Draft version May 30, 2024
Typeset using LaTeX **twocolumn** style in AASTeX631

Using Hybrid Machine Learning for Fundamental Stellar Parameter Inference

Sujay Shankar,[1] Michael Gully-Santiago,[1] and Caroline V. Morley[1]

[1] *Department of Astronomy, The University of Texas at Austin, Austin, TX 78712, USA*

ABSTRACT

The advent of machine learning (ML) methods has and continues to revolutionize numerous scientific disciplines, with a growing number of examples in astronomical spectroscopic inference. ML methods in spectroscopy are expected to perform with greater efficacy than conventional computational techniques. In this study we introduce a hybrid ML (HML) method that combines automatic differentiation (autodiff), interpolation, and Bayesian optimization to infer stellar parameters given stellar spectra. The stellar parameters we study are $T_{\rm eff}$, $\log(g)$, and [Fe/H], but this method has potential for extension to other properties such as [α/Fe], C/O, and f_{sed}. We first use the semi-empirical approach to treating spectra as sets of tunable spectral lines introduced in `blase` (Gully-Santiago & Morley 2022), contrasting with traditional data-driven approaches. `blase` is used on 1,314 spectra from the PHOENIX synthetic spectral model grid (Husser et al. 2013), identifying 128,723 unique spectral lines. For each of these lines, a manifold mapping stellar parameters to spectral line parameters is created using regular grid linear interpolation. These manifolds are aggregated to to create the PHOENIX generator, sporting parallelized reconstruction of spectra at continuously valued points within the PHOENIX subset. Gaussian Process minimization is then used to infer stellar parameters by minimizing the root-mean-square (RMS) loss between the input spectrum and PHOENIX generator spectra. From testing, the inference error in $T_{\rm eff}$ was 185 K, $\log(g)$ was 0.19, and [Fe/H] was 0.12 dex. Our products are threefold: first an archive of the state dictionaries of the `blase` clones of the PHOENIX subset, second the PHOENIX generator itself, and third, the reconstruction and inference algorithm. This study represents a rudimentary proof of concept showing that semi-empirical HML is a viable, albeit fledgling alternative to traditional approaches.

1. INTRODUCTION

Stellar spectra are exceedingly rich sources of information about the stars that produce them. Spectra encode fundamental properties such as temperature, surface gravity, and chemical composition via their seemingly countless absorption lines. Extrinsic properties such as the stellar radial velocity or the projected equatorial rotation further transform spectra. Our observations of the spectra degrade the otherwise pure astronomical signals, and so the resolution the observing instrument and other instrumental properties limit our ability to extract fundamental properties perfectly. Stellar spectra represent extremely complex, data, influenced by multiple parameters, that have gone through multiple transformations before reaching our detectors.

The challenge lies in extracting as much information as possible from a stellar spectrum to get back its stellar parameters. Spectroscopic surveys such as APOGEE develop their own in-house pipelines to extract stellar parameters from spectra (ASPCAP in the case of APOGEE), and these pipelines appear effective (Majewski et al. 2017; García Pérez et al. 2016). However, they all work on the core assumption that the data to be analyzed is a list of pixels. Analysis pipelines may be closed-source or limited to the scope of the survey itself, causing a sort of siloing effect among surveys. This motivates the development of more universal, instrument-agnostic open-source frameworks that can apply broadly to a range of spectral observations with relatively little tuning.

Multiple efforts have been made in this direction, treating spectra in different ways. The standard practice is to treat the wavelength and flux as simply two arrays and use bespoke algorithms tailored to a small number of well-calibrated spectral lines to obtain fundamental stellar parameters and chemical compositions. Other whole-spectrum fitting abstractions decompose model spectra into an eigenbasis, implementing the data-model comparison stage as a tractable regression, such as `starfish` (Czekala et al. 2015). However, `blase` takes an intermediate approach (Gully-Santiago & Morley 2022), treating spectra not as a set of pixels or a set of eigenbasis coefficients but as a set of spectral lines, specifically Voigt

profiles. Each of these approaches has tradeoffs, but one key distinction places `blase` in another category defined by the ability to fit the nonlinear spectral line parameters with autodiff. We see autodiff used in tools such as `exojax` and `wobble` (Kawahara et al. 2022; Bedell et al. 2019), but the recasting of spectra into sets of inherently nonlinear spectral lines positions `blase` as a unique and promising semi-empirical tool. The original `blase` paper demonstrated the ability to tune spectral lines with autodiff, but only on a few hand-picked fiducial models, not scaled out to an entire model grid.

Ideally this process could be monolithic, with the end-to-end spectral inference code powered by a single autodiffable machine learning framework, like PyTorch or JAX (Paszke et al. 2019; Bradbury et al. 2018). However, here we separate the problem into three pieces. First, the scaling out of the blase method to 1,314 precomputed synthetic spectral model clones, yielding a downloadable archive of pretrained machine learning models with 128,723 unique spectral lines. Second, the creation of manifolds mapping stellar parameters to spectral line parameters using regular grid linear interpolation. Finally, we show how reconstructions of the spectra using said manifolds can be used for spectral inference. An overview of this process is shown in Figure 1. Advancements to this underlying basis have the potential to be top tier options with further research and development.

We chose the widely-adopted PHOENIX synthetic spectral model grid (Husser et al. 2013) as the basis for this study. Our approach can be straightforwardly applied to any other model grid in the future, including substellar atmosphere grids such as Sonora (Marley et al. 2021; Karalidi et al. 2021; Morley et al. 2024; Mukherjee et al. 2024), but for now we limit our scope to a subset of the PHOENIX grid.

2. CLONING THE PHOENIX MODEL GRID

2.1. *The PHOENIX Subset*

For the purposes of this study, we did not consider the full range of the PHOENIX synthetic spectral model grid. Instead, we focused on a subset of the grid, with details given in Table 1.

Parameter	Symbol	Range
Alpha Element Abundance	α	[0] dex
Iron Abundance	[Fe/H]	[-0.5, 0] dex
Effective Temperature	$T_{\rm eff}$	[2300, 12000] K
Surface Gravity	$\log(g)$	[2, 6]
Wavelength	λ	[8038, 12849] Å

Table 1. The subset of the PHOENIX grid used in this study. These limits were imposed to reduce the computational cost of the algorithms and to ensure a rectilinear parameter space in order to work with `scipy`'s `RegularGridInterpolator` (Virtanen et al. 2020). The wavelength limits in particular roughly line up with that of the Habitable Zone Planet Finder (HPF) spectrograph (Mahadevan et al. 2012). This subset is comprised of 1,314 individual spectra: 73 unique $T_{\rm eff}$ values, 9 unique $\log(g)$ values, and 2 unique [Fe/H] values.

2.2. *Preprocessing with `gollum`*

First, the PHOENIX subset was programmatically retrieved with `gollum` (Shankar et al. 2024). The spectra were then put through a three-step preprocessing pipeline similar to that from Gully-Santiago & Morley 2022.

1. *Blackbody Division*: Since the $T_{\rm eff}$ of each spectrum is known, the according blackbody spectrum was divided out.

2. *Percentile Normalization*: The spectra were normalized by dividing them by their 99th percentile.

3. *Continuum Normalization*: The spectra were further normalized by dividing them by a $5^{\rm th}$ order polynomial continuum.

Mathematically, we can express the preprocessing as follows:

$$\bar{\mathsf{S}} = \frac{\mathsf{S}}{\mathsf{B}\mathsf{Q}_5 P_{99}} \tag{1}$$

where $\bar{\mathsf{S}}$ is the preprocessed spectrum, S is the original spectrum, B is the blackbody spectrum, Q_n is the $n^{\rm th}$ order polynomial continuum fit, and P_n is the $n^{\rm th}$ percentile function. Arithmetic operations between arrays are assumed to be elementwise in all notation from here on out.

2.3. *Line Identification with `blase`*

The next step was to convert the PHOENIX subset into a physically interpretable intermediate representation: a table of spectral line properties rather than a list of fluxes. This was done using `blase`, which detects spectral lines as Voigt profiles and tunes the profiles to mimic the original PHOENIX spectrum with back propagation. Four parameters were optimized: the line center μ, the log-amplitude $\ln(a)$, the Gaussian width σ, and the Lorentzian width γ. The optimization used the Adam optimizer with a learning rate of 0.05 over 100 epochs (Kingma & Ba 2017). In addition, we limited two custom parameters: wing cut to 6000 and prominence

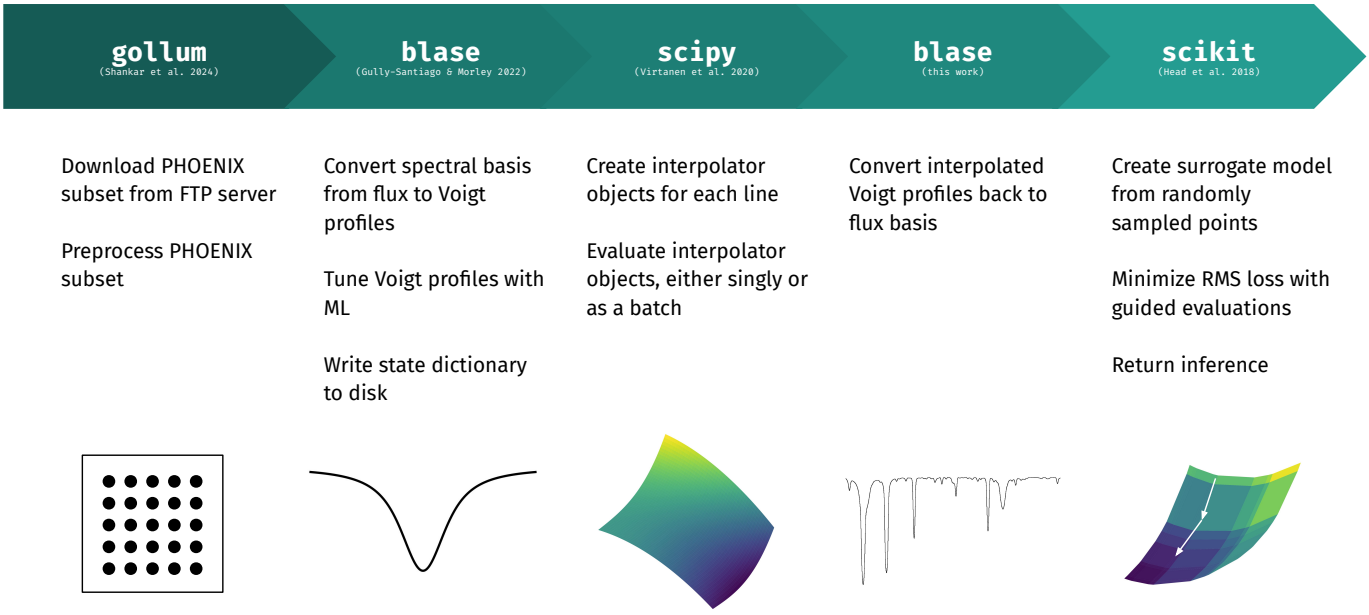

Figure 1. Overview of the process used in this study.

to 0.005. Wing cut is a parameter that determines the extent of the Voigt profile to evaluate, saving computational resources by not evaluating negligible line wings. Prominence sets a lower limit for the amplitude of detected lines, which also saves resources by disregarding shallow lines. In short, smaller values for wing cut and smaller values for prominence both decrease the computational cost of `blase`'s cloning process at the expense of decreasing its accuracy slightly. In addition, it should be mentioned that `blase` uses the pseudo-Voigt approximation, which saves on computational cost while remaining accurate to about 1% (Ida et al. 2000; Thompson et al. 1987). It is a weighted average of a Gaussian and Lorentzian as opposed to a convolution. `blase`'s pseudo-Voight profile implementation uses the following:

$$\tilde{\mathrm{V}}_\mu(\lambda) = a\left[\eta \mathbf{L}(\lambda - \mu'; f) + (1-\eta)\mathbf{G}(\lambda - \mu'; f)\right] \tag{2a}$$

$$\eta = \sum_{n=1}^{3} \mathbf{u}_n \left(\frac{2\gamma}{f}\right)^n \tag{2b}$$

$$f = 32 \sum_{n=0}^{5} \mathbf{v}_n \left(\sqrt{2\ln(2)}\sigma\right)^{5-n} (\gamma)^n \tag{2c}$$

$$\mathbf{u} = \begin{bmatrix} 1.36603 \\ -0.47719 \\ 0.11116 \end{bmatrix} \quad \mathbf{v} = \begin{bmatrix} 1 \\ 2.69269 \\ 2.42843 \\ 4.47163 \\ 0.07842 \\ 1 \end{bmatrix}$$

where $\mathbf{L}$ and $\mathbf{G}$ are abbreviations for Lorentzian and Gaussian profiles, respectively. Notice that we use μ' instead of μ in the formula. This is because `blase` optionally allows the line center to shift slightly during optimization, and it is this shifted center which is used in computation. The individual Voigt profiles are still indexed by μ for cross-model line identification, explained in the next section. Once optimization was complete, the list of identified lines, was saved to a 'state dictionary': a common representation for pre-trained machine learning models that can be stored to disk for reuse

later. These are stored in the `.pt` file format for each of the 1,314 PHOENIX subset grid points. The total disk space these files take up is 465 MB (382 MB when downloaded as a zip archive).

3. INTERPOLATING MANIFOLDS

3.1. *Cross-Model Line Identification*

As previously mentioned, `blase` tunes the line centers of detected lines. This means that from one PHOENIX spectrum to the next, the same line could have a slightly different line center. Since the goal of this study is to interpolate the properties of each line, we needed to identify the presence of a particular line across the PHOENIX subset, associating the same line with every occurrence. We decided to do this by using the line centers μ of the detected lines pre-optimization. Now with each spectral line indexed by μ, we had four parameters to interpolate: μ', $\ln(a)$, σ, and γ. Note that since we are dealing with the parameters of Voigt profiles, we can see in Equation 2 that even if the interpolation method is linear, a final spectral reconstruction will vary nonlinearly in flux.

There is also a second issue. Spectral lines were often only detected in some spectra from the PHOENIX subset. In Figure 2, we show that different grid points sport differing counts of detected spectral lines.

This missing-lines phenomenon can arise for three different reasons: Stellar atmospherse genuinely do not produce that line at detectable strength at the given temperature and surface gravity. Second, PHOENIX missed it, possibly accidentally. Or third, our line-finding and line-association algorithms missed it. Whatever the cause, these missing lines have immediate practical consequences. Rectilinear interpolation schemes break in regions where a line does not appear; you can't interpolate a quantity that simply doesn't exist.

To solve this, we artificially populated missing grid points with log-amplitudes of -1000, which retained interpolator stability while nullifying the evaluated line. An example of the appearance of missing sections in heatmaps where a line does not appear is shown in Figure 3. In total, across the entire PHOENIX subset, `blase` detected 128,723 individual spectral lines. Every one of these lines can be visualized as a manifold mapping a 3D stellar parameter vector to a 4D spectral line parameter vector, and every one must be interpolated.

3.2. *Continuously Evaluable Manifolds*

For each line, the inputs to the interpolator were the three input parameters $T_{\rm eff}$, $\log(g)$, and [Fe/H], and the output was a list of four parameters, μ', $\ln(a)$, σ, and γ. For each line, one of these interpolator objects was created using linear interpolation, and these interpolators were aggregated into a single list, which was then written to disk in the `.pkl` file storage format. These interpolators generate multiple manifolds representing the following mapping:

$$\begin{bmatrix} T_{\rm eff} \\ \log(g) \\ [\rm Fe/H] \end{bmatrix} \rightarrow \begin{bmatrix} \mu' \\ \ln(a) \\ \sigma \\ \gamma \end{bmatrix} \tag{3}$$

These interpolators could now be evaluated at any point lying within the domain of the PHOENIX subset, turning a discretely sampled PHOENIX subset into a continuously evaluable function, sometimes called a spectral emulator. With the given size of the PHOENIX subset, the interpolator list takes up 13.2 GB of disk space. This evaluation is able to reconstruct an existing PHOENIX spectrum or alternatively interpolate a new spectrum within the domain of the PHOENIX subset, so we call this the PHOENIX generator. In Figure 4, we show the same spectral line as in Figure 3, but now supersampled using the PHOENIX generator evaluated over the same slice.

The spectral reconstruction process is done by iterating over the PHOENIX generator, evaluating each interpolator at the given coordinates, then reshaping the data into the same format that PyTorch uses for state dictionaries. During the iteration, if the interpolated log-amplitude of the line is less than -100, the line is excluded from the state dictionary. We do this to avoid artifacts in the manifolds due to the artifical population of log-amplitudes of -1000 where grid points were missing.

Finally, the state dictionary is fed into `blase`'s `SparseLinearEmulator`, which reconstructs the spectrum by constructing a forward model based on the input state dictionary. Any `nan` values are set to 1 (which we can do because the spectra are all normalized), and the spectral reconstruction is complete.

4. BAYESIAN INFERENCE AND TESTING

4.1. *Spectral Reconstruction Time*

A typical use case for the PHOENIX generator may be to batch reconstruct spectra from an array of input stellar parameters. Therefore, there is some motivation to reduce the computational cost of this procedure to tractable levels. We evaluate the computational time needed to use the PHOENIX generator in two distinct ways. First, for a single input, which would be relevant in serial applications. Second, for an array of multiple inputs, which would be relevant in parallel applications. The `RegularGridInterpolator` API allows for

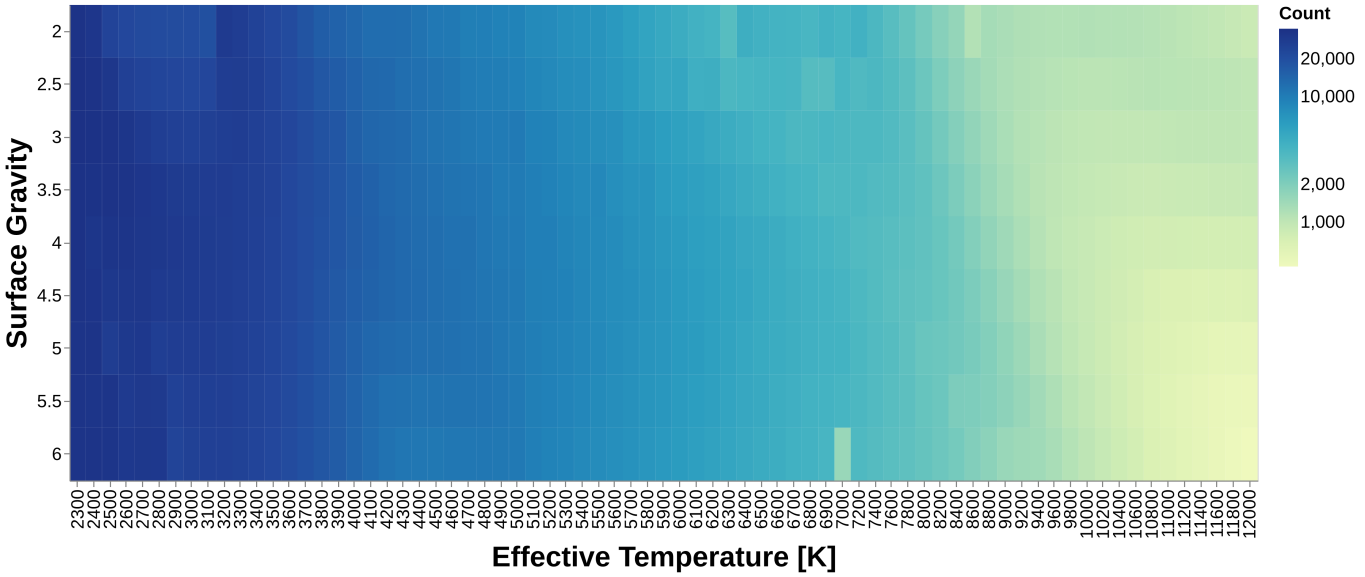

Figure 2. Number of detected spectral lines at each grid point of a slice of the PHOENIX subset at solar metallicity. We can see that the number of detected lines decreases with increasing $T_{\rm eff}$. Also note that from $T_{\rm eff}$ = 7000 K onward, the spacing between grid points increases from 100 K to 200 K.

the passing in of an entire array of input coordinates to be evaluated at once. However using `blase` to reconstruct the spectrum from our interpolated state dictionary is always done serially, leading to what is actually more of a psuedo-parallel evaluation, but extremely performant nonetheless. Performance results are shown in Figure 5, and we can see that the multi-reconstruction is much faster than a series of single reconstructions.

4.2. *Inference Algorithm*

We elected to use Bayesian optimization as the inference algorithm, specifically the `gp_minimize` function from the `scikit-optimize` library (Head et al. 2018). This algorithm uses a Gaussian Process to model the objective function, which in this case was the RMS (Root-Mean-Square) loss between the interpolated spectrum M and the true spectrum D, defined as:

$$\mathcal{L} = \left\langle (\mathsf{M} - \mathsf{D})^{\circ 2} \right\rangle^{1/2} \tag{4}$$

The optimizer was configured to first run 100 random evaluations to seed the surrogate model, then run 20 more evaluations now guided by the surrogate model. This totals to 120 evaluations, which was deemed sufficient for this study. One inference run takes on average just under 7.5 minutes to complete.

4.3. *Bayesian Optimizer Performance*

To test the performance of the inference algorithm, we used the PHOENIX subset itself. At first glance, this may seem circular, as the PHOENIX generator has memorized the PHOENIX subset, being able to reconstruct a PHOENIX spectrum when evaluated at that grid point. However, that stategy allows us to use the PHOENIX subset as test data in the context of Bayesian optimization. The `gp_minimize` surrogate model is seeded by random continuously sampled generator evaluations within the search space, *not* grid points of the PHOENIX subset, meaning the surrogate model has no memorization to speak of. If the optimizer had been a grid-based strategy, this would not have been possible, because then the surrogate model would be affected by memorization.

We know that in typical observational settings, a coarse estimate for $T_{\rm eff}$ tends to be fairly well-constrained from ancillary data prior to even looking at the spectra. So when testing the inference algorithm, we limited its search space to only include $T_{\rm eff}$ which lay within 500 K of the true value on either side. $\log(g)$ and [Fe/H] were allowed to vary freely. The test sample consisted of 9 unique $T_{\rm eff}$ values, 3 unique $\log(g)$ values, and 2 unique [Fe/H] values, totaling 54 unique spectra in the test set. $T_{\rm eff}$ ranged from 3000 K to 11000 K in increments of 1000 K, $\log(g)$ ranged from 2 to 6 in increments of 2, and [Fe/H] ranged from -0.5 to 0 in increments of 0.5.

The results of the inference algorithm are as follows: $T_{\rm eff}$ was off from the true result by an average of 185 K or 2.6%. $\log(g)$ was off by an average of 0.19 or 6.8%. [Fe/H] was off by an average of 0.12 dex, which is 24% of our search range. From this, we can see that $T_{\rm eff}$ was the most accurately inferred parameter, followed by $\log(g)$, and then [Fe/H].

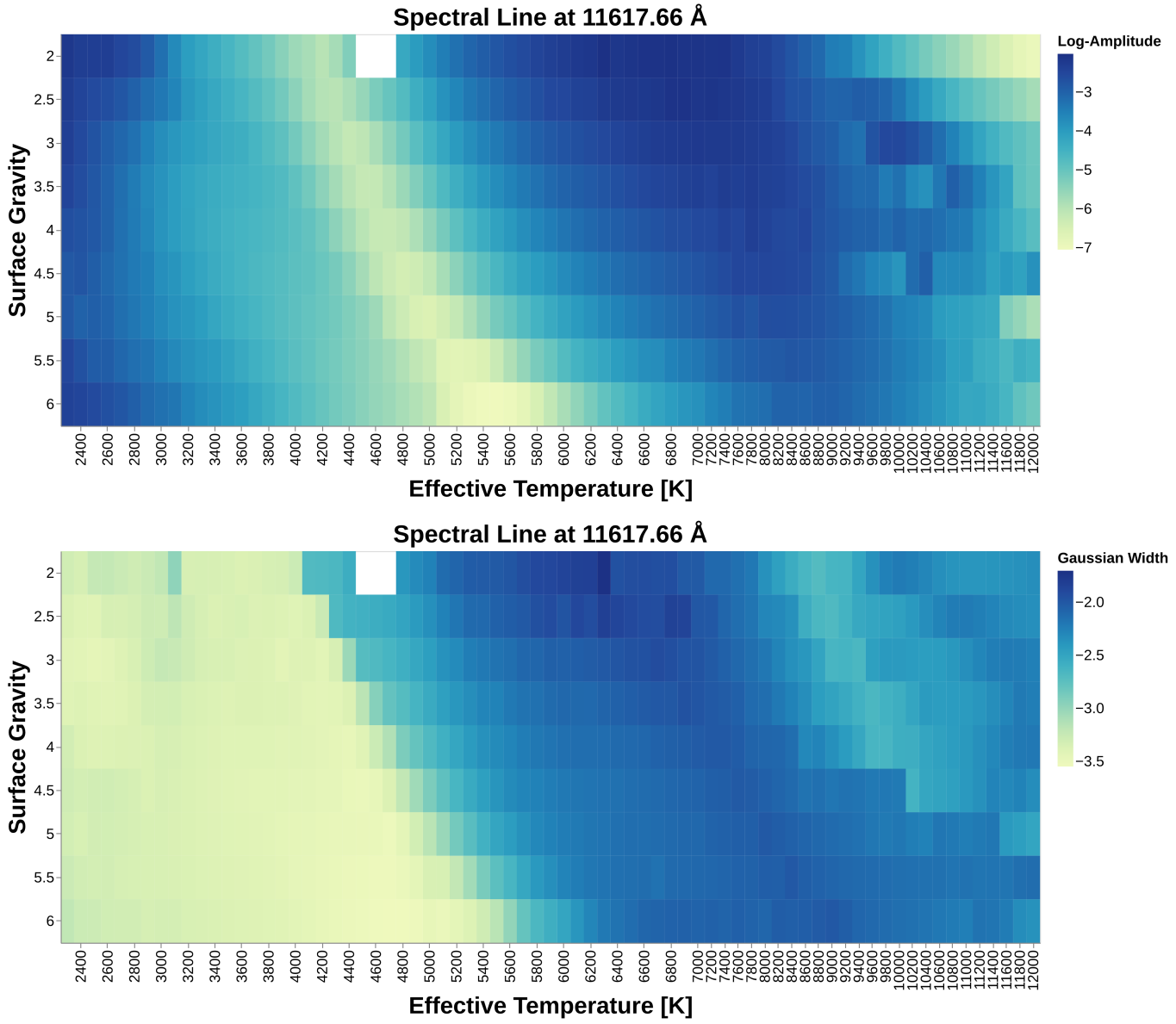

Figure 3. Heatmap showing how $\ln(a)$ and σ vary over the PHOENIX subset slice at solar metallicity. Notice the missing chunk in the top left of the figure; `blase` did not detect a spectral line here, but we have to artificially populate those points with lines that have $\ln(a) = -1000$. The change in spacing on the x-axis is owing to the step of $T_{\rm eff}$ in the PHOENIX grid changing from 100 K to 200 K after $T_{\rm eff} = 7000$ K.

5. FINAL CONSIDERATIONS

The computer used for this study, Triton, has the following specifications:

CPU	AMD EPYC 7513
RAM	256 GB
GPU	Nvidia A100 40GB (×2)

Table 2. This machine was used for all computations, but not for generating visualizations. The EPYC 7513 is a 32c/64t CPU with a boost clock of 3.65 GHz. The A100 has 6912 CUDA cores.

The interpolation scheme presented here represents a proof of concept, showing that leveraging the mapping between synthetic spectral lines and their inputs can yield a semi-empirical basis for data-model comparisons. There are numerous design considerations that could be improved upon with future work. These include but are not limited to the following:

- *Limited PHOENIX Subset*: The PHOENIX subset used in this study did not include the full PHOENIX grid, which expands the [Fe/H] range to [-4.0, 1.0] dex and the $\log(g)$ range to [0, 6], and also includes the alpha element abundance param-

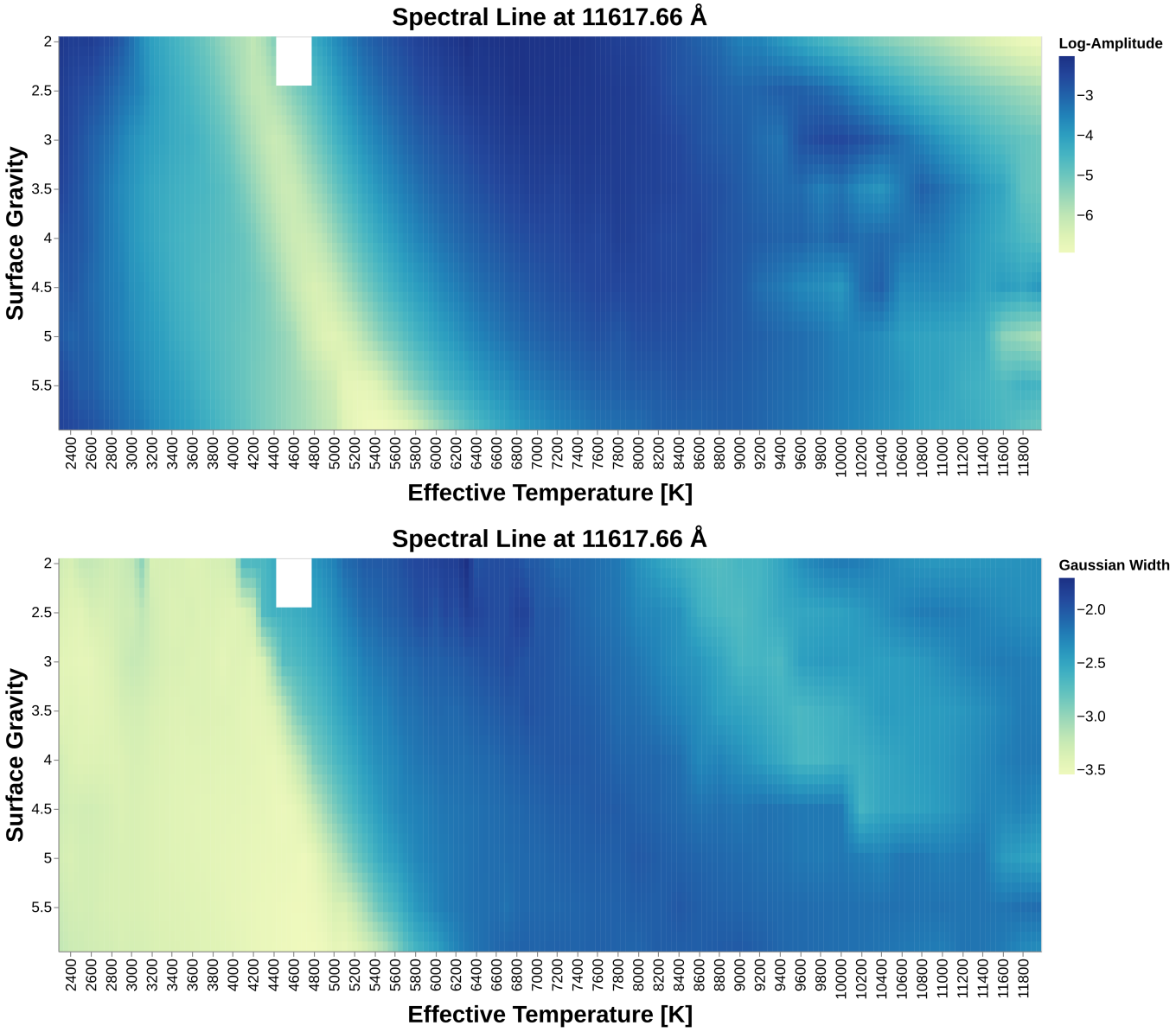

Figure 4. Heatmap showing how $\ln(a)$ and σ vary over the PHOENIX subset slice at solar metallicity, now supersampled with the PHOENIX generator. Notice that the missing chunk in the top left still exists and does not display any artifacts, as the artificially populated points are removed after interpolation to retain the model's integrity. Also see that the x-axis spacing is now uniform, as the PHOENIX generator was evaluated at constant step.

eter, which we elected to fix at 0 for this study. In addition to the actual stellar parameters, we also took a subset of the PHOENIX wavelength range, with the full [500, 55000] Å wavelength range also being left to future work.

- *Strict Wavelength Range*: Currently, the generator only supports inference on spectra whose wavelength limits are either equal to it, or encompass that of the generator and have been truncated to match. However, when the spectrum in question has a smaller wavelength range than the generator, currently there is no functionality to truncate the generator. This would require externally indexing the generator's individual interpolators by line center position and selectively evaluating those to eliminate wasteful computation.

- *Single Model Grid*: The PHOENIX grid is not the only model grid of synthetic spectra available, and it does not apply to all types of stars. Future work would extend the reach of this study's algorithm to encompass other model grids such as the Sonora series of substellar models. `blase` should be able

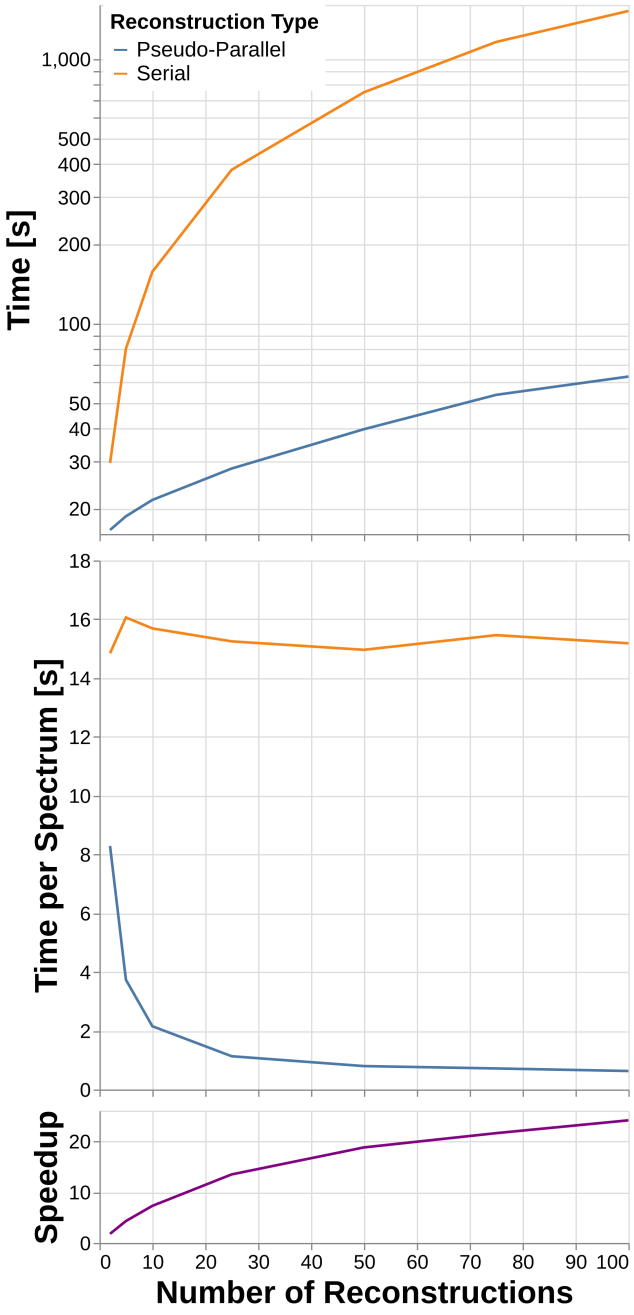

Figure 5. Line chart showing the time taken to reconstruct varying numbers of spectra using the PHOENIX generator (lower is better). We can see that the time taken per spectra for the serial implementation hovers around 15 seconds within run-to-run variance, while the pseudo-parallel implementation continually decreases in time taken per spectrum as the number of inputs increases. The speedup factor (higher is better) increases as more spectra are generated, which is also a desirable outcome.

to have an option for the user to input which model grid they would like to base the inference on, and to get even more advanced, perhaps even have the ability to intelligently determine which model grid to use automatically.

- *Memorization vs. Generalization*: The current design of the algorithm constructs manifolds using interpolation. This means that performance is good at points close to PHOENIX subset grid points, but is highly dependent on the type of interpolation used. As interpolators require memorization of the data, advanced interpolation becomes extremely expensive in terms of disk utilization. Future work would involve constructing manifolds using more advanced methods, which would allow for much better generalization and lower disk utilization at the expense of some accuracy.

- *Extrinsic Absence*: The current design of our algorithm does not account for extrinsic parameters that modify the appearance of spectra such as rotational broadening and Doppler shifting. Future work would need to develop ways to tune these extrinsic parameters alongside stellar parameters.

- *Framework Overhead*: As this algorithm is currently more proof of concept than practical, it uses convenience functions from various libraries, which naturally introduces some level of overhead and leaves performance on the table. Future work would involve writing custom functions expressly designed for `blase`, most likely a complete rewrite of the library from the ground up.

- *Pseudo-Interpretability*: Our algorithm boasts interpretability by considering spectral lines as the objects of interest as opposed to the rather uninterpretable flux values of other approaches. However, this is only a step in the direction of interpretability. True interpretability would decompose a spectrum not into a set of spectral lines, but into a set of species component spectra, which requires a much more advanced understanding of different species and their behavior, as well as direct access to a radiative transfer code as opposed to an off-the-shelf model grid. This approach would also extend the inference from just stellar parameters defined by a grid to any set of parameters accounted for in the radiative transfer model, down to specific species abundances. So while readers may be tempted to try to identify certain spectral lines seen in our figures, this is not the point.

`blase` is *agnostic* to the identity of the line that it is optimizing. We study these lines as `blase` sees them (i.e. their four shape parameters), because for the purposes of this study, that is the only information that is useful.

- *The Continuum Black Box*: Continuum normalization is a process that is not yet completely understood, and is currently done as a preprocessing step with a fairly simple algorithm. Future work would dive deeper into the science of continuums and develop more advanced methods that can discern continuums with greater accuracy and less modeling restrictions.

- *One Voigt Fits All*: The current assumption of `blase` is that every spectral line is a Voigt profile. This assumption is largely true, but there are situations where that is simply not enough. Future studies need to account for more advanced spectral line profiles and procedures to deal with phenomena such as ro-vibrational bands.

In short, it is quite clear that `blase` is a long way from being a powerhouse in spectral inference, however it represents a step down a road not yet traveled, and the potential for future growth is immense. At the end of the day, we aim to develop a tool that can analyze spectra with the growing power of physics-informed machine learning.

1 Text

Software: `altair` (VanderPlas et al. 2018; Satyanarayan et al. 2017), `astropy` (Astropy Collaboration et al. 2013, 2018, 2022), blasé/`blase` (Gully-Santiago & Morley 2022), CUDA (NVIDIA et al. 2020), `gollum` (Shankar et al. 2024), `matplotlib` (Hunter 2007), `numpy` (Harris et al. 2020), `pandas` (pandas development team 2020; Wes McKinney 2010), Python (Van Rossum & Drake 2009), PyTorch/`torch` (Paszke et al. 2019), scikit-optimize/`skopt` (Head et al. 2018), `scipy` (Virtanen et al. 2020), `tqdm` (da Costa-Luis 2019), `vegafusion` (Kruchten et al. 2022),

REFERENCES

Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., et al. 2013, A&A, 558, A33, doi: 10.1051/0004-6361/201322068

Astropy Collaboration, Price-Whelan, A. M., Sipőcz, B. M., et al. 2018, AJ, 156, 123, doi: 10.3847/1538-3881/aabc4f

Astropy Collaboration, Price-Whelan, A. M., Lim, P. L., et al. 2022, ApJ, 935, 167, doi: 10.3847/1538-4357/ac7c74

Bedell, M., Hogg, D. W., Foreman-Mackey, D., Montet, B. T., & Luger, R. 2019, The Astronomical Journal, 158, 164, doi: 10.3847/1538-3881/ab40a7

Bradbury, J., Frostig, R., Hawkins, P., et al. 2018, JAX: composable transformations of Python+NumPy programs, 0.3.13. http://github.com/google/jax

Czekala, I., Andrews, S. M., Mandel, K. S., Hogg, D. W., & Green, G. M. 2015, ApJ, 812, 128, doi: 10.1088/0004-637X/812/2/128

da Costa-Luis, C. O. 2019, Journal of Open Source Software, 4, 1277, doi: 10.21105/joss.01277

García Pérez, A. E., Allende Prieto, C., Holtzman, J. A., et al. 2016, AJ, 151, 144, doi: 10.3847/0004-6256/151/6/144

Gully-Santiago, M., & Morley, C. V. 2022, ApJ, 941, 200, doi: 10.3847/1538-4357/aca0a2

Harris, C. R., Millman, K. J., van der Walt, S. J., et al. 2020, Nature, 585, 357, doi: 10.1038/s41586-020-2649-2

Head, T., MechCoder, Louppe, G., et al. 2018, scikit-optimize/scikit-optimize: v0.5.2, v0.5.2, Zenodo, doi: 10.5281/zenodo.1207017

Hunter, J. D. 2007, Computing in Science & Engineering, 9, 90, doi: 10.1109/MCSE.2007.55

Husser, T. O., Wende-von Berg, S., Dreizler, S., et al. 2013, A&A, 553, A6, doi: 10.1051/0004-6361/201219058

Ida, T., Ando, M., & Toraya, H. 2000, Journal of Applied Crystallography, 33, 1311, doi: https://doi.org/10.1107/S0021889800010219

Karalidi, T., Marley, M., Fortney, J. J., et al. 2021, ApJ, 923, 269, doi: 10.3847/1538-4357/ac3140

Kawahara, H., Kawashima, Y., Masuda, K., et al. 2022, ExoJAX: Spectrum modeling of exoplanets and brown dwarfs, Astrophysics Source Code Library, record ascl:2206.003

Kingma, D. P., & Ba, J. 2017, Adam: A Method for Stochastic Optimization. https://arxiv.org/abs/1412.6980

Kruchten, N., Mease, J., & Moritz, D. 2022, in 2022 IEEE Visualization and Visual Analytics (VIS), 11–15, doi: 10.1109/VIS54862.2022.00011

Mahadevan, S., Ramsey, L., Bender, C., et al. 2012, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 8446, Ground-based and Airborne Instrumentation for Astronomy IV, ed. I. S. McLean, S. K. Ramsay, & H. Takami, 84461S, doi: 10.1117/12.926102

Majewski, S. R., Schiavon, R. P., Frinchaboy, P. M., et al. 2017, AJ, 154, 94, doi: 10.3847/1538-3881/aa784d

Marley, M. S., Saumon, D., Visscher, C., et al. 2021, ApJ, 920, 85, doi: 10.3847/1538-4357/ac141d

Morley, C. V., Mukherjee, S., Marley, M. S., et al. 2024, arXiv e-prints, arXiv:2402.00758, doi: 10.48550/arXiv.2402.00758

Mukherjee, S., Fortney, J. J., Morley, C. V., et al. 2024, arXiv e-prints, arXiv:2402.00756, doi: 10.48550/arXiv.2402.00756

NVIDIA, Vingelmann, P., & Fitzek, F. H. 2020, CUDA, release: 10.2.89. https://developer.nvidia.com/cuda-toolkit

pandas development team, T. 2020, pandas-dev/pandas: Pandas, latest, Zenodo, doi: 10.5281/zenodo.3509134

Paszke, A., Gross, S., Massa, F., et al. 2019, PyTorch: An Imperative Style, High-Performance Deep Learning Library. https://arxiv.org/abs/1912.01703

Satyanarayan, A., Moritz, D., Wongsuphasawat, K., & Heer, J. 2017, IEEE transactions on visualization and computer graphics, 23, 341

Shankar, S., Gully-Santiago, M., Morley, C. V., et al. 2024

Thompson, P., Cox, D. E., & Hastings, J. B. 1987, Journal of Applied Crystallography, 20, 79, doi: https://doi.org/10.1107/S0021889887087090

Van Rossum, G., & Drake, F. L. 2009, Python 3 Reference Manual (Scotts Valley, CA: CreateSpace)

VanderPlas, J., Granger, B., Heer, J., et al. 2018, Journal of Open Source Software, 3, 1057, doi: 10.21105/joss.01057

Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, Nature Methods, 17, 261, doi: 10.1038/s41592-019-0686-2

Wes McKinney. 2010, in Proceedings of the 9th Python in Science Conference, ed. Stéfan van der Walt & Jarrod Millman, 56 – 61, doi: 10.25080/Majora-92bf1922-00a