The Astrophysical Journal Letters, 910:L23 (5pp), 2021 April 1 https://doi.org/10.3847/2041-8213/abeb1b

Dalek: A Deep Learning Emulator for TARDIS

Wolfgang E. Kerzendorf[1,2], Christian Vogl[3,4], Johannes Buchner[5], Gabriella Contardo[6], Marc Williamson[7], and Patrick van der Smagt[8,9]
[1] Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48824, USA; wkerzendorf@gmail.com, wkerzend@msu.edu
[2] Department of Computational Mathematics, Science, and Engineering, Michigan State University, East Lansing, MI 48824, USA
[3] Max-Planck-Institut für Astrophysik, Karl-Schwarzschild-Str. 1, 85748 Garching, Germany
[4] Exzellenzcluster ORIGINS, Boltzmannstr. 2, 85748 Garching, Germany
[5] Max-Planck-Institut für extraterrestrische Physik, Giessenbachstrasse 1, D-85748 Garching bei München, Germany
[6] Center for Computational Astrophysics, Flatiron Institute, New York, NY 10010, USA
[7] Department of Physics, New York University, New York, NY 10003, USA
[8] Machine Learning Research Lab, Volkswagen AG, Munich, Germany
[9] Faculty of Informatics, Eötvös Loránd University, Budapest, Hungary
Received 2020 June 25; revised 2021 February 1; accepted 2021 February 10; published 2021 April 6

Abstract

Supernova spectral time series contain a wealth of information about the progenitor and explosion process of these energetic events. The modeling of these data requires the exploration of very high dimensional posterior probabilities with expensive radiative transfer codes. Even modest parameterizations of supernovae contain more than 10 parameters and a detailed exploration demands at least several million function evaluations. Physically realistic models require at least tens of CPU minutes per evaluation putting a detailed reconstruction of the explosion out of reach of traditional methodology. The advent of widely available libraries for the training of neural networks combined with their ability to approximate almost arbitrary functions with high precision allows for a new approach to this problem. Instead of evaluating the radiative transfer model itself, one can build a neural network proxy trained on the simulations but evaluating orders of magnitude faster. Such a framework is called an emulator or surrogate model. In this work, we present an emulator for the TARDIS supernova radiative transfer code applied to Type Ia supernova spectra. We show that we can train an emulator for this problem given a modest training set of 100,000 spectra (easily calculable on modern supercomputers). The results show an accuracy on the percent level (that are dominated by the Monte Carlo nature of TARDIS and not the emulator) with a speedup of several orders of magnitude. This method has a much broader set of applications and is not limited to the presented problem.

Unified Astronomy Thesaurus concepts: Supernovae (1668); Neural networks (1933); Radiative transfer (1335)

1. Introduction

Supernova spectra arise from a complex interplay of processes. Simulating them self-consistently is a computationally intensive endeavor ranging from single simulations taking several CPU minutes to thousands of CPU hours on large supercomputers.

The TARDIS (Kerzendorf & Sim 2014) supernova spectrum synthesis code can evaluate a single parameterized explosion model within ≈10 CPU minutes with some approximations that have a minor impact on the output. One of the goals of TARDIS is to perform a Bayesian parameter inference on spectral time series. However, even for a very simple model for a single supernova spectrum with a fixed density profile and 10 uniform abundances this results in a dozen parameters. Such parameter spaces require millions of evaluations for parameter searches, which is infeasible even for fast codes like TARDIS.

Emulators are a solution to this problem (see Czekala et al. 2015 for an early implementation of emulators). These constructs approximate simulations by using functions that are easy to fit to a grid of simulations and are fast to evaluate. Lietzau (2017) did attempt to emulate TARDIS using princpal component analysis (PCA) and Gaussian process (GP) regression. Lietzau's (2017) emulator worked on 11 abundances for (SN Ia) simulations with TARDIS but was not able to work on the full set of 13 parameters. Vogl et al. (2020) showed that using PCA and GP emulator techniques worked for the lower (five) dimensional space of (SN IIP) spectra.

Neural networks have been shown to be universal function approximators (Cybenko 1989; Hornik et al. 1989). TARDIS can be seen as a function that takes an input vector of parameters and transforms these into a spectral vector. We emulate an equivalent parameter space (our parameter space taking nuclear decay into account) to the work of Lietzau (2017) using neural networks.

In Section 2, we describe the methods used in this emulation attempt. Section 3 summarizes the performance of the emulator (accuracy and computational efficiency). We conclude the Letter in Section 4 and give an outlook of future work.

2. Methods

The aim of this Letter is to show that the neural network emulator technique is precise enough to be used for exploring supernova spectra with radiative transfer codes. As an example we choose a Type Ia supernova (SN Ia) spectrum at roughly 10 days before maximum using a uniform model for the abundances. We varied the abundances of nine elements, one isotope, the velocity of the inner boundary, and temperature of the inner boundary (see Kerzendorf & Sim 2014 for a description of these parameters). All other parameters of the model remain fixed. We chose the density profile `branch85_w7` (a power-law density profile; see Kerzendorf & Sim 2014 for details) and an outer boundary velocity of 20,000 km s^{-1}. The plasma calculation used the `nebular` setting for ionization and `dilute-lte` setting for excitation. We use the formal integral calculated spectrum for our emulation purposes.[10] The input TARDIS configuration file is available in the data cache linked to this Letter: doi:10.5281/zenodo.4552670.

There are several steps to construct an emulator for TARDIS: (1) selecting the training set of parameters covering the necessary parameters for the specific problem; (2) calculating the TARDIS spectra for the training set; (3) constructing a neural network architecture; (4) training the neural network architecture.

2.1. *Generating the Training Set*

We are trying to use a parameter space that is close to physically realistic values. SN 2002bo is one of the most well studied SNe Ia (309 results in ADS). Stehle et al. (2005) have done a detailed abundance tomography on this object, and Kerzendorf (2011) used this object for initial automated fitting attempts, which are a precursor to the work presented here. The assumptions made in TARDIS make it most accurate before maximum and we will focus on 8.9 days after explosion (roughly 10 days before maximum). We divide the creation of the training set into finding suitable abundance combinations and finding suitable inner boundary velocity and temperature combinations.

Stehle et al. (2005) model the spectrum at 8.9 days after explosion using $v_{\mathrm{inner}} = 13{,}900$ km s^{-1} and $T_{\mathrm{inner}} = 11{,}850$ K. We construct a uniformly spaced training grid with inner boundary temperatures (T_{inner}) between 10,000 and 14,000 K and inner velocities (v_{inner}) between 10,000 and 15,000 km s^{-1}. This grid safely contains the accepted values of the parameters presented in Stehle et al. (2005).

We rely on theoretical nucleosynthesis calculations given in the Heidelberg Supernova Model Archive (HESMA[11]) to find physically viable abundances. We use 62 spherically averaged isotopic models (presented in the following papers: Fink et al. 2010, 2018, 2014; Kromer et al. 2010, 2016, 2013a, 2013b, 2015; Pakmor et al. 2010, 2012; Sim et al. 2010, 2012, 2013; Röpke et al. 2012; Seitenzahl et al. 2013, 2016; Summa et al. 2013; Ohlmann et al. 2014; Marquardt et al. 2015; Noebauer et al. 2017) for the creation of the training set (the online data doi:10.5281/zenodo.4552670 contain the specific list of models). We only use abundances that are in cells with velocities above 10,000 km s^{-1} to be self-consistent with our choice of inner boundary velocities.

The training set is created with the abundances of O, C, Mg, Si, S, Ca, Ti, Cr, Fe (stable), and ^{56}Ni. We then calculate the location of the 20% and 80% quantile for each element excluding oxygen. We sample uniformly in $\log_{10}$-space between these two quantiles for all elements. Finally, we "fill" the remaining part of the abundance fraction with oxygen. Oxygen as a "filler" element is widely used in supernova fitting (e.g., Hachinger et al. 2017) due to the fact that spectra are normally not overly sensitive to changes of some percent in the oxygen mass fraction (see Hachinger 2011, Section 2.2.5.2).

We removed any combination of these parameters that would lead to an input luminosity of less than 1×10^{42} erg s^{-1}. The extent of the training parameter set can be seen in Table 1.

We experimented with several choices of number of packets for each Monte Carlo iteration and gauged the variation for the spectrum creation resulting from the Monte Carlo nature of TARDIS (see Kerzendorf & Sim 2014 for details of this process). The choice of 100,000 Monte Carlo packets for each iteration (opting for 30 iterations in total and increasing the number to 200,000 packets for the last iteration) resulted in spectra that had less than 1% intrinsic noise—far lower than the systematic uncertainties present in the comparison between data and spectra.

We calculated a training/validation data set with 98,000 samples and a test set with 19,930 samples on the Michigan State University (MSU) high-performance cluster provided by the Institute for Cyber-Enabled Research.

We resampled the spectra from TARDIS on a logarithmic grid between 3400 Å and 7600 Å to make the line structures across the spectrum have roughly equal pixels per structure. The final data set has 12 input parameters and 500 spectral data points. The input TARDIS file, parameters (abundances, inner boundary velocity/temperature), and spectra are available at 10.5281/zenodo.4552670.

2.2. *Neural Network Architecture and Training*

We split the group of 98,000 spectra into a set of 68,600 (=70%) for training the neural networks and 29,400 for cross-validation. Each data point consisted of 12 inputs (the "parameters") and 500 outputs (the "spectra"). Both input and output values were preprocessed by first taking the $\log_{10}$ of the values, after which the values were normalized by removing the mean and scaling to unit variance with STANDARDSCALER (SCIKIT-LEARN; Pedregosa et al. 2011).

We use feed-forward neural networks to efficiently approximate and generalize these data. Even though training a neural network may cost a few hours of computation time, inference with trained neural networks is very fast since it only involves a small number—for the architectures used in this Letter on the order of 10^6—of floating-point operations and a few hundred nonlinear function evaluations.

We trained a number of feed-forward neural networks of different topology on the data. The neural networks were implemented in KERAS on TENSORFLOW 1.14 or 2.0.

Good neural network architectures were found by hyperparameter search. We used cluster-based hyperparameter search using Polyaxon 0.5.6[12] on a cluster of IBM and Nvidia machines, each with multiple Tesla V100 GPUs. Training a single neural network lasts 4–7 hr on such an architecture, and we parallelized over 200 instances.

Table 2 lists the hyperparameters over which we searched, and their range of possible values.

We chose to train the network for 15,000 epochs for networks trained without dropout and 40,000 epochs with dropout. Both numbers were chosen with a considerable margin. From the approximately 4000 runs we selected the best results by analyzing their average loss (using mean squared error) over the cross-validation data set. The best found neural network architectures had a width of 200 to 400 neurons in one or two hidden layers, a softplus activation function, and Nesterov-adam

[10] See https://tardis-sn.github.io/tardis/physics/montecarlo/sourceintegration.html.
[11] https://hesma.h-its.org
[12] https://polyaxon.com/

Table 1
The Distributions of Elemental Abundances (Uniformly in $\log_{10}$-space) for the Training Set

	C	O	Mg	Si	S	Ca	Ti	Cr	Fe	^{56}Ni
min	6.6e-06	0.05	2.5e-05	0.031	0.012	0.0016	3.6e-06	0.00019	0.005	0.025
25%	8.6e-05	0.51	0.00014	0.056	0.022	0.003	6.7e-06	0.00031	0.011	0.052
50%	0.0011	0.63	0.00071	0.1	0.04	0.0055	1.2e-05	0.00049	0.023	0.11
75%	0.013	0.73	0.0038	0.19	0.071	0.01	2.2e-05	0.00078	0.05	0.22
max	0.16	0.92	0.021	0.34	0.13	0.018	4.1e-05	0.0012	0.11	0.46

Note. The two additional parameters have uniform distributions: $T_{\mathrm{inner}} = 10{,}000$–$14{,}000$ K and $v_{\mathrm{inner}} = 10{,}000$–$15{,}000$ km s^{-1}.

Table 2
A Short Explanation of the Hyperparameters

Parameter	Values
# hidden layers	2–6
# neurons/layer	100–500 in steps of 100
batch size	100, 500, 1000, 2000
activation function	tanh, relu, selu, elu, softplus
optimizer	adam, nadam, adadelta, adagrad
dropout rate	0–0.6 in steps of 0.2
batch normalization	after each layer / not at all
initializer	glorot_normal, he_normal

Note. Activation functions: tanh is a known mathematical function that keeps the output of the neuron between +1 and −1; "rectified linear unit" (relu) equals $\mathrm{relu}(x) \equiv \max(0, x)$. Softplus is a smooth version of relu and defined as $\ln(1 + \exp(x))$. The selu function is a normalized relu (Klambauer et al. 2017). The elu (Clevert et al. 2016) goes negative with $a(\exp(x) - 1)$ for $x < 0$. Adadelta (Zeiler 2012), adagrad (Duchi et al. 2011), and adam (Kingma & Ba 2015) are modern second-order optimization methods used in neural network training. Nadam is adam but with Nesterov gradients (Sutskever et al. 2013). Batch normalization (Ioffe & Szegedy 2015) normalizes the activations of a layer of neurons per batch and helps a lot in preventing overfitting. Dropout (Hinton et al. 2012) prevents overfitting by randomly switching hidden units off during training by the given rate. We never combined batch normalization with dropout. Early stopping is always done, by selecting that step in the optimization that has a low error on the cross-validation Set.

as the optimizer. Dropout never improved the results; batch normalization was not among the 10 best but in the 50 best networks (7% worse). Activation function softplus was in the top 30, but the difference in error with neural networks with elu, relu, selu, or tanh activations functions was not more than about 3%. The used batch size had little influence, nor did the choice of initializer.

We then selected the best neural networks for ensemble modeling (Opitz & Maclin 1999). The selected network architectures, those with the lowest loss on the cross-validation data, are listed in Table 3. Ensemble modeling was done by averaging over all listed neural networks.

3. Results

In the following, we used the predictions of the ensemble neural network when comparing with the TARDIS spectra from the test set (unless otherwise noted). We used both the maximum fractional error and mean fractional error metrics (see also Vogl et al. 2020) for comparisons:

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

Table 3
List of the Five Best Neural Network Architectures Used in the Ensemble Training

Depth	Optimizer	Activation	Width
4	nadam	softplus	200
4	adam	softplus	200
3	adam	softplus	300
3	nadam	softplus	400
4	nadam	softplus	200

Note. Network 1 and 5 are the same network architecture but were trained with different starting seeds resulting in different performances.

$$\mathrm{MaxFE} = \max_{i=0}^{N} \frac{|f_{\lambda,i}^{\mathrm{emu}} - f_{\lambda,i}^{\mathrm{test}}|}{f_{\lambda,i}^{\mathrm{test}}}, \tag{2}$$

with N being the number of pixels in our spectra (in our case 500) and $f_{\lambda,i}$ the flux at the ith pixel in the test set. For the training of the emulator we chose to use spectra in $\log_{10}$. However, for the evaluation of the emulator, we will use the linear space as any likelihood comparing the emulated spectrum to an observed spectrum will be in linear flux units.

The ensemble neural network emulator performs well in both metrics with 99% of predictions having a $\mathrm{MaxFE} < 0.049$ and $\mathrm{MeanFE} < 0.014$ and a median prediction of $\mathrm{MaxFe} = 0.016$ and $\mathrm{MeanFE} = 0.004$. Figure 1 shows the best and worst predictions in the test set including residuals.

Figure 2 shows the distribution and also compares the prediction uncertainty to the networks that make up the ensemble. The ensemble has roughly a 10% improvement in MeanFE over the individual networks.

We remind the reader that TARDIS is based on an iterative Monte Carlo algorithm. The method results in variations in the final spectrum given different random seeds. We have run the worst predicting parameter set (see Figure 1) with 100 different seeds to test the variation. Figure 3 shows that the prediction uncertainty of the emulator is close to the uncertainty of the Monte Carlo algorithm.

For the desired application both MeanFE and MaxFE of the emulation will not contribute significantly as the systematic uncertainties will be much larger (MeanFE for SN 2002bo, 18%; see Figure 5.5 in Kerzendorf 2011).

The main reason to use an emulator compared to TARDIS itself is the speedup. The mean and standard deviation runtime for all TARDIS runs during training set creation on a single CPU on the MSU HPCC cluster are 602 s ± 186 s with a minimum of 253 s and a maximum of 2054 s. Ensemble network evaluation takes 85 ± 13.7 ms, which is several thousand times faster than the TARDIS evaluation. A toy example of exploring likelihoods shows that a 20-dimensional problem needs 26 million evaluations (see

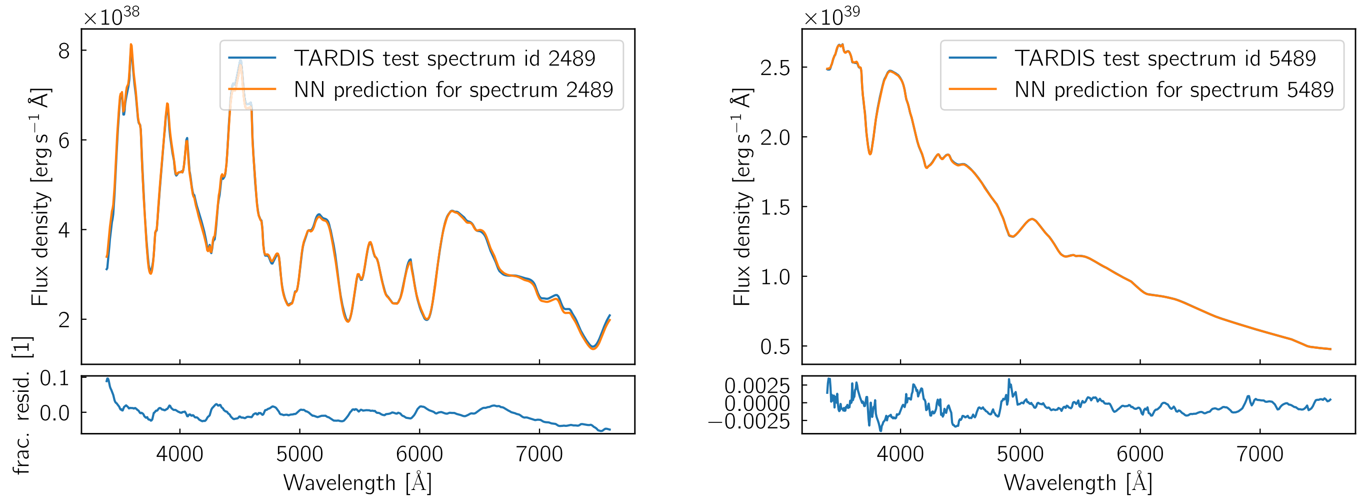

Figure 1. Comparison of a spectrum from the test set with a prediction from the ensemble emulator. We showcase the spectra with the highest and the lowest MaxFE from the set of test set predictions. Left: largest MaxFE from the test set (≈10%). Right: smallest MaxFE from the test set (≈0.4%).

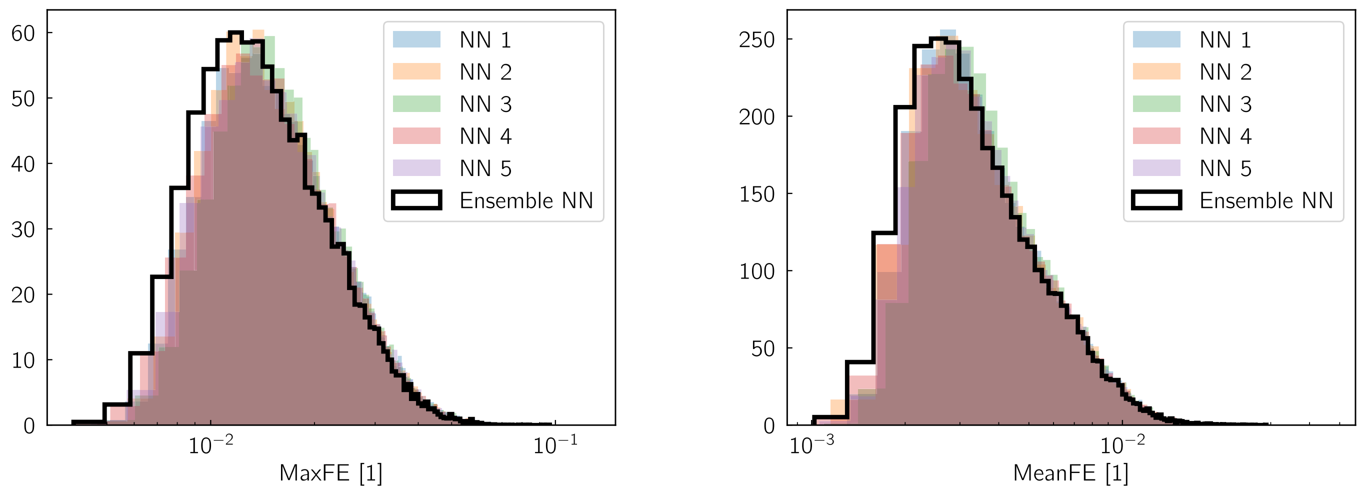

Figure 2. Histogram of prediction uncertainties for the test set using the MaxFE metric on the left and the MeanFE metric on the right.

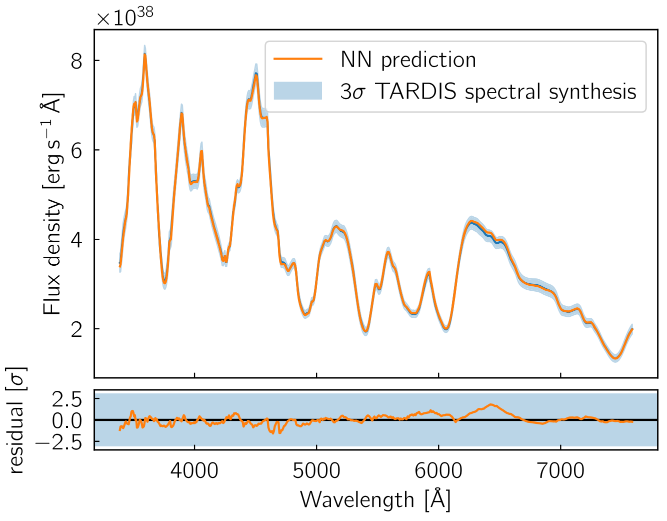

Figure 3. Comparison of the emulator prediction uncertainty with the Monte Carlo uncertainty.

algorithm radfriends in Table 1 in Buchner 2016), which with the emulator is possible within ≈25 days but not achievable without an emulator (≈420 yr). This can be improved by foregoing ensemble modeling and taking a 10% accuracy loss but having an evaluation time of 13.3 ± 0.46 ms, which would complete the exploration within ≈4 days.

4. Conclusion and Future Work

We present a 12-dimensional emulator for the TARDIS radiative transfer code. The emulator can predict the spectrum with an accuracy of on average 1% with a speedup of almost 10,000 in so-called ensemble mode and a speedup of almost 50,000 with a marginally lower accuracy in single mode. A major part of the prediction uncertainty is likely not the emulator itself but noise from the Monte Carlo method of TARDIS. The chosen parameter space is focused on the SNe Ia modeling. However, the general methodology can be applied to a much broader set of problems.

The presented emulator is useful for exploring single spectra with abundances that are uniform throughout the envelope. Initial fitting of supernova spectra including researching likelihoods that incorporate systematic uncertainties to account for the mismatch between TARDIS and observed spectra is already underway.

A complete reconstruction of an exploded object from spectral time series will have more than 100 parameters. This will require the development of more complex emulators. For such parameter spaces, we will need to use more constraining priors when generating the training set. The authors have already experimented with various schemes to find a training set (e.g., drawing from kernel density estimates of the Heidelberg Supernova Model Archive (HESMA) abundances) but such work is outside the current scope of exploring neural networks as function approximators for radiative transfer codes.

We have shown that emulators enable the exploration of high dimensional parameter spaces even with costly simulations. Such tools will be important assets for the data-rich era that astronomy is entering.

Contributor Roles

We use the CRT standard (see https://casrai.org/credit/) for reporting our contributor roles:

1. *Conceptualization*—Kerzendorf, Vogl.
2. *Data curation*—Kerzendorf.
3. *Formal Analysis*—Kerzendorf, PvdS.
4. *Investigation*—Kerzendorf, PvdS, Contardo, Buchner.
5. *Methodology*—Kerzendorf, PvdS.
6. *Resources*—PvdS, Kerzendorf.
7. *Software*—Kerzendorf, PvdS, Vogl, Williamson.
8. *Validation*—Kerzendorf, PvdS.
9. *Visualization*—Kerzendorf, PvdS.
10. *Writing—original draft*—Kerzendorf, PvdS.
11. *Writing—review and editing*—Kerzendorf, PvdS, Vogl, Williamson, Contardo, Buchner.

This research made use of TARDIS, a community-developed software package for spectral synthesis in supernovae (Kerzendorf & Sim 2014; Vogl et al. 2019; Kerzendorf et al. 2020). The development of TARDIS received support from the Google Summer of Code initiative and from ESA's Summer of Code in Space program. TARDIS makes extensive use of Astropy and PyNE. We would also like to thank Gautham Narayan for his thoughtful and constructive referee comments.

W.E.K. was and M.W. is part of SNYU, and the SNYU group is supported by the NSF CAREER award AST-1352405 (PI: Modjaz) and the NSF award AST-1413260 (PI: Modjaz). W.E.K was also supported by an ESO Fellowship and the Excellence Cluster Universe, Technische Universitat Munchen, Boltzmannstrasse 2, D-85748 Garching, Germany, for part of this work. W.E.K would also like to thank the Flatiron Institute for hosting him for some of this work. C.V. was supported for part of this work by the Excellence Cluster ORIGINS, which is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy—EXC-2094—390783311. We would also like to thank David Hogg and Dan Foreman-Mackey for fruitful discussions on the emulator and strategies.

This work was supported in part through computational resources and services provided by the Institute for Cyber-Enabled Research at Michigan State University. Neural network training was done at Volkswagen Group. This work made use of the Heidelberg Supernova Model Archive (HESMA), https://hesma.h-its.org.

ORCID iDs

Wolfgang E. Kerzendorf https://orcid.org/0000-0002-0479-7235
Christian Vogl https://orcid.org/0000-0002-7941-5692
Johannes Buchner https://orcid.org/0000-0003-0426-6634
Gabriella Contardo https://orcid.org/0000-0002-3011-4784
Marc Williamson https://orcid.org/0000-0003-2544-4516
Patrick van der Smagt https://orcid.org/0000-0003-4418-4916

References

Buchner, J. 2016, S&C, 26, 383
Clevert, D.-A., Unterthiner, T., & Hochreiter, S. 2016, arXiv:1511.07289
Cybenko, G. 1989, Math. Control Signals Syst., 2, 303
Czekala, I., Andrews, S. M., Mandel, K. S., Hogg, D. W., & Green, G. M. 2015, ApJ, 812, 128
Duchi, J., Hazan, E., & Singer, Y. 2011, J. Mach. Learn. Res., 12, 2121
Fink, M., Kromer, M., Hillebrandt, W., et al. 2018, A&A, 618, A124
Fink, M., Kromer, M., Seitenzahl, I. R., et al. 2014, MNRAS, 438, 1762
Fink, M., Röpke, F. K., Hillebrandt, W., et al. 2010, A&A, 514, A53
Hachinger, S. 2011, PhD thesis, TU München
Hachinger, S., Röpke, F. K., Mazzali, P. A., et al. 2017, MNRAS, 471, 491
Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. 2012, arXiv:1207.0580
Hornik, K., Stinchcombe, M., & White, H. 1989, NN, 2, 359
Ioffe, S., & Szegedy, C. 2015, arXiv:1502.03167
Kerzendorf, W., Sim, S., Vogl, C., et al. 2020, tardis-sn/tardis: TARDIS v3.0.dev3463, Zenodo, doi:10.5281/zenodo.3902923
Kerzendorf, W. E. 2011, PhD thesis, Australian National Univ.
Kerzendorf, W. E., & Sim, S. A. 2014, MNRAS, 440, 387
Kingma, D. P., & Ba, J. 2015, arXiv:1412.6980
Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. 2017, arXiv:1706.02515
Kromer, M., Fink, M., Stanishev, V., et al. 2013a, MNRAS, 429, 2287
Kromer, M., Fremling, C., Pakmor, R., et al. 2016, MNRAS, 459, 4428
Kromer, M., Ohlmann, S. T., Pakmor, R., et al. 2015, MNRAS, 450, 3045
Kromer, M., Pakmor, R., Taubenberger, S., et al. 2013b, ApJL, 778, L18
Kromer, M., Sim, S. A., Fink, M., et al. 2010, ApJ, 719, 1067
Lietzau, S. 2017, Master's thesis, Technical Univ. Munich, doi:10.5281/zenodo.1312512
Marquardt, K. S., Sim, S. A., Ruiter, A. J., et al. 2015, A&A, 580, A118
Noebauer, U. M., Kromer, M., Taubenberger, S., et al. 2017, MNRAS, 472, 2787
Ohlmann, S. T., Kromer, M., Fink, M., et al. 2014, A&A, 572, A57
Opitz, D., & Maclin, R. 1999, J. Artif. Intell. Res., 11, 169
Pakmor, R., Kromer, M., Röpke, F. K., et al. 2010, Natur, 463, 61
Pakmor, R., Kromer, M., Taubenberger, S., et al. 2012, ApJL, 747, L10
Pedregosa, F., Varoquaux, G., Gramfort, A., et al. 2011, J. Mach. Learn. Res., 12, 2825
Röpke, F. K., Kromer, M., Seitenzahl, I. R., et al. 2012, ApJL, 750, L19
Seitenzahl, I. R., Ciaraldi-Schoolmann, F., Röpke, F. K., et al. 2013, MNRAS, 429, 1156
Seitenzahl, I. R., Kromer, M., Ohlmann, S. T., et al. 2016, A&A, 592, A57
Sim, S. A., Fink, M., Kromer, M., et al. 2012, MNRAS, 420, 3003
Sim, S. A., Röpke, F. K., Hillebrandt, W., et al. 2010, ApJL, 714, L52
Sim, S. A., Seitenzahl, I. R., Kromer, M., et al. 2013, MNRAS, 436, 333
Stehle, M., Mazzali, P. A., Benetti, S., & Hillebrandt, W. 2005, MNRAS, 360, 1231
Summa, A., Ulyanov, A., Kromer, M., et al. 2013, A&A, 554, A67
Sutskever, I., Martens, J., Dahl, G., & Hinton, G. 2013, ICML, 28, 1139, http://proceedings.mlr.press/v28/sutskever13.html
Vogl, C., Kerzendorf, W. E., Sim, S. A., et al. 2020, A&A, 633, A88
Vogl, C., Sim, S. A., Noebauer, U. M., Kerzendorf, W. E., & Hillebrandt, W. 2019, A&A, 621, A29
Zeiler, M. D. 2012, arXiv:1212.5701