbital distance), as well as its stellar parameters (radius and effective temperature of the central star, stellar XUV) and atmospheric properties (the initial mixing ratios of the gases in the atmosphere, base temperature and number density of particles). All parameters related to the inclusion of reactions, the size of the model grid (series of stacked horizontal slices, each representing a vertical atmospheric layer), the zenith angle and other technical properties can also be specified.

As described earlier, the code divides the atmosphere into several one-dimensional vertical layers or cells from a user-defined point in the middle atmosphere to the exobase. The thickness of the cells is dynamically adjusted to match the expansion of the grid. That provides higher resolution near the base, where detailed phenomena are important, and lower resolution at higher altitudes. Each layer — all of which are treated as a finite volume — is assigned certain initial conditions (values) based on its altitude, including temperature, pressure, and chemical composition, reflecting the inherent processes of the planet in question at a given distance from the star. Layers are coupled by the vertical transport of gases, heat and momentum. The spatial resolution of the grid, namely the number of layers and the stretch factor for the thickness adjustment, is crucial: the higher the resolution, the more detailed, but also the more time-consuming the modeling will be.

At the beginning, the model updates the absorption *stellar XUV and IR radiation* for each layer, determining how much energy a layer absorbs from the stellar flux and how it

redistributes that energy within the atmosphere. Combined with non-thermal electron spectra calculations, this affects the thermal structure from one layer to the next.

Hydrodynamics is then modeled by simulating the bulk movement of gases that can redistribute heat and atmospheric chemistry.

Afterward, the concentrations of chemical species as individual parts of the gas in each layer are updated using prescribed reactions and diffusion processes to create dynamic *chemical changes* that are strongly dependent on the vertical thermal and radiation profile.

To determine the *temperature changes*, various heating effects such as stellar absorption, chemical reaction heat and Joule heating are also applied. At the same time, cooling effects due to IR emissions from key molecules (such as CO_2, H_2O) are calculated.
To ensure that the temperature gradients along the vertical atmospheric column are adjusted, the conductive heat transfer between neighboring layers is evaluated and modeled, taking into account the energy exchange processes for neutrals, ions, and electrons into which the gas has also been broken.

At the *exobase*, it calculates the conditions under which particles might escape the gravitational pull of a planet, based on the results of updated thermal and chemical properties of the upper layers.

Throughout the whole process, the mechanisms mentioned above are iterated, and *the atmospheric properties* of the layers (as well as their adjacent layers) are *periodically adjusted* based on information from previous cycles: physical and chemical processes occurring within and between layers.

At the end of the simulation, when a steady state is reached or it is manually stopped, the model outputs detailed temperature profiles, chemical composition, reaction rates and atmospheric escape characteristics in all layers, ready for potential analysis and comparison with observations. An example of such an output can be seen in Figure 1.4. Lastly, a *restart file* is created that can be used to continue a simulation that was previously run.

Exobase determination

One of the key features of the code is the detailed and dynamically varying process of exobase determination based on real-time simulation data, allowing more accurate modeling of escape processes under changing conditions. This allows us to study planets around different types of stars where traditional models may not be effective.

As described in Johnstone et al. (2018), the *exobase*, or **critical level**, in this code is the boundary where the mean free path of atmospheric particles becomes comparable to the scale height of the atmosphere. In other words, when the probability that the particle will fly away without further collisions becomes significant, signaling a transition to space.

Formally, this looks like this:

$$\frac{1}{\sigma_c n} = \frac{k_B T}{mg}, \tag{1.1}$$

where the left-hand side represents the mean free path (λ) and the right-hand side represents the scale height (H). The collision cross-section of a molecule — σ_c, n — the

number density of all atmospheric particles at an exobase distance, k_B — the Boltzmann constant, T — the atmospheric temperature, g — the gravitational acceleration, and m — the average mass of the molecules.

The atmospheric mass-loss is mainly explained by thermal processes such as **Jeans escape** since fewer assumptions are needed to estimate the lower limit of the escape loss rate before it becomes fully hydrodynamic (likely dominated by non-thermal processes). The critical part of the calculation is to determine the thermal velocities of the particles at the exobase, assuming that they are constantly moving in random directions, and their ability to overcome the gravitation[4]. (Van Looveren et al., 2024; Johnstone et al., 2018; Catling and Kasting, 2017)

If the Jeans coefficient is low, the kinetic energy of the particles at the exobase (*exo*) is high relative to the gravitational potential energy ($\Leftrightarrow v_{\text{thermal}} > v_{\text{escape}}$), what indicates an increased escape rate:

$$\lambda_{J,exo} \equiv \frac{G_{const} M_{planet} m_{mass\,of\,a\,species}}{r_{exo\,radius} k_B T_{exo}} \equiv \frac{r_{exo}}{H_{exo}} \equiv \frac{v_{escape}^2}{v_{thermal}^2} < 1 \tag{1.2}$$

The calculation of this parameter is necessary to predict how XUV affects the escape and evolution of atmospheres to clarify how stable they might be against future changes. (Catling and Kasting, 2017)

Limitations

Notwithstanding its advantages, the code has limitations in terms of **computational efficiency**. The complexity of the atmospheric models simulated by the KOMPOT code, due to the inclusion of a large number of physical processes and the desired accurate results, expectedly uses many computational resources and negatively affects the simulation time. This is why improving the performance of the code is a key point for future development to enable more extensive and detailed atmospheric studies.

1.4. Research Question, Hypothesis and Scope

Research Question

Based on the problem discussed above, the following research question was formulated:

«How might computational optimizations and machine learning techniques affect the efficiency of the KOMPOT atmospheric simulation code?»

[4]Jeans parameter as an indicator of the escape probability

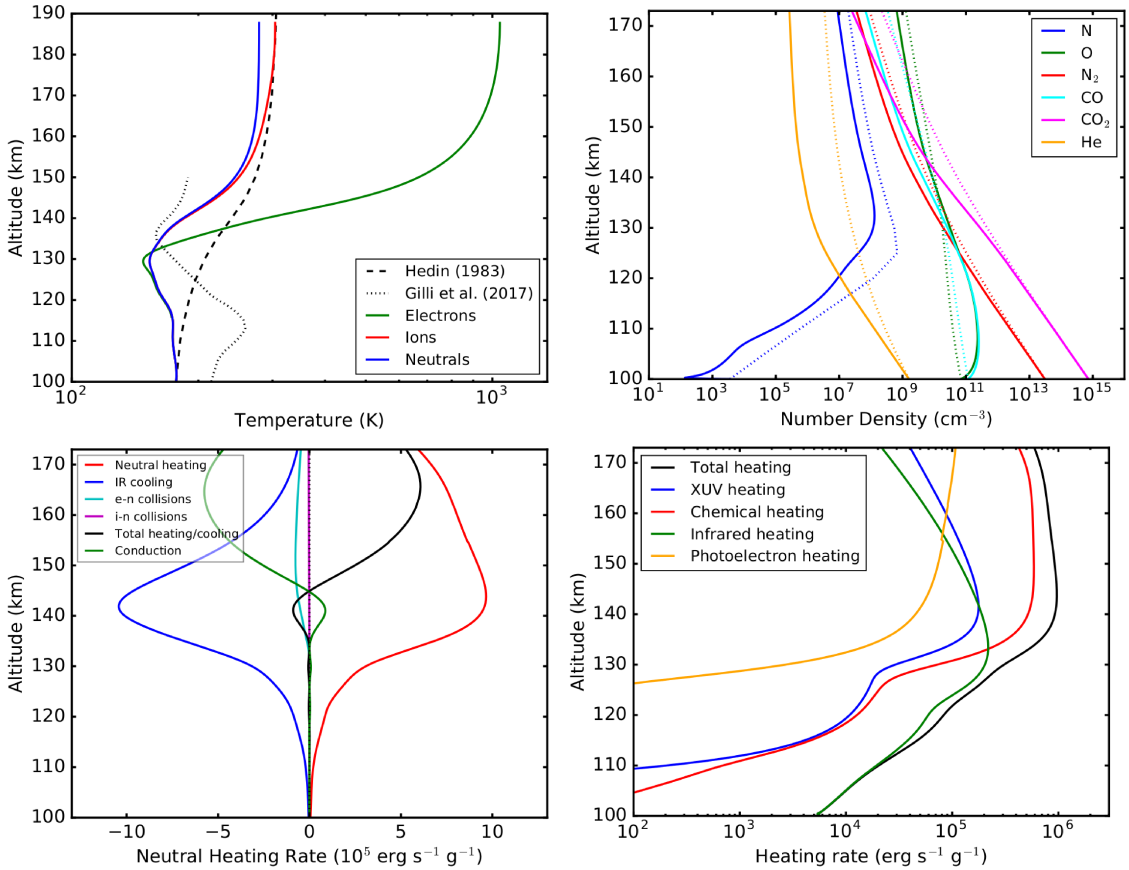

Figure 1.4.: **KOMPOT's example output: Venus model.** These figures are provided by Johnstone et al. (2018), which compares the KOMPOT output with results from other models, including the Hedin et al. (1983)'s empirical model and the 3D Venus model from Gilli et al. (2017). In the upper left corner are temperature profiles, and on the right are density profiles of selected relevant species. At the bottom, the cooling and heating processes of the neutral gas are shown first, and all heating processes are shown second.

Research Hypothesis

Furthermore, the following hypothesis was formulated:

«The integration of computational optimizations and machine learning techniques are hypothesized to significantly improve the efficiency of the KOMPOT code».

In particular, it is proposed that the use of initially generated KOMPOT simulation results with variations in input parameters as training data for the machine learning model will enable fast prediction of changes in atmospheric conditions (for instance, temperature profiles, chemical composition).

The predictions, converted into a KOMPOT restart file, can serve as the input base

for running the KOMPOT simulation, creating a *feedback loop* that iteratively corrects inaccuracies in the prediction results. Restarting the simulation in this way will also improve the efficiency of the simulation itself by significantly reducing the number of code iterations needed to achieve accurate results. Because it is also expected that such a completed KOMPOT simulation can be used again as a training data set for re-training the model to improve its predictive ability, this approach aims to be scalable.

Scope

This project focuses on code optimization, feature engineering and subsequent prediction of temperature profiles using deep learning techniques. The temperature profiles, as fundamental properties in atmospheric analysis, can tell a lot about atmospheric stability, climate (Wang et al., 2023); information about which can be used to further predict the atmospheric species behavior (Badhan et al., 2019), and to develop a complete synthesized model between code and machine learning for this work.

Considering the limited data available for machine learning tasks, the criterion of success will not only be the accuracy of the prediction of temperature profiles but rather the development of a methodology, a universal concept, that will lead to accurate and fast modeling as the amount of data on simulation models increases in the future.

The combination of the KOMPOT code and AI will be the fundamental basis for modeling the upper atmospheres of a large number of model templates for future observations. This can make a significant contribution to astroinformatics and to the mission of studying the diversity of exoplanet atmospheres.

1.5. Proposed Contribution

The author's proposed contribution lies in the following three aspects:

1. **Improved Code Efficiency**
   This study aims to significantly reduce the execution times of the KOMPOT code, making the simulations of atmosphere characteristics faster accessible to researchers.

2. **Command Line Interface (CLI) Tool for Predictions and Generating a Restart File for the KOMPOT**
   The next goal of this study is to create a comprehensible tool based on the ML models developed by the author to predict the temperature profiles' values, making the process easier and more automated.

   Expected automated features include (1) quick access to information about the ML models and data used for training; (2) prediction of only two exobase (last) values, altitude and temperature, based on the input file, and / or (3) prediction of complete temperature profiles and their visualization; (4) creation of a file for restarting the KOMPOT simulation based on previous predictions; (5) using a data analysis approach, exploration of temperatures and input planetary data from newly

added KOMPOT simulations, or (6) creation of predictions and comparison with these added simulations.

3. **Methodological Framework as a Jupyter Notebook for Future Research** This project also aims to create a Jupyter notebook that will not only allow one to see in detail the analysis of the training data and the hyperparameter selection process for ML models but, more importantly, it can be used to reanalyze and search for hyperparameters and retrain models when more KOMPOT atmosphere simulations are available, making this approach scalable and strengthening its future use.

2. Related Work

The master's thesis of Rosenberger (2021) forms the basis of comparison for the present study, in which he focused on the temperature profiles parameterization of terrestrial planets by developing a Multilayer Perceptron Neural Network (MLP NN) for their predictions (from the 50 km base to the top of the simulation domain at 500 km, represented by 104 temperature values at different altitudes). This neural network was trained and tested on more than 1000 atmospheric profiles generated by the KOMPOT code, which was simplified to improve simulation performance. Simplification was achieved by reducing chemical complexity, assuming a hydrostatic atmosphere, simplifying the XUV radiation transport by treating all photons as having the same energy, and focusing only on CO_2 for cooling processes, with fixed molecular gas mass and atomic oxygen mixing ratio as constants throughout the simulation.
Profiles were varied for planet size, CO_2 abundance and stellar XUV flux, with values chosen randomly within predetermined limits:

- planet mass: $0.4\,M_\oplus$ to $10\,M_\oplus$
- CO_2 mixing ratio: 10^{-4} to 10^{-3}
- stellar XUV flux: 1 to 100 $\mathrm{erg\,s^{-1}\,cm^{-2}}$.

These 3 initial parameters were chosen based on the paper of Johnstone (2021), where their significant influence on thermal and chemical processes in the upper atmosphere was argued. The developed models were evaluated using the cross-validation method and MSE as a quality estimator. The result of Rosenberger's work on accurate prediction of temperature profiles clearly demonstrates the great potential of using machine learning along with, unfortunately, mostly very time-consuming numerical models.

One of the *major differences* in the approach of this work is that the original, not-simplified code is used to generate the data set. Although the simplified code approach is much faster, it can lead to missing critical atmospheric processes and interactions, which, in turn, can negatively affect the simulation accuracy of temperature profiles and dynamics. However, using the full code has its consequences: a single simulation can take 10 or more hours, even days, depending on the input planetary and atmospheric parameters. Due to time constraints, only 136 profiles are used, which is expected to significantly affect the ability of the model to generalize well. The choice of input parameters is entirely based on feature engineering. The remaining details can be found in the following Methodology section.

The exoplanet atmospheric science has already made significant advances by integrating machine learning techniques to retrieve and predict different atmospheric properties, each with unique contributions.

Examples include the study published by Gebhard et al. (2024), which used Convolutional Neural Networks, CNNs, *as in this work*, and MLPs to retrieve pressure-temperature profiles. One of the data sets was the synthetic PyATMOS data set, consisting of 124,314 profiles of Earth-like planets, covering altitudes up to 80 km (Chopra et al., 2023); the other was a data set of 11,293 profiles for 89 hot Jupiters with various atmospheres simulated by Goyal et al. (2020). First, the authors ran the profiles through a CNN encoder (only during training) with MLP at the end, which produced a vector representation of the profiles. They then used a simpler decoder with MLP (during training and retrieval) to predict final pressure-temperature values based on the profile representations, and trained these two networks simultaneously. The new combined approach with fewer input parameters demonstrated 3.2-fold performance and high retrieval accuracy, outperforming basic methods on average — providing a computationally efficient alternative to traditional retrieval methods with comparable accuracy.

Furthermore, studies on Earth's pressure-temperature profiles were considered relevant and reviewed as well. But it should be noted that all of the works described below use large empirical data sets, *in contrast to the limited data set used in this master's thesis.*

In the study by Malmgren-Hansen et al. (2019) deep CNNs were used to process Earth's spectral data from the Infrared Atmospheric Sounding Interferometer on the MetOp satellites. Their model successfully predicts atmospheric temperature and moisture profiles at 90 different altitudes in the troposphere and stratosphere (inferred from the provided plots) using as input spectral data reduced from the original 8461 spectral channels to 125 features using Minimum Noise Fraction transformation. The authors achieved $+32 - 34\%$ improved prediction accuracy through deep learning compared to linear regression models. This demonstrates the high potential of CNN applications in atmospheric modeling.

In another paper conducted by Shi (2001), the authors used Backpropagation NN (BPNN) trained on data obtained with the NOAA-15 Advanced Microwave Sounding Unit AMSU-A and processed into brightness temperatures[1] to retrieve profiles. The outputs included temperatures at 26 pressure levels from the surface to 10 hPa, surface temperature, and tropopause values (temperature, altitude and pressure). Unlike a linear regression approach, their NN retrieval approach showed outstandingly better results at all atmospheric layers up to the lower stratosphere.

Similarly, and much later, Wang et al. (2021) used BPNN to retrieve temperature profiles from Earth's hyperspectral microwave data, focusing on 50∼70 GHz brightness temperatures. They used 10 principal components from the Principal Component Analysis of 2000 channel temperature values as input data to predict profiles at 99 altitudes, achieving a higher predictive ability in the stratosphere than the statistical inversion approach.

Using a machine learning approach similar to this thesis, Tan et al. (2024)'s study combines CNN and attention mechanisms to improve the retrieval of atmospheric pressure-temperature profiles using a data set sourced from the SeeborV5 atmospheric profile database and the GRAPES database. By calculating and focusing on brightness temperatures, based on information content near 60 GHz, 118 GHz and 425 GHz, as inputs,

[1] radiance intensity indicators

the model achieves especially enhanced predictions at the middle and upper layers in the mesosphere. The integration of the local attention mechanism significantly improved the efficiency and accuracy of the model, compared to separate 1D-CNN and Attention, BPNN, eXtreme Gradient Boosting, and Support Vector Machine results, and setting a new standard for its use in determining atmospheric temperature profiles.

Finally, motivated by the concept of sequence-to-sequence models, the authors of Gkioxari et al. (2016) use a *similar approach to making predictions*, but in the domain of structured vision. They have shown that the use of multiple neural networks, where the predicted values depend not only on a defined set of initial data, but also on previously predicted values, is a powerful method for solving complex, applied problems. The authors used CNNs for image processing together with a multiscale deconvolutional architecture for prediction at each stage.

3. Methodology

According to the description of the types of research study in Baban et al. (2009); Hassani (2017), this research is **applied and exploratory** with a strong focus on predictive modeling; aimed at making a practical contribution to the field of atmospheric modeling. Firstly, applied research is carried out to develop specific procedures to improve the computational efficiency of the code. In parallel, exploratory research is taken in the initial stages of the research, profiling the code to identify potential problems, and then exploring the data provided for predictive goals, toward which the thesis shifts as the project progresses. This shift is evident in the development and application of AI models to predict the temperature profiles of planets (altitude and temperature values up to the exobase), which brings us back to the applied nature of this thesis.

The author of this thesis adopted a **mixed-method approach based on the *engineering cycle*** (or *regulative cycle* described in Wieringa (2009)) structure model of rational problem solving for iterative and systematic inquiry — it is a list of activities including problem investigation, solution design and validation, solution implementation and evaluation. This methodology has been successfully applied for decades in various fields such as product development, systems and software engineering. (Wieringa et al., 2006; Wieringa, 2009)

The major methods used in this work are discussed and reasoned in this chapter. For convenience, characteristics, for example, the detailed architecture of a chosen NN, or modifications, such as the specific optimizations made, are discussed directly in the next two chapters: Code optimization and AI-Driven Modeling and Its Integration into the KOMPOT Code.

1. *Stage:* Investigation and Analysis of Practical Problems

• *Question*: **Which bottlenecks exist in the code?**

⇒ *Action*: Performance analysis using the ***gprof*** Fortran profiling tool (Graham et al., 2004) to identify bottlenecks – the parts of the code that slow down execution the most and may have the greatest potential for optimization. The main reason for choosing this tool is a detailed view of program execution, showing the time taken to execute each function.

• **What does the data provided for training and testing future AI models indicate?** (parallel independent process)

⇒ **Data analysis**. The data set focuses on planets of the TRAPPIST-1 type, an ultracold dwarf star, and covers most of them. This star was discovered in 2017 with the TRAnsiting Planets and PlanetesImals Small Telescope and is located about 40

light-years from Earth. The scientific interest is justified by the fact that this star has 7 rocky exoplanets around it, 4 of which are Earth-sized and in the habitable zone (Gillon et al., 2017). Studying and modeling the atmospheres of these planets will help us understand whether they can retain an atmosphere long enough for life to have a chance to exist and possibly evolve (Van Looveren et al., 2024). Figure 1.1 provides an overview of these planets.

Since the KOMPOT simulations are very time-consuming, the data consists of a limited data set (138 entries), courtesy of my co-supervisor Gwenaël Van Looveren (Van Looveren et al., 2024). The term «data set» refers to the KOMPOT simulation data, where each simulation or each row of tabulated data corresponds to a representation of a planet. The study focuses only on the initial input parameters and the results in the form of temperature profiles. The choice of constants for standardization was based on Earth parameters that may be suitable for Earth-like planets: the chosen gas temperature at the base of the atmosphere is 258.2 K, the corresponding number density value of $1.73077 \times 10^{16}\ cm^{-3}$ at the same altitude, and the zenith angle at which the simulated segment lies is 66 degrees (the same values were chosen as in Johnstone et al. (2018)). For this data set, planets with H_2O-dominated atmospheres (which could potentially exist in the TRAPPIST-1 system) were not considered because, according to Johnstone (2020), even in habitable zones, conditions around M dwarfs may be severe enough to prevent such atmospheres from stabilizing; however, planets ranging from N_2-dominated like Earth to CO_2-dominated like Venus were considered, representing a large fraction of likely atmospheric compositions. The range of atmospheres considered extends from 50 km to $10,000\ km$ (Van Looveren et al., 2024). The orbital distances were chosen in the range of 0.27 AU to 1.0 AU, which covers a wide range of irradiance cases. The largest possible number of altitude / temperature values per planet is 403. The photon energy values start from the ultraviolet part of the spectrum and end at the high energy end of the XUV spectrum. The other parameters were chosen as a compromise between resolution and computational efficiency. These data are detailed in Table 3.1.

The author of this thesis used exploratory data analysis at the beginning to better understand the data, detect patterns, and identify anomalies, which will later help in selecting the transformation techniques needed in the preprocessing stage and in selecting a machine learning method. Exploratory data analysis, developed in the 1970s, is still a widely used gold standard method in the data discovery process (Data et al., 2016).

Figure 3.1 shows the data set in more detail: the changes in *Exobase Altitudes* and *Temperatures* as a function of XUV flux densities and planet size. As can be seen, higher XUV flux densities lead to higher *Exobase values*. CO_2-rich planets have lower *Exobase Altitudes / Temperatures*, while N_2-rich planets tend to have higher values. This is because CO_2 can act as a coolant due to its radiative infrared cooling (through the 15 μm line), which is also why the atmospheres with higher CO_2 levels tend to lose their particles more slowly (Van Looveren et al., 2024).

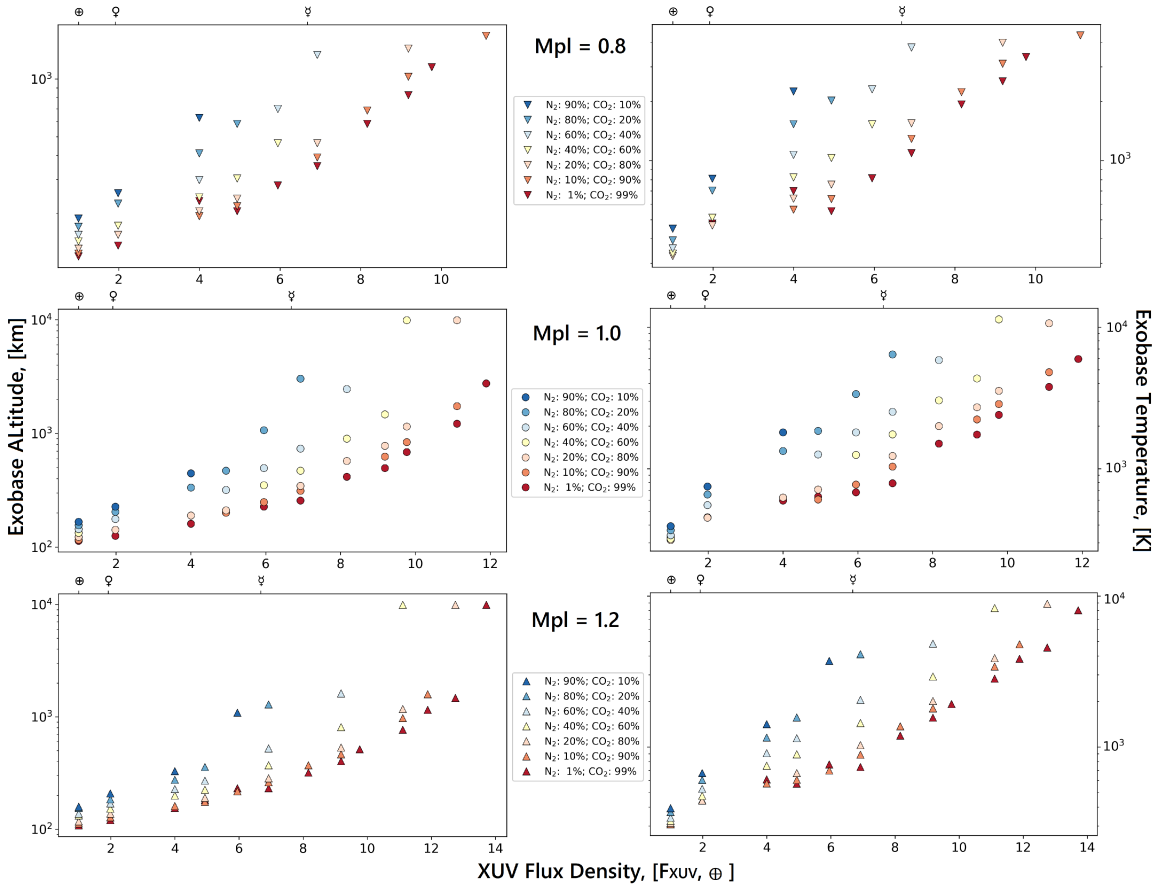

Figure 3.1.: **Exobase Values vs. XUV Flux Density for 0.8, 1.0 and 1.2 $M_\oplus$ planets with different bulk compositions**: Altitudes are shown on the left, temperatures on the right, bulk compositions represented by colors. The top ticks correspond to the XUV flux received by the solar system planets. The y-axis is log-scaled for convenience (code courtesy of Gwenaël Van Looveren).

2. Specification of Solution Designs

• **What tools should be used?**

⇒ Python was chosen for the thesis because of its readability and extensive support for data science libraries. The main libraries used were: the intuitive PyTorch for modeling neural networks; NumPy, Pandas and SciPy for efficient data manipulation; Scikit-learn for testing more traditional machine learning algorithms; Plotly for interactive and detailed visualization, and Matplotlib for static plots.

• **What optimization techniques for identified bottlenecks can be used?**

⇒ **Compiler optimizations**. The very first thing that can be checked is the compilation process of a Fortran code. Even without changing its functionality, compiler optimizations can decrease the execution time while reducing resource consumption. Loop

transformations, such as loop unrolling or loop fusion, which can eliminate function call overhead, and vectorization for array operations, where multiple data points are processed simultaneously as array statements, rather than in loops are typically key optimizations. Code performance can already be significantly improved by allowing only these two compiler flags. (Eijkhout, 2015)

⇒ **Scientific libraries**. The second important point is to check the code for mathematical operations, such as solving linear equations, which can be accelerated by replacing their direct programming with existing functions of fast scientific libraries. For such procedures it was decided to use the ***OpenBLAS*** library. OpenBLAS, one of the fastest open-source scientific libraries available today, is an optimized BLAS, that includes fast assembler code and can be parallelized with OpenMP support (X. Zhang, 2012). Although Intel's MKL currently outperforms OpenBLAS, at least on Intel processors, OpenBLAS was chosen primarily because of its open license and fewer restrictions. This makes it easier to share code and collaborate in an academic environment. This library is much easier to integrate into an existing project, and it is also possible to examine the source code, which is not the case with the MKL library.

⇒ **Parallel computing** is another important aspect of high-performance computing. It means dividing a certain code section into independent subparts so that each such part is processed simultaneously. Not all tasks can be performed in parallel, but for those that can, it becomes possible through the use of multiple cores or processors. The author chose to use the ***OpenMP*** API because of its open-source nature, ease of use and scalability, allowing to work on both shared-memory multicore and multiprocessor systems. (de Supinski et al., 2018)

• **How data should be pre-processed?**

⇒ The heterogeneity of the data records, manifested by the fact that one planet may have calculations, for example, reaching its exobase at 200 km with only 220 values and another at 1000 km with a hypothetical 350 values, requires the implementation of methods for handling «missing»[1] data for successful NNs training: we have no missing data up to 107.51 km, and the overall percentage of missing values in our data frame with temperature profiles is equal to 21.10%. Given the limited non-uniform data available, it can be concluded that methods such as amputation and replacing missing values with zeros are impractical, as each entry of data is meaningful. Therefore, an **interpolation strategy** is considered and chosen so that each data row contains a maximum of 403 records in total (this fixed number of records ranges from 50 km to 10,000 km for altitudes or their corresponding temperatures). The standard *linear interpolation function* of the *SciPy* library is selected due to its simplicity and effectiveness, and is also adjusted to ensure that the interpolated values accurately match the altitude-temperature values shape defined by the actual data points. However, such a *SciPy* function involves replacing the true values with approximations, which can affect the reliability of the data — after

[1] Different planets have different numbers of records depending on the location of their exobase, so the length of record entries is not uniform: some planets have fewer records than others → have «missing» values.

interpolation, this problem is solved by replacing the nearest interpolated points with real values.

Subsequent manipulations include normalization of the data. In this thesis are used: NN with attention for exobase predictions and CNN for whole temperature profile predictions (the reasons for their choice and the description of their operation can be found below, after the description of the feature engineering). In accordance with empirical findings, for the **CNN**, it is decided to use original / **non-normalized data** because its convolutional nature appears to make it quite robust to changes in data scale. On the other hand, it is assumed that, since the data is evaluated separately according to its perceived importance by attention mechanisms and has «outliers» (in this work, these are a few cases with very high *Exobase values*, which are not experimental errors), the **NN with attention** is sensitive to a range and distribution of values — so scikit-learn's **min-max normalization** is applied to inputs and outputs in order to obtain a uniform range from 0 to 1 that does not distort the value relationships, thereby stabilizing the training process and accelerating convergence (also empirically proven). The exobase predictions are then converted back to the original scale to use the values in the CNN input set.

• What feature engineering techniques can be used to select the input values for future predictions?

⇒ The following techniques were used to identify and select input parameters (features) for NNs: correlation and multicollinearity analysis, sorting the features by importance using thre tree-based model XGBRegressor.

The **correlation matrix** calculates the correlation coefficients, in this case the Pearson's correlation coefficients, to study the linear relationships, both between the features and between the features and the target variables (*Exobase Altitude* and *Exobase Temperature*). It helps to set the priority for each input parameter. A zero indicates that the variables are not correlated at all. However, collinearity can exist between multiple variables even when no pair of variables is particularly correlated. **Multicollinearity** refers to the case where several of the independent variables under consideration are correlated, and if this rate is high, it can cause problems when solving regression problems — it is beneficial that the $Variance\ Inflation\ Factor$ (VIF) can be used to measure the degree to which the variance of the estimated regression coefficient is increased by multicollinearity. It is calculated first for each parameter by regressing on all the remaining variables and then measuring how much the variance is increased. In other words, unlike the correlation matrix mentioned above, the multicollinearity test evaluates each independent variable with a group of others, not in a bivariate way. A coefficient of 5 means that the variance of the estimated coefficient is 400% higher than would be expected if the parameters were not collinear. Values greater than this tend to indicate problematic features that may be better eliminated by focusing on more independent / informative features. (James, 2023)

In addition, the **importance of features** was ranked using the Extreme Gradient Boosting Regressor (***XGBRegressor***[2]) tree model. Tree models are one of the machine learning algorithms based on decision trees that are used to make predictive decisions.

[2] `https://xgboost.readthedocs.io/`

XGBRegressor was chosen for its convenient built-in modes. For example, you can quantify the contribution of each feature to the prediction using a seemingly simple method such as the **weight mode**, which counts the number of times a feature used for a data split appears in the tree.

- **What ML methods based on EDA will be tested?**

⇒ Recognizing complex associations between input and output data is an important goal, but also a difficult task for many prediction problems. Therefore, in this master's thesis, **neural networks** (NNs) are chosen for predicting temperature profiles, as they often outperform traditional statistical models in such complex tasks and have long been successfully used in a variety of fields, from text recognition to medical diagnostics. A neural network is an adaptive statistical model, a machine learning algorithm that mimics the way biological neural networks work in our brains. Typically, such a network has layers consisting of neurons / nodes connected by weighted edges, where these connections analogously play the role of synapses. These neurons update the received or, as in the input layer, initially assigned signals / values using a given activation function. They then pass these updated signals to the next or previous (for example, hidden) layer, and so on, until the final output layer. In this way, the network is trained by adjusting the weights of the connections between neurons, for example, by iteratively comparing predicted and actual data, as seen in supervised learning. The goal of such an algorithm is to minimize the chosen loss function used for this comparison. When the number of hidden layers is greater than two, the neural model is usually referred to as a deep learning model. Figure 3.2 shows an example of a simple neural network architecture. (Abdi et al., 1999; Bishop, 2006; Jo, 2021)

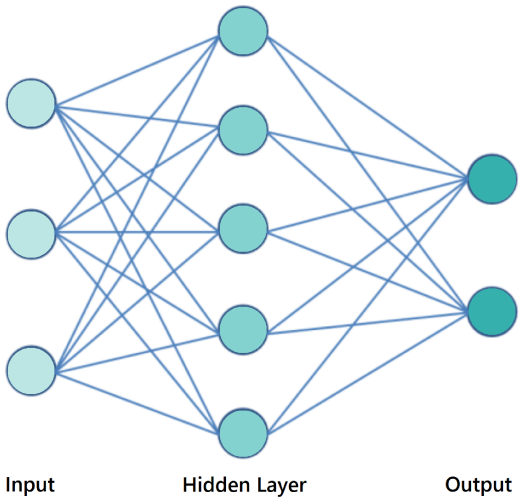

Figure 3.2.: **Example of a simple neural network:** 3 inputs, 2 outputs and one hidden layer of 5 neurons. (Danilov and Karpov, 2018)

⇒ **Dual-stage cascade ensemble learning approach**

Given the small amount of data available, it is decided to use the cascading technique of ensemble learning. The idea behind this approach is quite simple: use the strengths of multiple neural networks to improve the accuracy of the prediction results. As described in de Zarzà et al. (2023), this approach is explained by the concept of «wisdom of the crowd», where weaker players can beat a stronger player by compensating for each other's mistakes. Cascading implies a hierarchical structure in which the predictions of the first neural network are used as input parameters for the next one. The modulated approach is primarily motivated by reducing task complexity, simplifying the learning process and increasing manageability, which

refers to independent updates and scalability. Narrowly targeted learning increases the accuracy of predictions, because each task can be optimized separately — this minimizes the error propagation across tasks. This technique is well suited for complex problems and has been proven, for example, in image recognition in medicine and time series forecasting in banking systems. Although temperature profiles are not time series data, they show a similar sequential dependences, which argues for the use of this technique in this thesis. (Dietterich, 2000; de Zarzà et al., 2023; Pławiak et al., 2019)

Based on the analysis of the available data, the author decided to use 2 neural networks, dividing the main goal into 2 subgoals: (1) predicting the altitude and temperature of the exobase using NN with attention, (2) taking into account past input data and 2 predicted values, predicting the «shape» of the profiles using CNN, that is, a set of 806 (403x2) values between the established base (258.2 K at an altitude of 50 km) and the previously predicted exobase.

As described in the Introduction 1.4, these temperature profiles predictions are later used to create a restart file for KOMPOT, which will accurately produce a whole atmospheric model in fewer iterations than without a restart file. The entire process is illustrated in more detail in Figure 3.3.

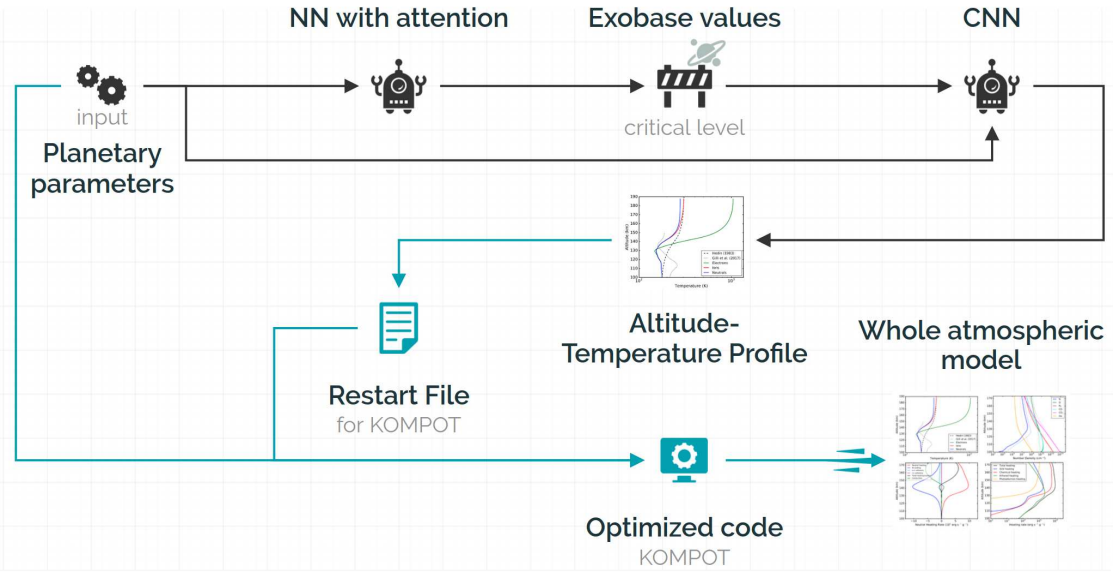

Figure 3.3.: **Method for embedding AI predictions into optimized KOMPOT code to obtain a whole physically consistent model in fewer iterations.**

⇒ **Neural networks with attention mechanisms** (for *Exobase values* predictions) Given a limited data set, standard machine learning models, while generally excellent at solving nonlinear problems, can be ineffective due to the high likelihood of overfitting, where the model has learned too well the details and possible noise in the data used for training, and therefore has poor predictive ability when working with new data. However, neural networks with attention mechanisms chosen for the task of exobase value prediction can mitigate this problem: attention mechanisms help identify and learn the

parts of the data that are most informative at different times, because each input feature may be critical to obtaining accurate results. As a result, the model does not waste resources on less important aspects, being more efficient compared to traditional networks. Such networks also increase the interpretability of the model, which is no less important information than the predictions themselves. (Vaswani et al., 2017)

The chosen NN architecture consists of a fully connected attention layer, followed by a feed-forward network that processes this data from the attention layer through additional fully connected hidden and output layers. For even more adjusted training, a dropout layer is applied after the hidden layer, which minimizes overfitting probability by randomly setting some activations to zero during training. If we visualize the path to the result as a single formula, it will look like this:

$$\hat{\mathbf{y}} = FC_{\text{output}}\left(\text{Dropout}\left(\text{ReLU}\left(FC_{\text{hidden}}\left(\text{ReLU}(FC_{\text{combine}}\mathbf{c}' + \mathbf{b}_{\text{combine}})\right) + \right.\right.\right. \\ \left.\left.\left. + \mathbf{b}_{\text{hidden}}\right) + \mathbf{b}_{\text{output}}\right), \quad (3.1)$$

where:

- $\hat{y} \in \mathbb{R}^2$ represents the final output vector of 2 *Exobase values* (prediction),
- $c = ax^T$ is an attention context vector, a batch matrix multiplication result between the attention weights $\mathbf{a} \in \mathbb{R}^{attn\_n}$ and the input vector $\mathbf{x} \in \mathbb{R}^{input\_n}$, in other words, it is an input feature vector weighted by their previously defined attention scores / relevance. $c' = \text{Flatten}(c)$ is a smoothing function needed to fit the context into the next layer, the attention combination layer,
- $a = \text{softmax}(FC_{\text{attn}}x + b_{\text{attn}})$ is attention weights calculation by applying a fully connected (linear) layer to the input x, where $FC_{\text{attn}} \in \mathbb{R}^{attn\_n \times input\_n}$ is a learned weight matrix, and $b_{\text{attn}} \in \mathbb{R}^{attn\_n}$ is the bias term for the attention layer. This assigns an importance to each linear transformation of the input data. The softmax function is applied to each sample of data in the batch along $dim = 1$, normalizing the weights so that their sum equals 1,
- $h = ReLU(FC_{\text{combine}}c' + b_{\text{combine}})$ is an attention combination layer, where c is passed through a fully connected layer to transform it into the hidden layer's dimensionality, followed by a ReLU activation function;
  where $FC_{\text{combine}} \in \mathbb{R}^{hidden\_n \times (attn\_n \cdot input\_n)}$ and $b_{\text{combine}} \in \mathbb{R}^{hidden\_n}$.
  ReLU (Rectified Linear Unit) is chosen as the activation function. This decision is based on the fact that, in practice, NNs using ReLU tend to achieve convergence faster due to simplified mathematical operations and often perform better than, for example, the hyperbolic tangent activation function. The simple function introduces nonlinearity into the network to better learn complex patterns:
  $rectifier(x) = max(0, x)$, meaning that if x is positive, then x is returned, if negative — 0 (neuron deactivated). This means that only some part of neurons is active at any given time, making the process more efficient (Glorot et al., 2011; Bengio et al., 2017; James, 2023),

- $ReLU(FC_{\text{hidden}}h + b_{\text{hidden})}$ is then reapplied to ensure that the complex relationships between the dimensionally reduced data are preserved, where $FC_{\text{hidden}} \in \mathbb{R}^{hidden\_n \times hidden\_n}$ and $b_{\text{hidden}} \in \mathbb{R}^{hidden\_n}$,
- As discussed earlier, the $Dropout()$ regularization layer is used to prevent severe overfitting so that the model is forced to learn more robust features,
- $FC_{\text{output}} \in \mathbb{R}^{2 \times hidden\_n}$ and $b_{\text{output}} \in \mathbb{R}^2$ are the weights and bias of the output layer, the final linear transformation.

⇒ **Convolutional neural networks (CNNs)** (for altitude-temperature profile predictions)
CNN is an artificial neural network, part of deep learning, and originally used for more efficient object detection (Girshick et al., 2014) and classification problems (Krizhevsky et al., 2012), or for image virality analysis (Alameda-Pineda et al., 2017).
The network is structured with no feedback — CNN can have signals traveling only in one direction — and with many convolutional layers, that usually alternate with pooling layers, generally followed by several fully-connected layers and a softmax layer with a loss function (Lathuilière et al., 2019). The convolution operation consists, that each piece of input data (for example, an image) is multiplied by a convolution matrix element-wise, then the result is usually summed and written to the same position of the output element (in this case, an image) (Bengio et al., 2017).
Such networks can also be used to solve regression problems (Belagiannis et al., 2015; Zeng et al., 2016), in this case the classification softmax layer is usually replaced by a fully-connected regression layer with an activation function (Lathuilière et al., 2019). Its univariate version is also effective in regression analysis of sequential data such as time series (Durairaj and Mohan, 2022).
Therefore, it was hypothesized that this type of network, because of its ability to find complex (for example, temporal) patterns, is suitable for predicting temperatures whose values have spatial correlations and vary with altitude. (Tokareva, 2019)

The full expression for selected structure of the CNN is ($\hat{y} \in \mathbb{R}^{806}$):

$$\hat{\mathbf{y}} = FC_{\text{output}} \left(\text{Dropout} \left(\text{ReLU} \left(FC_{\text{hidden}} \left(\text{ReLU} \left(\text{BN}(FC_{\text{conv1d}} * \mathbf{x} + \mathbf{b}_{\text{conv1d}})\right)\right)'\right.\right.\right. + \\ + \mathbf{b}_{\text{hidden}} + \mathbf{b}_{\text{output}}, \quad (3.2)$$

which repeats the structure of the first neural network, but fundamentally contains a one-dimensional convolution:

$$FC_{\text{conv1d}} * \mathbf{x} + \mathbf{b}_{\text{conv1d}} \quad (3.3)$$

which represents a set of feature maps by sliding the filter over the input vector $x \in \mathbb{R}^{1 \times input\_n}$,where $*$ is the convolutional operation, $FC_{\text{conv1d}} \in \mathbb{R}^{filter\_n \times 1 \times kernel\_n}$ is a set of learned filters, and $b_{\text{conv1d}} \in \mathbb{R}^{filter\_n}$ is a corresponding vector of bias terms. Then, to speed up convergence, the batch normalization (BN) function is applied to this convolutional output, which stabilizes learning by adjusting the mean and variance to reduce internal covariate shift (Bengio et al., 2017).

3. Validation of solutions

• **What solution design can be chosen?**

⇒ Implement and test code optimizations.

Before each new optimization step, the code is tested to avoid loss of computational accuracy.

⇒ Implement the selected ML solution and find the best hyperparameters.

Hyperparameters are important properties of a machine learning model that define its structure and influence the behavior of the learning process, which then determines the performance of the model. Improperly chosen parameters (over- or underestimated values) can lead to overfitting, when a model generalizes poorly, in other words, does not work well with new data, or underfitting, when a model does not work well even with the data used for training. This can also increase the consumption of computational resources and the convergence time. The main hyperparameters used in this work are: (1) batch size that specifies how many samples a model is trained once; (2) number of epochs, which specifies the number of passes through the entire data set by a neural network; (3) learning rate or step size that determines how much the model's weights (feature weights) are adjusted after each batch of data; (4) number of layers, and (5) number of neurons in each layer; (6) activation function, which decides whether a particular neuron should be activated depending on the input data and weights, thereby affecting the output values and introducing nonlinearity into the model to allow capturing more complex patterns between input and output data; (7) scheduling techniques that make the learning rate dynamic to improve convergence (for instance, *StepLR*, which is used in this work, reduces this number by a given factor after a certain number of epochs). Another important parameter is the loss function (8), which periodically evaluates the predictive ability of the considered model by calculating the error margin between the predicted and the actual value. For example, the *Huber loss* function used in the CNN model combines the positive aspects of mean squared and absolute errors, making it less sensitive to outliers and more robust than using only one of these errors separately:

$$L_\delta(a) = \begin{cases} \frac{1}{2}(a)^2 & \text{for } |a| \le \delta, \\ \delta \cdot (|a| - \frac{1}{2}\delta) & \text{otherwise,} \end{cases} \tag{3.4}$$

where a is the difference between the actual and predicted parameter, and δ determines the «switch» between the first, quadratic case (ideal for smaller errors) and the second, linear case (ideal for larger errors). (Bengio et al., 2017; Julian, 2018; Friedman, 2017)

A tool called ***optuna*** is used to efficiently search for optimal hyperparameters of both NNs; optuna is an open source, automated, next-generation hyperparameter optimization framework. The objective function to be minimized or maximized can be viewed as follows:

$$objective(x) = ModelEvaluationFunction(x), \tag{3.5}$$

where x is the hypermarameters' vector to be optimized.

The key feature is the define-by-run approach, which dynamically constructs the search

space during the optimization process itself — this allows dealing with complex hyperparameters that depend on previous choices. The method also incorporates advanced techniques for sampling the next set of parameters to test, such as the TPE (Tree-structured Parzen Estimator, which uses probability distributions and focuses on configurations with promising results), and pruning techniques to reduce computational overhead such as the Asynchronous Successive Halving algorithm (first evaluates many configurations with a small amount of resources, then allocates more resources only to promising configurations), speeding up the process by cutting off poorly performing tests early on. This framework is easy to integrate and use due to its intuitive interface and the ability to visually analyze an optimization process using the optuna dashboard. (Akiba et al., 2019)

⇒ Train and test the NN models with available data.
The data is split 85/15% (117/21 planets) to allow both artificial neural models to better learn the currently limited data. The widely used $train\_test\_split()$ function from *scikit-learn*[3] is responsible for randomization and representative sampling in the training and test sets, which also supports accurate reproducibility of the split — important for code debugging.

4. Evaluation of Outcomes

• **How can results be evaluated?**

⇒ Comparing pre- and post-optimization performance metrics: to evaluate code performance, the author of this thesis chose a direct quantity metric — **execution time** — that is easy to understand and compare, measuring the total time it takes to execute a program from start to finish. First, the execution time of a given simulation (original code) is measured to establish a baseline, with as few third-party processes active as possible. After optimizations, this step is repeated with the same initial conditions. The measurements are repeated several times and the average of the attempts is used for later comparison to ensure consistency. (Suh et al., 2017)

⇒ Evaluate AI models' accuracy using error metrics in order to assess the predictive ability of the models as accurately as possible, and considering the purpose of further applying this methodology to a large data set, several methods typical for regression problems (described in Bengio et al. (2017); Gelman (2007); Chatterjee and Hadi (2015); Bishop (2006); James (2023)) are used:

1. Mean Squared Error (MSE) to measure the average squared difference between the true and predicted values; where large errors are seen more than small errors — therefore, the smaller the values, the better the model predicts. The ideal value is considered to be when the error is 0. In this work, the **Root Mean Squared Error** ($RMSE$) is used instead because it also reflects the average error, but in units of the target variable, which can help to better interpret the results.

2. **Mean Absolute Error** (MAE) is the mean absolute difference, which is more robust to outliers. As with the MSE: the lower the values, the better the performance

[3] `https://scikit-learn.org/`

of the model under consideration.

3. **Coefficient of determination** ($\boldsymbol{R^2}$), or goodness of fit, is a measure of the overall accuracy of the model, measuring the proportion of variance in the dependent values that is explained by the independent values. When this value is 1, the model is a perfect fit to the actual data. While this estimation method works well for 1-value predictions (for example, when predicting *Exobase values*, but measuring the model's predictive ability separately for an altitude and for a temperature, as in this thesis), it is not a good indicator of model accuracy when estimating predictions of multiple values (403 altitude-temperature value pairs) because it does not directly account for the interactions of simultaneously predicting multiple values without considering the sum of each predictor. In this case, $Weighted\ R^2$ is used to estimate the CNN model, where all scores are averaged, weighted by the variances in each individual atmospheric model.

4. **Mean Relative Error** gives an indication of how much, on average, the predicted results differ from the true results as a percentage of the true values themselves, and helps to understand the overall accuracy of the predictions as well. The smaller the percentage, the better the accuracy of the model.

5. **Mean Bias Deviation** is useful for understanding prediction bias by measuring the average deviation of the predicted values from the actual / true values. The ideal value is 0. If the value is above zero, it indicates that the model tends to overestimate the true values, if it is negative — the model underestimates the results.

6. For predicting *Exobase values*, **visual analysis** is also performed, including scatter plots, residual plots, histograms, and relative error plots to understand patterns in the model (such as heteroscedasticity or the presence of outliers) for further analysis. The scatter plot directly shows the relationship between predicted and true values. The residual plot visualizes the difference between the values and determines the independence of the errors. The plot is expected to show no obvious patterns, otherwise it could mean that the model is missing some aspect of the data. A histogram, in turn, reflects the error distribution by grouping the data into bins and calculating the frequency of points for each bin. Ideally, a symmetric distribution around zero is expected. The visualization of the relative error of each prediction shows the consistency of the errors across the range of predictions. In this work, errors within $10 - 20\%$ are acceptable errors because, as mentioned earlier, the KOMPOT code can successfully correct for them during a restarted simulation.
   In the case of the second CNN model, which predicts multiple values, it was decided to visualize the predicted and true temperature profiles on a single plot, as well as scatter plots for thermal profiles separately for small, medium, and relatively large altitude values to identify possible patterns, and to understand which altitudes and temperatures the model predicts best.

5. Implementation of Selected Design

• **How can the selected design be implemented?**

⇒ Create a simple Jupyter notebook to track the EDA process and the creation and training of AI models to simplify subsequent optimizations.

⇒ Introduce AI models into the pre-simulation process of optimized KOMPOT code by converting temperature profile predictions into the KOMPOT's restart file, creating a clear, modular and simple command line tool with the ability to read input parameters from different file types and perform visual analysis of the predictions for further study (including simplified visual comparison of these predictions with KOMPOT code simulations if KOMPOT simulations are provided).

⇒ Write a short documentation for the CLI tool.

0. Start of a New Regulative Cycle

⇒ As new data becomes available, the cycle process can be repeated, starting with data analysis but skipping solution and evaluation method selection steps, with the option of using the aforementioned Jupyter notebook.

Initial Parameter	Definition	Unit	Tested Range
Planetary Parameters			
Mpl	Planetary mass	Earth masses	[0.8, 1.0, 1.2]
Rpl	Planetary radius	Earth radii	[0.93, 1.0, 1.06]
aOrbit	Orbital distance	AU	[0.27, 0.28, 0.29, 0.3, 0.32, 0.33, 0.35, 0.38, 0.41, 0.45, 0.5, 0.71, 1.0]
Atmospheric Parameters			
$\boldsymbol{X(N_2)}$	Mixing ratio of N_2, with $X(N_2) = 1 - X(CO_2)$	-	[0.01, 0.1, 0.2, 0.4, 0.6, 0.8, 0.9]
$\boldsymbol{X(CO_2)}$	Mixing ratio of CO_2	-	[0.1, 0.2, 0.4, 0.6, 0.8, 0.9, 0.99]
zMin	Minimum altitude	km	[50.0]
zMax	Maximum altitude	km	[10000.0]
ZenithAngle	Angle between model direction and direction pointing at star	° (degrees)	[66.0]
TgasBase	Base temperature	K	[258.2]
nBase	Density number at the base	cm^{-3}	[1.73077e+16]
$\boldsymbol{X(H_2O)}$	Mixing ratio of H_2O	-	[0.0]
nBinsXUV	Number of XUV bins	-	[10.0]
nPhotoelectronBins	Number of photoelectron bins	-	[100.0]
EmaxPhotoelectron	Maximum energy for photoelectron spectrum	eV	[1000.0]
EminXUV	Lower energy for spectrum	eV	[3.11]
EmaxXUV	Upper energy for spectrum	eV	[1239.8]

Table 3.1.: **Data set initial parameters overview**: The table summarizes the range of initial parameters used to build atmospheric models of planets by the KOMPOT code, their definitions and units.

4. Code Optimization

This chapter describes in more detail the code manipulations performed to optimize the KOMPOT code. The results of the experiment can be found in Chapter 6.

4.1. Compiler Optimizations

In the context of CPU-based computations, the free-to-use ***gfortran*** is still the recommended Fortran compiler because it supports most of the latest standards, is easy to install and use, and is actively developed. The optimization potential can be activated by using the compiler options. The following is a list of options, or flags, for optimizations used in this thesis after careful analysis of the code content[1]:

- The easiest way to enable the largest number of optimizations is to add the **-O3** flag. This advanced option includes about 100 optimizations that can reduce program execution time, but is expected to slow down the compilation process itself.
- The next flag is **-funroll-all-loops**. It unrolls all loops, and uses the so-called time-memory trade-off technique, where by increasing the code size, the execution time is reduced. An example of loop unrolling can be seen in Listing 4.1. Despite the fact that today's version of the KOMPOT benefits from using this option, it is recommended to reconsider it when changing the code in the future, because there is a big risk of slowing down the code, since it transforms all loops, regardless of whether we know the number of their iterations or not. In such a case the author would try the *-funroll-loops option*, when only loops with known number of iterations are deployed, or manually change the loops suitable for unwinding. In the example 4.1, we reduce the number of iterations in the loop by a factor of 4 by performing 4 tasks at once in a single iteration.
- **-ftree-vectorize** was used for vectorization of loops and scalar operations (using a tree data structure), which means processing one operation on multiple operands at once, which is possible on systems that support the SIMD (Single Instruction, Multiple Data) processing type, which includes most modern processors. An example of a loop suitable for vectorization is shown in Listing 4.2, where each iteration in the loop is independent of the others.

[1]The descriptions are based on information taken from the book Eijkhout (2015) and the official GNU Compiler Collection website: `https://gcc.gnu.org/onlinedocs/gfortran/`

```
! Original Loop:                    ! Partially Unrolled Loop:
do i = 1, 100                       do i = 1, 100, 4
  array(i) = array(i) + 3             array(i) = array(i) + 3
end do                                array(i+1) = array(i+1) + 3
                                      array(i+2) = array(i+2) + 3
                                      array(i+3) = array(i+3) + 3
                                    end do
```

Listing 4.1: **Fortran code example of a partially unrolled loop**.

```
do i = 1, 100
  c(i) = a(i) * b(i)
end do
```

Listing 4.2: **Example of Fortran code suitable for vectorizing**.

- The cache, a temporary storage, management optimization option **-fprefetch-loop-arrays** was also used. The operands of some instructions, such as those in the same loop, are usually located contiguously in memory and can be detected by the compiler as regular patterns of cache misses: when computations wait for data to be delivered. Once detected, data is pre-loaded from memory into the appropriate cache lines before it is ever used. This helps to eliminate such delays for all but the first few elements needed to directly generate the pattern being searched for. Figure 4.1 shows an example of such a instruction. Again, despite the success of using this flag for today's version of the KOMPOT, it is necessary to re-test the option if the code is changed significantly. This is necessary because, for example, the problem of cache pollution can arise when wrong candidates are unloaded by incorrect predicted patterns due to shared cache mechanisms in a multi-core environment (Choi et al., 2021).

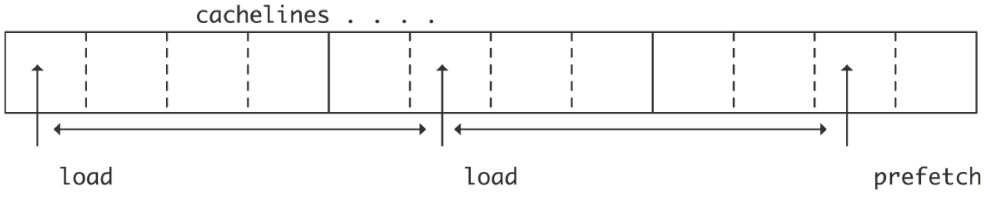

Figure 4.1.: **Illustration of a prefetch stream generated by equally spaced requests** (Eijkhout, 2015).

The figure shows the cache memory segments where the CPU loads data from main memory. «Load» indicates the points where data from the cache is already loaded into the CPU's registers for the computation process. First, the compiler calculates the optimal distance, or step, to the current execution point, and then generates

a request that takes this step into account to load data before it is needed, which speeds up the task because accessing main memory takes more time.

4.2. Bottlenecks Identification

Before proceeding with profiling, the completion criterion for experimental validity was set: since the standard parameters (baseline) used in the Johnstone et al. (2018) to study the effect of CO_2 content and changes in the Sun's XUV spectrum on the upper structure of the Earth's atmosphere, stable results were observed after 650,000 iterations, and this number of iterations was set as the criterion. The ***gprof*** tool was used to identify performance bottlenecks in the KOMPOT, with the most significant ones visualized in descending order. Figure 4.2 shows a a notable trend for the processes involved in solving **chemical equations** concerning chemical evolution. The **Gaussian Elimination** function, which is also the most frequently called function during the simulation and takes up almost 80% of the total run time, offers considerable optimization potential. The next most frequently called functions are those that solve chemical ODEs, such as *get_kCoeff_RB*, which calculates the coefficients needed to calculate gas temperatures for the next time step by solving linearized equations; or *get_chem_dndt_point*, which calculates the rate of change of each species as a result of chemical reactions for a single cell.

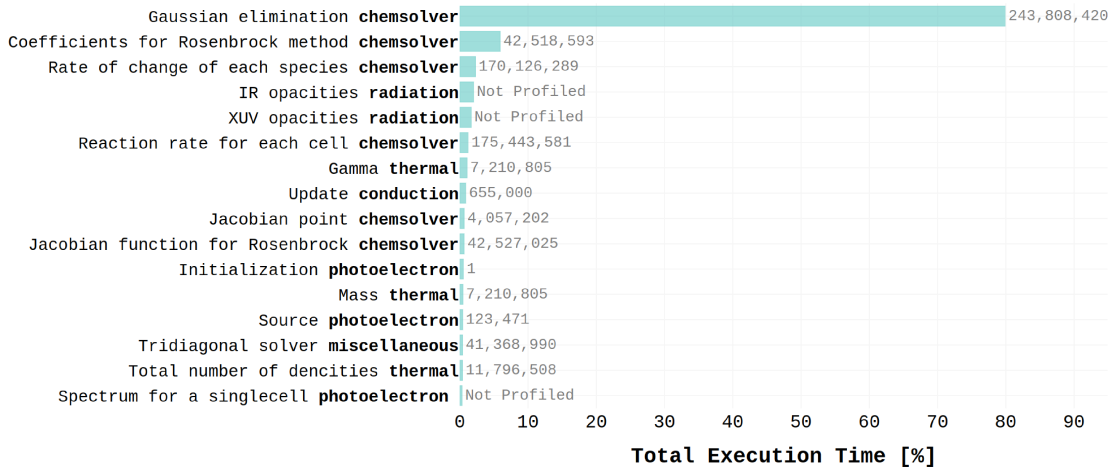

Figure 4.2.: **Function performance analysis with call counts**: x-axis represents the percentage of the total simulation runtime, and the y-axis — the functions, sorted in descending order of their total percentage usage. The number of times that function was called during the baseline simulation is displayed next to the bar.

4.3. Gaussian Elimination and LU Decomposition with Partial Pivoting[2]

Gaussian Elimination was chosen as the main candidate for optimization. What is this function? **Gaussian elimination** is a method for solving linear equations of the form $\boldsymbol{A \cdot x = b}$, where A is a matrix, x is the vector of values to be searched for, and b is the vector resulting from the multiplication (Solomon, 2015).

First, as illustrated in Algorithm 1, the elements of the first row (the first row of the matrix combined with the first element of the result vector) are normalized by dividing each value in the row by the first diagonal element (pivot) so that the first diagonal element becomes equal to 1. Further, each row below the first row is replaced by the result of subtracting the normalized first row multiplied by the first element of the minuend row (the element of the decrementing row that is in the same column as the diagonal element in question) from the minuend row, so that all elements in the first column below the first diagonal element are equal to 0. The second row is normalized by the second diagonal element and the subtraction process is repeated, and so for each row, until all values below the diagonal become 0, and we obtain the upper triangular matrix. Then we equate the vector x with the values of the result vector b.

Using backward substitution, starting from the penultimate row to the first row, the value of each element of x is replaced by the result of subtracting from the product of the corresponding element in the matrix A and the element of the next row in the vector x. At the same time, the corresponding row in the matrix A is corrected by the result of subtracting from it the product of the same element in the matrix A and the elements of the next row in A.

At the end of all the necessary iterations, we obtain the solution of the equations in the vector x. In summary: this method works because each step simplifies the problem, aiming at isolating each variable.

This method is used at the beginning of the initial calculation and subsequent updates of the *energy exchange* between neutral, ion and electron gases. Using the implicit Crank-Nicolson method, multiple simplified time-dependent equations of gas temperatures are solved at once: involving the interactions of different pairs of species, which can be reduced to the form of a linear equation, where A will consist of known pre-calculated exchange coefficients, b — known values calculated on the basis of these exchange coefficients and component temperatures at the current time step, x — the result of energy exchange correspondingly — unknown component temperatures for the next time step, which must be calculated at each grid cell.

As part of the implicit Rosenbrock multistep method, the Gaussian method is also used in solving *chemical ordinary differential equations (ODEs)* for various species to model chemical kinetics: the calculation of species' densities in the upper atmosphere. In this case, the matrix A is based on a Jacobian matrix, which ultimately reflects the interaction density change rates between different species. The resulting vector b consists of values based on density changes due to multiple chemical reactions at the current time

[2]This section is based primarily on the books: Solomon (2015) and Burden et al. (2015)

step; and k, or x, is the desired vector consisting of the changed densities of the chemical species during the evolution, specifically, for the next time step.

Data: A, B
Result: Solution X

Let $X = B$

Do Forward Substitution:

for $i = 1$ **to** N **do**
  Normalize the pivot row:
  for $j = i$ **to** N **do**
    $A_{i,j} = A_{i,j}/A_{i,i}$
  end
  $X_i = X_i/A_{i,i}$
  Eliminate elements below the pivot:
  for $k = i + 1$ **to** N **do**
    $factor = A_{k,i}$
    for $j = i$ **to** N **do**
      $A_{k,j} = A_{k,j} - factor \cdot A_{i,j}$
    end
    $X_k = X_k - factor \cdot X_i$
  end
end

Do Backward Substitution:

for $i = N - 1$ **downto** 1 **do**
  for $j = i + 1$ **to** N **do**
    $X_i = X_i - A_{i,j} \cdot X_j$
  end
  Normalize the result:
  $X_i = X_i/A_{i,i}$
end

Algorithm 1: Gaussian elimination for solving $\boldsymbol{A \cdot X = B}$ (Solomon, 2015; Burden et al., 2015)

Standard Gaussian elimination is not numerically stable, because if the matrix A is ill-conditioned, any small perturbation in the input data can lead to incorrect results in the end; or if the matrix consists of very small values on the diagonal, it also risks large round-off errors. Considering also that the elements of the Jacobian matrix in the KOMPOT are often zero, namely, the matrix is very sparse, the use of LU decomposition with partial pivoting is a suitable solution to these problems, although it has the same

computational complexity as Gaussian elimination: $O\left(number\,of\,species^3\right)$. This means that the completion time of the algorithm is proportional to the cubic value of the number of species. (Lindquist et al., 2022)

The **LU decomposition with partial pivoting** aims to find such a lower triangular and an upper triangular matrix whose product would equal the matrix A multiplied by the matrix P representing the raw swaps (permutation matrix): $P \cdot A = L \cdot U$.

First, the matrix L is assigned the values of the matrix with 1 on the diagonal (identity matrix), like P, and the matrix U is assigned the values of the matrix A. Then the row with the largest absolute value in the first column is swapped with the first row (with the row containing the pivot element), the rows with the same indices in both matrix P and matrix L are swapped from the second iteration, but only the parts of the rows below the diagonal are swapped. Then, each value of the first column of matrix L, starting from the second, is updated by the results of the divisions of each element of the first column U by the current diagonal pivot element: the first element of matrix U. Let us call the results of these divisions the coefficients of L.

Then, starting from the second row, each row in U is replaced by the values obtained by subtracting the product of the corresponding coefficients of L and the first row in U from the considered row in U. The process is repeated for each remaining column. Once we have the matrices L and U, we solve an equation of the form $L \cdot y = P \cdot b$ (here, as in the following linear equation, the computational complexity is $O\left(n^2\right)$), where the rows are swapped according to the instructions in the matrix P. Since the matrix L has zeros above the diagonal, we use the method of forward substitution — already described here as backward substitution, but in the opposite direction — to find the vector y, starting from the second row. Finally, we use backward substitution to find x in the equation: $U \cdot x = y$ (see Algorithm 2).

The optimal solution was to use a specialized scientific library for this purpose: after manual installation of 4 threads-parallel ***OpenBLAS*** — which is recommended by the author of this thesis in order to configure the library as optimized as possible for the specific system used —, the function **dgesv** was introduced [3]. To call the function, the following input information is required: (1) number of equations, (2) number of columns in the matrix B, (3) matrix A, (4) leading dimension of the A: number of rows if the matrix is stored in column-major order (column by column), as in Fortran, but if the row-major order — the number of columns, (5) a vector of size equal to the number of equations as a permutation vector, (6) a vector of results b, which will be overwritten by the calculated values of x, (7) the leading dimension of b, and (8) an integer variable to assign the success status of the computation at the end.

This approach is numerically more stable compared to Gaussian elimination, thanks to the pivoting step, and faster due to the fact that the decomposition requires fewer operations (in comparison, Gaussian elimination is prone to repeated operations on rows) and due to the optimized multi-threaded OpenBLAS.

[3] `https://www.netlib.org/lapack/explore-html/`

Data: A, B
Result: Matrices P, L, U and solution X

Let $P = I$, $L = I$, $U = A$

Decompose A to $P \cdot A = L \cdot U$:

for $c = 1$ **to** $N - 1$ **do**
    Do pivoting:
    select r **where** $U_{r,c} = \max(|U_{:,c}|)$
    swap $U_{c,c:n}$ **with** $U_{r,c:n}$
    swap $L_{c,1:c-1}$ **with** $L_{r,1:c-1}$
    swap $P_{c,:}$ **with** $P_{r,:}$
    Decompose U and form L:
    for $r = c + 1$ **to** N **do**
        Define elements below the diagonal:
        $L_{r,c} = U_{r,c}/U_{c,c}$
        Update U:
        $U_{r,c:n} = U_{r,c:n} - L_{r,c} \cdot U_{c,c:n}$
    end
end

Apply row permutation to B:

$B = permute\,(B, P)$

Do Forward Substitution to solve $L \cdot Y = B$:

for $r = 1$ **to** N **do**
    for $c = 1$ **to** $r - 1$ **do**
        $Y_r = Y_r - L_{r,c} \cdot Y_c$
    end
end

$X = Y$

Do Backward Substitution as in Gaussian elimination (see Algorithm 1) to solve $U \cdot X = Y$

Algorithm 2: LU decompositon with partial pivoting for solving $\boldsymbol{A \cdot X = B}$ (Solomon, 2015; Burden et al., 2015)

4.4. Parallelization and Minor Optimizations

Initially, the code was already partially parallelized, but after a code revision, additional loops were identified that were suitable for parallelization using ***OpenMP***. The following functions were considered: *in the chemical evolution and ODEs solver*: (1) a function used to calculate the rate of change of all species for a single cell (2) a function used to calculate the Jacobian matrix; *in the thermal calculations*: (3) a function to compute the average mass of each grid point, and (4) a function to calculate the average specific heat capacity ratio of each grid point, (5) calculations of the total number density of each grid point; *in the radiative calculations*: (6) a subroutine of calculations of the IR opacity for each energy bin, (7) and the same function by the target but with XUV; *in photoelectron chemistry*: (8) function initializing photoelectrons, (9) calculations of the photoelectron source function and (10) spectrum for a single cell. Algorithm 3 shows an example (5) of using OpenMP in the KOMPOT code.

This algorithm starts by identifying the region that the author wanted to parallelize, which implies simplifying the task by dividing its execution into several parts or threads that can be executed independently and simultaneously, for example, using multiple processor cores. The declaration of *private variables* at the beginning is necessary to ensure that each thread has its own copy of the independent variables, and that multiple threads do not modify the same area of memory during computation, potentially leading to inaccurate computations (race conditions). *Shared variables*, on the other hand, declare variables that are available to all threads at all times. Note that these variables are used for reading only, like the $nSpecies$ variable, or for storing the final accumulation results, like $nTotal$. Next, the accumulators, independent variables, are initialized for the later accumulation of density values of the gas, which is done separately for neutrons, electrons and ions. In the loop, which iterates over each species, each thread calculates the corresponding part of the total density. After the loop, the update of the total number of densities array occurs in the *critical section*, which allows only one thread at a time to update it. Although this slows down the parallel process, it is necessary to avoid errors. (Chapman et al., 2007)

In some functions, only minor calculations were also simplified, mostly to eliminate redundancy. For example, in the function responsible for the *tridiagonal matrix algorithm* (used in solving energy equations and semi-static hydrodynamic equations), in loops where the calculation of the *denominator* used to update several variables is repeated, an additional variable was introduced that is calculated only once at the beginning of the loop as $1/denominator$ and is used in other calculations in the loop by multiplication. This helper variable increases the readability of the code and has a positive effect on its performance, since multiplication is less computationally expensive than division (Bailey, 1993).

To maintain scalability, several options have been developed to compile without using OpenMP, as well as using the built-in *BLAS* or *LAPACK* libraries as options if for some reason OpenBLAS cannot be installed properly. Although the author of this thesis recommends using all of the above methods, compiler optimizations and usage of the

dgesv function, even using standard BLAS and without OpenMP, can also accelerate the KOMPOT simulations.

Data: Shared Variables (described below)
Result: $nTotal$

Begin OpenMP (Parallel processing)

Declare Private Variables: $iSpecies$, $nTotal\_private\_N(eutrons)$, $nTotal\_private\_I(ons)$, $nTotal\_private\_E(lectrons)$
Declare Shared Variables: $n(umber\,density\,of\,each\,species)$, $nTotal$, $chargeSpecies$, $namesSpecies$, $n(umber\,of)Species$, $iTemp(erature)N$, $iTempE$, $iTempI$

Initialize Private Accumulators for each thread:

$nTotal\_private\_N = 0.0$
$nTotal\_private\_I = 0.0$
$nTotal\_private\_E = 0.0$

Parallel loop over each species:

for $iSpecies = 1$ **to** $nSpecies$ **do**
  if $chargeSpecies(iSpecies) == 0.0$ **then**
    Accumulate neutral species contributions:
    $nTotal\_private\_N = nTotal\_private\_N + \{calculations\}$
  end
  else if $chargeSpecies(iSpecies) == 1.0$ **then**
    Accumulate positively charged species contributions for $nTotal\_private\_I$
  end
  else if $chargeSpecies(iSpecies) == -1.0$ **then**
    Accumulate negatively charged species contributions for $nTotal\_private\_E$
  end
end

Begin Critical Section:
Aggregate private accumulators into shared array:

$nTotal(iTempN, :)$ += $nTotal\_private\_N$
$nTotal(iTempI, :)$ += $nTotal\_private\_I$
$nTotal(iTempE, :)$ += $nTotal\_private\_E$

End Critical Section

End OpenMP

Algorithm 3: Example of parallelizing the *get_ nTotal* subroutine that computes the total densities for neutrals, ions and electrons separately.

5. AI-Driven Modeling and Its Integration into the KOMPOT Code

This chapter describes selected configurations for AI models that predict altitude-temperature values of atmospheres from 50 km/258.2 K up to the exobase: selected input parameters, optimal hyper- and training parameters, as well as a description of the transformation process of temperature profiles into a KOMPOT restart file, and a description of the CLI tool creation. More specific results on the performance of ML models and the ML-KOMPOT combination are presented in Chapter 6.

5.1. Feature Selection

The choice of input parameters for NN models is fundamental to modeling, as incorrectly chosen features (which is characterized, for example, by having low-informative features) or too many trainable parameters (Ayinde et al., 2019) can increase the complexity of a deep NN model, and the likelihood of overfitting, which is already high due to the limited data set used in this work. First, the parameter interactions were examined using the correlation matrix shown in Figure 5.1. The following observations were important for consideration:

- As the *Rpl* values in the data set used were chosen so that the planetary density remains constant at $\sim 5.5\ g/cm^3$ (Earth's density), *Rpl* and Mpl parameters are expected to show a perfect positive correlation (100%). This suggests to use only one of the two features because they carry similar information. Both parameters are weakly positively related to *Exobase Altitude* parameter.
- The parameters $X(CO_2)$ and $X(NO_2)$ were chosen with the expectation that $X(N_2) = 1 - X(CO_2)$, which explains the polarity of the also strong correlation between them. A weaker correlation is observed between these two parameters and *aOrbit*. Their barely noticeable correlation can also be observed with *Exobase values*, which increases their importance in the choice of features.
- A negative correlation between *aOrbit* and target / output parameters can be observed, especially with *Exobase Temperature*, which is consistent with the physical interpretation of the relationship and indicates that *aorbit* is an important parameter for predictions.
- Constant parameters were excluded due to their non-informative value and to simplify the model.

Figure 5.1.: **Pearson correlation heatmap of data set features and targets**: colors indicate the type of relationship. ExoAlt means *Exobase Altitude* and ExoTemp — *Exobase Temperature*.

Figure 5.2 shows the ranking of features according to their importance using XGBoost[1]. This is made possible by a tree-building process at the root of this method, where the parameters under consideration are chosen to partition decision trees in such a way that, for example, the quadratic error used as the evaluation criterion is minimized Chen (2016). The more often a parameter is used in such decisions, the more «important» it becomes. The data were split in a 80/20% scheme for the training and testing data sets, respectively; and normalized using the MinMax method so that all values, including the target values, were in the range between 0 and 1 to accelerate convergence.

The most influential variable in predicting target *Exobase values* was found to be *aOrbit*. $X(N_2)$ is shown as the second most important feature. *Mpl* was also frequently involved in function minimization. The remaining parameters that were insignificant or not used at all in the calculations, XGBRegressor proposes to neglect.

Based on the results of the correlation analysis and importance ranking, it was decided to use the following **input parameters for the Attention-based NN model: *Mpl*,**

[1]XGBoost documentation: `https://xgboost.readthedocs.io/`

aOrbit* and *X(CO$_2$). Since $X(CO_2)$ has a linear inverse relationship with $X(N_2)$ — a parameter that also distinguished its importance in the tests — and a high physical relevance, consisting in its key participation in the atmospheric cooling processes, it was preferred to $X(N_2)$. The importance of the *Mpl* parameter can be explained by the relationship between a planet's mass and its gravitational pull, because, for example, planets with larger masses and smaller radii tend to have stronger gravitational fields. This makes it possible to retain atmospheres, in other words, to prolong their survival time, because of the lower probability of their gases escaping into space (Kubyshkina et al., 2020), which directly affects the exobase determination. And the orbital distance, *aOrbit*, determines how close a planet is to a star by implicating the amount of radiation coming from that star, which plays one of the most important roles in atmospheric temperature gradients: for example, all else being equal, planets close to their stars tend to absorb more radiation, potentially accelerating hydrodynamic atmospheric escape, especially for planets with lower gravitational potential (Mo, 2018) — this physically justifies the inclusion of this feature in the input parameter set.

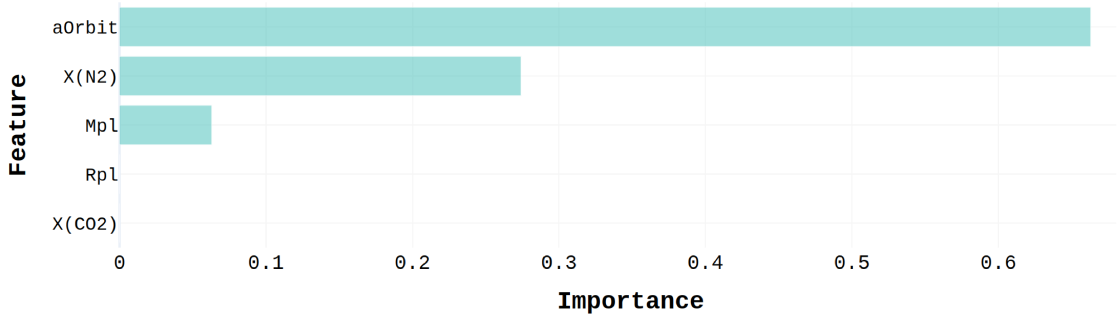

Figure 5.2.: **XGBoost Regressor feature importance analysis**: parameters are listed in descending order of importance.

Finally, an additional collinearity analysis of the VIF between these three parameters was performed, as shown in the Table 5.1, in which magnitudes of multicollinearity can be considered acceptable, statistically confirming the reasonableness of the choice. These coefficients, which now indicate a moderately problematic[2] collinearity, may have been overestimated due to the lack of data and, consequently, due to overfitting. In general, the ambiguity caused by collinearity can be reduced by the regularization methods for NN models. (Miles, 2005; Fox, 2019)

Using cascading, a case of ensemble learning in order to obtain better predictive performance, these three features were also used for the second ML model, specifically a **CNN model**, but in conjunction with the *predicted Exobase values* from the first model (***Mpl***, *aOrbit*, ***X(CO$_2$)***, ***Predicted Exobase Altitude* and *Predicted Exobase Temperature***) to set upper bounds on the predictions of the full temperature profiles,

[2]Problematic in the sense that strong correlation of variables can lead to redundant information, making the ML model less reliable.

Name of input feature	VIF coefficient
$\boldsymbol{Mpl}$	10.573
$\boldsymbol{aOrbit}$	5.387
$\boldsymbol{X(CO_2)}$	5.141

Table 5.1.: **VIF multicollinearity analysis results**.

thus reducing the complexity of the modeling problem. Additional feature engineering techniques, such as creating new features based on initial features and representing the relationship of multiple features simultaneously (for example, by multiplying them) were considered redundant because deep neural networks are better able to recognize complex parameter relationships, as opposed to simple models such as Random Forest, where feature manipulations are potentially more advantageous and worth considering. (Yu et al., 2015)

5.1.1. Feature considerations for future data sets

Hypothetically, with more diverse and real-world data in the future, three features will not be enough to describe all the complex relationships that lie in exobasic values. In such a case, the author suggests expanding the range of features to include:

- **Rpl** for a more holistic understanding of the influence of planetary characteristics on the behavior of atmospheres. The radius can be used as a basis for determining the scale height of the planet (via the gravitational acceleration formula), which describes the decrease in atmospheric pressure with height (Formula 3.5). For example, a study by Chin et al. (2024) discusses that for rocky exoplanets the atmospheric escape rate can be represented as a nonmonotonic function of the planetary radius, confirming the importance of this parameter in modeling atmospheres. In the present data set, *Rpl* was excluded only because of the strong direct dependence on the parameter Mpl, which negatively affects model performance (Farrar, 1967).
- **Tbase** and **nbase** (initial temperature and atmospheric density at the base) to delineate the boundaries of the temperature profiles predictions in case of diversity of these values. The actual constancy of the parameters served as an exclusion factor in the selection of features for the current data.
- **Additional mixing ratios of species** in the case of a larger initial number of gases in the atmosphere, such as methane (CH_4), water (H_2O), or sulphur dioxide (SO_2).
- A neater and more accurate representation of the effects of a star's radiation by multiple flux points at different wavelengths — flux is the amount of energy from a star per m^2 of surface area considered, or a measure of the integration over the angle at which the radiation rays fall on that surface of all intensity contributions (brightnesses) — each responsible for a particular bin of the spectrum, after it was

divided into such bins. All current KOMPOT models use the solar spectrum, so the correlation between points from different models is the same, since they differ only in the parameter *aOrbit* (due to the inverse square law, where the *intensity* $\propto 1/distance^2$). In other words, the influence of the spectrum changes only with distance. Therefore, the approach of including additional parameters of different fluxes is fraught with multicollinearity and was not used with this limited data set. However, for future data sets, it is worth using multiple values, because having only one parameter representing the spectrum in the input parameters can greatly overestimate the atmospheric heating. This can happen because the spectrum of a Sun-like star grouped at one point is dominated by the near-UV (due to its longer wavelengths), which does not contribute much to the upper atmospheric processes, participating instead in the surface heating. But the more distant UV already begins to interfere, for example with oxygen, even more distant UV (far-UV) with CO_2 and water vapor in the atmosphere; and another large group of emission lines past Lyman-alpha towards the X-ray (XUV) mainly affect the *Exobase Temperatures*. (Le et al., 2011; France et al., 2022)
In Frey (2023)'s master's thesis, the effect of using different numbers of XUV spectrum bins of the star GJ176 (between 10 and 1000 bins) on the speed and accuracy of the KOMPOT code was tested (see Figure 5.3). Two cases with two different water contents were considered: 1% and 0.5%. The results of his experiments show that there is almost no difference in accuracy between using 250 and 1000 bins. 100 bins still perfectly characterize the spectrum, and the simulation stabilizes much faster compared to 1000 bins. It is recommended to include **at least 10 flux points** in the input parameters of re-constructed prediction models for future data sets, which, according to the above-mentioned experiments, represent the full spectrum much better than a single point, even if they tend to overestimate the flux in some regions compared to 100 bins, as shown in Figure 5.3.

5.2. Hyperparameter Optimization

To find the best hyperparameters, the tool *optuna* was used, which automates the search process. Tables 5.2 and 5.3 show the range of parameters tested and the final results obtained by optuna, which were then used to construct and train NN models (the constructs are shown in Listings 5.1 and 5.2). The models and data sets were transferred as tensors to a GPU[3] for efficient resource usage and faster testing.

Upper and lower bounds on the hyperparameters for finding optimal configurations were chosen based on the complexity of the problem and the size of the trained data set, with the goal of creating a model that is not over-regularized and that ideally effectively captures the patterns of predictors with a lower probability of overfitting, based on empirically tested value ranges and recommendations from the following articles and books: Bengio et al. (2017); Pandey (2022); Pauls (2018); Vaswani et al. (2017); Kilichev (2023); Zhu et al. (2019); Wojciuk et al. (2024); Jiang et al. (2022); Fatyanosa (2020).

[3]NVIDIA GeForce MX150

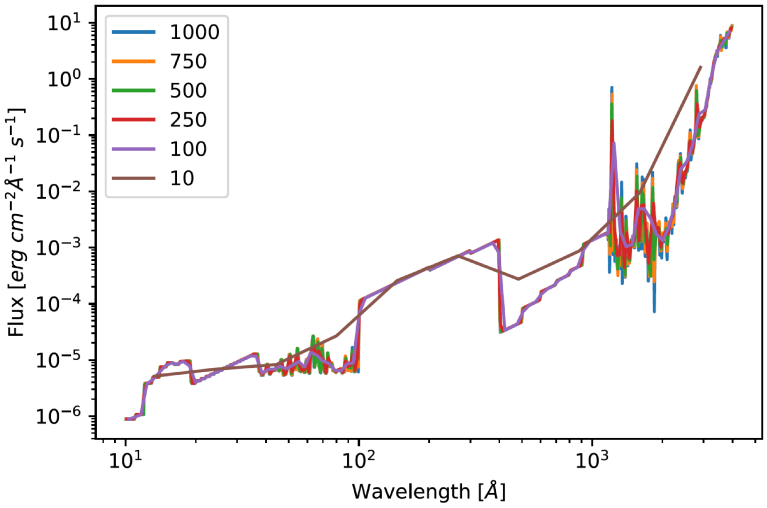

Figure 5.3.: **XUV spectrum of the star GJ176 with different numbers of bins studied for their effect on the speed and accuracy of KOMPOT simulations.** (Frey, 2023)

For the future larger data set, the author recommends increasing the upper value of the ranges of such searched hyperparameters as number of filters, hidden sizes, kernel, attention and step sizes, with less fear of overfitting, but taking into account the fact that it will affect the speed of finding optimal solutions.

```
NeuralNetWithAttention(
  (attention_weights): Linear(in_features=3, out_features=28, bias
   =True)
  (attention_combine): Linear(in_features=84, out_features=187,
   bias=True)
  (hidden_layer): Linear(in_features=187, out_features=187, bias=
   True)
  (dropout): Dropout(p=0.020295468412210447, inplace=False)
  (output_layer): Linear(in_features=187, out_features=2, bias=
   True)
)
```

Listing 5.1: **Attention-based NN architecture with detailed layer configurations**: 3 inputs (Mpl, $aOrbit$, $X(CO_2)$) and 2 outputs (*Exobase Altitude* and *Exobase Temperature*).

[4]Intel(R) 4-Core (TM) i7-8565U CPU @ 1.80GHz, 16GB RAM, SSD

Name of Parameter	Tested Range	Results / Best Parameters Used for Training
Hidden size	16 to 256	187
Attention size	1 to 30	28
Learning rate	1e-5 to 1e-2	0.00464
Dropout rate	0.01 to 0.5	0.02030
Step size (Scheduler)	1 to 500	132
Gamma (Scheduler)	0.1 to 0.9	0.54294
Epoch	Optuna: 1000	Training: 11000
Optimizer	Adam	Adam
Scheduler	StepLR	StepLR
Loss Function	MSELoss	MSELoss

Table 5.2.: **Optuna optimization and resulting parameters used to train the Attention-based NN model**: test run of 6 hours[4] with 100 hyperparameter combinations (n_trials) and MedianPruner configuration to stop unpromising trials early. The highlighted column shows the best test results, which were used to construct and train the NN network. The rows of the second column after the double line contain information about the parameters of the test itself, and the rows of the third column contain information about the training parameters, including the number of epochs, empirically chosen from the following range of values: [100, 500, 1000, 2000, 5000, 10000, 11000, 15000].

```
SimpleCNN(
  (conv1): Conv1d(1, 34, kernel_size=(2,), stride=(1,))
  (bn1): BatchNorm1d(34, eps=1e-05, momentum=0.1, affine=True,
   track_running_stats=True)
  (hidden_layer): Linear(in_features=136, out_features=253, bias=
   True)
  (dropout): Dropout(p=0.061575615417825504, inplace=False)
  (output_layer): Linear(in_features=253, out_features=806, bias=
   True)
)
```

Listing 5.2: **CNN architecture with detailed layer configurations**: 5 inputs (Mpl, $aOrbit$, $X(CO_2)$, predicted *Exobase Altitude* and *Exobase Temperature*) and 806 outputs (403 profiles: 403 altitudes and 403 corresponding temperatures, from 50 km base up to exobase).

5.3. Model Training

The data was split 85/15% with random shuffling (for avoiding bias) to allow the model to learn as much of the data as possible. Train and test data sets were normalized within

Name of Parameter	Tested Range	Results / Best Parameters Used for Training
Hidden size	16 to 256	253
Number of filters	16 to 128	34
Kernel size	1 to 5	2
Learning rate	1e-5 to 1e-2	0.00086
Dropout rate	0.01 to 0.5	0.06158
L1 regularization	1e-5 to 1e-2	3.70983e-05
L2 regularization	1e-5 to 1e-2	0.00082
Delta (HuberLoss)	0.1 to 3	1.00168
Epoch	Optuna: 1000	Training: 10000
Optimizer	Adam	Adam
Loss Function	HuberLoss	HuberLoss

Table 5.3.: **Optuna optimization and resulting parameters used to train the CNN model**: a 10-hour test run[4] with 100 hyperparameter combinations. The description of the content corresponds to the further description of Table 5.2.

the $(0, 1)$ range and separately using *Scikit-learn's MinMaxScaler* to avoid data leakage. This would be the case during training, when the neural model has access to additional, unexpected information from the test data, leading to an overestimation of the model's predictive ability by the evaluation (Kaufman et al., 2012). The reason for this decision is the different variability of input and output data, as shown in Figure 5.4, because neural networks, including those with attention mechanisms, benefit from scaling that ensures an equal contribution of all input features, which prevents, for example, features with a large magnitude from dominating during training (LeCun et al., 2002).

Despite the fact that in Figure 5.4 we have obvious outliers[5] in the output parameters, MinMax was a successful choice as the experiments showed. However, if there are still more cases of large outliers in the data, it is recommended to try a less outlier-sensitive scaling strategy, such as Scikit-learn's RobustScaler (Buitinck et al., 2018). In this work, no significant difference was found between using MinMax and RobustScaler through experiments.

Normalized data was converted to tensors so that PyTorch NN models could efficiently manipulate the data, and there was a quick option to switch to GPU or back to CPU when needed. For the CNN work, the decision was made not to normalize the data, as no differences were found between using MinMax, Robust scaled, or unscaled data. The reason for this may be that the use of the Batch Normalization (BN) method, which normalizes the output data layers in each batch during training (which can mitigate problems caused by non-scaling), proved to be sufficient for this model (Ioffe, 2015). Then the training process of the two models was started, taking into account the details

[5]The outliers in this data set represent the lack of such cases, not the incorrectness

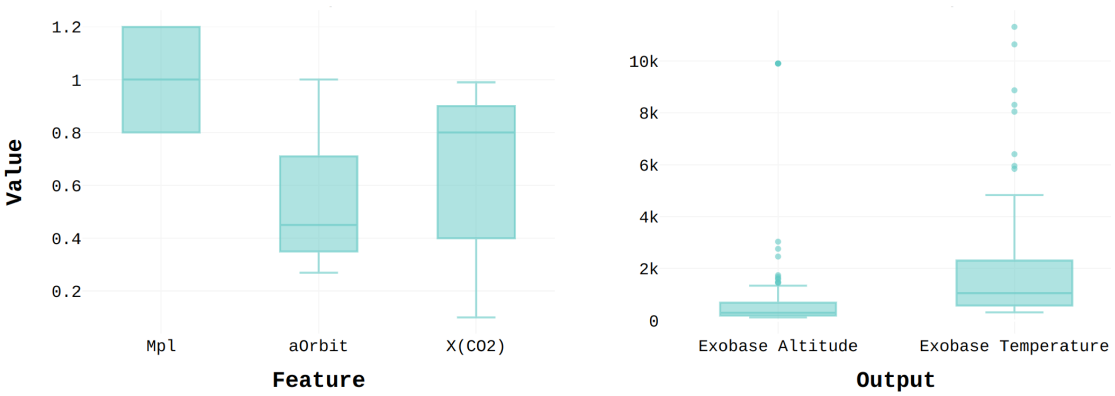

Figure 5.4.: **Distribution of input and output variables**: shaded boxes represent the interquartile range (25th to 75th percentiles), whiskers show the range within 1.5 times the IQR, and small bullet points indicate outliers.

presented in Tables 5.2 and 5.3. It is important to note that the CNN model, like the Attention-based NN model, was trained on previously known (true) *Exobase values*, but was evaluated after training on **predicted Exobase values** from the Attention-based NN. The results of the training process are shown in Figures 5.5 and 5.6.

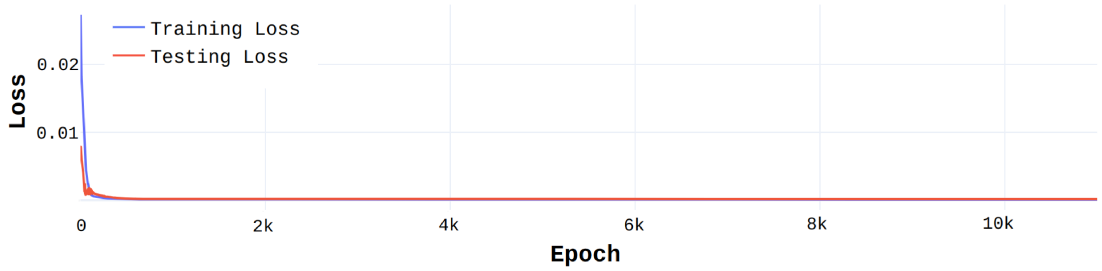

Figure 5.5.: **Attention-based NN model: Train and test losses over epochs**.

Both training processes show fast convergence and both models achieve good accuracy results even in the early stages of training. When Figure 5.5 is zoomed in, minimal overfitting of the NN with attention can be observed. This was predictable due to the limited data for this task and the use of intentionally complex models that, more scientifically significant for the goals of this work, should simultaneously fit future large and diverse data (but where the hyperparameter tuning process will have to be repeated).

The CNN performs marginally better on the unseen data than on the training data (although the difference appears to be slightly larger here compared to the Attention-based NN training due to the different scales used), confirming that the configurations chosen for the model are sufficiently optimal. The author of this thesis suggests that the reason for this may also be due to the previously discussed BN (in combination with a small data set), which uses mini-batch statistics and adds a small random value for numerical stability in

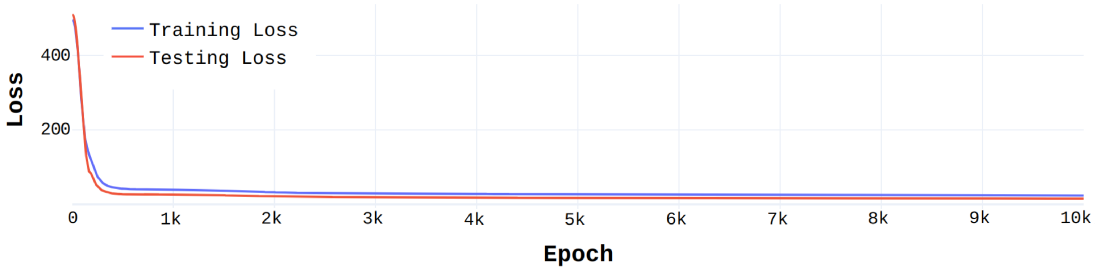

Figure 5.6.: **CNN model: Train and test losses over epochs.**

training, but not in the evaluation process, where BN uses the whole population statistics (exponential moving average) (Huang et al., 2022; Wu and Johnson, 2021), which may happen to be more representative than mini-batch statistics due to the small batch sizes used during training.

In both cases, despite fast convergence, training was not terminated earlier to give the models time to fine-tune the weights of their layers, to better recognize complex patterns of parameter interactions in the limited data set used, and to test the robustness of the models' performance. This approach can cause the model to learn the training data too well and predict new, unseen data poorly (Jo, 2021). In this work, empirical testing of training showed that extended training improved rather than degraded model performance. The optimal number of epochs was chosen by comparing the evaluation metrics (the results of which can be found in Chapter 6) of the models' predictive abilities on test data. The validity of this approach should be re-examined for future new data sets.

5.4. KOMPOT's Restart File

The purpose of creating a restart file is to start the simulation not from initial conditions, but from a reference point based on initially known planetary atmosphere parameters and previously predicted altitude-temperature values (ML-informed initial estimates), in order to correct for inaccuracies in these predictions and to speed up KOMPOT's subsequent simulation of the chemistry and atmospheric escape processes — a complete physically consistent model.

The header of this file contains information about the total number of atmospheric layers simulated and the number of species considered. Each of these atmospheric layers is described by its:

- radial distance from planet center
- total atmospheric mass density
- momentum density

- neutron, ion and electron total energy (meaning thermal and kinetic) densities of gases
- and by the density of each species in this layer.

The code provided for use by Gwenaël Van Looveren first reads the species names and their masses and specific heat ratios, or gammas, from a prepared file as input parameters. The initial densities for each species at the 50 km base are calculated using the input mixing ratios, the input number density value and species masses. This is followed by the calculation of species densities for each successive layer of the atmosphere up to the exobase, taking into account changes in gravitational acceleration and temperatures that reflect how quickly atmospheric pressure decreases with altitude using the scale height definition. The changes in momentum values are based on the sum of these densities multiplied by their molecular mass (total mass density) and the pre-calculated Parker wind velocities for each layer. Taking into account all the previous calculations, the total energy for each particle in each layer is calculated, also taking into account the input specific heat ratios of the gases.

5.5. ExoTempPro_predictor tool

The functionality described in Proposed Contribution 1.5 has been programmed in the command line interface (CLI) tool «ExoTempPro_predictor», using a modular architecture that divides the project into manageable parts (see the project structure on the next page). This methodical organization has already been described by Parnas (1972) and remains relevant because of the possibility of independent development and testing, which makes the project scalable. Version control is also possible for collaborative programming using Git (Loeliger, 2012), which is facilitated by the presence of the necessary documentation written by the author of this thesis. Dependencies (libraries used) can be easily and conveniently managed with the help of created conda / pip environment configuration files, making the builds of this software tool reproducible (Moraila et al., 2014).

```
... src/
... ... data_preparation/ # Module for dataset handling
... ... model/            # ML models and utilities
... ... prediction/       # ML prediction functions
... ... restart_file/     # Restart file creation
... ... utils/            # Helper utilities
... ... visualization/    # Visualization scripts
... data/                 # Sample datasets
... ... example_inputs/   # Example input datasets
... ... simulations/      # KOMPOT models for comparing
... results/              # Storage for outputs
... notebooks/            # Methodological Jupyter notebook
... ExoTempPro.py         # Main executable
... README.md             # Project documentation
... requirements.txt      # Pip dependencies
... environment.yml       # Conda dependencies
```

Listing 5.3: **Python Project Directory Structure and Logic**: demonstrating the modularity of the ExoTempPro_predictor CLI tool.

6. Findings and Discussion

This chapter presents the results of code optimizations, AI-driven altitude-temperature profile modeling, and the symbiosis of ML and KOMPOT code in the context of predictive and computational performance.

6.1. Code Optimization Results

For this performance test, the author of this thesis ran the KOMPOT simulation with 4 threads, using the standard parameters described in Johnstone et al. (2018), for both the initial original code and the optimized code without ML models (each run was repeated 3 times to ensure consistency): $Mpl = 1.0$, $Rpl = 1.0$, $aOrbit = 1.0$, $X(N_2) = 0.78110$, $X(O_2) = 0.20955$, $X(Ar) = 9.343e{-}3$, $X(He) = 5.2417e{-}6$, $X(H_2O) = 6.0e{-}6$, and $X(CO_2) = 4.0e{-}4$. The experiment showed (see Table 6.1) that maintaining the same level of code accuracy and stability, there is a notable **improvement in performance**, with a **3.63-fold increase**.

KOMPOT Type	Average Execution Time, [h]	Speedup
Original	3.09	**3.63**
Optimized	0.852	

Table 6.1.: **Computational performance comparison of baseline and optimized KOMPOT**: execution times of default KOMPOT simulation after 275,000 iterations in parallel using 4 threads and the final speedup of the code optimizations performed[1].

It should also be noted that the **original 1-thread simulation** reached 250,000 iterations in 8.7 hours, which is **10.21 times slower than the optimized code on 4 threads** (including the original and additional parallelization parts).

After additional exclusion tests, it was found that the increase in code performance is mainly due to the use of the OpenBLAS *dgesv* function as a replacement for the Gaussian elimination subroutine implementation, along with the use of compiler flags. With the same number of Gaussian Elimination calls (243,808,420 calls as shown in Figure 4.2), the percentage of total run-time used by this function dropped from 79.94% to 0.25%. However, the chemistry part remains the main bottleneck in the code, consuming approximately 52% of the total execution time, especially the function for calculating the coefficients for the Rosenbrock method and the function for calculating the rate of

[1] Intel(R) 4-Core (TM) i7-8565U CPU @ 1.80GHz, 16GB RAM, SSD

change of each species after chemical reactions, making them prime candidates for further optimizations. The next most computational resource-consuming process is the conduction process, namely, the function of updating the energy density of the atmospheric gas.

6.1.1. Limitations

Despite the fact that using the OpenBLAS scientific library accelerates the simulation process, the problem of portability and achieving the acceleration factor previously demonstrated in this Chapter arises when considering the parallelization of KOMPOT through OpenMP, which promises a significant reduction in execution time for similar platforms[1], and when using at least 4 threads, which is not always possible for an end user. However, even using 2 threads would imply a notable performance improvement.

6.2. ML Modeling Results

Based on the results demonstrated in Tables 6.2 and 6.3, **both NN models** show high fit in training and testing, with $\boldsymbol{R^2 \simeq 0.99}$ for both measures: altitudes and temperatures. The performance of the NN with attention degrades on the test set, as also evidenced by the increase in the mean bias deviation metric, suggesting the possibility of overfitting and that the model slightly overestimates values on average, especially altitudes. The CNN also tends to slightly overestimate the true values, but more so due to the overestimated Attention-based NN's predictions used in the input feature set. The CNN itself (without predictions but with true *Exobase values*) shows better performance and robustness in predicting the altitude set, and to a lesser extent the temperature set, where temperatures tend to be weakly underestimated. These results prove the appropriateness of choosing these architectures, especially CNNs for dealing with time series like data.

Data set	**RMSE**	**MAE**	**Mean Relative Error, [%]**	**Mean Bias Deviation**	**R^2**
Altitudes					
Train	143.969	58.088	6.298	-9.545	0.994
Test	212.621	75.234	5.909	37.351	**0.9891**
Temperatures					
Train	170.35	82.29	4.446	-3.231	0.993
Test	174.732	108.221	5.284	14.236	**0.9929**

Table 6.2.: **Performance metrics of the Attention-based NN model**.

To examine the predictive ability of the NN with attention model in more detail, the residual analysis in Figure 6.1 and the relative error analysis in Figure 6.2 were performed. The main observations are that the **prediction errors** are relatively small, **in the ±10% error range**, which is considered acceptable for the purposes of this work, and are consistently distributed around ideal values (around zero). The presence of expected outliers is explained by the lack of training data for these cases.

Data set	RMSE	MAE	Mean Relative Error, [%]	Mean Bias Deviation	Weighted R^2
Altitudes					
Train	1.484	0.938	0.89	0.15	0.999
Test (True Exo)	1.742	1.11	0.952	0.104	**0.999**
Test (Pred Exo)	53.743	10.859	2.703	4.709	**0.9892**
Temperatures					
Train	38.275	16.539	4.668	2.281	0.998
Test (True Exo)	55.565	23.447	4.746	-5.185	**0.998**
Test (Pred Exo)	103.85	48.417	6.135	3.156	**0.9932**

Table 6.3.: **Final result: Performance metrics of the CNN with true Exobase values vs. with Exobase values predicted by the attention-based NN in input predictors**. One test set is considered for both cases: «True Exo» — using actual Exobase values, «Pred Exo» — using Exobase predictions.

This Attention-based NN model is very sensitive to changes in orbital distances, and these distances, *aOrbits*, are the most important features in the NN's prediction decisions, especially for temperatures, as shown in Figure 6.3, which is consistent with the physical interpretation of the phenomenon. This tendency also explains the slight overestimation of the values by the model, emphasizing the importance and the insufficiency of using only one value to represent the influence of stellar irradiance. As previously described in Chapter 5, the choice of a single parameter is associated with a small set of available data, which will change as the number of atmospheric models increases, increasing the need to use multiple values to better represent the full spectrum. The second feature in terms of influence is $X(CO_2)$, which is more involved in decision making after orbital distance, especially in altitude predictions. The model considered the influence of mass, Mpl, on Exobase definition as tertiary. The explainable XGBoost model used to select initial features had a similar decision path (see Figure 5.2).

The use of the SHAP method can then be used to find threshold values for parameter interactions, such as between $X(CO_2)$ and $aOrbit$, beyond which the behavior and composition of the atmosphere changes abruptly and it begins to lose mass more rapidly. This could be informative for planets that are, for example, farther away from their stars but still have a chance of having an ocean due to the greenhouse effect if the CO_2 concentration is sufficient. Thus, this NN model, trained on a larger and more diverse data set, can be useful in determining the limits of the habitable zone and in understanding star-planet interactions. An example of a visual representation of the SHAP method is shown in Figure 6.3.

Regarding the CNN model, 2 examples of atmospheric profiles with different irradiance levels are shown in Figure 6.4. In the first case, the true *Exobase values* were used to evaluate the predictive power of this model alone. In the second case, the *Exobase values* predicted by the first, Attention-based NN model were already used in the input parameter set for the CNN, evaluating the combined predictive ability of the 2 models

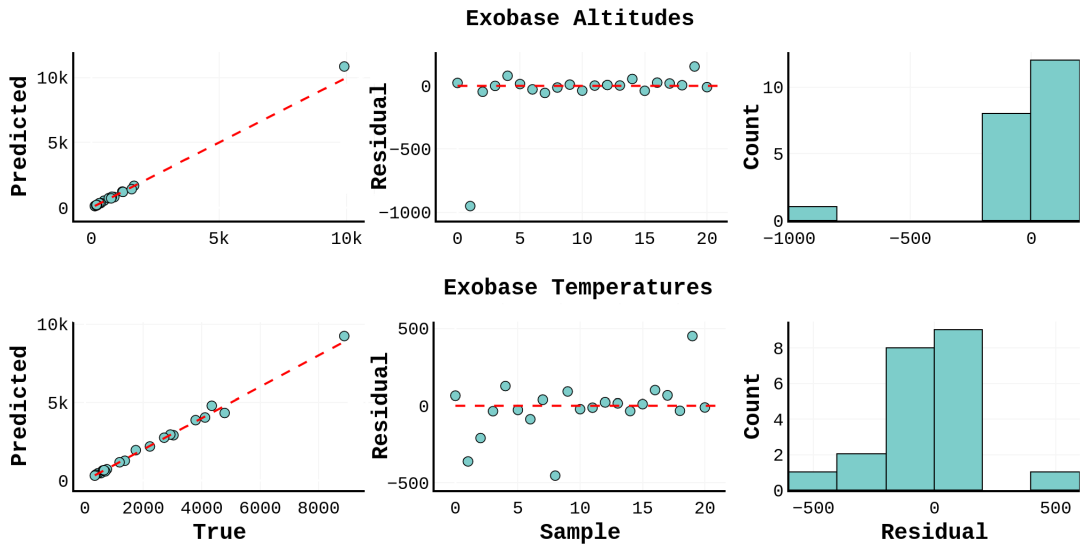

Figure 6.1.: **Residual plots for evaluating the attention-based NN model's performance (Test data set)**: the first row evaluates the predictions of *Exobase Altitudes* (km), the second row — *Temperature* predictions (K). The scatter plots on the left compare predicted and actual values, while the middle plots show the prediction errors, which are detailed in the histogram on the right. The middle «Sample» axis represents one model / planet. Red dashed lines serve as a reference point for perfect predictions.

simultaneously. As can be seen in these examples, the predicted profiles are in strong agreement with real profiles, but the overall quality of the predictions depends on the accuracy of the Exobasic predictions of the first NN model.

In the Appendix, you can see the evaluation results of the NN models for the training set (Figures A.1-A.3).

To better estimate the CNN's prediction power visually, scatter plots were created to compare the predicted values with the real ones in cases where the Exobase lies between 0-200 km, then between 200-800 km, 800-1700 km, and in the range of 10,000 km.

As you can see on Figure 6.5 and 6.6, the models synergy has a high prediction accuracy at altitudes of 50-200 km with an absolute error $\leq$ 40 km and their corresponding temperatures of 100-700 K with an absolute error $\leq$ 100 K, reflecting a good understanding of the process of temperature degradation observed in the mesosphere layer and temperature inversion starting in the thermosphere. Stronger deviations are observed for planets with an Exobase of 800-1700 km and 2700-5000 K. On the parameter side, the dual-NN concept so far best predicts temperature profiles on average for planets with Earth radius and mass, an average *aOrbit* of 0.77 AU and a carbon dioxide concentration of about 56%, slightly worse in the range of absolute errors $\leq$ 130 km and $\leq$ 300 K for the same planetary conditions but with a lower *aOrbit* of 0.38 AU and a higher $X(CO_2)$ of 91% on average. When both *Mpl* and *Rpl* begin to increase, and the CO_2 concentration becomes less than 70%, at an *aOrbit* of $\sim$ 0.71 AU, the predictive ability of the model begins to

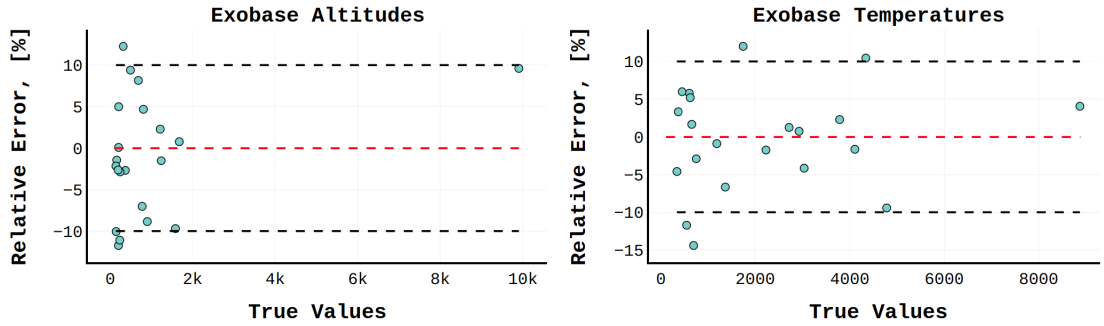

Figure 6.2.: **Relative errors distribution across true values for evaluating the attention-based NN model's performance (Test data set)**: the first plot shows prediction errors for *Exobase Altitudes* (km), and the second plot — for *Exobase Temperatures* (K). The red dashed line indicates perfect predictions, and the black dashed lines — error boundaries from −10% to +10%.

degrade more significantly with maximum absolute errors by ∼ 200 km and ∼ 700 K. The worst models' performance can be seen for a planet with an Exobase value pair of ∼ 10,000 km-9000 K, where the mass of the planet is 1.2 Earth masses, the radius — 1.06 Earth radii, the *aOrbit* is 0.28 AU, and the CO_2 concentration level — 80% (absolute errors ≤ 1000 km for altitudes and ≤ 600 K for temperatures), which suggest creating more cases with more extreme conditions, including higher irradiance cases, to re-train the networks.

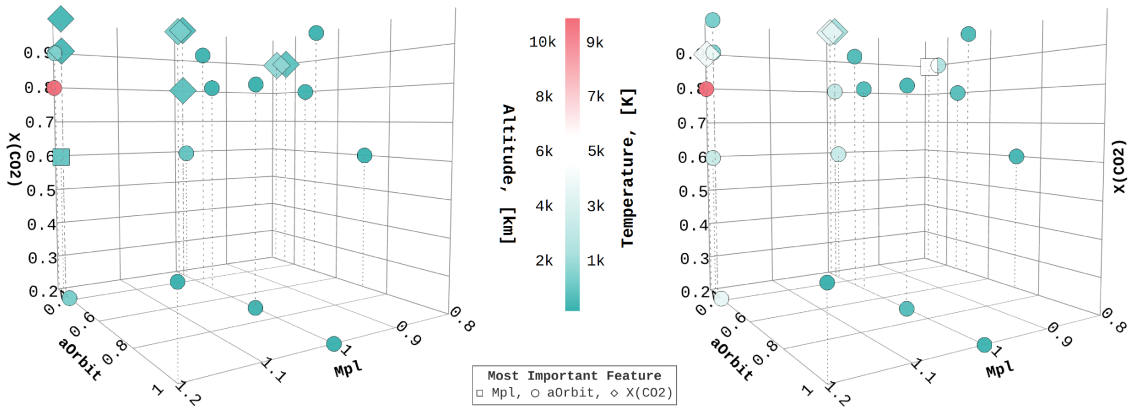

Figure 6.3.: **Importance of features and their correlations with targets in attention-based NN's predictions (Test data set)**: 3D plots show the planetary features and their corresponding *Predicted Exobase Altitudes* (left) and Temperatures (right) using color gradations. Each point's shape indicates the most influential feature during model predicting based on the game theoretic SHAP approach. Dashed lines are projections.

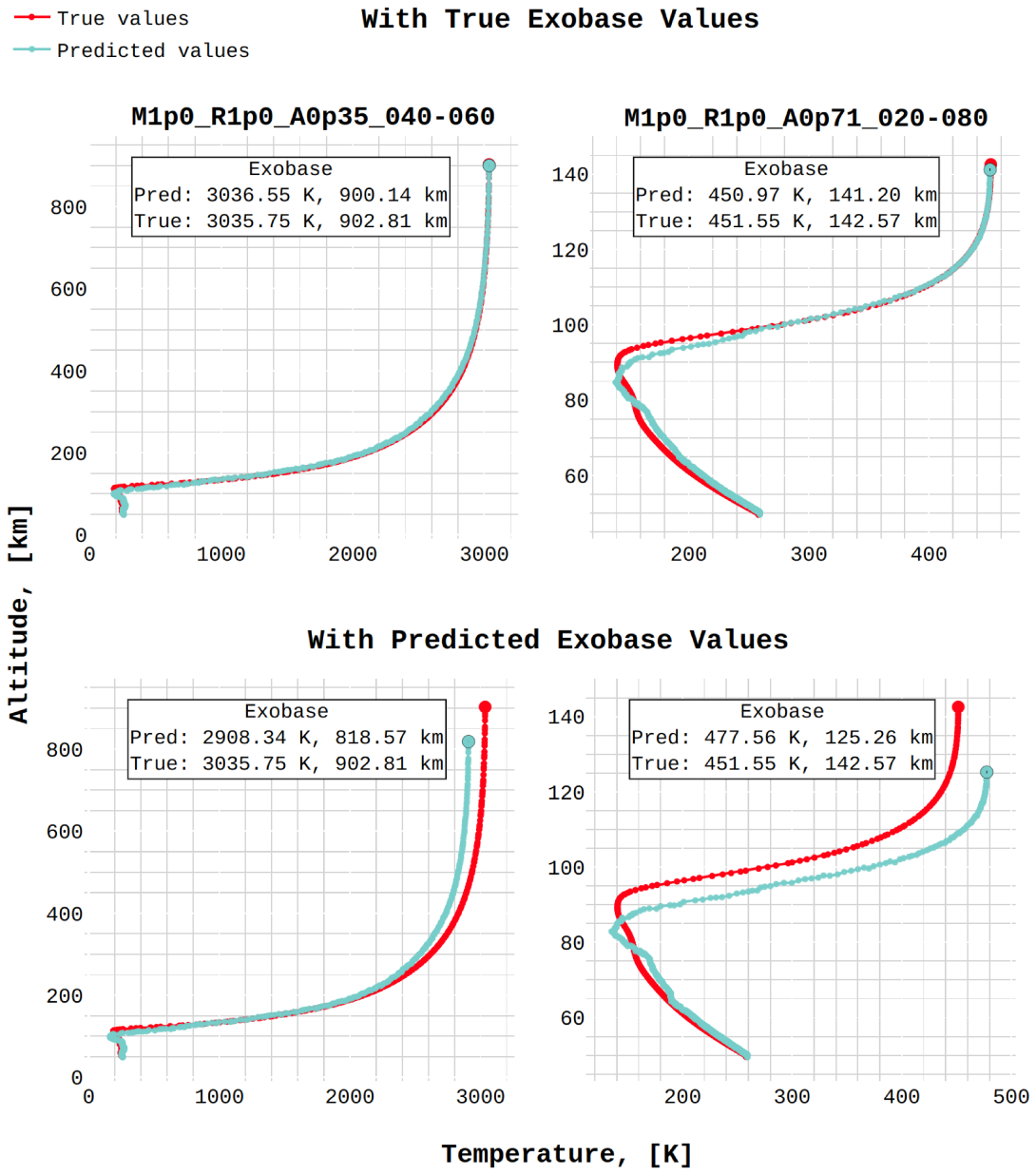

Figure 6.4.: **Altitude-temperature profiles' CNN predictions of 2 atmospheric models from Test data set with differing irradiation levels**: the first row shows the predictions made using *true Exobase values*, the second row — using previously *predicted Exobase values* by the attention-based NN as part of the input features for CNN. 2 items in the first column represent the same model, similarly in the second column. Red dots are true values, turquoise — predicted values. In the *Exobase values* box, «True» means *actual Exobase values*, «Pred» — final predicted last values as *Exobase values* by CNN. The atmospheric model name is formatted as follows: «p» — point, «M» — mass, «R» — radius, «A» — orbital distance, followed by the initial mixing ratio of N_2, and concluding with $X(CO_2)$.

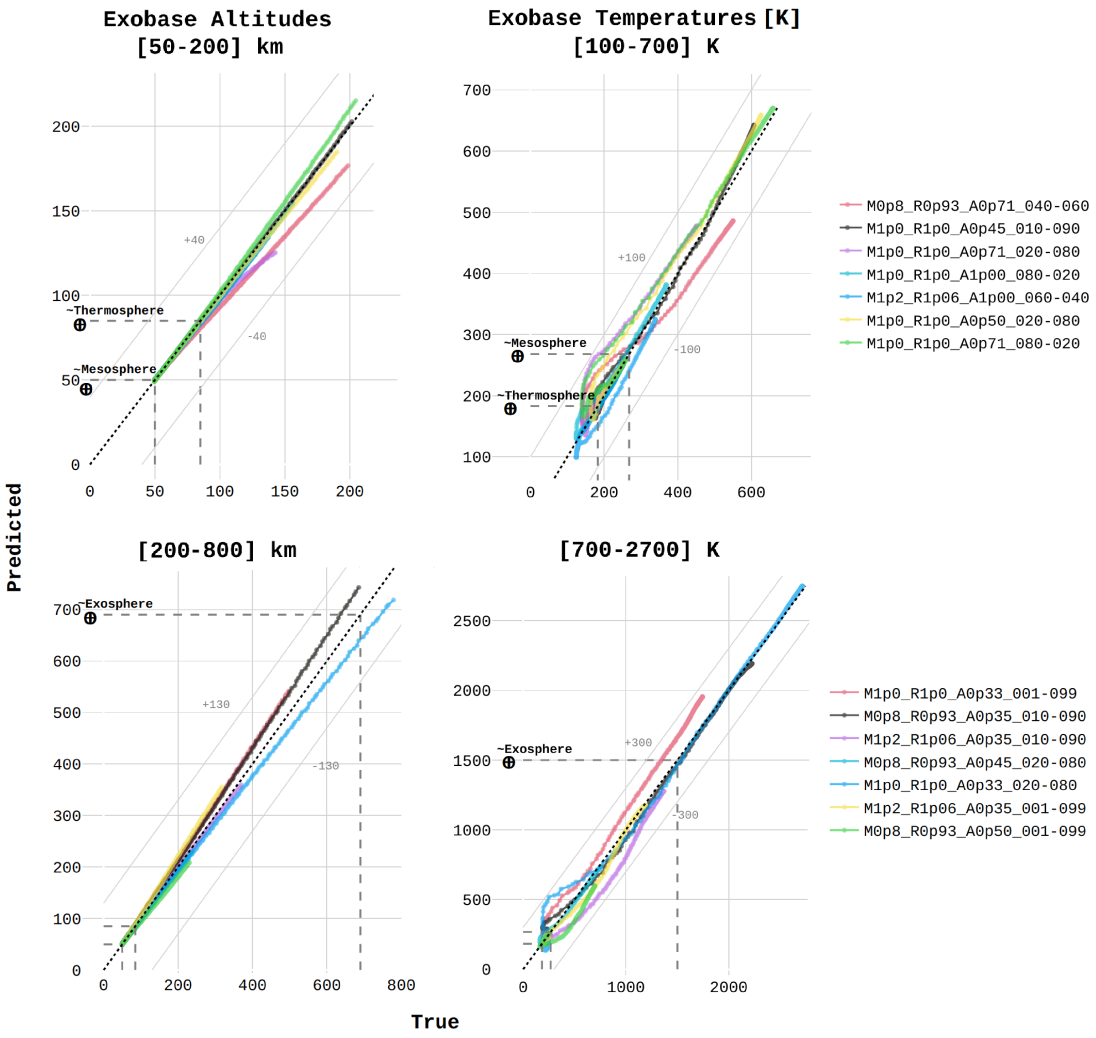

Figure 6.5.: **CNN model evaluation scatter plots: predicted vs. true altitude-temperature profiles with Exobase up to 800 km / 2700 K (Test data set with *Exobase values* predicted by the Attention-based NN)**: the first row shows thermal profile comparisons for atmospheric models with Exobase up to 200 km, the second row — from 200 to 800 km. The plots on the left show Altitude predictions from 50 km against the actual values, while the plots on the right — Temperatures comparisons from 258.2 K. Dashed diagonal lines indicate perfect predictions. Other dashed lines mark the approximate beginning of the atmospheric layers according to Figure 1.2: the first line counting from zero — Earth's mesosphere, the second — Earth's thermosphere, and the third — Earth's exosphere. The last profiles dots are *Exobase values*. In the second column, the approximate atmospheric Earth's temperatures at the beginning of the indicated layers are shown. Two grey lines parallel to the diagonal delineate the prediction errors, labeled with their maximum magnitude, where points in the «+» positive area indicate CNN model's overestimations, and in the «-» negative — underestimations. Different colors represent different atmospheric models. The first top plot legend refers to the first two plots, and the bottom legend — to the last two plots.

The average prediction times measured 1000 times for 1000 models are as follows: Attention-based NN — 0.0019 seconds, CNN — 0.0054 seconds. The **prediction of 1 model for both networks** is also **within the range of 1e-3 seconds**.

6.2.1. Limitations

These NNs, while showing strong predictive power, are still slightly overfitted due to the limited number of atmospheric simulations available for the given deep learning task, which means that the models perform well on similar scenarios, but are expected to perform worse on larger planets with higher potential to be more irradiated or on planets orbiting, for example, a K-type star, requiring the creation of a larger and more diverse data set, and re-building and re-training the NN models to obtain a more robust solution. In any case, poorly predicted thermal structure profiles can accelerate the KOMPOT code as well, although not as much as if the profiles were predicted more accurately.

6.3. Combined Impact on KOMPOT: Code + ML Estimates

In order to reliably quantify the performance changes resulting from the integration of the ML altitude-temperature profiles predictions into the optimized KOMPOT, the average runtime of 2 types of simulations that were run until the KOMPOT code reached stable results (each run was also repeated 3 times to ensure consistency) was calculated: a *low irradiation case* similar to Earth *(1 AU)* and a *high irradiation case* similar to Mercury *(0.28 AU)*. The other parameters were the same for both simulations to maintain controlled conditions: $Mpl = 1.0$, $Rpl = 1.0$, $X(CO_2) = 0.20$; and $aOrbit = 1.0/0.28$.

Simulation type	**Average Execution Time, [h] / Speedup**		
	Non-Optimized Code (extrap.)	**Optimized Code + ML estimates**	**Optimized Code**
Low Irradiation (similar to Earth)	15.12	3.54	4.2
Speedup		1.2	
	4.3		
High Irradiation (similar to Mercury)	43.7	6.69	12.14
Speedup		1.8	
	6.5		

Table 6.4.: **Computational performance comparison of baseline and optimized KOMPOT code with and without ML estimations**: execution times of 2 types of simulations in parallel using 4 threads, and final speedup compared to non-optimized (extrapolated) and optimized environments. The highlighted column shows the final result considering all optimization manipulations, the highlighted cells / rows — the speedup[1].

The provided experiments indicate that the use of machine learning-informed initial estimates allows the code to reduce computational time significantly. As illustrated in the Table 6.4, *for a low irradiation simulation*, the time has been reduced from about 15.12 hours with non-optimized KOMPOT (used as a baseline without compiler optimizations and scientific libraries), or from 4.2 hours with optimized code, to 3.5 hours with optimized code and ML predictions. In the case of a *high irradiation simulation*, the results are even clearer, with a reduction in computation time from about 43.7 hours with baseline KOMPOT, or from 12.14 hours with optimized KOMPOT, to 6.69 hours with accelerated code and ML estimates. These results indicate **accelerations by a factor of 1.2 to 1.8 with optimized KOMPOT**.
The ***final result of all the code optimisations and the integration of ML predictions***, that is, if we compare it with the non-optimised KOMPOT with 4 threads, is an ***acceleration by a factor of 4.3 to 6.5*** for the low and high irradiation cases, respectively. A comparison of the final optimized code on 4 threads with the initial code on a 1-thread would indicate a speedup factor of approximately 12.1 to 18.3.

A minor advantage of using predictions as a restart file is that only one simulation is required for KOMPOT to reach the desired high irradiance level, whereas without predictions one would have to first run a simulation with a lower orbital distance value, then use the restart file generated by KOMPOT itself after completion of that first simulation to run the code for a higher irradiance case, and repeat the process until the desired irradiance level is reached. This is because the isothermal assumptions become inadequate due to significant temperature gradients within the simulation caused by the high irradiance, which in the worst case can lead to the termination of the simulation. But with a restart file created from the predicted temperature profiles, it is possible to run such a simulation immediately without any additional steps.

6.3.1. Limitations

It is important to note that the ML models used only predict the thermal structure of the atmosphere. The initial estimate for the KOMPOT still uses an altitude-homogeneous chemical composition, which is currently the largest time sink in the reported values. By training another CNN on the chemical profile in the same way as the thermal profile, the initial estimate will be even closer to the stable solution. This is expected to result in another reduction in computation time, on the same order as the factors found in this work (1.2 to 1.8).

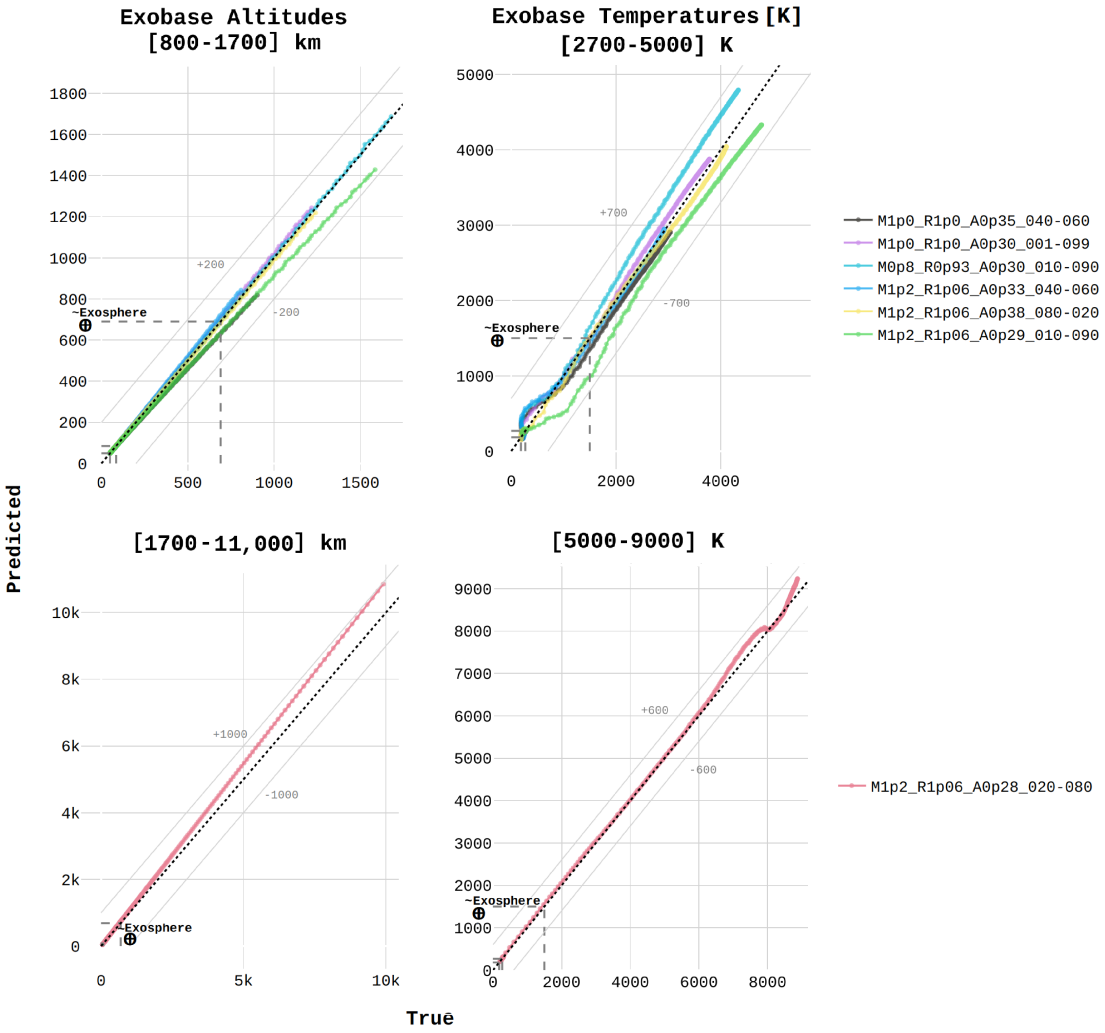

Figure 6.6.: **CNN model evaluation scatter plots: predicted vs. true altitude-temperature profiles with Exobase from 800 km to 11,000 km / from 2700 K to 9,000 K (Test data set with *Exobase values* predicted by the Attention-based NN)**: the first row shows thermal profile comparisons for atmospheric models with Exobase from 800 km to 1700 km, the second row — from 1700 km to 11,000 km. The rest of the plots is the same as described in Figure 6.5.

7. Conclusion

The study and modeling of planetary atmospheres remains our primary tool for peeking into the past of planets and opening doors that may lead to habitable worlds. Modern technologies allow us to experience a unique boom in the discovery of new exoplanets and their characteristics.

For example, recent data from JWST confirm the importance of non-equilibrium effects on processes in the upper atmospheres, suggesting that computational solutions that incorporate photochemistry into chemical and thermal profile calculations are worth considering when studying these delicate matters (Dyrek et al., 2024; Tsai et al., 2023). This is what distinguishes the self-consistent KOMPOT code from others. As the main advantage, it is also the main disadvantage of the code in terms of computational performance. The **main aim** of this thesis **was to accelerate the code without compromising the accuracy** of its results.

Rosenberg's previous research (Rosenberger, 2021) has shown the great potential of using machine learning to replace numerical calculations of thermal profiles, the first step in characterizing exoplanet atmospheres. Despite its high predictive accuracy, his MPL NN solution, while trained on a large number of atmospheric models, was created with a simplified KOMPOT by reducing chemical complexity, which can lead to the omission of critical processes.

In this thesis, it was decided to use the unsimplified code to create a grid of models covering most of the planets of the TRAPPIST-1 star, which is of great interest to the scientific community because it is the first planetary system discovered that has seven Earth-size planets around a single star (Gillon et al., 2017). The **hypothesis** (see Introduction 1.4) that the **use of high performance computing** techniques and the integration of **pre-built machine learning models** into the simulation process will significantly improve the code's performance **was proposed and subsequently confirmed** by experiments.

Computational optimizations were achieved by profiling the code with gprof and identifying performance bottlenecks, using compiler optimization flags, replacing manually programmed Gaussian elimination with LU with partial pivoting from OpenBLAS, and additional code parallelization with OpenMP. The second part of the methodology, the *AI-driven part*, used cascading techniques from deep ensemble learning that combined different types of NNs to leverage their individual strengths for more accurate predictions. The modulated approach reduced task complexity, simplified the learning process, and increased manageability, which refers to independent updates and scalability, and minimized error propagation across tasks. (1) In the first step of the approach, an Attention-based Neural Network was used to predict the *Exobase Temperature* and *Altitude* by fitting the hyperplane to the array of input parameters. This model focused on significant features

by weighting the importance of various inputs differently, which is crucial for capturing complex nonlinear interactions in atmospheric data. Features selected through feature engineering techniques for this stage included planetary mass, CO_2 mixing ratio, and orbital distance. (2) These *predicted Exobase values*, combined with the previously used input parameters, were passed through a second network, namely a Convolutional Neural Network, to generate the complete altitude-temperature profile by predicting the «shape» of the profiles from the established base to the previously *predicted Exobase values*.

Furthermore, previous work was advanced by using these predictions as ML-informed KOMPOT initial estimates, which were converted into a KOMPOT restart file, and served as the input base for running the simulations. This created a feedback loop that not only iteratively corrected inaccuracies in the prediction results but also led to finding a whole physically consistent model with fewer iterations needed to achieve stability than without using NN models.

It was found that the calculation of chemical processes is the main performance bottleneck of the code. As a result of computational optimizations, the code managed to speed up 3.63 times, or 10.21 times if we compare the optimized code on 4 threads with the code without parallelization on 1 thread.

Despite a slight expected tendency to overestimate the predictions, both NNs showed a high level of goodness of fit, where the R^2 was on average 0.99 for both networks, and the RMSE was no greater than 213 for exobase and no greater than 104 for full thermal profiles. The relative prediction errors of the *Exobase values* varied within an acceptable range of $\pm 10\%$. The CNN separately showed better results, neatly covering even the interval of temperature inversions at lower altitudes, but its results depend directly on the quality of the first model's predictions. The thermal profile predictions take about $1e-3$ minutes for 1-1000 models. The use of machine learning-informed initial estimates allows the code to reduce computational time significantly: by a factor of 1.2 for the low irradiation case (similar to Earth) and by a factor of 1.8 for the high irradiation case (similar to Mercury).

In the end, after all the optimization manipulations, **the code runs 4.3 and 6.5 times faster for the low irradiation and high irradiation cases**, respectively. If we **compare it with the code running on 1 thread**, the **resulting factors** will be in the approximate interval: **from 12.1 to 18.3.**

This thesis significantly extended the previous research of Rosenberger, which was described in Chapter 2, in the context of predicting temperature profiles. It was demonstrated that even with a limited data set generated by the original, non-simplified code (136 profiles compared to the Rosenberger's 1000 profiles simulated by the simplified KOMPOT), it was possible to accurately predict temperature profiles using a cascaded dual-NN approach, setting a new standard in atmospheric simulation methodology.

The **main achievement** of this project **is a scalable methodology**, and its practical embodiment in a CLI tool for automated use of current models (including documentation), as well as a universal framework for further retraining and optimization of models for future data sets.

Most notably, this is the first study to the knowledge of the thesis's author, where the

concept of dual cascade ensemble deep learning using NN with attention and CNN in exoplanet science was utilized.

In conclusion, this thesis has significant implications for the field of astroinformatics and exoplanetary science. In a local sense, the neural network for exobasic value predictions and the SHAP explainer can be used to identify patterns and relationships between initial parameters, predictors, and final altitude-temperature values, and to find the thresholds of parameter interactions, beyond which the atmosphere, for example, begins to lose its mass faster. In a global sense, the accelerated code will be able to reconstruct stellar and planetary evolutionary histories more quickly and constrain the possible habitability conditions of the future space mission PLATO exoplanet sample in 2026 (Rauer et al., 2021), and together with the characteristics of habitable zones, will more quickly calculate the optimal distances of exoplanets from their host stars to find stars that are likely to contain planets with liquid surface water and, importantly, sufficiently capable of sustaining their atmospheres, which may help in the selection of targets for space missions such as Ariel in 2029 (Pascale et al., 2018).

However, some **limitations** are worth noting. First — although without code parallelization a notable reduction in simulation time is expected — a similar level of acceleration is achievable if it is possible to run the code on at least 2 threads, which indicates a lack of portability. Second, with the strong predictive power of NN models, they are still slightly overtrained and overestimate values in the mean due to the small amount of data available for such a deep learning task, and will perform worse on planets more different from those used in the current training data set. Although the code still benefits even from using less accurate thermal atmospheric models, better predictions should reveal noticeably more of the code's optimization potential. Also, it is important to note that the predictions are limited to thermal structures, and the code still uses an altitude-homogeneous chemical composition, which computationally still complicates the search for a stable solution.

Given the above limitations, **recommendations for future work** include:

1. Revisit the compiler optimizations and parallelization parts when significant code changes occur.

2. Create a larger and more diverse data set to rebuild and retrain the NN models to create a more robust solution.

3. In addition to a larger data set, it is also recommended to consider including in the initial parameters: planet radius, initial temperature and atmospheric density at the base, additional mixing ratios of species, and more parameters characterizing stellar irradiance, since using a single spectrum parameter (or spectrum-represented parameter such as orbital distance) can significantly overestimate upper atmospheric heating, as it captures mostly near-UV, which mainly affects surface heating. Previous studies by Frey (2023) have shown that even using 10 values can accurately represent broader spectral impacts, representing the full spectrum much better than a single point, even if they tend to overestimate flux in some regions compared to 100 bins.

4. Create a similar CNN and train it on chemical profiles so that the initial estimates will be even closer to the stable KOMPOT solution. This is expected to further reduce the computation time by a factor of 1.2 to 1.8.

5. Using CUDA implementations, nvfortran, and Nsight Systems for profiling (for example, nsys profile -t nvtx,cuda –stats=true) and using -cublas (a CUDA-optimized BLAS library) instead of OpenBLAS to achieve significant performance gains on NVIDIA GPU architectures (this would be particularly useful for using the code on Vienna Scientific Cluster (VSC) computations). It has already been demonstrated by Vorobyov et al. (2023) that it is possible to achieve an acceleration factor of 200 using GPU programming.

6. To create a large data set, it would be interesting to consider the possibilities of quantum computing in the calculation of photochemical reactions by introducing, for example, a quantum-classical Variational Quantum Eigensolver (Armaos et al., 2024) that can calculate molecular ground state energies (properties and stability of atmospheric gases), which can be used to determine potential energy surfaces (how molecules interact and react as they absorb stellar radiation energy) that are related to the rates at which chemical reactions proceed (Kao et al., 2019) by calculating activation energies and Gibbs free energies of activation (the energy barriers that must be overcome for a photochemical reaction to proceed) for the Transition State Theory (TST) equations, using the Amazon Braket system, which may be possible with the approval of the AWS Public Sector Cloud Credit for Research program committee.

The author of this thesis anticipates that future research, by building on the achievements of this work and taking into account the recommendations, will make a significant contribution to our understanding of exoplanets and their atmospheres, and will bring us closer to identifying potentially habitable worlds.

Bibliography

Hervé Abdi, Dominique Valentin, and Betty Edelman. *Neural networks*. Number 124. Sage, 1999.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

Xavier Alameda-Pineda, Andrea Pilzer, Dan Xu, Nicu Sebe, and Elisa Ricci. Viraliency: Pooling local virality. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6080–6088, 2017.

V Armaos, Athanassios A Argiriou, and Ioannis Kioutsioukis. Quantum computing and atmospheric dynamics: Exploring the lorenz system. *arXiv preprint arXiv:2401.03475*, 2024.

Babajide O Ayinde, Tamer Inanc, and Jacek M Zurada. Regularizing deep neural networks by enhancing diversity in feature extraction. *IEEE transactions on neural networks and learning systems*, 30(9):2650–2661, 2019.

Serwan MJ Baban, Patricia Mohammed, Peter Baberstock, Clement Sankat, William Boyd, Bruce Laukner, David Lloyd, and Serwan MJ Baban. *The Journey from Pondering to Publishing*. University of the West Indies Press, 2009.

Mahmuda Afrin Badhan, Eric T Wolf, Ravi Kumar Kopparapu, Giada Arney, Eliza M-R Kempton, Drake Deming, and Shawn D Domagal-Goldman. Stellar activity effects on moist habitable terrestrial atmospheres around m dwarfs. *The Astrophysical Journal*, 887(1):34, 2019.

David H Bailey. Algorithm 719: Multiprecision translation and execution of fortran programs. *ACM Transactions on Mathematical Software (TOMS)*, 19(3):288–319, 1993.

Maxim D Ballmer and Lena Noack. The diversity of exoplanets: from interior dynamics to surface expressions. *Elements: An International Magazine of Mineralogy, Geochemistry, and Petrology*, 17(4):245–250, 2021.

Dalya Baron. Machine learning in astronomy: A practical overview. *arXiv preprint arXiv:1904.07248*, 2019.

Vasileios Belagiannis, Christian Rupprecht, Gustavo Carneiro, and Nassir Navab. Robust optimization for deep regression. In *Proceedings of the IEEE international conference on computer vision*, pages 2830–2838, 2015.

Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning*, volume 1. MIT press Cambridge, MA, USA, 2017.

Lorenzo Biasiotti, Paolo Simonetti, Giovanni Vladilo, Laura Silva, Giuseppe Murante, Stavro Ivanovski, Michele Maris, Sergio Monai, Erica Bisesi, Jost von Hardenberg, et al. Eos-estm: a flexible climate model for habitable exoplanets. *Monthly Notices of the Royal Astronomical Society*, 514(4):5105–5125, 2022.

Christopher M Bishop. Pattern recognition and machine learning. *Springer google schola*, 2:1122–1128, 2006.

Andrea P Buccino, Guillermo A Lemarchand, and Pablo JD Mauas. Ultraviolet radiation constraints around the circumstellar habitable zones. *Icarus*, 183(2):491–503, 2006.

L Buitinck, G Louppe, M BLondel, F Pedregosa, A Mueller, O Grisel, V Niculae, P Prettenhofer, A Gramfort, J Grobler, et al. Robustscaler: Scikit-learn documentation, 2018.

R.L. Burden, J.D. Faires, and A.M. Burden. *Numerical Analysis*. Cengage Learning, 2015. ISBN 9781305465350. URL `https://books.google.dk/books?id=9DV-BAAAQBAJ`.

David C Catling and James F Kasting. *Atmospheric evolution on inhabited and lifeless worlds*. Cambridge University Press, 2017.

Barbara Chapman, Gabriele Jost, and Ruud Van Der Pas. *Using OpenMP: portable shared memory parallel programming*. MIT press, 2007.

Benjamin Charnay and Pierre Drossart. Characterization and modelling of exoplanetary atmospheres. *Comptes Rendus. Physique*, 24(S2):1–11, 2023.

Samprit Chatterjee and Ali S Hadi. *Regression analysis by example*. John Wiley & Sons, 2015.

Guestrin Carlos Chen, Tianqi. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

Laura Chin, Chuanfei Dong, and Manasvi Lingam. Role of planetary radius on atmospheric escape of rocky exoplanets. *The Astrophysical Journal Letters*, 963(1):L20, 2024.

Sae-Gyeol Choi, Jeong-Geun Kim, and Shin-Dug Kim. Adaptive granularity based last-level cache prefetching method with edram prefetch buffer for graph processing applications. *Applied Sciences*, 11(3):991, 2021.

Aditya Chopra, Aaron C Bell, William Fawcett, Rodd Talebi, Daniel Angerhausen, Atılım Güneş Baydin, Anamaria Berea, Nathalie A Cabrol, Christopher Kempes, and Massimo Mascaro. Pyatmos: A scalable grid of hypothetical planetary atmospheres. *arXiv preprint arXiv:2308.10624*, 2023.

Charles S Cockell, Tim Bush, Casey Bryce, Susana Direito, Mark Fox-Powell, Jesse Patrick Harrison, Helmut Lammer, H Landenmark, Javier Martin-Torres, N Nicholson, et al. Habitability: a review. *Astrobiology*, 16(1):89–117, 2016.

Mikhail Danilov and Arkadi Karpov. A classification of meteor radio echoes based on artificial neural network. *Open Astronomy*, 27(1):318–325, 2018.

MIT Critical Data, Matthieu Komorowski, Dominic C Marshall, Justin D Salciccioli, and Yves Crutain. Exploratory data analysis. *Secondary analysis of electronic health records*, pages 185–203, 2016.

Bronis R de Supinski, Thomas RW Scogland, Alejandro Duran, Michael Klemm, Sergi Mateo Bellido, Stephen L Olivier, Christian Terboven, and Timothy G Mattson. The ongoing evolution of openmp. *Proceedings of the IEEE*, 106(11):2004–2019, 2018.

I de Zarzà, J de Curtò, Enrique Hernández-Orallo, and Carlos T Calafate. Cascading and ensemble techniques in deep learning. *Electronics*, 12(15):3354, 2023.

L Drake Deming and Sara Seager. Illusion and reality in the atmospheres of exoplanets. *Journal of Geophysical Research: Planets*, 122(1):53–75, 2017.

Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

Dr M Durairaj and BH Krishna Mohan. A convolutional neural network based approach to financial time series prediction. *Neural Computing and Applications*, 34(16):13319–13337, 2022.

Achrène Dyrek, Michiel Min, Leen Decin, Jeroen Bouwman, Nicolas Crouzet, Paul Mollière, Pierre-Olivier Lagage, Thomas Konings, Pascal Tremblin, Manuel Güdel, et al. So2, silicate clouds, but no ch4 detected in a warm neptune. *Nature*, 625(7993):51–54, 2024.

Victor Eijkhout. *Introduction to High Performance Scientific Computing*. Lulu. com, 2015.

Glauber Robert R Farrar, Donald E. Multicollinearity in regression analysis: the problem revisited. *The Review of Economic and Statistics*, pages 92–107, 1967.

Aritsugi Masayoshi Fatyanosa, Tirana Noor. Effects of the number of hyperparameters on the performance of ga-cnn. In *2020 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)*, pages 144–153. IEEE, 2020.

François Forget, Robin Wordsworth, Ehouarn Millour, J-B Madeleine, Laura Kerber, Jérémy Leconte, Emmanuel Marcq, and Robert M Haberle. 3d modelling of the early martian climate under a denser co2 atmosphere: Temperatures and co2 ice clouds. *Icarus*, 222(1):81–99, 2013.

Charles D Fortenbach and Courtney D Dressing. A framework for optimizing exoplanet target selection for the james webb space telescope. *Publications of the Astronomical Society of the Pacific*, 132(1011):054501, 2020.

Jonathan J Fortney. Modeling exoplanetary atmospheres: An overview. *Astrophysics of Exoplanetary Atmospheres: 2nd Advanced School on Exoplanetary Science*, pages 51–88, 2018.

John Fox. *Regression diagnostics: An introduction*. Sage publications, 2019.

Kevin France, Brian Fleming, Allison Youngblood, James Mason, Jeremy J Drake, Ute V Amerstorfer, Martin Barstow, Vincent Bourrier, Patrick Champey, Luca Fossati, et al. Extreme-ultraviolet stellar characterization for atmospheric physics and evolution mission: motivation and overview. *Journal of Astronomical Telescopes, Instruments, and Systems*, 8(1):014006–014006, 2022.

Thorben Alexander Johannes Frey. Atmospheric evolution of the habitable zone sub-neptune k2-18 b. Masterarbeit, Universität Wien, 2023.

Jerome Friedman. *The elements of statistical learning : data mining,iInference, and prediction*. Springer Series in Statistics. Springer-Verlag, New York, NY, second edition edition, 2017. ISBN 9780387848587.

Siddharth Gandhi and Nikku Madhusudhan. Genesis: new self-consistent models of exoplanetary spectra. *Monthly Notices of the Royal Astronomical Society*, 472(2): 2334–2355, 2017.

Jonathan P Gardner, John C Mather, Randy Abbott, James S Abell, Mark Abernathy, Faith E Abney, John G Abraham, Roberto Abraham, Yasin M Abul-Huda, Scott Acton, et al. The james webb space telescope mission. *Publications of the Astronomical Society of the Pacific*, 135(1048):068001, 2023.

Muriel Gargaud, William M Irvine, Ricardo Amils, Philippe Claeys, Henderson J Cleaves, Maryvonne Guerin, Daniel Rouan, Tilman Spohn, Stéphane Tirard, and Michel Viso. Encyclopedia of astrobiology. 2023.

Timothy D Gebhard, Daniel Angerhausen, Björn S Konrad, Eleonora Alei, Sascha P Quanz, and Bernhard Schölkopf. Parameterizing pressure–temperature profiles of exoplanet atmospheres with neural networks. *Astronomy & Astrophysics*, 681:A3, 2024.

Andrew Gelman. *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press, 2007.

A Gettelman, MJ Mills, DE Kinnison, RR Garcia, AK Smith, DR Marsh, S Tilmes, F Vitt, CG Bardeen, J McInerny, et al. The whole atmosphere community climate model version 6 (waccm6). *Journal of Geophysical Research: Atmospheres*, 124(23): 12380–12403, 2019.

Gabriella Gilli, Sebastien Lebonnois, Pedro Machado, and Ruben Gonçalves. The puzzling transition region of venus atmosphere studied by a ground-to-thermosphere 3d model. In *AAS/Division for Planetary Sciences Meeting Abstracts# 49*, volume 49, pages 502–05, 2017.

Michaël Gillon, Amaury HMJ Triaud, Brice-Olivier Demory, Emmanuël Jehin, Eric Agol, Katherine M Deck, Susan M Lederer, Julien De Wit, Artem Burdanov, James G Ingalls, et al. Seven temperate terrestrial planets around the nearby ultracool dwarf star trappist-1. *Nature*, 542(7642):456–460, 2017.

Flavio Giobergia, Alkis Koudounas, and Elena Baralis. Reconstructing atmospheric parameters of exoplanets using deep learning. In *2023 IEEE 17th International Conference on Application of Information and Communication Technologies (AICT)*, pages 1–6. IEEE, 2023.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

Georgia Gkioxari, Alexander Toshev, and Navdeep Jaitly. Chained predictions using convolutional neural networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 728–743. Springer, 2016.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

Jayesh M Goyal, Nathan Mayne, Benjamin Drummond, David K Sing, Eric Hébrard, Nikole Lewis, Pascal Tremblin, Mark W Phillips, Thomas Mikal-Evans, and Hannah R Wakeford. A library of self-consistent simulated exoplanet atmospheres. *Monthly Notices of the Royal Astronomical Society*, 498(4):4680–4704, 2020.

Susan L Graham, Peter B Kessler, and Marshall K McKusick. Gprof: A call graph execution profiler. *ACM Sigplan Notices*, 39(4):49–57, 2004.

Amelie Gressier. *Observations and modelling of exoplanet atmospheres: from the top of hot-Jupiters to the bottom of temperate terrestrial planets, investigating the transition from Super-Earth to Sub-Neptune with a Hubble survey.* PhD thesis, Université Paris-Saclay, 2022.

Hossein Hassani. Research methods in computer science: The challenges and issues. *arXiv preprint arXiv:1703.04080*, 2017.

Alan E Hedin, HB Niemann, WT Kasprzak, and A Seiff. Global empirical model of the venus thermosphere. *Journal of Geophysical Research: Space Physics*, 88(A1):73–83, 1983.

TC Hinse, R Michelsen, UG Jørgensen, K Goździewski, and S Mikkola. Dynamics and stability of telluric planets within the habitable zone of extrasolar planetary systems-numerical simulations of test particles within the hd 4208 and hd 70642 systems. *Astronomy & Astrophysics*, 488(3):1133–1147, 2008.

Lei Huang, Yi Zhou, Tian Wang, Jie Luo, and Xianglong Liu. Delving into the estimation shift of batch normalization in a network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 763–772, 2022.

Szegedy Christian Ioffe, Sergey. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

Željko Ivezić, Andrew J Connolly, Jacob T VanderPlas, and Alexander Gray. *Statistics, data mining, and machine learning in astronomy: a practical Python guide for the analysis of survey data*, volume 8. Princeton University Press, 2020.

Gareth James. *An introduction to statistical learning : with applications in Python*. Springer texts in statistics. Springer, Cham, Switzerland, 2023. ISBN 9783031387463.

Ben Jiang, Hongwei Gong, Haosen Qin, and Mengjie Zhu. Attention-lstm architecture combined with bayesian hyperparameter optimization for indoor temperature prediction. *Building and Environment*, 224:109536, 2022.

Taeho Jo. Machine learning foundations. *Machine Learning Foundations. Springer Nature Switzerland AG. https://doi. org/10.1007/978-3-030-65900-4*, 2021.

Colin Johnstone. The kompot code: first-principles upper atmosphere modelling and the evolution of planetary atmospheres. In *European Planetary Science Congress*, pages EPSC2018–1054, 2018.

Colin P Johnstone. Hydrodynamic escape of water vapor atmospheres near very active stars. *The Astrophysical Journal*, 890(1):79, 2020.

Colin P Johnstone. Stellar winds and planetary atmospheres. *arXiv preprint arXiv:2105.11243*, 2021.

Colin P Johnstone, Manuel Güdel, Helmut Lammer, and Kristina G Kislyakova. Upper atmospheres of terrestrial planets: Carbon dioxide cooling and the earth's thermospheric evolution. *Astronomy & Astrophysics*, 617:A107, 2018.

Colin P Johnstone, Helmut Lammer, Kristina G Kislyakova, Manuel Scherf, and Manuel Güdel. The young sun's xuv-activity as a constraint for lower co2-limits in the earth's archean atmosphere. *Earth and Planetary Science Letters*, 576:117197, 2021.

David Julian. *Deep learning with pytorch quick start guide: learn to train and deploy neural network models in Python*. Packt Publishing Ltd, 2018.

E. Kalinicheva, Valery Shematovich, and Yaroslav Pavlyuchenkov. О тепловом убегании атмосферы π men c (On the thermal atmospheric escape of π men c). *Астрономия и исследование космического пространства.—Екатеринбург, 2021*, pages 117–120, 01 2021. doi: 10.15826/B978-5-7996-3229-8.28.

Der-you Kao, Marko Gacesa, Renata M Wentzcovitch, Shawn Domagal-Goldman, Ravi K Kopparapu, Stephen J Klippenstein, Steven B Charnley, Wade G Henning, Joe Renaud, Paul Romani, et al. Impacts of quantum chemistry calculations on exoplanetary science, planetary astronomy, and astrophysics. *arXiv preprint arXiv:1904.07161*, 2019.

Shachar Kaufman, Saharon Rosset, Claudia Perlich, and Ori Stitelman. Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(4):1–21, 2012.

Kim Wooseong Kilichev, Dusmurod. Hyperparameter optimization for 1d-cnn-based network intrusion detection using ga and pso. *Mathematics*, 11(17):3724, 2023.

KG Kislyakova, CP Johnstone, Manuel Scherf, M Holmström, II Alexeev, H Lammer, ML Khodachenko, and M Güdel. Evolution of the earth's polar outflow from mid-archean to present. *Journal of Geophysical Research: Space Physics*, 125(8):e2020JA027837, 2020.

Takanori Kodama, Hidenori Genda, Ryouta O'ishi, Ayako Abe-Ouchi, and Yutaka Abe. Inner edge of habitable zones for earth-sized planets with various surface water distributions. *Journal of Geophysical Research: Planets*, 124(8):2306–2324, 2019.

Thaddeus D Komacek, Wanying Kang, Jacob Lustig-Yaeger, and Stephanie L Olson. Leveraging models to constrain the climates of rocky exoplanets. *arXiv preprint arXiv:2108.08386*, 2021.

TT Koskinen, RV Yelle, DS Snowden, P Lavvas, BR Sandel, FJ Capalbo, Y Benilan, and RA West. The mesosphere and lower thermosphere of titan revealed by cassini/uvis stellar occultations. *Icarus*, 216(2):507–534, 2011.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Daria Kubyshkina, Aline A Vidotto, Luca Fossati, and Eoin Farrell. Coupling thermal evolution of planets and hydrodynamic atmospheric escape in mesa. *Monthly Notices of the Royal Astronomical Society*, 499(1):77–88, 2020.

Helmut Lammer, James F Kasting, Eric Chassefière, Robert E Johnson, Yuri N Kulikov, and Feng Tian. Atmospheric escape and evolution of terrestrial planets and satellites. *Space Science Reviews*, 139:399–436, 2008.

Jonathan Langton and Gregory Laughlin. Hydrodynamic simulations of unevenly irradiated jovian planets. *The Astrophysical Journal*, 674(2):1106, 2008.

Stéphane Lathuilière, Pablo Mesejo, Xavier Alameda-Pineda, and Radu Horaud. A comprehensive analysis of deep regression. *IEEE transactions on pattern analysis and machine intelligence*, 42(9):2065–2081, 2019.

Huijun Le, Libo Liu, Han He, and Weixing Wan. Statistical analysis of solar euv and x-ray flux enhancements induced by solar flares and its implication to upper atmosphere. *Journal of Geophysical Research: Space Physics*, 116(A11), 2011.

Jérémy Leconte, Rémi Soummer, Sasha Hinkley, Ben R Oppenheimer, Anand Sivaramakrishnan, Douglas Brenner, Jeffrey Kuhn, James P Lloyd, Marshall D Perrin, Russell Makidon, et al. The lyot project direct imaging survey of substellar companions: Statistical analysis and information from nondetections. *The Astrophysical Journal*, 716(2):1551, 2010.

Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer, 2002.

Neil Lindquist, Mark Gates, Piotr Luszczek, and Jack Dongarra. Threshold pivoting for dense lu factorization. In *2022 IEEE/ACM Workshop on Latest Advances in Scalable Algorithms for Large-Scale Heterogeneous Systems (ScalAH)*, pages 34–42. IEEE, 2022.

McCullough Matthew Loeliger, Jon. *Version Control with Git: Powerful tools and techniques for collaborative software development.* " O'Reilly Media, Inc.", 2012.

Nikku Madhusudhan. Atmospheric retrieval of exoplanets. *arXiv preprint arXiv:1808.04824*, 2018.

Nikku Madhusudhan. Exoplanetary atmospheres: Key insights, challenges, and prospects. *Annual Review of Astronomy and Astrophysics*, 57:617–663, 2019.

Nikku Madhusudhan, Marcelino Agúndez, Julianne I Moses, and Yongyun Hu. Exoplanetary atmospheres—chemistry, formation conditions, and habitability. *Space science reviews*, 205:285–348, 2016.

David Malmgren-Hansen, Valero Laparra, Allan Aasbjerg Nielsen, and Gustau Camps-Valls. Statistical retrieval of atmospheric profiles with deep convolutional neural networks. *ISPRS journal of photogrammetry and remote sensing*, 158:231–240, 2019.

SM Matthews, CA Watson, EJW de Mooij, TR Marsh, M Brogi, SR Merritt, KW Smith, and D Steeghs. Doppler tomography as a tool for characterising exoplanet atmospheres ii: an analysis of hd 179949 b. *Monthly Notices of the Royal Astronomical Society*, page stae906, 2024.

Saroj K Meher and Ganapati Panda. Deep learning in astronomy: a tutorial perspective. *The European Physical Journal Special Topics*, 230(10):2285–2317, 2021.

João M Mendonça, Simon L Grimm, Luc Grosheintz, and Kevin Heng. Thor: A new and flexible global circulation model to explore planetary atmospheres. *The Astrophysical Journal*, 829(2):115, 2016.

Jeremy Miles. Tolerance and variance inflation factor. *Encyclopedia of statistics in behavioral science*, 2005.

Michiel Min, Chris W Ormel, Katy Chubb, Christiane Helling, and Yui Kawashima. The arcis framework for exoplanet atmospheres-modelling philosophy and retrieval. *Astronomy & Astrophysics*, 642:A28, 2020.

Jian-heng Guo Mo, Yang. Effect of orbital distance on the atmospheric escape of exoplanets. *Chinese Astronomy and Astrophysics*, 42(1):81–100, 2018.

Shankaran-Akash Moraila, Gina, Zuoming Shi, and Alex M Warren. Measuring reproducibility in computer systems research. *PLoS Comput Biol*, 9:37, 2014.

Giuseppe Morello, Achrène Dyrek, and Quentin Changeat. Is binning always sinning? the impact of time-averaging for exoplanet phase curves. *Monthly Notices of the Royal Astronomical Society*, 517(2):2151–2164, 2022.

Sagnick Mukherjee, Natasha E Batalha, Jonathan J Fortney, and Mark S Marley. Picaso 3.0: a one-dimensional climate model for giant planets and brown dwarfs. *The Astrophysical Journal*, 942(2):71, 2023.

Camryn Mullin, Ruobing Dong, Jarron Leisenring, Gabriele Cugno, Thomas Greene, Doug Johnstone, Michael R Meyer, Kevin R Wagner, Schuyler G Wolff, Martha Boyer, et al. Jwst/nircam imaging of young stellar objects. iii. detailed imaging of the nebular environment around the hl tau disk. *The Astronomical Journal*, 167(4):183, 2024.

F Murgas, G Chen, E Pallé, L Nortmann, and G Nowak. The gtc exoplanet transit spectroscopy survey-x. stellar spots versus rayleigh scattering: the case of hat-p-11b. *Astronomy & Astrophysics*, 622:A172, 2019.

NOAA & Mysid. Layers of the atmosphere. Wikimedia Commons, August 2 2014. URL `https://commons.wikimedia.org/wiki/File:Atmosphere_layers-en.svg`. [SVG].

James E Owen. Atmospheric escape and the evolution of close-in exoplanets. *Annual Review of Earth and Planetary Sciences*, 47:67–90, 2019.

James E Owen and Yanqin Wu. Atmospheres of low-mass planets: the "boil-off". *The Astrophysical Journal*, 817(2):107, 2016.

Rautaray-S.S. Pandey, M. *Machine Learning: Theoretical Foundations and Practical Applications*. Studies in Big Data. Springer Nature Singapore, 2022. ISBN 9789813365209. URL `https://books.google.at/books?id=IL8IzwEACAAJ`.

David Lorge Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053–1058, 1972.

Enzo Pascale, Paul Eccleston, and Giovanna Tinetti. The ariel space mission. In *2018 5th IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, pages 31–34. IEEE, 2018.

Yoder-J Pauls, Alexander. Determining optimum drop-out rate for neural networks. In *Midwest Instructional Computing Symposium (MICS)*, 2018.

Mark Peplow. Light from alien planets confirmed. *Nature*, 2005. `https://doi.org/10.1038/news050321-9`.

Paweł Pławiak, Moloud Abdar, and U Rajendra Acharya. Application of new deep genetic cascade ensemble of svm classifiers to predict the australian credit scoring. *Applied Soft Computing*, 84:105740, 2019.

K Rajendran, SR Lewis, and MR Patel. Tracking trace gas plumes on mars with a lagrangian particle dispersion model. In *Ninth International Conference on Mars*, volume 2089, page 6449, 2019.

Heike Rauer, Isabella Pagano, Miguel Mas-Hesse, Conny Aerts, Magali Deleuil, Laurent Gizon, Marie-Jo Goupil, Ana María Heras, Giampaolo Piotto, Don Pollacco, et al. The plato mission: Overview and status. In *European Planetary Science Congress*, pages EPSC2021–90, 2021.

Miriam Rengel and Jakob Adamczewski. Radiative transfer and inversion codes for characterizing planetary atmospheres: an overview. *Frontiers in Astronomy and Space Sciences*, 10:1176740, 2023.

Markus Rosenberger. Parameterizing atmospheric temperature profiles of terrestrial planets using modern machine learning regression methods. Master's thesis, Universität Wien, 2021.

MS Rumenskikh, IF Shaikhislamov, ML Khodachenko, H Lammer, IB Miroshnichenko, AG Berezutsky, and L Fossati. Global 3d simulation of the upper atmosphere of hd189733b and absorption in metastable he i and lyα lines. *The Astrophysical Journal*, 927(2):238, 2022.

Sara Seager and Drake Deming. Exoplanet atmospheres. *Annual Review of Astronomy and Astrophysics*, 48:631–672, 2010.

Lei Shi. Retrieval of atmospheric temperature profiles from amsu-a measurement using a neural network approach. *Journal of Atmospheric and Oceanic Technology*, 18(3): 340–347, 2001.

Aneta Siemiginowska, Gwendolyn Eadie, Ian Czekala, Eric Feigelson, Eric B Ford, Vinay Kashyap, Michael Kuhn, Tom Loredo, Michelle Ntampaka, Abbie Stevens, et al. The next decade of astroinformatics and astrostatistics. *Bulletin of the American Astronomical Society*, 51(3):355, 2019.

Justin Solomon. *Numerical algorithms: methods for computer vision, machine learning, and graphics*. CRC press, 2015.

Young-Kyoon Suh, Richard T Snodgrass, John D Kececioglu, Peter J Downey, Robert S Maier, and Cheng Yi. Emp: execution time measurement protocol for compute-bound programs. *Software: Practice and Experience*, 47(4):559–597, 2017.

Gabrielle Suissa, Avi M Mandell, Eric T Wolf, Geronimo L Villanueva, Thomas Fauchez, and Ravi kumar Kopparapu. Dim prospects for transmission spectra of ocean earths around m stars. *The Astrophysical Journal*, 891(1):58, 2020.

Xiangyang Tan, Kaixue Ma, and Fangli Dou. A convolutional neural network and attention-based retrieval of temperature profile for a satellite hyperspectral microwave sensor. *Atmosphere*, 15(2):235, 2024.

Axinya Tokareva. Training deep convolutional networks for protein similarities on a pfam data set. Bachelor's thesis, Universität Wien, 2019.

DeepL Translate, DeepL Write (AI), and Grammarly. These tools were used during the writing of this thesis for grammar correction and rephrasing certain sentences. Provided by the University of Vienna and external sources, 2024.

Shang-Min Tsai, Elspeth KH Lee, Diana Powell, Peter Gao, Xi Zhang, Julianne Moses, Eric Hébrard, Olivia Venot, Vivien Parmentier, Sean Jordan, et al. Photochemically produced so2 in the atmosphere of wasp-39b. *Nature*, 617(7961):483–487, 2023.

Shang-Min Tsai, Vivien Parmentier, João M Mendonça, Xianyu Tan, Russell Deitrick, Mark Hammond, Arjun B Savel, Xi Zhang, Raymond T Pierrehumbert, and Edward W Schwieterman. Global chemical transport on hot jupiters: Insights from the 2d vulcan photochemical model. *The Astrophysical Journal*, 963(1):41, 2024.

Gwenaël Van Looveren, Manuel Güdel, Sudeshna Boro Saikia, and Kristina Kislyakova. Airy worlds or barren rocks? on the survivability of secondary atmospheres around the trappist-1 planets. *arXiv preprint arXiv:2401.16490*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Eduard I Vorobyov, James McKevitt, Igor Kulikov, and Vardan Elbakyan. Computing the gravitational potential on nested meshes using the convolution method. *Astronomy & Astrophysics*, 671:A81, 2023.

Danlei Wang, Ling Tong, Xun Gong, Xin Guan, Peicheng Wang, and Bo Gao. Retrieval of atmospheric temperature profiles from hyperspectral microwave radiative data based on the neural network. In *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, pages 7095–7098. IEEE, 2021. ISBN 9781665403696.

Hongkun Wang, Dong Liu, Yingwei Xia, Wanyi Xie, and Yiren Wang. Retrieval of atmospheric temperature profile from historical data and ground-based observations by using a machine learning algorithm. *Remote Sensing*, 15(11):2717, 2023.

Roel Wieringa. Design science as nested problem solving. In *Proceedings of the 4th international conference on design science research in information systems and technology*, pages 1–12, 2009.

Roel Wieringa, Neil Maiden, Nancy Mead, and Colette Rolland. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requirements engineering*, 11:102–107, 2006.

Mikolaj Wojciuk, Zaneta Swiderska-Chadaj, Krzysztof Siwek, and Arkadiusz Gertych. Improving classification accuracy of fine-tuned cnn models: Impact of hyperparameter optimization. *Heliyon*, 2024.

Eric T Wolf, Ravi Kopparapu, Vladimir Airapetian, Thomas Fauchez, Scott D Guzewich, Stephen R Kane, Daria Pidhorodetska, Michael J Way, Dorian S Abbot, Jade H Checlair, et al. The importance of 3d general circulation models for characterizing the climate and habitability of terrestrial extrasolar planets. *arXiv preprint arXiv:1903.05012*, 2019.

Yuxin Wu and Justin Johnson. Rethinking" batch" in batchnorm. *arXiv preprint arXiv:2105.07576*, 2021.

Q. Wang X. Zhang. Openblas: a high performance blas library on loongson 3a cpu. *IEEE 18th International Conference on Parallel and Distributed Systems (ICPADS)*, 22(zk2): 208–216, 2012.

Dong Yu, Li Deng, Dong Yu, and Li Deng. Feature representation learning in deep neural networks. *Automatic speech recognition: A deep learning approach*, pages 157–175, 2015.

Haoyang Zeng, Matthew D Edwards, Ge Liu, and David K Gifford. Convolutional neural network architectures for predicting dna–protein binding. *Bioinformatics*, 32 (12):i121–i127, 2016.

Xi Zhang. Atmospheric regimes and trends on exoplanets and brown dwarfs. *Research in Astronomy and Astrophysics*, 20(7):099, 2020.

Wenbo Zhu, Weichang Yeh, Jianwen Chen, Dafeng Chen, Aiyuan Li, and Yangyang Lin. Evolutionary convolutional neural networks using abc. In *Proceedings of the 2019 11th International Conference on Machine Learning and Computing*, pages 156–162, 2019.

A. Appendix

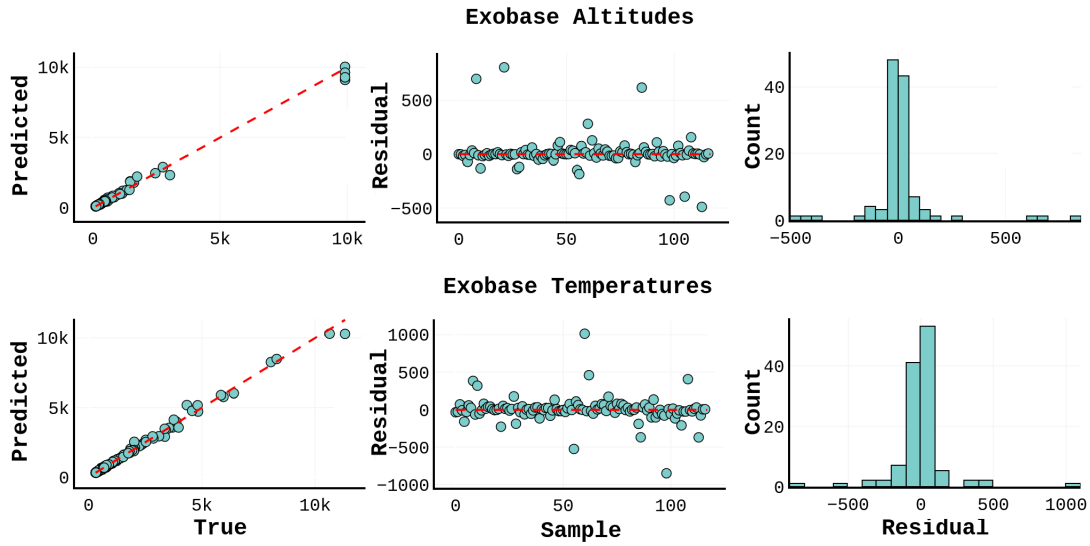

Figure A.1.: **Residual plots for evaluating the attention-based NN model's performance (Train data set)**: the first row evaluates the predictions of *Exobase Altitudes* (km), the second row — *Temperature* predictions (K). The scatter plots on the left compare predicted and actual values, while the middle plots show the prediction errors, which are detailed in the histogram on the right. The middle «Sample» axis represents one model / planet. Red dashed lines serve as a reference point for perfect predictions.

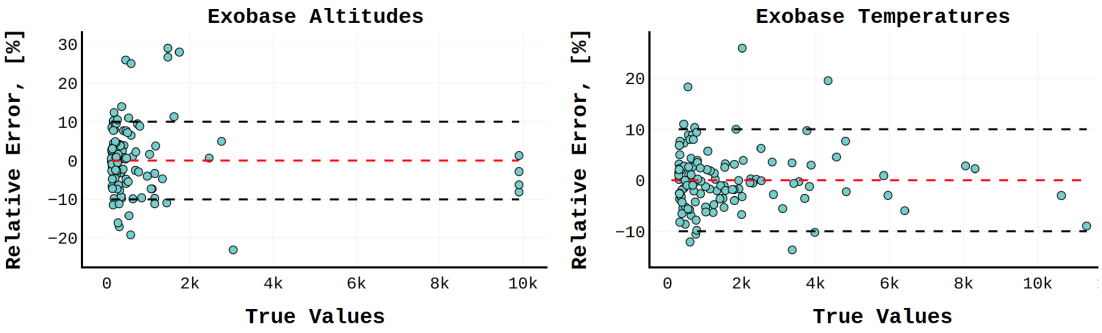

Figure A.2.: **Relative errors distribution across true values for evaluating the attention-based NN model's performance (Train data set)**: The first plot shows prediction errors for *Exobase Altitudes* (km), and the second plot — for *Exobase Temperatures* (K). The red dashed line indicates perfect predictions, and the black dashed lines — error boundaries from -10% to +10%.

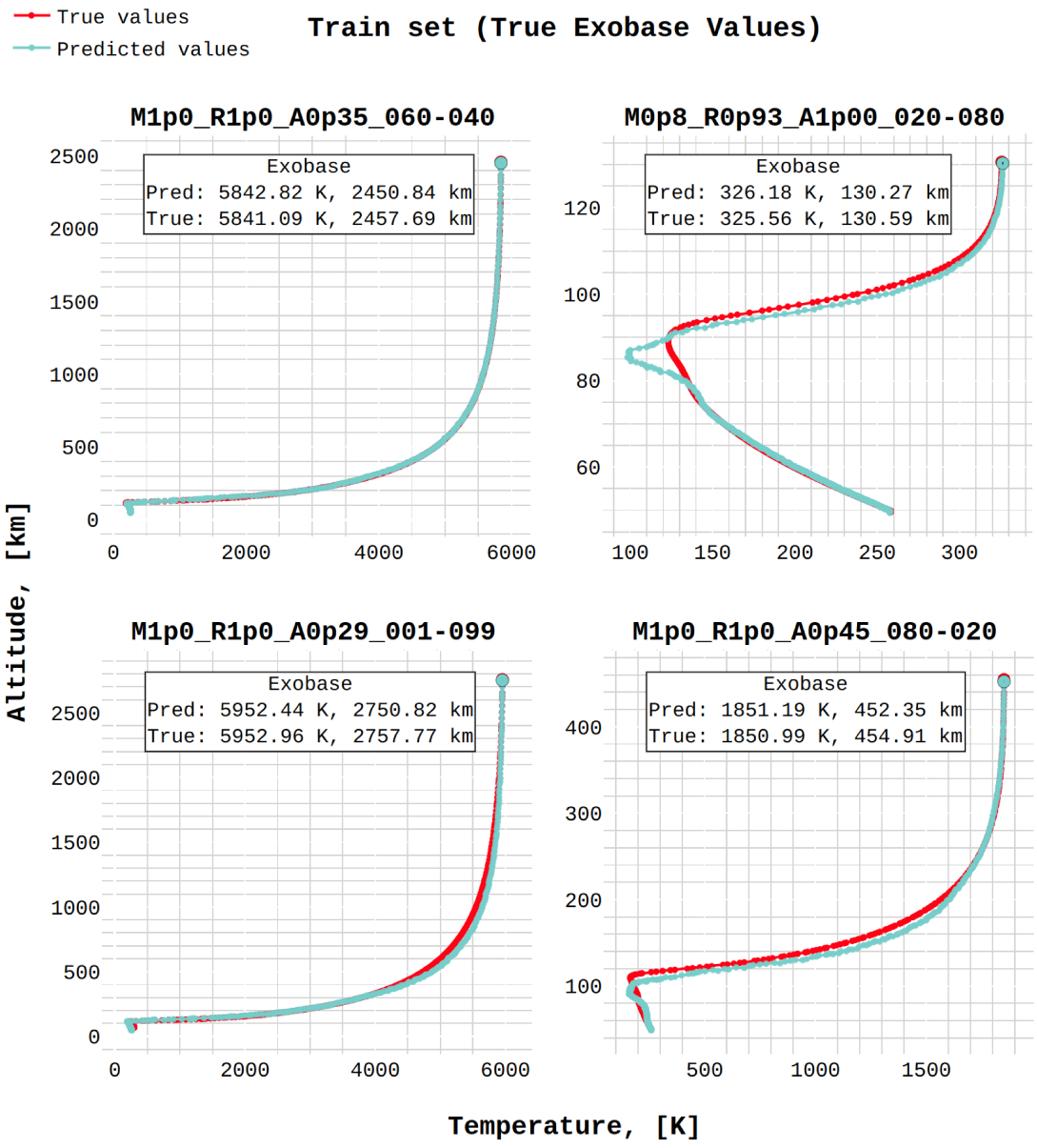

Figure A.3.: **Altitude-temperature profiles' CNN predictions of 4 atmospheric models from Train data set with differing irradiation levels**: the plots show the predictions made using *true Exobase values* as input features for CNN. Red dots are true values, turquoise — predicted values. In the *Exobase values* box, «True» means *actual Exobase values*, «Pred» — final predicted last values as *Exobase values* by CNN. The atmospheric model name is formatted as follows: «p» — point, «M» — mass, «R» — radius, «A» — orbital distance, followed by the initial mixing ratio of N_2, and concluding with $X(CO_2)$.