














# Accurate Machine-learning Atmospheric Retrieval via a Neural-network Surrogate Model for Radiative Transfer

Michael D. Himes<sup>1</sup> , Joseph Harrington<sup>2</sup> , Adam D. Cobb<sup>3</sup> , Atılım Güneş Baydin<sup>3</sup> , Frank Soboczenski<sup>4</sup> , Molly D. O’Beirne<sup>5</sup> , Simone Zorzan<sup>6</sup> , David C. Wright<sup>1</sup> , Zacchaeus Scheffer<sup>1</sup> , Shawn D. Domagal-Goldman<sup>7</sup> , and Giada N. Arney<sup>7</sup> 

<sup>1</sup> Planetary Sciences Group, Department of Physics, University of Central Florida, USA; [mhimes@knights.ucf.edu](mailto:mhimes@knights.ucf.edu)

<sup>2</sup> Planetary Sciences Group, Department of Physics and Florida Space Institute, University of Central Florida, USA

<sup>3</sup> Department of Engineering Science, University of Oxford, UK

<sup>4</sup> SPHES, King’s College London, UK

<sup>5</sup> Department of Geology and Environmental Science, University of Pittsburgh, USA

<sup>6</sup> ERIN Department, Luxembourg Institute of Science and Technology, Luxembourg

<sup>7</sup> NASA Goddard Space Flight Center, Greenbelt, MD, USA

Received 2020 March 4; revised 2021 January 22; accepted 2021 February 4; published 2022 April 25

## Abstract

Atmospheric retrieval determines the properties of an atmosphere based on its measured spectrum. The low signal-to-noise ratios of exoplanet observations require a Bayesian approach to determine posterior probability distributions of each model parameter, given observed spectra. This inference is computationally expensive, as it requires many executions of a costly radiative transfer (RT) simulation for each set of sampled model parameters. Machine learning (ML) has recently been shown to provide a significant reduction in runtime for retrievals, mainly by training inverse ML models that predict parameter distributions, given observed spectra, albeit with reduced posterior accuracy. Here we present a novel approach to retrieval by training a forward ML surrogate model that predicts spectra given model parameters, providing a fast approximate RT simulation that can be used in a conventional Bayesian retrieval framework without significant loss of accuracy. We demonstrate our method on the emission spectrum of HD 189733 b and find good agreement with a traditional retrieval from the Bayesian Atmospheric Radiative Transfer (BART) code (Bhattacharyya coefficients of 0.9843–0.9972, with a mean of 0.9925, between 1D marginalized posteriors). This accuracy comes while still offering significant speed enhancements over traditional RT, albeit not as much as ML methods with lower posterior accuracy. Our method is  $\sim 9\times$  faster per parallel chain than BART when run on an AMD EPYC 7402P central processing unit (CPU). Neural-network computation using an NVIDIA Titan Xp graphics processing unit is  $90\times$ – $180\times$  faster per chain than BART on that CPU.

*Unified Astronomy Thesaurus concepts:* [Exoplanet atmospheres \(487\)](#); [Bayesian statistics \(1900\)](#); [Posterior distribution \(1926\)](#); [Convolutional neural networks \(1938\)](#); [Neural networks \(1933\)](#)

## 1. Introduction

Over the past decades, exoplanet studies have expanded from their detection to include characterization of their atmospheres via retrieval (see reviews by Seager & Deming 2010; Deming & Seager 2017). Retrieval is the inverse modeling technique whereby forward models of a planet’s spectrum are compared to observational data in order to constrain the model parameters (see review by Madhusudhan 2018). These typically include the shape of the thermal profile, abundances of species, and condensate properties. While some solar system objects can be characterized with simpler approaches (such as Levenberg–Marquardt minimization) due to their high signal-to-noise ratios (e.g., Koskinen et al. 2016), retrieval on noisy exoplanet spectra require Bayesian methods to provide a distribution of models that can explain the observed data. The posterior distribution resulting from a Bayesian retrieval places limits on each model parameter (within some range, an upper or lower limit, or equally probable for all values considered), informing the statistical significance of the result.

Bayesian retrieval methods involve evaluating thousands to millions of spectra, integrating over the observational

bandpasses, and comparing to observations. Depending on model complexity, this requires hundreds to thousands of parallelizable compute hours, resulting in hours to days of runtime. Calculating the model spectra by solving the radiative transfer (RT) equation takes the vast majority of compute time.

Machine learning (ML) encompasses algorithms that learn representations of and uncover relationships within a collection of data samples. Deep learning (Goodfellow et al. 2016) is a subfield of ML that is based on neural networks, which are highly flexible differentiable functions that can be fit to data. Neural networks can classify images (e.g., Krizhevsky et al. 2012; Simonyan & Zisserman 2015; Szegedy et al. 2015; He et al. 2016; Huang et al. 2017), recognize speech (e.g., Chorowski et al. 2014; Amodei et al. 2016; Chan et al. 2016; Xiong et al. 2016), and translate between languages (e.g., Cho et al. 2014; Bahdanau et al. 2015; Ranzato et al. 2016; Sennrich et al. 2016; Wu et al. 2016). Neural networks consist of a hierarchy of layers that contain nodes performing weighted (non)linear transformations of their inputs, through a series of hidden layers, to the desired output. For example, for a retrieval, one might have the input layer receive the observed spectrum, hidden layers extract features, and the output layer predict the underlying atmospheric parameters. Neural-network training conventionally uses gradient-based optimization, iteratively adjusting the weights of the connections between



Original content from this work may be used under the terms of the [Creative Commons Attribution 4.0 licence](#). Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

nodes to minimize the error between the neural network’s prediction and the desired output (Rumelhart et al. 1986).

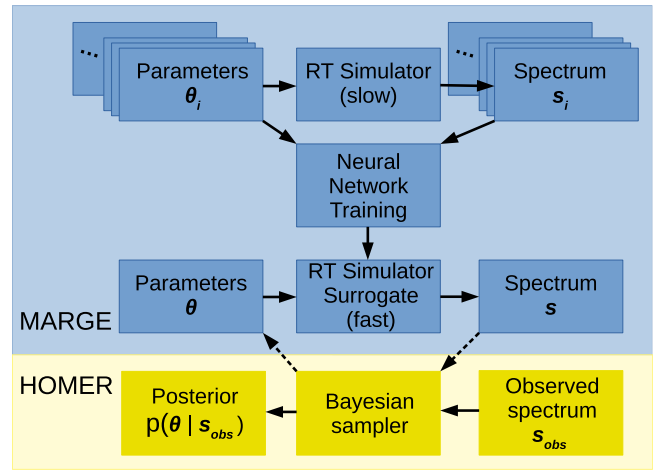
Recent applications of ML to atmospheric retrieval reduced compute time from hundreds of hours to minutes or less. Márquez-Neila et al. (2018) presented a random forest of regression trees to build predictive distributions comparable to the posterior distributions of traditional Bayesian retrievals. Zingales & Waldmann (2018) utilized a generative adversarial network (GAN; Goodfellow et al. 2014) to retrieve distributions for model parameters. Waldmann & Griffith (2019) used a convolutional neural network (CNN) to map spatial and spectral features across Saturn. In Cobb et al. (2019), we introduced *plan-net*, an ensemble of Bayesian neural networks that uses parameter correlations to inform the uncertainty on retrieved parameters. Hayes et al. (2020) demonstrated a new approach to ML retrieval by applying *k*-means clustering to a principal component analysis of the observed spectrum to inform a standard Bayesian retrieval. Johnsen & Marley (2020) showed that a dense neural network can provide quick estimations of atmospheric properties.

While these approaches are promising, all except Hayes et al. (2019) suffer from a common deficiency: the reduction in computational time is accompanied by a reduction in posterior accuracy because they make significant approximations when performing Bayesian inference. For ML to become an integral part of atmospheric retrieval, the accuracy of the posterior approximation must be preserved.

The solution lies in simulation-based inference methods (Cranmer et al. 2019). While directly using a simulator (e.g., RT code) requires a consistent amount of compute time for each new inference (e.g., retrieval), surrogate models that emulate the simulator (e.g., neural networks) allow new data to be quickly evaluated after an upfront computational cost to train the surrogate (Munk et al. 2019; Kasim et al. 2022). ML- and simulation-based inference approaches have been successfully applied to a variety of tasks ranging from quantum chemistry (Gilmer et al. 2017) to particle physics (Brehmer et al. 2018; Baydin et al. 2019), resulting in significant reductions in compute cost with minimal loss in accuracy.

Similar approaches have been used by the Earth science community to reduce the computational burden of forward modeling of spectra, retrieval of surface conditions, and atmospheric correction (e.g., Atzberger 2004; Garcia-Cuesta et al. 2009; Rivera et al. 2015; Verrelst et al. 2015; Gómez-Dans et al. 2016; Verrelst et al. 2016, 2017; Chernetskiy et al. 2018; Yin et al. 2018; Vicent et al. 2018; Bue et al. 2019).

Here we present a novel application of this approach to retrieval, which uses a neural-network model of RT within a Bayesian framework, apply it to the emission spectrum of HD 189733 b, and compare the results to a classical retrieval using the RT code that trained the surrogate model. Our general method is to (1) generate a data set over some parameter space, (2) train a surrogate forward model on the generated data, and (3) infer the inverse process via a Bayesian sampler (Figure 1). Our approach circumvents the existing limitations of ML retrieval methods, which seek to directly learn the inverse process, by learning the forward, deterministic process (RT) and using the simulator surrogate in a standard inference pipeline. This approach preserves the accuracy of the Bayesian inference and, while slower than direct ML retrieval, is still much faster than computing RT.



**Figure 1.** Schematic diagram of our inverse modeling method, color-coded based on the scope of our software packages. MARGE (Section 2.3.1) generates a data set based on a deterministic, forward process (e.g., RT) and trains a surrogate model to approximate that process. Using the trained surrogate, HOMER (Section 2.3.2) infers the inverse process (e.g., atmospheric retrieval) by simulating many forward models and comparing them to the target data (e.g., an observed spectrum) in a Bayesian framework.

In Section 2, we describe our approach in detail as well as introduce the software packages that implement the method. Section 3 discusses the results. Finally, Section 4 presents conclusions.

## 2. Methods

### 2.1. Model Training

To train a neural network for our approach (Figure 1), we generate a data set of spectra using the Bayesian Atmospheric Radiative Transfer (BART) code (Blecic et al. 2022; Cubillos et al. 2022; Harrington et al. 2022).

The atmospheric models consist of 100 log-uniform layers spanning pressures from  $10^{-8}$  to 100 bar, and we assume that the planet radius corresponds to a pressure of 0.1 bar. We use the five-parameter temperature–pressure profile,  $T(p)$ , the parameterization of Line et al. (2013);  $\kappa$ , the Planck mean infrared opacity;  $\gamma_1$  and  $\gamma_2$ , the ratios of the Planck mean visible and infrared opacities for each of two streams;  $\alpha$ , which controls the contribution of the two streams; and  $\beta$ , which represents albedo, emissivity, and energy recirculation. We allow the radius ( $R_p$ ), mass ( $M_p$ ), and semimajor axis ( $a$ ), adjusts the temperature at the top of the atmosphere due to stellar irradiation) of the planet to vary to encompass a range of hot Jupiters. We also include a free parameter for each of the uniform vertical abundance profiles of  $\text{H}_2\text{O}$ ,  $\text{CO}_2$ ,  $\text{CO}$ , and  $\text{CH}_4$ .

We allow a wide range of values without regard for physical plausibility, except by enforcing that (1) the  $\text{H}_2/\text{He}$  ratio remains constant, (2) the total relative abundances of molecules in the atmosphere equals 1, and (3) the  $T(p)$  profile does not exceed the temperature range of the line lists. For example, this could lead to models with  $\text{H}_2\text{O}$  at conditions where it dissociates (Arcangeli et al. 2018), though in the case of HD 189733 b, such models would be rejected with a high probability due to a poor fit. We note that these are not fundamental constraints of our approach; other constraints (e.g., enforcing thermochemical equilibrium, keeping elemental ratios within some range) may be used when generating the data set to train the surrogate model.

For opacities, we use HITEMP for  $\text{H}_2\text{O}$ ,  $\text{CO}$ , and  $\text{CO}_2$  (Goorvitch 1994; Tashkun et al. 2003; Barber et al. 2006;

**Table 1**  
Forward Model Parameter Space

Parameter	Minimum	Maximum
$\log \kappa$	-5.0	1.0
$\log \gamma_1$	-2.0	2.0
$\log \gamma_2$	-1.3	1.3
$\alpha$	0.0	1.0
$\beta$	0.7	1.3
$R_p (R_J)$	0.8	1.5
$M_p (M_J)$	0.8	1.5
$a$ (AU)	0.2	0.4
$\log \text{H}_2\text{O}$	-13	-0.5
$\log \text{CO}_2$	-13	-0.5
$\log \text{CO}$	-13	-0.5
$\log \text{CH}_4$	-13	-0.5

Rothman et al. 2010), HITRAN for  $\text{CH}_4$  (Niederer et al. 2008; Boudon et al. 2010; Nikitin et al. 2010, 2011; Brown et al. 2013; Campargue et al. 2013; Daumont et al. 2013; Niederer et al. 2013; Nikitin et al. 2013; Rothman et al. 2013), and collision-induced absorptions of  $\text{H}_2\text{--H}_2$  and  $\text{H}_2\text{--He}$  (Borysow et al. 2001; Borysow 2002; Abel et al. 2012; Richard et al. 2012). While there are newer line lists available with a greater number of lines (e.g., Hargreaves et al. 2020), these tests are meant to demonstrate consistency between neural-network-based and non-ML retrievals; we therefore use the setup described in Harrington et al. (2022), which uses this set of line lists to compare with previous studies. As our approach learns RT from a data set of spectra, it is not tied to any specific line lists.

To train our neural-network surrogate model, we generate 3,458,432 spectra, which are subdivided into 2,446,784 spectra ( $\sim 70\%$ ) for training, 689,536 spectra ( $\sim 20\%$ ) for validation, and 322,112 spectra ( $\sim 10\%$ ) for testing (for considerations about data set size, see Appendix B). Model parameters come from the uniform distribution bound by the limits listed in Table 1. Each spectrum spans  $280\text{--}7100\text{ cm}^{-1}$  at a resolution of  $1.0\text{ cm}^{-1}$  and corresponds to the planet’s emitted flux in  $\text{erg s}^{-1}\text{ cm}^{-1}$ .

When processing the BART inputs/outputs for our neural network, we simplify the neural-network inputs by transforming the planet mass into the surface gravity, because this is a factor in the integration to calculate the spectrum. We assume a host star of radius  $0.756 R_\odot$  with a temperature of 5000 K to calculate the  $T(p)$  profiles; because  $\beta$  acts as a scaling factor on the related term (Equation (15) of Line et al. 2013), it can compensate for different stellar fluxes.

We normalize the input and output data by (1) taking the logarithm of the output spectra, (2) standardizing the inputs and (log) outputs by subtracting the training mean and dividing by the training standard deviation, and (3) scaling the standardized inputs and outputs to be in the range  $[-1, 1]$ . The neural network’s input layer corresponds to the 12 inputs described above, with surface gravity replacing planetary mass. The hidden layers consist of Conv1d(256)L(0.05)—Dense(4096)L(0.05)—Dense(4096)L(0.05)—Dense(4096)L(0.05)—Dense(4096)L(0.05). Conv1d ( $n$ ) indicates a 1D convolutional layer with  $n$  feature maps and a kernel size of 3. L( $m$ ) indicates a leaky rectified linear unit (ReLU) activation function with slope  $m$  for  $x < 0$ . The dense output layer has 6821 nodes, corresponding to the emitted spectrum over the defined wavenumber grid, with a ReLU activation function. For details on our model selection process, see Appendix A.

We train with a batch size of 64 using a mean-squared error loss function, the Adam optimizer, and early stopping with a patience of 30 epochs based on the validation loss. We employ a cyclical learning rate that increases from  $8 \times 10^{-6}$  to  $5 \times 10^{-3}$  over four epochs, then decreases over the same window. After each complete cycle (eight epochs), the maximum learning rate decays by half the difference between the maximum and minimum learning rates (triangular2 policy; Smith 2015). The boundaries were chosen according to the method described in Smith (2015), except that we consider the loss instead of accuracy (see Appendix A for details). To evaluate the model’s performance, we compute the root-mean-squared error (RMSE; comparable to the standard deviation of the differences between the predicted and true values) and the coefficient of determination ( $R^2$ , measures the linear correlation between the predicted and true values) between the data and the predictions, both for the full high-resolution output and the band-integrated spectra corresponding to the observations of HD 189733 b.

## 2.2. Retrieval

Following the setup of Harrington et al. (2022), we perform a retrieval of the dayside atmosphere of HD 189733 b based on the measurements by the Hubble Space Telescope Near Infrared Camera MultiObject Spectrograph (Swain et al. 2009); Spitzer Space Telescope Infrared Spectrograph (IRS; Grillmair et al. 2008); Spitzer InfraRed Array Camera (IRAC) channels 1 and 2 values of  $0.1533 \pm 0.0029\%$  and  $0.1886 \pm 0.0071\%$  (M. Line 2021, private communication); IRAC channel 3, IRS  $16\text{ }\mu\text{m}$  photometry, and Multiband Imaging Photometer for Spitzer (Charbonneau et al. 2008); and IRAC channel 4 (Agol et al. 2010). We use a K2 solar-abundance Kurucz stellar model for the host star’s emission (Castelli & Kurucz 2003). Using the differential evolution Markov chain with snooker updating algorithm of ter Braak & Vrugt (2008), 2,500,000 iterations are spread across 10 parallel chains, with a burn in of 50,000 iterations per chain. When retrieving, we fix the semimajor axis to 0.031 au and the planetary radius and gravity at  $0.1\text{ bar}$  to  $1.138 R_J$  and  $2187.762\text{ cm s}^{-2}$ , respectively. The remaining neural-network input parameters are allowed to freely vary over the entire training space.

We compute the Bhattacharyya coefficient (Bhattacharyya 1943; Aherne et al. 1998) to compare the similarity of the 1D marginalized posteriors, where a value of 0 indicates no overlap and a value of 1 indicates identical distributions. We choose this metric over others, such as the Kullback–Leibler divergence, because it is both intuitive to understand and defined for all distributions, even those that do not overlap.

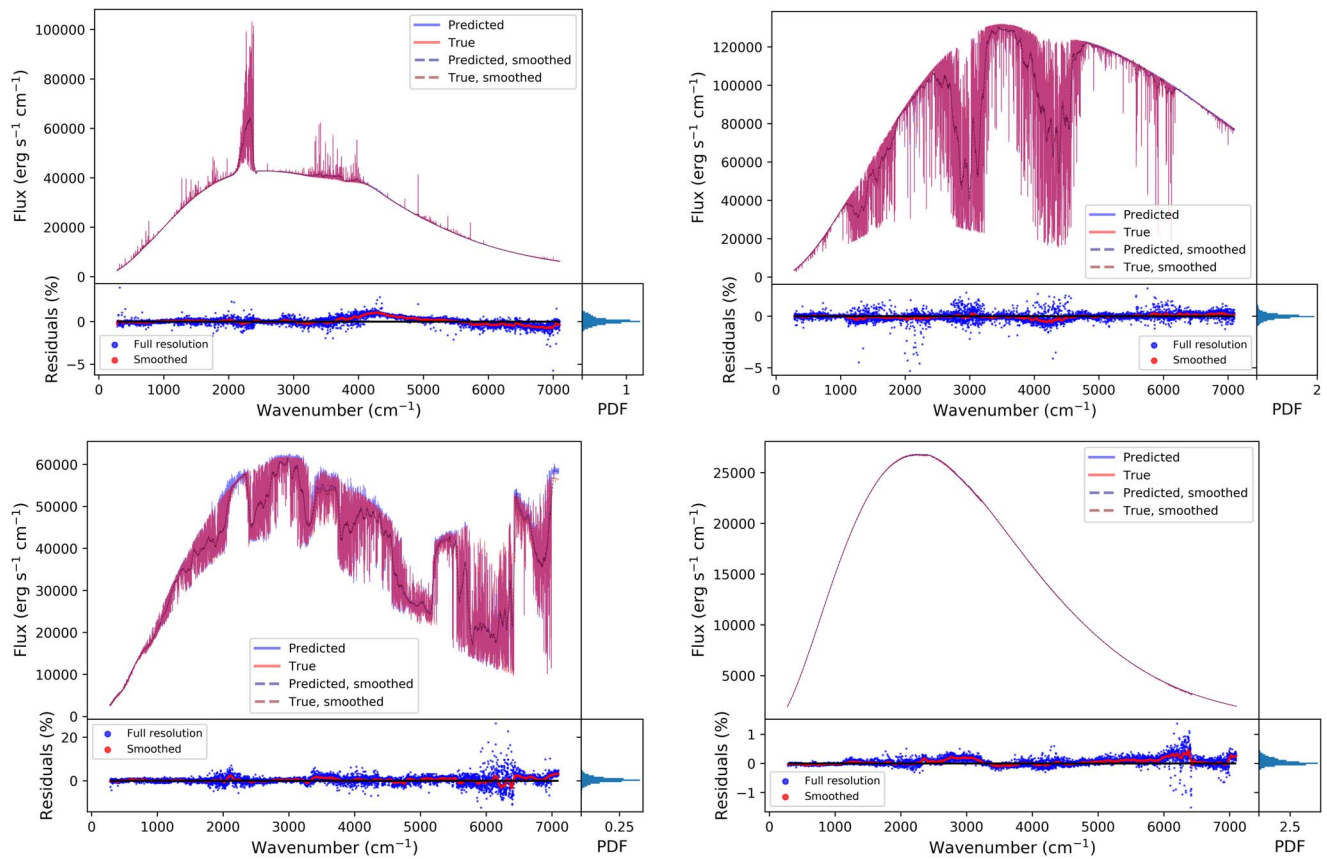
For this investigation, we focus on a neural network as a faster replacement for an RT code for retrieval; we therefore only compare the results of BART and the neural-network approximation. For a discussion of these results in the context of previous retrievals of HD 189733 b’s dayside atmosphere, see Harrington et al. (2022).

## 2.3. Software

We have developed two Python packages for this investigation. Both are open-source software, with full documentation, under the Reproducible Research Software License.<sup>8</sup> We

<sup>8</sup> <https://planets.ucf.edu/resources/reproducible-research/software-license/>





**Figure 2.** Four comparisons of planetary emission spectra predicted by MARGE and calculated by BART. The smoothed curves use a Savitzky–Golay filter with a third-order polynomial across a window of 101 elements ( $100\text{ cm}^{-1}$ ). The purple color arises due to a detailed match between the red and blue spectra at high resolution. For the residuals, a black line is plotted at 0 to show regions where the neural network consistently over- or underpredicts the spectrum. A histogram of the high-resolution residuals appears to the right of the residual scatter plot, where the x-axis shows the probability density function (PDF) for the range of residual percentages. Top left: case with  $T(p)$  profile that increases in temperature with altitude, with  $\text{H}_2\text{O}$  and  $\text{CO}_2$  emission lines. Top right: case with  $T(p)$  profile that decreases in temperature with altitude, with absorption primarily due to  $\text{CH}_4$  and  $\text{H}_2\text{O}$ . Bottom left: cases with  $T(p)$  profile that has an inversion around 0.1 bar, with  $\text{CH}_4$ ,  $\text{CO}$ , and  $\text{CO}_2$  absorption and emission features. Bottom right: case with  $T(p)$  profile that is nearly isothermal at the pressures with sensitivity.

encourage users to contribute to the code via pull requests on Github.

### 2.3.1. MARGE

The Machine learning Algorithm for Radiative transfer of Generated Exoplanets<sup>9</sup> (MARGE, Figure 1) (1) generates a data set based on a user-supplied function, (2) processes the generated data using a user-supplied function, and (3) trains, validates, and tests a user-specified neural-network architecture on a data set. The software package allows independent execution of any of the three modes, enabling a wide range of applications beyond exoplanet retrieval.

MARGE’s design allows it to be applied to any deterministic model. For 1D data (such as spectra), MARGE’s desired format is NumPy binary (.npy) files of 2D arrays, where each row corresponds to a single case. Each row is a data vector of the input parameters followed by the output data (e.g., spectrum). MARGE currently includes data-generation and -processing functions for BART as well as a data-processing function for the `pypsg`<sup>10</sup> Python interface (Soboczenski et al. 2018) for the NASA Planetary Spectrum Generator (Villanueva et al. 2018). We encourage users to contribute code via pull request to

handle the processing of the inputs/outputs of other software packages.

We implement neural-network model training in Keras (version 2.2.4, Chollet et al. 2015), using a Tensorflow (version 1.13.1, Abadi et al. 2016) backend. MARGE enables early stopping by default to prevent overfitting, and the user can halt or resume training. MARGE allows for cyclical learning rates for more efficient training (Smith 2015; see also Appendix A). Users specify the model architecture details and the data location, and the software handles the data normalization, training, validation, and testing. MARGE preprocesses the data into Tensorflow’s TFRecords format for efficient handling. Users have multiple options when preprocessing the data, which include taking the logarithm of the inputs and/or outputs, standardizing the data according to its mean and standard deviation, and/or scaling the data to be within a specified range. The mean and standard deviation of the data set are computed using Welford’s method (Welford 1962) to avoid the need to load the entire data set into memory at once. MARGE computes the RMSE and  $R^2$  for predictions on the validation and test sets to evaluate model performance; these metrics can optionally be calculated over integrated filter bandpasses. Finally, users may specify cases from the test set to plot the predicted and true spectra, with residuals (e.g., Figure 2).

<sup>9</sup> MARGE is available at <https://github.com/exosports/marge>.

<sup>10</sup> <https://gitlab.com/frontierdevelopmentlab/astrobiology/pypsg>

### 2.3.2. HOMER

The Helper Of My Eternal Retrievals<sup>11</sup> (HOMER) utilizes a MARGE-trained model to infer the underlying inputs corresponding to some observed outputs (Figure 1). For its Bayesian framework, HOMER uses a Python wrapper for Markov Chain and nested-sampling algorithms. The user specifies data, uncertainties, observational filters, a parameter space, and a few related inputs, which are passed to the Bayesian sampler to perform the inference. If available, a graphics processing unit (GPU) calculates neural-network predictions, though the central processing unit (CPU) can do this at the cost of increased runtime. For each iteration of the Bayesian inference, the trained neural network predicts on the proposed input parameters, which are modified as necessary (descale, denormalize, divide by the stellar spectrum, unit conversions, and/or integrated over bandpasses).

HOMER produces plots of the best-fit spectrum, 1D marginalized posteriors, 2D pairwise posteriors, and parameter history traces. The best-fit spectrum plot contains the data (with observational bandpasses indicated by uncertainties in  $x$ ) and, if the Datasets<sup>12</sup> library is installed, the  $1\sigma$ ,  $2\sigma$ , and  $3\sigma$  spectra. We use the streaming quantiles method of Karnin et al. (2016) as implemented in DataSketches to compute the  $1\sigma$ – $2\sigma$ – $3\sigma$  spectra. This approach avoids needing to load all of the evaluated models at once, which could exceed system memory.

HOMER calculates the steps per effective independent sample (SPEIS) and effective sample size (ESS) as described in Harrington et al. (2022). Markov chains make small, correlated steps; while a chain may perform 100,000 iterations, if it takes 5000 steps to materialize a completely independent sample (steps per effective independent sample, SPEIS), then there have only been 20 effective samples. SPEIS is calculated from the autocorrelation function of each parameter for each chain; as a conservative estimate, we use the highest SPEIS value when calculating the ESS of the Bayesian inference to ensure we do not underestimate credible region uncertainties. By rearranging Equation (1) of Harrington et al. (2022), an uncertainty  $s_{\hat{c}}$  can be calculated on a given credible region  $\hat{C}$  based on the ESS:

$$s_{\hat{c}} \approx \sqrt{\frac{\hat{C}(1 - \hat{C})}{\text{ESS}}} \quad (1)$$

For example, if the ESS is 20, then the determined 68.27% credible region is actually the  $68.27 \pm 10\%$  credible region; running the inference for more iterations would increase the ESS and accordingly decrease the uncertainty on that credible region.

For easy comparison with other retrieval results, HOMER can overplot the 1D and 2D posteriors for multiple retrievals (e.g., Figure 3) and compute the Bhattacharyya coefficients between the 1D posteriors.

## 3. Results and Discussion

The normalized RMSE, normalized  $R^2$ , and denormalized  $R^2$  metrics for the MARGE-trained model on the test set for the high-resolution and band-integrated spectra are detailed in Tables 2 and 3, respectively. The normalized RMSE  $\ll 1$  and  $R^2 \sim 1$  indicate an accurate model for RT over the parameter

space. Rather than waiting for early stopping to engage, we manually stopped training at 130 epochs because there was an insignificant improvement in the loss for dozens of epochs. For considerations on how this affects model performance, see Appendix B.

Figure 2 shows example comparisons between the spectra predicted by MARGE and true spectra calculated by BART. While residuals tend to be around a few percent, they generally fluctuate around 0; when band-integrated over the observational filters, these errors usually cancel, as shown by the lower normalized RMSE and higher denormalized  $R^2$  metrics (Tables 2 and 3). We observe that in some cases, there are regions where the spectrum is consistently over- or underestimated by a few percent (e.g., the top-left panel of Figure 2 around  $4250 \text{ cm}^{-1}$ ), thereby introducing error in the band-integrated value. However, the small deviations appear to have only a minor effect on this retrieval’s result; see Section 3.1 for considerations when retrieving at high spectral resolutions or in cases where a traditional retrieval result is not available for comparison.

When applying HOMER to the emission spectrum of HD 189733 b, the results are consistent with BART. The retrieved  $T(p)$  profiles (bottom-left panel Figure 3) agree in the regions probed by the observations ( $<1$  bar, bottom-right panel Figure 3) and only begin to deviate deeper in the atmosphere, where little to no signal is measured according to the contribution functions. By nature, HOMER cannot calculate contribution functions, as the MARGE model does not solve RT. While they could be included for each case in the training set, this would require significantly more compute resources. Computing the contribution functions for the single best-fit case using the RT code that trained MARGE more efficiently uses compute resources.

Table 4 compares HOMER’s retrieved 68.27% (“ $1\sigma$ ”), 95.45% (“ $2\sigma$ ”), and 99.73% (“ $3\sigma$ ”) credible regions with BART’s retrieved credible regions. All regions closely agree, with differences attributable to a combination of uncertainty from a finite ESS and the neural network’s imperfect nature (Figure 3, top-right panel). For CO, both BART and HOMER favor large abundances, though BART finds a greater probability for log abundances  $\geq -2$  (Figure 3, top-right panel). Despite this, the credible regions agree (Table 4). Similarly, HOMER favors lower values for  $\gamma_1$  and  $\alpha$ , though the resulting thermal profiles agree (Figure 3, bottom-left panel).

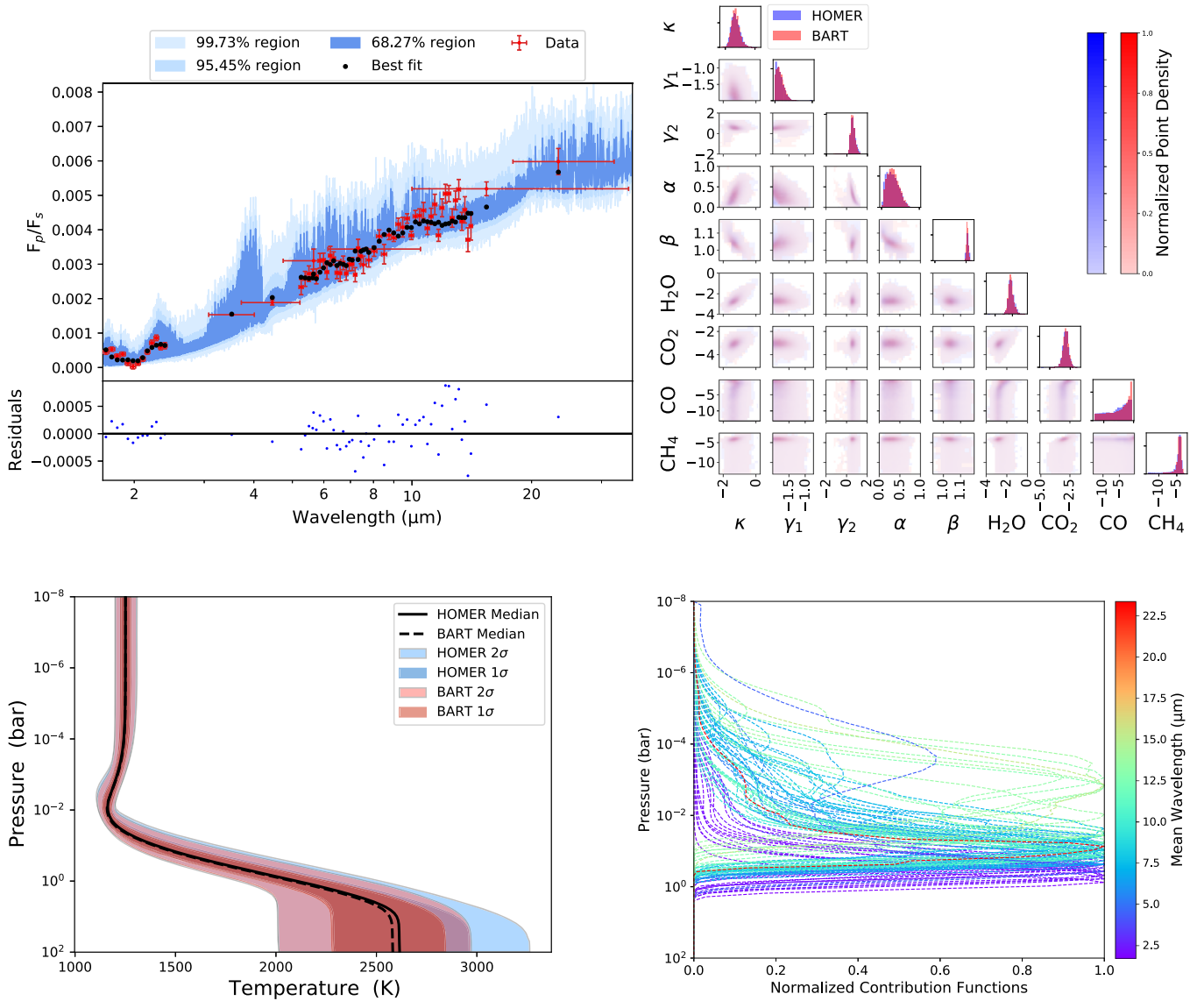
Table 5 compares the SPEIS, ESS values, and associated uncertainties in the  $1\sigma$ ,  $2\sigma$ , and  $3\sigma$  credible regions for HOMER and BART. HOMER yields an SPEIS that is less than BART’s, attributable to the conservative estimate of SPEIS as being the greatest among all chains and parameters. The highest SPEIS values fluctuate between runs, though the median SPEIS remains relatively constant. HOMER’s median SPEIS of 627 and BART’S 615 better reflect the close agreement between the two retrievals. The Bhattacharyya coefficients between the 1D marginalized posteriors of HOMER and BART indicate agreement in the range 0.9843–0.9972, with a mean of 0.9925 (Table 6).

### 3.1. Limitations

HOMER’s accuracy is, by nature, bound by the accuracy of the neural-network model. Model inaccuracies may slightly bias the results, as seen in the minor differences between the

<sup>11</sup> HOMER is available at <https://github.com/exosports/homer>.

<sup>12</sup> <https://datasketches.apache.org/>



**Figure 3.** Comparisons between HOMER and BART posteriors. Top left: best-fit spectrum of HOMER, with  $1\sigma$ ,  $2\sigma$ , and  $3\sigma$  regions. Top right: normalized probability density functions of the 2D marginalized pairwise posteriors retrieved for HD 189733 b, with the 1D marginalized posteriors along the diagonal. The purple color arises from the close match between HOMER and BART. Bottom left: posterior median,  $1\sigma$  from the median, and  $2\sigma$  from the median  $T(p)$  profile. In the regions with sensitivity, HOMER closely matches BART, with a slightly greater uncertainty. Bottom right: normalized contribution functions, which show the pressure range each filter probes, for the best-fit BART model.

**Table 2**  
Model Evaluation: High-resolution Spectra

Metric	Min.	Median	Mean	Max.
Norm. RMSE	0.00153	0.00224	0.00247	0.01040
Norm. $R^2$	0.99999	1.00000	1.00000	1.00000
Denorm. $R^2$	0.99885	0.99993	0.99990	0.99997

**Note.** RMSE and  $R^2$  are calculated for each of the 6821 outputs corresponding to the wavenumber grid of  $280\text{--}7100\text{ cm}^{-1}$  with a resolution of  $1.0\text{ cm}^{-1}$ . For conciseness, we present statistics about these values. The  $R^2$  values are slightly less than 1, but they round to 1 at the reported precision.

posteriors of HOMER and BART. In our application, this discrepancy does not significantly affect the scientific conclusions at the spectral resolution of these observations for the current neural-network accuracy. However, this does not necessarily hold for all cases. It is possible that at higher

**Table 3**  
Model Evaluation: Band-integrated Spectra

Metric	Min.	Median	Mean	Max.
Norm. RMSE	0.00123	0.00147	0.00148	0.00183
Norm. $R^2$	1.00000	1.00000	1.00000	1.00000
Denorm. $R^2$	0.99995	0.99997	0.99997	0.99998

**Note.** Same as Table 2, except integrated over the 66 bandpasses corresponding to the referenced observations of HD 189733 b.

resolutions this neural network’s minor inaccuracies can drive the Bayesian sampler to radically different results. While in theory MARGE works for any spectral resolution, users will need to carefully select the model architecture to ensure that it can accurately model the spectra over the desired phase space. In situations lacking a physics-based retrieval to compare with,

**Table 4**  
Retrieved Credible Regions

Parameter	Code	68.27%	95.45%	99.73%
log $\kappa$	HOMER	[−1.63, −1.06]	[−1.84, −0.71]	[−2.07, −0.33]
	BART	[−1.58, −1.09]	[−1.81, −0.79]	[−1.99, −0.46]
log $\gamma_1$	HOMER	[−1.98, −1.65]	[−1.99, −1.34]	[−1.99, −1.06]
	BART	[−1.98, −1.62]	[−2.00, −1.33]	[−2.00, −1.07]
log $\gamma_2$	HOMER	[0.34, 0.77]	[0.21, 1.10]	[0.11, 1.29]
	BART	[0.35, 0.73]	[0.21, 1.02]	[−0.07, 1.30]
$\alpha$	HOMER	[0.07, 0.39]	[0.03, 0.60]	[0.01, 0.74]
	BART	[0.11, 0.42]	[0.06, 0.60]	[0.02, 0.74]
$\beta$	HOMER	[1.01, 1.07]	[0.99, 1.12]	[0.96, 1.15]
	BART	[1.01, 1.06]	[0.99, 1.10]	[0.97, 1.15]
log H <sub>2</sub> O	HOMER	[−3.11, −2.37]	[−3.37, −1.82]	[−3.70, −1.27]
	BART	[−3.12, −2.44]	[−3.37, −1.92]	[−3.63, −1.41]
log CO <sub>2</sub>	HOMER	[−3.39, −2.73]	[−3.78, −2.36]	[−4.26, −2.01]
	BART	[−3.33, −2.71]	[−3.66, −2.32]	[−4.05, −2.03]
log CO	HOMER	[−6.89, −0.51]	[−12.02, −0.51]	[−12.90, −0.51]
	BART	[−6.60, −0.50]	[−12.55, −0.50]	[−12.90, −0.50]
log CH <sub>4</sub>	HOMER	[−5.16, −3.53]	[−10.25, −3.20]	[−12.95, −3.12]
	BART	[−4.71, −3.67]	[−10.53, −3.14]	[−12.73, −3.07]

**Table 5**  
Credible Region Accuracy

Code	SPEIS	ESS	1 $\sigma$ Uncertainty	2 $\sigma$ Uncertainty	3 $\sigma$ Uncertainty
HOMER	1668	1199	1.34%	0.60%	0.15%
BART	2084	959	1.50%	0.67%	0.17%

**Table 6**  
Bhattacharyya Coefficients

Parameter	Value
$\kappa$	0.9948
$\gamma_1$	0.9972
$\gamma_2$	0.9950
$\alpha$	0.9909
$\beta$	0.9879
H <sub>2</sub> O	0.9968
CO <sub>2</sub>	0.9968
CO	0.9888
CH <sub>4</sub>	0.9843
Mean	0.9925

we advise testing to ensure that forward models are reasonably accurate over the retrieval’s phase space, as some regions may not be sufficiently sampled for accurate predictions.

### 3.2. Compute Cost

The performance differences between HOMER and BART highlight HOMER’s computational benefits. For a single Markov Chain iteration, BART requires around 1.8 s per parallel chain on an AMD EPYC 7402P CPU, and multiple chains parallelize linearly across CPUs. By comparison, a single iteration with HOMER on the same CPU—which includes preprocessing (e.g., input normalization), prediction, postprocessing (e.g., output denormalization, scaling according to the stellar spectrum), and band integration—requires just  $\sim 0.2$  s for any number of chains fewer than 32. For a single chain, this is thus a  $\sim 9\times$  speedup. In our setup, we considered 10 parallel chains, translating to a  $\sim 90\times$  speedup for the function evaluated at each step of the Markov chain. Using an

NVIDIA Titan Xp for predictions, the model evaluations at each Markov chain step require 0.01–0.02 seconds, a  $10\times$ – $20\times$  speedup over predictions with the aforementioned CPU and, when using 10 parallel chains, a  $900\times$ – $1800\times$  speedup over the same function evaluation in BART. We note that if BART were capable of utilizing a GPU, this speedup factor would be much less. Further investigation is necessary to determine whether HOMER offers speed improvements over a GPU-accelerated RT code. Nevertheless, the CPU results emphasize the speed improvements of our approach; the significant reduction in compute time enables retrievals to be executed on an average laptop. The memory footprints for both approaches were comparable, though the parameters of each approach can strongly affect the required memory.

Here, the upfront compute cost to generate a data set and train a MARGE model is greater than the time to execute a single BART retrieval. In our example, we generated around  $1.5\times$ – $2\times$  the number of spectra typically computed during a BART retrieval with small credible region uncertainties, plus a few dozen hours to train the neural network. However, additional retrievals within the trained parameter space execute in around 30 minutes on our GPU (less when neglecting to compute spectral quantiles). Thus, when carrying out even two retrievals within some shared phase space, the compute cost of MARGE+HOMER is less than two classical retrievals. In certain circumstances, such as where the radius and mass of the planet do not need to be varied (e.g., retrievals on different data sets of the same exoplanet), the number of spectra required to approximate the phase space accurately would be less than in our example, which may lead to MARGE+HOMER requiring less compute time than a single BART retrieval. Beyond the scope of retrieval, this approach could also provide a benefit to situations where it is advantageous to trade one set of



computing resources for another. For example, spaceflight missions may be limited by thermal, power, and/or onboard computational resources; it may be advantageous to increase the total compute time, if it can decrease the power, thermal, and/or onboard computing required for the calculation.

Another benefit of our approach is that the compute-cost scaling is less than linear: increasing from 10 to 256 chains results in just a  $\sim 12.5\times$  increase in compute cost per iteration when using a GPU, compared to  $25.6\times$  as much for BART. Additional chains enable faster exploration of the parameter space, and, if executed for the same number of iterations per chain, increases the ESS, which reduces the uncertainty in the bounds of credible regions (Harrington et al. 2022). Thus, the combination of MARGE and HOMER saves valuable compute resources when performing retrievals and reduces total runtime when performing multiple retrievals.

#### 4. Conclusions

This paper presents a novel technique for ML retrieval that uses a neural-network model of RT within a Bayesian framework to reduce the runtime of a retrieval. Our open-source codes, MARGE and HOMER, provide the community with an easy-to-use implementation of this approach, and they are readily applicable to any forward model and its inversion—not strictly BART or even RT. They are available on Github with full user documentation.

Our method enables fast retrievals that are consistent with algorithms that solve the RT equation. The approach circumvents limitations of current ML retrieval models by using an RT surrogate in place of the RT code found in classical retrieval algorithms, thereby preserving the accuracy of the Bayesian inference. Like BART, MARGE and HOMER work at both the low resolutions of Spitzer and the high spectral resolutions of advanced ground-based spectrographs.

On our hardware, HOMER reduces the runtime of each MCMC iteration by  $\sim 9\times$  per parallel chain using a CPU and  $90\times$ – $180\times$  per chain using a GPU, compared to BART. For the case of HD 189733 b, the Bhattacharyya coefficients of the 1D marginalized posteriors of BART and HOMER are  $>0.984$ , indicating a close match. This reduction in compute time enables using more realistic (and computationally expensive) RT models, such as those including scattering and condensates. Additionally, 3D retrievals with  $\sim 200$  cells could be completed in a matter of days.

Our approach is particularly well suited to planning studies for future observations, telescopes, and instruments, like the James Webb Space Telescope and the Large UltraViolet Optical InfraRed Surveyor (e.g., Rocchetto et al. 2016; Feng et al. 2018). Using a single MARGE model trained over the desired parameter space, HOMER can perform dozens to hundreds of retrievals in the time it takes to run a single retrieval with an RT solver.

More generally, our technique and tools can be applied to problems beyond the scope of this investigation. MARGE provides a generalized method to train a neural network to model any deterministic process, while HOMER uses a MARGE-trained model to infer the inverse process. MARGE models could be trained for cloud/haze formation or photochemistry within general circulation models, for example. MARGE and HOMER could also be used to map gravitationally lensed galaxies (e.g., Perreault Levasseur et al. 2017).

With the plethora of ML retrieval algorithms that have emerged in recent years, standard data sets should be created and used for benchmarking. Ideally, such a data set would cover a wide range of wavelengths at high resolution and include all available opacity sources, scattering, clouds/hazes, and, in the case of terrestrial planets, surface properties. This would allow easy comparisons among current and future ML retrieval codes.

The Reproducible Research Compendium for this work is available for download.<sup>13</sup> It includes all of the code, configuration files, data, and plots used in support of this work.

We gratefully acknowledge Jon Malkin, Lee Rhodes, and Edo Liberty for valuable contributions to the streaming quantiles method used in this work. We thank James Mang and Nicholas Susemihl for useful feedback on the software developed for this work. We thank Michael Lund and the NASA Exoplanet Archive for preparing and hosting the online RRC. We thank Jennifer Adams for helpful discussions on radiative transfer emulation in Earth science. We also thank contributors to the Datasets library, NumPy, SciPy, Matplotlib, Tensorflow, Keras, the Python Programming Language, the free and open-source community, and the NASA Astrophysics Data System for software and services.

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research. This research was supported by the NASA Fellowship Activity under NASA Grant 80NSSC20K0682 and NASA Exoplanets Research Program grant NNX17AB62G. We thank FDL (<http://www.frontierdevelopmentlab.org/>) and SETI (<https://www.seti.org>) for making this collaboration possible.

#### Appendix A Determining Model Architecture

To select an ideal neural-network architecture, a grid search must be carried out. This includes varying the types of hidden layers, number of hidden layers, number of nodes per layer, activation functions for each hidden layer, parameter(s) for each activation function, and learning rate.

We carried out a grid search by training each model on a subset of the total data set (171,456 training, 66,432 validation) for 20 epochs using a batch size of 64. We considered three to five dense and convolutional+pooling hidden layers, 64–4096 nodes, rectified linear unit (ReLU), leaky ReLU, exponential linear unit (ELU), hyperbolic tangent (tanh), and sigmoid activation functions. The convolutional layers use a kernel size of 3, and pooling layers use a size of 2. We consider four learning rate policies: (1) a cyclical rate ranging from  $8 \times 10^{-6}$ – $5 \times 10^{-3}$  where the maximum is reduced by half of the difference with the minimum every eight epochs, (2) as before but ranging from  $10^{-5}$ – $10^{-3}$ , (3) a constant learning rate of  $10^{-5}$ , and (4) a constant  $10^{-3}$ . Policies 3 and 4 are only considered for models that do not include tanh or sigmoid activations.

Table A1 presents the minimum validation loss for each architecture considered. There is some randomness to the minimum validation loss due to the shuffling of the training data, so models with comparable minimum validation losses can be considered equivalent in performance. We chose to

<sup>13</sup> Available at <https://exoplanetarchive.ipac.caltech.edu/docs/marge-homer.html>.



**Table A1**  
Model Grid Search, 20 Epochs

#	Hidden Layers	Min. Val. Loss ( $\times 10^5$ )			
		LR1 <sup>a</sup>	LR2 <sup>b</sup>	LR3 <sup>c</sup>	LR4 <sup>d</sup>
1	D(512) <sup>e</sup> R <sup>f</sup> -D(512)R-D(512)R	17.4	43.8	491	19.2
2	D(1024)R-D(1024)R-D(1024)R	9.61	24.6	291	12.3
3	D(2048)R-D(2048)R-D(2048)R	7.60	13.3	179	8.28
4	D(4096)R-D(4096)R-D(4096)R	8.56	7.17	106	6.54
5	D(512)R-D(512)R-D(512)R-D(512)R	13.0	31.3	382	16.2
6	D(512)S <sup>g</sup> -D(512)S-D(512)S-D(512)S	68.0	951	...	...
7	D(512)T <sup>h</sup> -D(512)T-D(512)T-D(512)T	487	47.0	...	...
8	D(512)S-D(1024)S-D(2048)S-D(4096)S	48.5	932	...	...
9	D(512)T-D(1024)T-D(2048)T-D(4096)T	1390	68.8	...	...
10	D(1024)R-D(1024)R-D(1024)R-D(1024)R	8.43	16.8	238	10.1
11	D(2048)R-D(2048)R-D(2048)R-D(2048)R	7.46	9.13	125	8.01
12	D(4096)R-D(4096)R-D(4096)R-D(4096)R	8.23	5.05	62.5	6.14
13	D(4096)S-D(4096)S-D(4096)S-D(4096)S	1350	197	...	...
14	D(4096)T-D(4096)T-D(4096)T-D(4096)T	1740	46.0	...	...
15	D(4096)E(0.05) <sup>i</sup> -D(4096)E(0.05)-D(4096)E(0.05)-D(4096)E(0.05)	220	5.10	64.6	5.55
16	D(4096)E(0.1)-D(4096)E(0.1)-D(4096)E(0.1)-D(4096)E(0.1)	227	5.03	70.3	5.93
17	D(4096)E(0.15)-D(4096)E(0.15)-D(4096)E(0.15)-D(4096)E(0.15)	206	5.13	74.5	7.13
18	D(4096)E(0.2)-D(4096)E(0.2)-D(4096)E(0.2)-D(4096)E(0.2)	238	5.39	81.7	6.18
19	D(4096)L(0.05) <sup>j</sup> -D(4096)L(0.05)-D(4096)L(0.05)-D(4096)L(0.05)	7.32	5.21	67.2	7.33
20	D(4096)L(0.05)-D(4096)L(0.05)-D(4096)L(0.05)-D(4096)L(0.05)	232	4.96	65.5	6.82
21	D(4096)L(0.1)-D(4096)L(0.1)-D(4096)L(0.1)-D(4096)L(0.1)	270	5.09	72.2	6.13
22	C(64) <sup>k</sup> E(0.05)-M <sup>l</sup> (2)-D(4096)E(0.05)-D(4096)E(0.05)	5.89	9.69	123	6.34
23	C(64)E(0.05)-M(2)-D(4096)E(0.05)-D(4096)E(0.05)-D(4096)E(0.05)	4.71	5.67	70.6	4.43
24	C(64)E(0.05)-M(2)-D(4096)E(0.05)-D(4096)E(0.05)-D(4096)E(0.05)-D(4096)E(0.05)	4.76	4.58	51.7	5.79
25	C(64)L(0.05)-M(2)-D(4096)L(0.05)-D(4096)L(0.05)	6.06	9.61	118	6.41
26	C(64)L(0.05)-M(2)-D(4096)L(0.05)-D(4096)L(0.05)-D(4096)L(0.05)	4.55	5.56	71.5	4.71
27	C(64)L(0.05)-M(2)-D(4096)L(0.05)-D(4096)L(0.05)-D(4096)L(0.05)-D(4096)L(0.05)	149	4.57	49.3	4.24
28	C(128)E(0.05)-M(2)-D(4096)E(0.05)-D(4096)E(0.05)	5.94	8.49	105	5.04
29	C(128)E(0.05)-M(2)-D(4096)E(0.05)-D(4096)E(0.05)-D(4096)E(0.05)	4.77	5.19	63.5	4.40
30	C(128)E(0.05)-M(2)-D(4096)E(0.05)-D(4096)E(0.05)-D(4096)E(0.05)-D(4096)E(0.05)	5.46	4.29	45.6	4.88
31	C(128)L(0.05)-M(2)-D(4096)L(0.05)-D(4096)L(0.05)	5.99	8.62	105	5.18
32	C(128)L(0.05)-M(2)-D(4096)L(0.05)-D(4096)L(0.05)-D(4096)R	5.34	5.18	61.1	5.86
33	C(128)L(0.05)-M(2)-D(4096)L(0.05)-D(4096)L(0.05)-D(4096)L(0.05)	4.48	5.18	61.1	4.61
34	C(128)L(0.05)-M(2)-D(4096)L(0.05)-D(4096)L(0.05)-D(4096)L(0.05)-D(4096)L(0.05)	4.23	4.28	44.9	4.86
35	C(256)E(0.05)-M(2)-D(4096)E(0.05)-D(4096)E(0.05)	6.46	8.18	93.0	5.05
36	C(256)E(0.05)-M(2)-D(4096)E(0.05)-D(4096)E(0.05)-D(4096)E(0.05)	5.40	4.98	56.5	5.12
37	C(256)E(0.05)-M(2)-D(4096)E(0.05)-D(4096)E(0.05)-D(4096)E(0.05)-D(4096)E(0.05)	5.98	4.08	41.4	5.28
38	C(256)L(0.05)-M(2)-D(4096)L(0.05)-D(4096)L(0.05)	6.14	7.97	93.6	5.29
39	C(256)L(0.05)-M(2)-D(4096)L(0.05)-D(4096)L(0.05)-D(4096)L(0.05)	4.68	5.00	56.0	5.60
40	C(256)L(0.05)-M(2)-D(4096)L(0.05)-D(4096)L(0.05)-D(4096)L(0.05)-D(4096)L(0.05)	4.32	4.10	41.7	4.94
41	C(256)S-M(2)-D(4096)S-D(4096)S-D(4096)S-D(4096)S	11700	1400	...	...
42	C(256)T-M(2)-D(4096)T-D(4096)T-D(4096)T-D(4096)T	11800	49.5	...	...

**Notes.** Models trained for 20 epochs in batches of 64.

<sup>a</sup> Triangular2 learning rate policy ranging from  $8 \times 10^{-6}$ – $5 \times 10^{-3}$  with a complete cycle spanning eight epochs.

<sup>b</sup> Like LR1, but ranging from  $10^{-5}$ – $10^{-3}$ .

<sup>c</sup> Constant learning rate of  $10^{-5}$ .

<sup>d</sup> Constant learning rate of  $10^{-3}$ .

<sup>e</sup> Dense layer with  $n$  nodes, D( $n$ ).

<sup>f</sup> ReLU activation.

<sup>g</sup> Sigmoid activation.

<sup>h</sup> tanh activation.

<sup>i</sup> ELU activation  $E(\alpha)$ , with scaling parameter  $\alpha$ .

<sup>j</sup> Leaky ReLU activation  $L(m)$ , with a slope of  $m$  for  $x < 0$ .

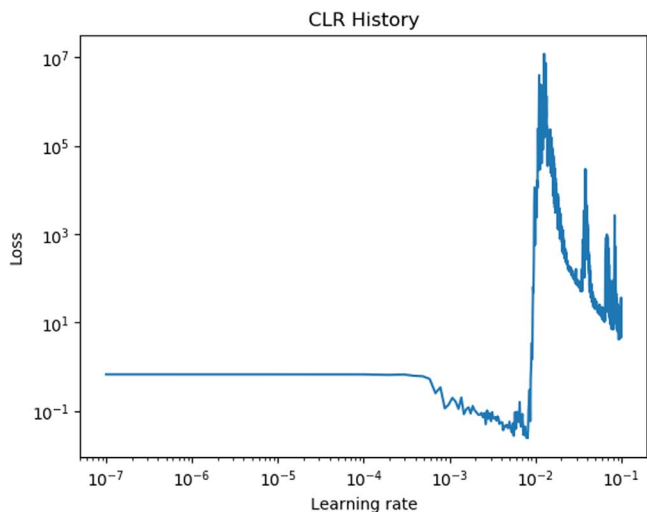
<sup>k</sup> Convolution1D layer with a kernel size of 3 and  $n$  nodes, C( $n$ ).

<sup>l</sup> MaxPooling1D layer M( $s$ ), with a pooling size  $s$ .

perform a more exhaustive grid search than is typical to emphasize certain points that can guide future investigations.

In general, we find that models with 4+ hidden layers with ReLU, leaky ReLU, and ELU activations achieve the lowest validation loss for this problem. The best-performing models

all have a 1D convolutional layer as the first hidden layer, while the worst-performing models use tanh or sigmoid activations. Additional layers generally lead to reductions in the loss. Cases where this does not occur can be attributed to the learning rate policy (e.g., models 25–27, LR1 versus LR2), highlighting the



**Figure A1.** Example of a range test. The learning rate begins at a value too small to make noticeable changes to the weights of the model. At a learning rate of  $\sim 4 \times 10^{-4}$ , the loss begins to decrease, indicating that the model has begun learning. However, at a learning rate of  $\sim 5 \times 10^{-3}$ , the loss begins to behave erratically, and it becomes very large at a learning rate of  $10^{-2}$ . From this, a learning rate policy varying between  $6 \times 10^{-4}$  and  $4 \times 10^{-3}$  would likely perform well for this architecture.

importance of properly selecting the policy (described below). Minor variations (e.g., models 28–30 LR1) can be attributed to training randomness. ReLU and leaky ReLU activations tend to have similar performance; leaky ReLU with a small parameter tends to perform equivalently or better than ReLU (e.g., models 32 and 33). While these results point to deep architectures as optimal configurations for this application of ML RT, tests varying the spectral resolution, wavelength range, etc. are necessary to definitively confirm if such variations change the optimal architecture(s). A future investigation should consider this in more detail.

Based on this grid search, we selected model 40. While a similar architecture with ELU activations performed equivalently (model 37), it took longer to train per epoch. Additionally, we found that the retrieval accuracy did not significantly change below some threshold validation loss (see Appendix B), so training time is a more important consideration than minor differences in minimum validation loss.

Our results show that, when the learning rate range is properly chosen, cyclical learning rates outperform constant learning rates, confirming the findings of Smith (2015) for this particular problem. Select models do not follow this trend (e.g., models 31, 35, 38), which is likely attributable to the small number of epochs considered in this grid search.

In Section 2, we state that we make the learning rate policy selection as described in Smith (2015), except based on the loss instead of the accuracy. Our selection process is to perform a “range test” by training the model over a few epochs using a learning rate policy that constantly increases from a very small rate (e.g.,  $10^{-7}$ ) to a large rate (e.g.,  $10^{-1}$ ). Looking at a plot of loss versus learning rate (e.g., Figure A1), the learning rate range can be deduced based on when the loss begins decreasing (minimum learning rate) and when the loss begins increasing (maximum learning rate). In practice, we find more efficient training using a range that is slightly interior to the extrema

determined via the plot. This is analogous to the method described in Smith (2015), except that it is more straightforward to determine the learning rate boundaries.

## Appendix B Data Set Size Considerations

To briefly investigate the effect of data set size for our problem, we consider three models in addition to that presented in Section 2 (“Main”). The additional models are trained on around 25% of the total data set (614,208 training, 171,584 validation, 78,848 testing). Two models were trained according to the same learning rate policy as described in Section 2, for 187 and 500 epochs (“Sub1a” and “Sub1b”, respectively). The third model was trained according to the LR2 learning rate policy described in Appendix A for 500 epochs (“Sub2”). All other setup parameters (e.g., data normalization) match those described in Section 2.












Table B1 compares the normalized RMSE and denormalized  $R^2$  test-set metrics over the high-resolution spectra, as well as the Bhattacharyya coefficients for the retrieved 1D marginalized posteriors. Based on the differences between models Sub1a and Sub1b (which only differ in the number of epochs trained), it can be concluded that manually stopping training once the loss begins to negligibly change does not have a major effect on the model performance. The differences in performance between models Sub1b and Sub2 (which only differ in learning rate policies) illustrate the importance of selecting the learning rate policy. However, both of these effects are negligible compared to those of the data set size: the differences among models Sub1a, Sub1b, and Sub2 are smaller than the differences between model Main and Sub2 (the best-performing Sub model). While Main and Sub2 underwent similar numbers of total training steps, Main outperforms Sub2. These results motivate the generation of large, comprehensive data sets of spectra to train surrogate RT models, though further research into how data set size, number of inputs/outputs, and architecture complexity influence model performance is needed to inform the optimal data set sizes for future investigations.

**Table B1**  
Model Comparison

Metric	Model	Min.	Median	Mean	Max.
Normalized RMSE	Main	0.00153	0.00224	0.00247	0.01040
	Sub1a	0.00271	0.00373	0.00407	0.01846
	Sub1b	0.00264	0.00365	0.00398	0.01679
	Sub2	0.00224	0.00331	0.00366	0.01721
Denormalized $R^2$	Main	0.99885	0.99993	0.99990	0.99997
	Sub1a	0.99646	0.99980	0.99974	0.99991
	Sub1b	0.99696	0.99981	0.99975	0.99992
	Sub2	0.99740	0.99983	0.99977	0.99994
Bhattacharyya coeff.	Main	0.9843	0.9948	0.9925	0.9972
	Sub1a	0.9585	0.9858	0.9853	0.9991
	Sub1b	0.8919	0.9933	0.9655	0.9976
	Sub2	0.9783	0.9940	0.9918	0.9984

**Note.** See text for model descriptions.

## ORCID iDs

Michael D. Himes  <https://orcid.org/0000-0002-9338-8600>  
 Joseph Harrington  <https://orcid.org/0000-0002-8955-8531>  
 Adam D. Cobb  <https://orcid.org/0000-0003-2868-6983>  
 Atılım Güneş Baydin  <https://orcid.org/0000-0001-9854-8100>  
 Frank Soboczenski  <https://orcid.org/0000-0001-8185-6094>  
 Molly D. O’Beirne  <https://orcid.org/0000-0001-9011-4420>  
 Simone Zorzan  <https://orcid.org/0000-0003-0550-3224>  
 David C. Wright  <https://orcid.org/0000-0003-1562-4679>  
 Zacchaeus Scheffer  <https://orcid.org/0000-0002-1177-113X>  
 Shawn D. Domagal-Goldman  <https://orcid.org/0000-0003-0354-9325>  
 Giada N. Arney  <https://orcid.org/0000-0001-6285-267X>

## References

- Abadi, M., Barham, P., Chen, J., et al. 2016, *OSDI*, 16, 265
- Abel, M., Frommhold, L., Li, X., & Hunt, K. L. C. 2012, *JChPh*, 136, 044319
- Agol, E., Cowan, N. B., Knutson, H. A., et al. 2010, *ApJ*, 721, 1861
- Aherne, F. J., Thacker, N. A., & Rockett, P. I. 1998, *Kybernetika*, 34, 363, <http://hdl.handle.net/10338.dmlcz/135216>
- Amodei, D., Ananthanarayanan, S., Anubhai, R., et al. 2016, *Proc. of Machine Learning Research*, 48, 173, <https://proceedings.mlr.press/v48/amodei16.html>
- Arcangeli, J., Désert, J.-M., Line, M. R., et al. 2018, *ApJL*, 855, L30
- Atzberger, C. 2004, *RSEnv*, 93, 53
- Bahdanau, D., Cho, K., & Bengio, Y. 2015, arXiv:1409.0473
- Barber, R. J., Tennyson, J., Harris, G. J., & Tolchenov, R. N. 2006, *MNRAS*, 368, 1087
- Baydin, A. G., Heinrich, L., Bhimji, W., et al. 2019, in *NeurIPS Proc. 32, Advances in Neural Information Processing Systems*, ed. H. Wallach et al. (Red Hook, NY: Curran Associates, Inc.), <https://papers.nips.cc/paper/2019/hash/6d19c113404cee55b4036fce1a37c058-Abstract.html>
- Bhattacharyya, A. 1943, *Bull. Calcutta Math. Soc.*, 35, 99
- Blecic, J., Harrington, J., Cubillos, P., et al. 2022, *PSJ*, 3, 82
- Borysow, A. 2002, *A&A*, 390, 779
- Borysow, A., Jorgensen, U. G., & Fu, Y. 2001, *JQSRT*, 68, 235
- Boudon, V., Pirali, O., Roy, P., et al. 2010, *JQSRT*, 111, 1117
- Brehmer, J., Cranmer, K., Louppe, G., & Pavez, J. 2018, *PhRvD*, 98, 052004
- Brown, L., Sung, K., Benner, D., et al. 2013, *JQSRT*, 130, 201
- Bue, B. D., Thompson, D. R., Deshpande, S., et al. 2019, *AMT*, 12, 2567
- Campargue, A., Leshchishina, O., Wang, L., Mondelain, D., & Kassi, S. 2013, *JMoSp*, 291, 16
- Castelli, F., & Kurucz, R. L. 2003, in *IAU Symp. 210, Modelling of Stellar Atmospheres, Poster Contributions*, ed. N. Piskunov, W. W. Weiss, & D. F. Gray (San Francisco, CA: ASP), A20
- Chan, W., Jaitly, N., Le, Q. V., & Vinyals, O. 2016, in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP) (Piscataway, NJ: IEEE)*, 4960, doi:10.1109/ICASSP.2016.7472621
- Charbonneau, D., Knutson, H. A., Barman, T., et al. 2008, *ApJ*, 686, 1341
- Chernetskii, M., Gobron, N., Gómez-Dans, J., et al. 2018, *AdSpR*, 62, 1654
- Cho, K., van Merriënboer, B., Gulcehre, C., et al. 2014, in *Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (Stroudsburg PA: Association for Computational Linguistics)*, 1724, doi:10.3115/v1/D14-1179
- Chollet, F., et al. 2015, *Keras*, GitHub, <https://github.com/keras-team/keras>
- Chorowski, J., Bahdanau, D., Cho, K., & Bengio, Y. 2014, *NIPS 2014 Workshop on Deep Learning*, December 2014
- Cobb, A. D., Himes, M. D., Soboczenski, F., et al. 2019, *AJ*, 158, 33
- Cranmer, K., Brehmer, J., & Louppe, G. 2019, arXiv:1911.01429
- Cubillos, P., Harrington, J., Blecic, J., et al. 2022, *PSJ*, 3, 81
- Daumont, L., Nikitin, A., Thomas, X., et al. 2013, *JQSRT*, 116, 101
- Deming, L. D., & Seager, S. 2017, *JGRE*, 122, 53
- Feng, Y. K., Robinson, T. D., Fortney, J. J., et al. 2018, *AJ*, 155, 200
- García-Cuesta, E., de la Torre, F., & de Castro, A. J. 2009, in *Machine Learning Approaches for the Inversion of the Radiative Transfer Equation*, ed. S.-I. Ao, B. Rieger, & S.-S. Chen (Dordrecht: Springer), 319
- Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. 2017, in *Proc. of the 34th International Conference on Machine Learning*, Vol. 70, 1263, doi:10.5555/3305381.3305512
- Gómez-Dans, J., Lewis, P., & Disney, M. 2016, *RemS*, 8, 119
- Goodfellow, I., Bengio, Y., & Courville, A. 2016, *Deep Learning* (Cambridge, MA: MIT Press)
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., et al. 2014, *Advances in Neural Information Processing Systems*, 2672
- Goorvitch, D. 1994, *ApJS*, 95, 535
- Grillmair, C. J., Burrows, A., Charbonneau, D., et al. 2008, *Natur*, 456, 767
- Hargreaves, R. J., Gordon, I. E., Rey, M., et al. 2020, *ApJS*, 247, 55
- Harrington, J., Himes, M., Cubillos, P., et al. 2022, *PSJ*, 3, 80
- Hayes, J. J. C., Kerins, E., Awiphan, S., et al. 2020, *MNRAS*, 494, 4492
- He, K., Zhang, X., Ren, S., & Sun, J. 2016, in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (Piscataway, NJ: IEEE)*, 770, doi:10.1109/CVPR.2016.90
- Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. 2017, in *2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (Piscataway, NJ: IEEE)*, 2261, doi:10.1109/CVPR.2017.243
- Johnsen, T. K., Marley, M. S., & Gulick, V. C. 2020, *PASP*, 132, 044502
- Karnin, Z., Lang, K., & Liberty, E. 2016, in *2016 IEEE 57th Annual Symp. on Foundations of Computer Science (FOCS) (Piscataway, NJ: IEEE)*, 71, doi:10.1109/FOCS.2016.17
- Kasim, M., Watson-Parris, D., Deaconu, L., et al. 2022, *MLS&T*, 3, 015013
- Koskinen, T. T., Moses, J. I., West, R. A., Guerlet, S., & Jouchoux, A. 2016, *GeoRL*, 43, 7895
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. 2012, *Proc. of the 25th Int. Conf. on Neural Information Processing Systems*, 1097, doi:10.1145/3065386
- Line, M. R., Wolf, A. S., Zhang, X., et al. 2013, *ApJ*, 775, 137
- Madhusudhan, N. 2018, *Handbook of Exoplanets* (Cham: Springer), 104
- Márquez-Neila, P., Fisher, C., Sznitman, R., & Heng, K. 2018, *NatAs*, 2, 719
- Munk, A., Scibior, A., Baydin, A. G., et al. 2019, arXiv:1910.11950
- Niederer, H.-M., Albert, S., Bauerecker, S., et al. 2008, *CHIMIA International Journal for Chemistry*, 62, 273
- Niederer, H.-M., Wang, X.-G., Carrington, T., et al. 2013, *JMoSp*, 291, 33
- Nikitin, A., Brown, L., Sung, K., et al. 2013, *JQSRT*, 114, 1
- Nikitin, A., Daumont, L., Thomas, X., et al. 2011, *JMoSp*, 268, 93
- Nikitin, A., Lyulin, O., Mikhailenko, S., et al. 2010, *JQSRT*, 111, 2211
- Perreault Lévassieur, L., Hezaveh, Y. D., & Wechsler, R. H. 2017, *ApJL*, 850, L7
- Ranzato, M., Chopra, S., Auli, M., & Zaremba, W. 2016, arXiv:1511.06732
- Richard, C., Gordon, I., Rothman, L., et al. 2012, *JQSRT*, 113, 1276
- Rivera, J., Verrelst, J., Gómez-Dans, J., et al. 2015, *RemS*, 7, 9347
- Rocchetto, M., Waldmann, I. P., Venot, O., Lagage, P. O., & Tinetti, G. 2016, *ApJ*, 833, 120
- Rothman, L. S., Gordon, I. E., Babikov, Y., et al. 2013, *JQSRT*, 130, 4
- Rothman, L. S., Gordon, I. E., Barber, R. J., et al. 2010, *JQSRT*, 111, 2139
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. 1986, *Natur*, 323, 533
- Seager, S., & Deming, D. 2010, *ARA&A*, 48, 631
- Sennrich, R., Haddow, B., & Birch, A. 2016, in *Proc. of the 54th Annual Meeting of the Association for Computational Linguistics*, Vol. 1: Long Papers (Berlin: Association for Computational Linguistics), 1715, doi:10.18653/v1/P16-1162
- Simonyan, K., & Zisserman, A. 2015, arXiv:1409.1556
- Smith, L. N. 2015, arXiv:1506.01186
- Soboczenski, F., Himes, M. D., O’Beirne, M. D., et al. 2018, arXiv:1811.03390
- Swain, M. R., Vasisht, G., Tinetti, G., et al. 2009, *ApJL*, 690, L114
- Szegedy, C., Wei, Liu, Yangqing, Jia, et al. 2015, in *2015 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (Piscataway, NJ: IEEE)*, 1, doi:10.1109/CVPR.2015.7298594
- Tashkun, S., Perevalov, V., Teffo, J.-L., Bykov, A., & Lavrentieva, N. 2003, *JQSRT*, 82, 165
- ter Braak, C. J. F., & Vrugt, J. A. 2008, *Statistics and Computing*, 18, 435
- Verrelst, J., Dethier, S., Rivera, J. P., et al. 2016, *IGRSL*, 13, 1012
- Verrelst, J., Rivera Caicedo, J., Muñoz-Marí, J., Camps-Valls, G., & Moreno, J. 2017, *RemS*, 9, 927
- Verrelst, J., Rivera, J. P., Gómez-Dans, J., Camps-Valls, G., & Moreno, J. 2015, *2015 IEEE Int. Geoscience and Remote Sensing Symp. (IGARSS) (Piscataway, NJ: IEEE)*, 633, doi:10.1109/IGARSS.2015.7325843
- Vicent, J., Verrelst, J., Rivera-Caicedo, J. P., et al. 2018, *IJSTA*, 11, 4918
- Villanueva, G. L., Smith, M. D., Protopapa, S., Faggi, S., & Mandell, A. M. 2018, *JQRST*, 217, 86
- Waldmann, I. P., & Griffith, C. A. 2019, *NatAs*, 3, 620
- Welford, B. P. 1962, *Technometrics*, 4, 419

Wu, Y., Schuster, M., Chen, Z., et al. 2016, arXiv:[1609.08144](https://arxiv.org/abs/1609.08144)

Xiong, W., Droppo, J., Huang, X., et al. 2016, IEEE/ACM Transactions on Audio, Speech, and Language Processing (Piscataway, NJ: IEEE), 2410, doi:[10.1109/TASLP.2017.2756440](https://doi.org/10.1109/TASLP.2017.2756440)

Yin, F., Gomez-Dans, J., & Lewis, P. 2018, 2018 IEEE Int. Geoscience and Remote Sensing Symp. (IGARSS) (Piscataway, NJ: IEEE), 1804, doi:[10.1109/IGARSS.2018.8517466](https://doi.org/10.1109/IGARSS.2018.8517466)

Zingales, T., & Waldmann, I. P. 2018, *AJ*, **156**, 268