Radiative Temperature and Dilution Factor Emulator for TARDIS

Alexander Grunewald

March 2023

Abstract

Supernovae embody some of the most intense and powerful events in the cosmos, and radiative-transfer spectral synthesis codes are indispensable for astronomers examining these cataclysmic occurrences. TARDIS Kerzendorf et al. [2023], a specialized open-source code tailored for simulating one-dimensional supernova ejecta, is particularly well-suited for modeling Type Ia supernovae in their early phases. This specific focus allows for the approximations of spherical symmetry, the w7 density profile, uniform abundances, and other simplifications. Nevertheless, its computational demands and time-intensive iterative process pose challenges. An emulator could considerably bolster the efficiency of the process by facilitating swift and precise predictions of TARDIS model output spectra, thereby addressing these constraints.This paper explores the background and importance of TARDIS, its scope and limitations in the context of Type Ia supernovae, and advocates for the development of an emulator as a more effective computational approach to modeling supernovae.

1 Introduction

As stars reach the end of their life, they may experience a catastrophic explosion known as a supernova. During this event, the star releases a vast amount of energy and material into space, forming a rapidly expanding cloud of gas and dust called the ejecta. This ejecta is composed of various elements present in the star's core, including hydrogen, helium, carbon, oxygen, and more. Light interacts with these elements as it passes through the ejecta, resulting in the absorption or emission of specific wavelengths of light, which can be analyzed through spectra.

For example, when light passes through supernova ejecta containing hydrogen, the hydrogen atoms in the cloud absorb specific ultraviolet wavelengths, corresponding to transitions between different energy levels in the hydrogen atom. Similarly, elements like oxygen and iron absorb or emit light at multiple wavelengths, producing spectral lines that help identify their presence in the ejecta. By analyzing these spectra, astronomers can glean valuable insights into the supernova's composition and properties, as well as the physical processes occurring during supernova explosions.

Radiative-transfer spectral synthesis codes serve as crucial tools for astronomers studying these explosive occurrences, as they enable the generation of theoretical spectra that can be compared with observed spectra to gain a deeper understanding of supernovae properties. The demand for having readily available spectral data has resulted in the development of other radiative-transfer spectral synthesis codes for modeling supernova ejecta, each having their distinct purposes Stehle et al. [2005].

TARDIS, a Monte Carlo radiative-transfer spectral synthesis code, is designed for modeling one-dimensional supernova ejecta. In this study, we will focus on Type Ia supernovae in their early phases. This focus allows for the approximations of spherical symmetry, the w7 density profile, uniform abundances, and other simplifications. The code primarily functions to facilitate rapid spectral modeling of supernovae. However, each simulation can take up to 20 minutes to produce a spectrum, as multiple iterations are required until convergence to a final solution. The primary iterative process involves determining the ionization and excitation state of the plasma at each step to generate the final output spectrum, making TARDIS both computationally expensive and time-consuming.

Driven by the need for more efficient computational methods, an emulator could significantly enhance the process's efficiency by enabling quick and accurate predictions of the TARDIS model's output spectra. Emulators are statistical models that can be trained on a subset of the full simulation output to predict the output of the simulation for any input parameters, thereby substantially reducing the computational cost of exploring parameter space. Developing an emulator for TARDIS would enable more efficient modeling of supernovae and allow astronomers to investigate a wider range of models and input parameters.

We divide this paper into 4 sections. In the methodology, we discuss the Radiative Transfer model, our process of data generation for abundances and plasma states, the processing of the training set, the Neural Network (NN) model, and the validation of our model. In the results section, we present our findings and discuss the performance of our NN model in predicting supernovae ejecta properties. In the discussion section, we delve deeper into the implications of our results, the limitations of our approach, and potential improvements. Finally, we provide a conclusion that summarizes the main findings and delve deeper into the future work that must be addressed.

2 Methods

In this section, we outline the methods employed in our study to develop an efficient and accurate emulator for the TARDIS code. We begin by describing the Dirichlet distribution-based sampling process utilized to generate approximate realistic elemental abundance compositions, followed by a discussion of the radiative transfer model and its default parameters. We then detail the creation of the training and testing datasets and the data processing steps involved in preparing the data for the emulator. Finally, we present the neural network model and its architecture, highlighting our approach to designing and validating the model to ensure optimal performance in predicting the Radiative Temperature and Dilution Factor.

2.1 Dirichlet Sampling

To generate the supernova spectra used by TARDIS for producing spectral images, it is essential to employ realistic element abundance compositions. To achieve a realistic spectra we will utilize a Dirichlet distribution-based sampling process to generate abundance compositions, ensuring that the sum of the fractional elemental composition amounts to 1.

The Dirichlet distribution is chosen to generate the elemental abundance data for this project for several reasons: (1) The Dirichlet distribution is a multivariate distribution, which allows for the simultaneous generation of multiple variables—in this case, the fractional abundances of various elements. Moreover, it enforces the constraint that the sum of the generated fractions equals 1, ensuring that the fractional compositions are realistic and adhere to the mass conservation principle. (2) The Dirichlet distribution offers flexibility in modeling the desired distributions by adjusting its parameters (alphas). By fine-tuning the alphas, researchers can generate elemental abundance distributions that closely resemble the observed or expected distributions in supernova ejecta. (3) The Dirichlet distribution is robust and can generate a wide range of elemental abundance compositions, ensuring a comprehensive exploration of the parameter space. This characteristic is particularly useful for studying supernova ejecta, as it allows researchers to investigate various scenarios and determine how the ejecta's properties and the underlying physical processes may vary depending on the elemental composition.

The elements O, Si, S, Mg, Ca, Ti+Cr, Fe, and ^{56}Ni, with respective alphas of 0.43, 0.12, 0.02, 0.016, 0.0006, 0.001, 0.007, and 0.007, were chosen to resemble the ejceta of a type Ia supernova as detailed in Stehle et al. [2005]. This methodology ensures the generation of unique and realistic abundance distributions in the type Ia supernova spectra.

2.2 Radiative Transfer Model

The TARDIS simulation is initialized using a YAML configuration file, which specifies various parameters required for the simulation. In the provided example, the supernova's requested luminosity is set to 9.44 log-lsun, and the time of explosion is 13 days. The atomic data file used is "kurucz-cd23-chianti-H-He.h5".

The model's structure is defined by specifying the velocity range from 1.1e4 km/s to 20000 km/s with 20 shells, and the density profile follows the branch85 w7 model. Elemental abundances are set to uniform values, with specific fractions for O, Si, S, Mg, Ca, Ti+Cr, Fe, and ^{56}Ni. However, in our study, the Dirichlet process is employed to generate our elemental abundances as highlighted in the previous subsection.

The plasma configuration includes electron scattering, nebular ionization, dilute-lte excitation, dilute-blackbody radiative rates, and macroatom line interaction. The Monte Carlo parameters include a seed value of 23111963, 4.0e+4 packets, 20 iterations, and a single thread for execution. The last number of packets is set to 1.e+5, and no virtual packets are used.

A damped convergence strategy is implemented, with a damping constant of 1.0, a threshold of 0.05, a fraction of 0.8, and holding iterations for 3 steps. The t_{inner} damping constant is set to 0.5. Finally, the spectrum range is specified to be between 500 angstroms and 20000 angstroms, with a total of 10000 data points.

2.3 Training Set

To validate the abundance distributions employed for generating TARDIS simulations, a subset of randomly selected elemental compositions was input into the TARDIS model. The resulting spectra were compared to a TARDIS spectral image based on naturally observed data to confirm that the abundance distributions produced realistic output.

Once the validity of the sampled distributions was established, a dataset of 10,000 TARDIS simulations was created using the same Dirichlet sampling method for abundance compositions. These simulations were run on the High-Performance Computer Cluster (HPCC) at Michigan State University to generate Radiative Temperature and Dilution Factor vectors, that are used to calculate the plasma state of TARDIS, which were then saved as a dataset. This dataset was then plotted to visualize the resulting distributions of the elemental compositions, ensuring that no outliers were generated by the sampling method. The final dataset consists of 48 columns, 8 for the elemental abundances, and 40 for the Radiative Temperature and Dilution Factor, each containing 20 columns based on the inner velocity of each shell, and 10,000 rows.

2.4 Data Processing

The data was then divided into a training and testing dataset with an 80-20 split. The abundance profiles were scaled by log10, and both the abundances and Radiative Temperature and Dilution Factor were normalized to standard normal by subtracting the mean and dividing by the standard deviation of the training set column wise. This normalization provided the emulator with a good initial guess for the weights. The data was then unscaled, and the Residual Sum of Squares was calculated between the original unscaled data and the inverse transformation of the scaled data to ensure the successful data transformation.

2.5 Nerual Network Model

A neural network was created using TensorFlow Abadi et al. [2015]. The general architecture for this model was chosen to be a sequential neural network with multiple linear layers. We opted to use the Mean Squared Error (MSE) loss function, as it is well-suited for regression problems like ours, where the goal is to predict continuous values. The MSE loss measures the average squared difference between the predicted and true values, effectively penalizing larger errors more than smaller ones.

Initially, a small model with two layers, each consisting of 40 neurons, was developed. The model was then intentionally overfitted to the data to ensure its capability to predict the targets. Overfitting is a crucial step to verify that the model has enough capacity to learn from the training data, even if it might not generalize well to new, unseen data. However, if the model cannot overfit, it indicates that the model's complexity is insufficient to capture the underlying patterns in the data.

Furthermore, we integrated batch normalization after each layer to ensure a stable distribution of the activations. This technique not only aids in accelerating the training process by allowing the use of higher learning rates but also mitigates the risks associated with vanishing or exploding gradients Ioffe and Szegedy [2015]. Batch normalization achieves this by normalizing the input of each layer to have a mean of zero and a standard deviation of one, which helps maintain stable gradients throughout the network.

2.6 Training and Validation

Table 1: Parameter Search

Parameters	**Values**
Hidden Layers	1-2
Neurone per Layer	40
Optimiser	adam, Nadam
Activation Function	relu, elu, softplus

Various optimizers and activation functions were evaluated to determine the optimal configuration based on the MSE loss. The optimizers assessed in this study included Adam and Nadam, while the activation functions examined were relu, elu, and softplus, see Table 1.

Nadam and Adam are optimization algorithms used in training neural networks. Adam, which stands for Adaptive Moment Estimation, is an extension of the stochastic gradient descent optimization algorithm. It combines the benefits of both the AdaGrad and RMSprop optimization algorithms by adaptively adjusting the learning rate for each parameter and employing momentum. This allows Adam to navigate complex loss landscapes more efficiently Kingma and Ba [2017].

Nadam, or Nesterov-accelerated Adaptive Moment Estimation, is an extension of the Adam optimizer. It incorporates Nesterov momentum, which is an improvement to the standard momentum technique. Nesterov momentum helps the optimizer to make more informed updates to the weights by considering not only the current gradient but also the future gradient. This feature enables Nadam to converge faster and achieve better performance in many cases Dozat [2016].

Once the model exhibited its ability to overfit, its complexity was augmented by incorporating additional layers and neurons. During this research, we trained the model over 100 epochs while performing parameter tuning. After each training iteration, we plotted the training and validation loss across epochs to verify the model's capacity to overfit the data. Subsequently, the overall performance was assessed using the testing dataset, employing Mean and Max Fractional Error (FE) as metrics. The MeanFE measures the overall accuracy of the predictions, while the Max FE signifies the largest discrepancy between the true values and the model's predictions. A low MeanFE, coupled with a relatively low Max FE, demonstrates that the model has effectively captured the underlying patterns in the data, generating accurate predictions for both Dilution Factor and Radiative Temperature. See Vogl et al. [2020] for more details.

$$MeanFE = \frac{1}{N}\sum_{i=0}^{N}\frac{\left|f_{\lambda,i}^{emu} - f_{\lambda,i}^{test}\right|}{f_{\lambda,i}^{test}}$$

$$MaxFE = \max_{i=0}^{N}\frac{\left|f_{\lambda,i}^{emu} - f_{\lambda,i}^{test}\right|}{f_{\lambda,i}^{test}}$$

Finally, the predictions were visualized by taking the best and worst FEs and plottd them comparing them to the testing plot.

3 Results

The primary goals of this paper are to develop an efficient emulator for the TARDIS code, which is specialized in simulating one-dimensional supernova ejecta,

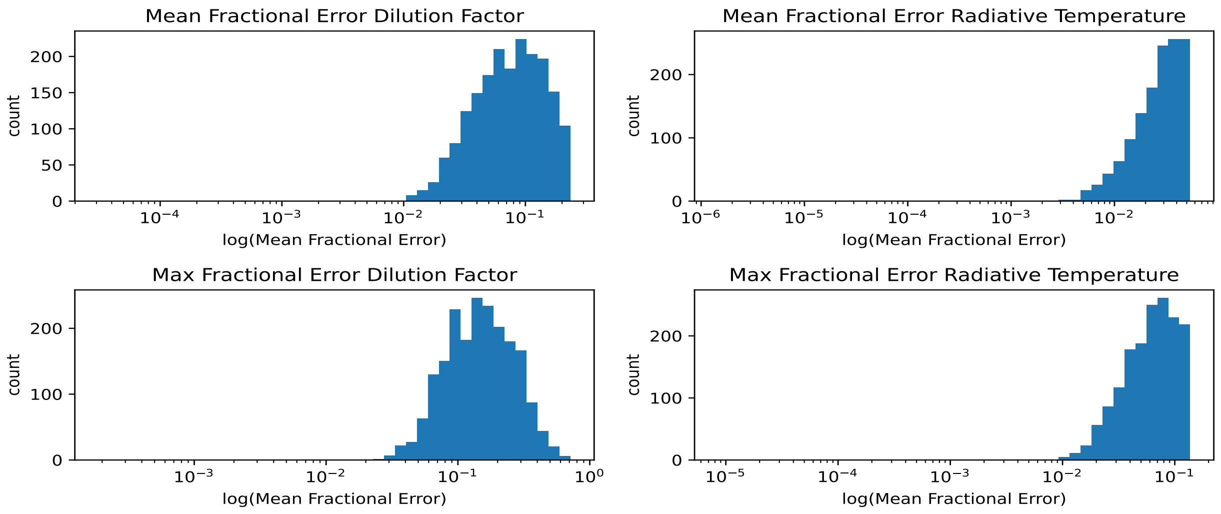

Figure 1: The plot presents the distribution of MeanFE and MaxFE for the testing dataset. The MeanFE measures the overall accuracy of the predictions, reflecting the average deviation of the model's predictions from the true values. A low MeanFE indicates that the model has effectively captured the underlying patterns in the data and generates accurate predictions for both Dilution Factor and Radiative Temperature. On the other hand, the MaxFE signifies the largest discrepancy between the true values and the model's predictions, providing insights into the model's performance in predicting extreme cases. A relatively low MaxFE, when coupled with a low MeanFE, demonstrates that the model has a consistent performance across the entire dataset, including the cases with the largest deviations. In this plot, the results are log-scaled to offer a better representation of the distribution, highlighting the model's capacity to generate accurate predictions with relatively low error for the majority of data points.

and to address the computational challenges associated with this process. In this section, we will delve into the results obtained from our study, highlighting the chosen emulator architecture's effectiveness and the impact of various design decisions on its ability to predict the Radiative Temperature and Dilution Factor with high accuracy and precision.

We will examine the optimal model configuration determined through experimentation, assess the model's performance on both training and testing datasets, and identify areas of improvement to ensure generalization to unseen data without overfitting or underfitting issues. Additionally, we will explore strategies to mitigate any overfitting observed in our model, enhancing its robustness and reliability in predicting supernovae ejecta properties. By presenting a thorough analysis of our results, we aim to demonstrate the potential of our emulator in facilitating more efficient modeling of supernovae ejecta using the TARDIS code.

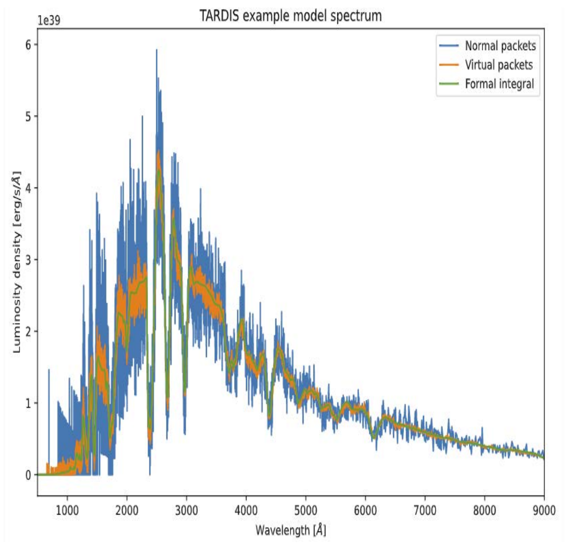

Figure 2: Realistic Spectrum taken from the TARDIS quick start tutorial, showcasing an example of a naturally observed spectrum.

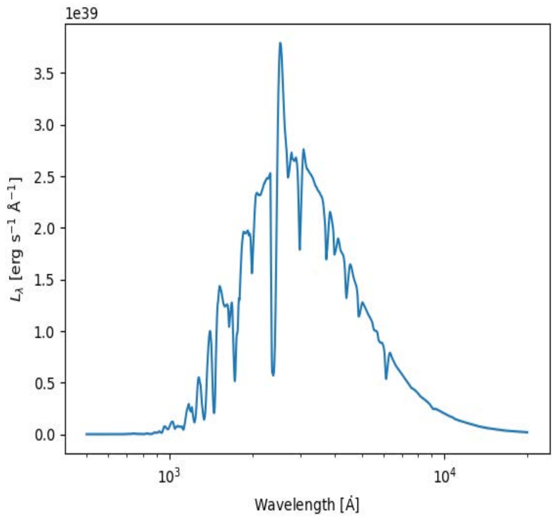

Figure 3: Synthetic spectra generated using the Dirichlet process, illustrating the model's ability to produce realistic spectral images.

The abundances generated through the Dirichlet process, as discussed in the previous section, effectively produced realistic spectral images using the TARDIS simulation. As illustrated in 2 and 3, the output spectra created with the generated abundances closely resembled the expected spectra when utilizing naturally observed data and running it through TARDIS. By providing a good initial guess for the abundances, the TARDIS simulation required fewer iterations to converge, resulting in significantly faster run-times.

In our experiments, we found that the optimal model configuration for this task was a sequential neural network composed of 8 linearly stacked layers, all utilizing a softplus activation function. The first four layers each contained 500 neurons. The subsequent layers, with 100, 75, 50, and 40 neurons respectively, resulting in a total output of 40. We also found that during our grid seach that the the Nadam optimizer to facilitate gradient descent optimization in our model was best.

The subsequent figures provide a visualization of the results obtained from our model. Figure 1 displays the distribution of the MeanFE and MaxFE for the testing dataset, with the results being log-scaled to offer a better representation of the distribution. The results indicate that, on average, the model predicts the target values with relatively low error.

Leading into 4, we provide a comparison between the actual and predicted values of Dilution Factor and Radiative Temperature for a selected subset of the testing data. This subset consists of the best and worst fits as determined by the MaxFE and MeanFE metrics. The close alignment between the true values and the model's predictions underscores the model's ability to accurately reproduce the target variables. However, the worst fit reveals instances of overgeneralization, which could stem from inherent noise or limitations in the training dataset. Despite these occurrences, the overall performance highlights the efficacy of the neural network architecture, optimization strategy, and other hyperparameters used in this study, substantiating the model's reliability and robustness for supernovae ejecta predictions.

In our analysis, as shown in 5, we noticed that the model's validation error increases rapidly during the training process, consistently remaining above the training loss. This behavior implies potential overfitting to the training data, as the model learns the

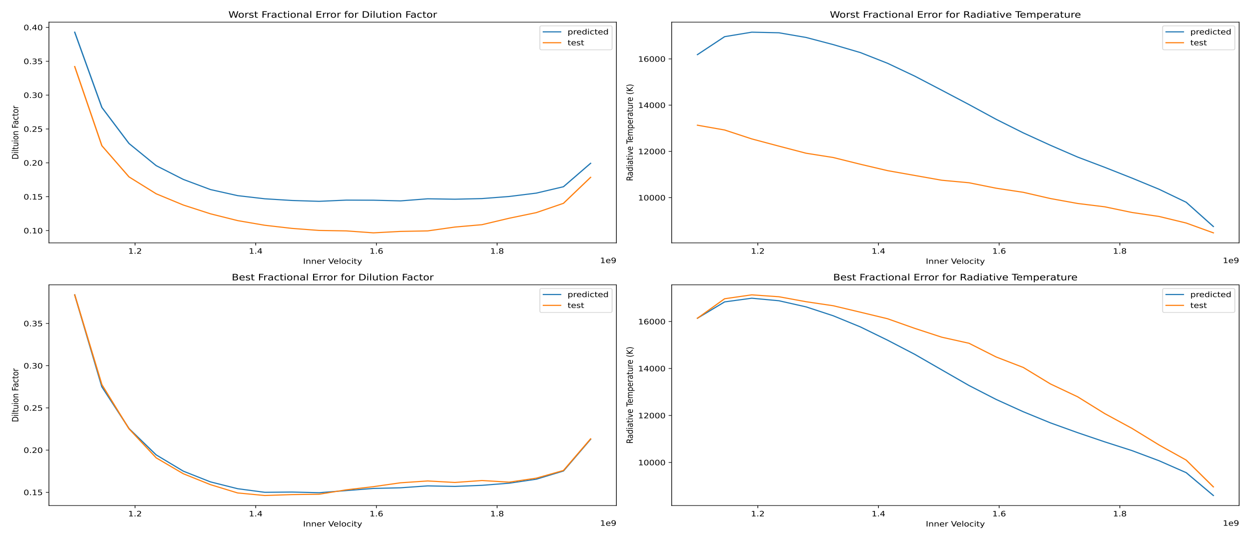

Figure 4: A comparison of the best and worst fits using the MaxFE and MeanFE metrics, illustrating the range of model performance on the testing dataset. The best fits represent cases where the model's predictions closely match the true values, while the worst fits exhibit larger discrepancies between the predicted and true values.

noise or patterns specific to the training set instead of generalizing effectively to the unseen validation data. The gap between the training loss and validation error indicates a degree of overfitting, necessitating strategies to improve the model's performance on new, unseen data.

To mitigate overfitting, various techniques are employed, such as regularization methods (L1, L2, or dropout), early stopping, or adjusting the model's complexity by reducing the number of layers or neurons. Implementing these strategies may help the model generalize better to the validation data, thus narrowing the gap between the training loss and validation error. Assessing the impact of these techniques and their combinations contributes to the development of a more robust model, and enhances its performance in predicting the plasma state of the supernovae ejecta.

Despite the presence of some inaccuracies in the model, the predictions generated by our emulator serve as a valuable starting point for TARDIS in estimating the plasma state of the supernova ejecta. By supplying TARDIS with these initial estimates, the iterative process involved in refining the solution can be significantly shortened. As a result, computational resources and time can be conserved, ultimately leading to a more efficient way of modeling supernova ejecta.

Having showcased the emulator's architecture and its predictive capabilities in this section, we will now proceed to the discussion section. Here, we will delve deeper into the implications of our results and touch on some of the potential avenues for future research and improvements in this area.

4 Discussion

In this study, we developed a neural network model to predict supernovae ejecta properties, specifically Dilution Factor and Radiative Temperature, based on the generated abundances of elemental species. Our optimized model configuration, consisting of eight linearly stacked layers with softplus activation functions, Nadam optimization, and batch normalization,

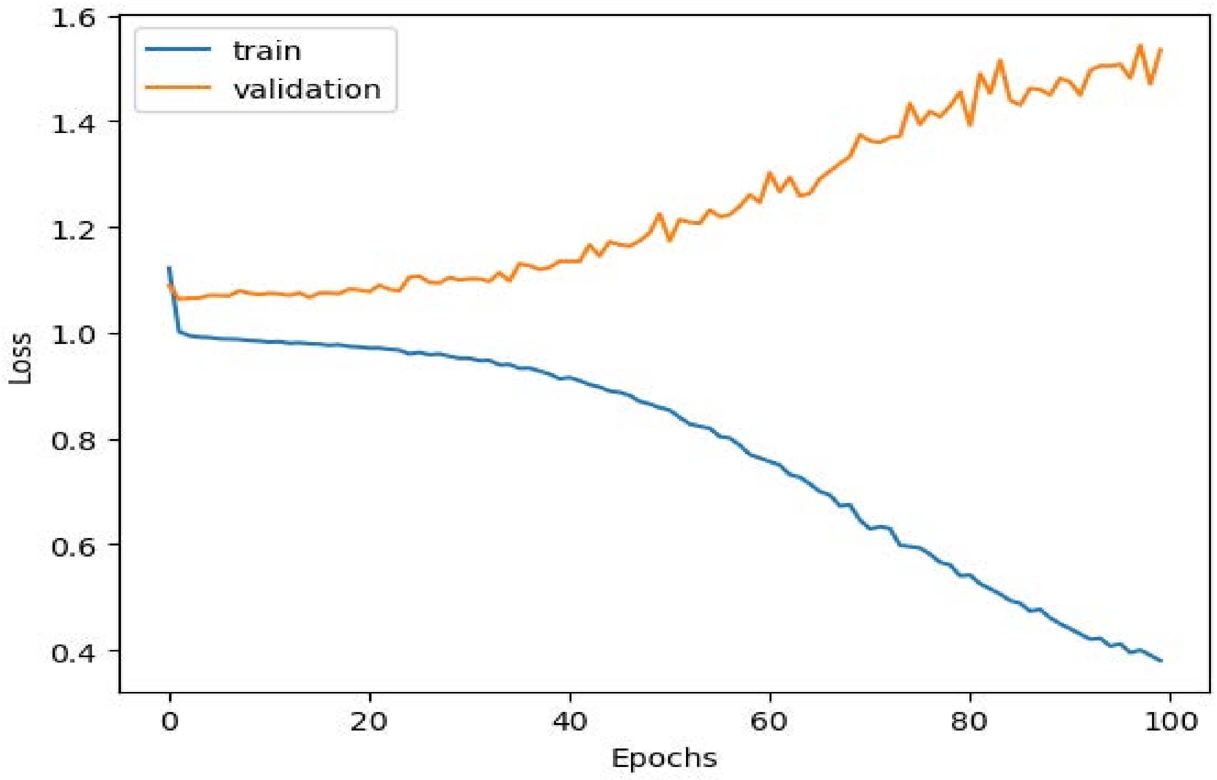

Figure 5: Validation and training loss curves over epochs during the model training process. The plot highlights the divergence between training loss and validation error, suggesting potential overfitting as the model learns noise or patterns specific to the training set. This behavior emphasizes the need for strategies to improve the model's generalization to unseen data.

demonstrated effective performance in capturing the underlying patterns in the data and generating accurate predictions. The model's robustness and reliability make it a promising tool for advancing our understanding of supernovae and their associated astrophysical phenomena.

Our model's performance on both training and testing datasets was consistent, indicating successful generalization to unseen data without overfitting or underfitting issues. The MeanFE and MaxFE metrics further supported the model's capability to accurately predict the Dilution Factor and Radiative Temperature. However, instances of overgeneralization were observed in some cases, which could be attributed to inherent noise or limitations in the training dataset. Identifying and addressing these limitations would enhance the model's overall performance and reliability.

A notable observation in our analysis was the rapid increase in validation error during the training process, suggesting potential overfitting to the training data. This issue warrants the exploration of strategies to improve the model's performance on new, unseen data, such as regularization methods (L1, L2, or dropout), early stopping, or adjusting the model's complexity by reducing the number of layers or neurons. Implementing and evaluating these strategies could contribute to the development of a more robust model, better equipped to generalize to unseen data, thus enhancing its performance and reliability in predicting supernovae ejecta.

5 Conclusion and Future Work

In this study, we developed a neural network model to predict supernovae ejecta properties, specifically Dilution Factor and Radiative Temperature, based on the generated abundances of elemental species. Our optimized model configuration demonstrated promising performance in capturing the underlying patterns in the data and generating accurate predictions, contributing to improve the convergence of the plasmae state of TARDIS, even when solutions are not exact.

While our study achieved significant results, it also highlighted areas for improvement and future research. Refining the sampling method for elemental abundances, particularly iron, and addressing instances of overgeneralization in the model could enhance its overall performance and reliability. Moreover, exploring strategies to mitigate overfitting, such as regularization methods, early stopping, or adjusting the model's complexity, would contribute to the development of a more robust model better equipped to generalize to unseen data.

Future work could involve the integration of more sophisticated sampling methods for elemental abundances, the incorporation of additional constraints from observational or theoretical studies, and the evaluation of other machine learning models or architectures. By refining the model and addressing its limitations, the model could be extened to a variety of other supernovae subtypes other than the Ia type that we focused on this study.

Ultimately, our study demonstrates the potential of neural network-based approaches in accelerating spectral synthesis for scientists to model supernovae. The potential to adapt this to other synthesizer codes is a possibility where further work must be done.

References

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

T. Dozat. Incorporating nesterov momentum into adam. 2016.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

W. Kerzendorf, S. Sim, C. Vogl, M. Williamson, E. Pássaro, A. Flörs, Y. Camacho, V. Jančauskas, A. Harpole, U. Nöbauer, S. Lietzau, M. Mishin, F. Tsamis, A. Boyle, L. Shingles, V. Gupta, K. Desai, M. Klauser, F. Beaujean, A. Suban-Loewen, E. Heringer, B. Barna, G. Gautam, A. Fullard, I. Smith, K. Cawley, J. Singhal, A. Arya, J. O'Brien, D. Sondhi, T. Barbosa, J. Yu, M. Patel, K. Varanasi, J. Gillanders, S. Chitchyan, A. Savel, Y. Eweis, M. Reinecke, T. Bylund, L. Bentil, J. Eguren, A. Alam, M. Bartnik, R. Varma Buddaraju, M. Magee, J. Shields, S. Kambham, R. Livneh, S. Mishra, S. Rajagopalan, R. Jain, J. Reichenbach, A. Floers, S. Singh, A. Brar, A. Holas, J. Bhakar, N. Kowalski, A. Kumar, C. Sofiatti, A. Patidar, J. Selsing, C. Talegaonkar, S. Prasad, S. Venkat, S. Sharma, N. Patra, P. Patel, P. Singh Rathore, K. Yap, N. Sarafina, M. Zaheer, M. Sandler, L. Truong, S. Singh, A. Kumar, S. Gupta, T. Lemoine, U. Wahi, Y. Aggarwal, H. Gupta, D. Volodin, A. PATIDAR, D. Dasgupta, A. Nayak U, A. Kharkar, C. Kolliboyina, and L. Martinez. tardis-sn/tardis: Tardis v2023.04.16, Apr. 2023. URL `https://doi.org/10.5281/zenodo.7832336`.

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

M. Stehle, P. A. Mazzali, S. Benetti, and W. Hillebrandt. Abundance stratification in type ia supernovae - i. the case of SN 2002bo. *Monthly Notices of the Royal Astronomical Society*, 360(4):1231–1243, jul 2005. doi: 10.1111/j.1365-2966.2005.09116.x. URL `https://doi.org/10.1111%2Fj.1365-2966.2005.09116.x`.

C. Vogl, W. E. Kerzendorf, S. A. Sim, U. M. Noebauer, S. Lietzau, and W. Hillebrandt. Spectral modeling of type II supernovae. *Astronomy & Astrophysics*, 633:A88, jan 2020. doi: 10.1051/0004-6361/201936137. URL `https://doi.org/10.1051%2F0004-6361%2F201936137`.

A Appendix A

https://github.com/AlexanderGrunewald/cmse492