Draft version February 5, 2025
Typeset using LATEX **twocolumn** style in AASTeX631

A New Hybrid Machine Learning Method for Stellar Parameter Inference

Sujay Shankar,[1] Michael A. 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) is revolutionary to numerous scientific disciplines, with a growing number of examples in astronomical spectroscopic inference, as ML is expected to be more powerful than traditional techniques. Here we introduce a hybrid ML (HML) method combining automatic differentiation, interpolation, and Bayesian optimization to infer stellar parameters given stellar spectra. We study $T_{\rm eff}$, $\log(g)$, and [Fe/H], but this method **could be extended** to other **parameters** such as [α/Fe] **(alpha element abundance)**, C/O **(carbon-oxygen ratio)**, and f_{sed} **(sedimentation efficiency)**. We first use `blase`'s nontraditional semi-empirical approach, **recasting** spectra **into** sets of tunable spectral lines. `blase` is **run** on 1,314 spectra from **a rectilinear subset of** the PHOENIX synthetic spectral model grid **([Fe/H]: [-0.5, 0] dex, $T_{\rm eff}$: [2300, 12000] K,** $\log(g)$**: [2, 6])**. For each of the **128,723** lines, we continuously map stellar parameters to spectral line parameters using regular grid linear interpolation. These manifolds are aggregated to create the PHOENIX generator, **enabling** parallelized reconstruction of spectra given stellar parameters. Gaussian Process minimization is then used to infer stellar parameters by minimizing the root-mean-square (RMS) loss between input 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 an archive of the `blase` **models** of the PHOENIX subset, as well as the spectral reconstruction and inference algorithms themselves. This study is a proof of concept showing that semi-empirical HML is a viable 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 numerous absorption lines. Extrinsic properties such as the stellar radial velocity or the projected equatorial rotation shift the wavelengths of lines and broaden their widths, respectively. Observing the spectrum alters it again; the resolution, bandwidth, and other properties of the instrument change the fidelity at which we observe the spectrum, and can limit our ability to extract fundamental properties precisely. Observed stellar spectra thus represent extremely complex, data, influenced by multiple parameters, that have gone through multiple transformations before reaching our detectors.

Modern astronomical spectroscopy **takes advantage of intuitive, performant spectral models such as KORG** (Wheeler et al. 2023)**, however it** would **further** benefit from **the paradigm of interpretability**. The determination of fundamental properties, the creation of Extreme Precision Radial Velocity (EPRV) templates, and the application of composite spectral fitting could all leverage such an innovation. This ambitious aim may be broadly referred to as a "foundational spectroscopy model for astrophysics", in reference to the same category of models used for large language models (LLMs) in artificial intelligence (AI). There are many challenges with creating such a foundational spectral model for astrophysics. Physical inputs into the spectral modeling process are imperfect**, and** simplifying assumptions in stellar (and substellar) atmospheres are necessarily inexact **due to factors such as asymmetries and unknown physics**. Computational costs **make** the training of such models **challenging; however, advances have been made with attention-based** (Różański et al. 2023) **and transformer-based** (Leung & Bovy 2024) **models, among others**. Most prevailing solutions have had to choose either a model driven approach **(such as this work)**, in which precomputed models are taken as gospel, or a data

Corresponding author: Sujay Shankar
sujays2001@gmail.com

driven/empirical approach **(such as Lux (Horta et al. 2025))**, in which our knowledge of stellar physics is ignored or treated phenomenologically, depending on the application. **Hybrid solutions exist and have been implemented, such as those presented in Leung & Bovy 2019 and Rains et al. 2024. Although a lingering challenge is always how to ideally balance the fusion of model driven and data driven paradigms.**

Spectroscopic surveys such as APOGEE (**Majewski et al. 2017**) use their own in-house pipelines to extract stellar parameters from spectra (ASPCAP (**García Pérez et al. 2016**) in the case of APOGEE), and these pipelines appear effective. The core assumption for the vast majority of pipelines is 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 (López-Valdivia et al. 2023; Rayner et al. 2009). 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).

Ideally, we want a system that can self-consistently learn, a genuine AI foundational spectral model. Such a system would enable the assignment of accurate stellar parameters, and could yield re-usable interpretable spectral models. `blase`, first presented in Gully-Santiago & Morley 2022, took an important, albeit limited, step in this direction, treating spectra not as a set of pixels or a set of eigenbasis coefficients but as a set of interpretable and traceable spectral lines, specifically Voigt profiles. Each of these approaches has tradeoffs, but one key scientific advantage of `blase` comes from its ability to adapt to new information, while preserving some adherence to physics-based models. This intelligent capability stems from its ability to fit a theoretically unlimited number of nonlinear spectral line parameters with automatic differentiation (autodiff). Autodiff is a technology that tracks transformations made to data using the chain rule, even being able to differentiate control flow transformations involving if-else blocks, for example. With the gradient obtained from autodiff's chain rule, we can optimize model parameters by just going against the direction of the gradient (because we usually want to minimize something such as a loss function, we go in the direction of greatest decrease, which is always the negative of the gradient). Autodiff has been used successfully in other astrophysical contexts such as `exojax` (**Kawahara et al. 2022**) and `wobble` (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 **and 'clone' spectra, recasting them as ML models defined by interpretable sets of tuned spectral lines, however it was** restricted to a pre-selected static synthetic model.

Here we introduce the next logical step in the sequence of expanding interpretable spectroscopic machine learning from operating on a single grid point and towards an entire 3D grid of precomputed spectra. We rebrand this augmentation as `blase3D`.

Ideally this process would be monolithic, with the end-to-end spectral inference code powered by a single autodiffable machine learning framework, like PyTorch **(Paszke et al. 2019)** or JAX (Bradbury et al. 2018). However, here we separate the problem into three pieces, only the first of which is currently differentiable. **In section 2,** we scale out 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. **In section 3,** we **then** fit manifolds mapping stellar parameters to uniformly-derived spectral line parameters using regular grid linear interpolation. Finally, **in section 4,** we show how reconstructions of the spectra using said manifolds can be used for inferring stellar fundamental properties from spectra. This final step resembles the aims of "atmospheric retrievals", but in principle should be faster, adaptive, and reusable. An overview of this process is shown in Figure 1.

2. CLONING THE PHOENIX MODEL GRID

2.1. *The PHOENIX Subset*

For the purposes of this study, we chose the widely-adopted PHOENIX synthetic spectral model grid (Husser et al. 2013). 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 **rectilinear** subset of the PHOENIX grid, focusing on near solar metallicities and a broad range of

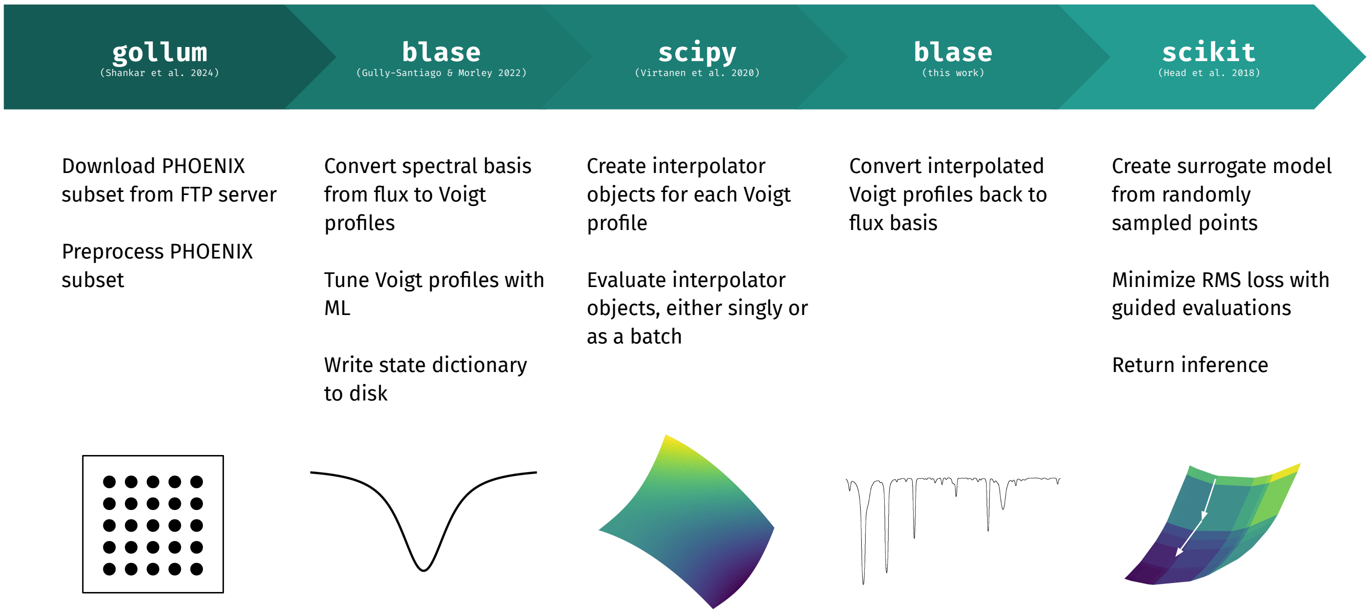

Figure 1. Overview of the process used in this study.

effective temperature and surface gravity, with details given in Table 1. **This approach is taken due to the computational cost of interpolation algorithms. Future versions with more advanced models will be able to bypass the limitation of a rectilinear subset.**

2.2. *Preprocessing with* `gollum`

First, the PHOENIX subset was programmatically retrieved with `gollum` (**Shankar et al. 2024**), which downloaded **spectra** from the PHOENIX FTP server. 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_{eff} of each spectrum is known, the **corresponding** blackbody spectrum was divided out.

2. *Percentile Normalization*: The spectra were normalized by dividing them by their 99th percentile in order to collapse the dynamic range of flux and only look at relative features.

3. *Continuum Normalization*: The spectra were further normalized by dividing them by a 5^{th} order polynomial continuum fit using a peak-finding filter in order to eliminate curvature that would inhibit line modeling.

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^{th} order polynomial continuum fit, and P_n is the n^{th} percentile function. Arithmetic operations between arrays are assumed to be elementwise in all following notation.

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**n** array of fluxes. We used `blase`, which models spectral lines as Voigt profiles and tunes the profiles to mimic the original PHOENIX spectrum with back propagation. Back propagation, put simply, is the process of moving in the autodiff gradient field to update ML model parameters (like mentioned earlier, usually against the gradient because we minimize loss functions). 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 (**Kingma & Ba 2017**) with a learning rate of 0.05 over 100 epochs. **Gully-Santiago & Morley 2022 recommends this**

Parameter	Symbol	Interval	Sampling
Alpha Element Abundance	α	[0] dex	**N/A**
Iron Abundance	[Fe/H]	[-0.5, 0] dex	**0.5 dex**
Effective Temperature	$T_{\rm eff}$	[2300, 12000] K	**100 K until 7000 K, then 200 K**
Surface Gravity	$\log(g)$	[2, 6]	**0.5**
Wavelength	λ	[8038, 12849] Å	$\mathbf{R = 500,000}$

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 $T_{\rm eff}$ values, 9 $\log(g)$ values, and 2 [Fe/H] values.

setup as the minimum, and for a proof-of-concept implementation, we found it best to leave it as is. In addition, we limited two custom parameters: wing cut to 6000 and prominence to 0.005. Wing cut **(in pixels)** is a parameter that determines the extent of the Voigt profile to evaluate, saving computational resources by not evaluating negligible line wings. Prominence **(in normalized flux counts)** sets a lower limit for the amplitude of detected lines, which saves resources by disregarding shallow lines, so in our case we disregard lines **with amplitude** < 0.05. In short, our choices for wing cut and prominence decrease the computational cost of `blase`'s cloning process at the expense of decreasing its accuracy slightly. `blase` uses the pseudo-Voigt approximation, which saves on computational cost while remaining accurate to about 1% (Ida et al. 2000). The pseudo-Voigt approximation uses a weighted average of a Gaussian and Lorentzian as opposed to a convolution. `blase`'s pseudo-Voigt profile implementation uses the following:

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

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

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

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

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). For reference, the storage space the PHOENIX subset takes up on disk is approximately 8.1 GB. This represents a data compression factor of around 20 just by recasting the spectrum with `blase`. **The state dictionaries are available on Zenodo at https://zenodo.org/records/11246174 (Shankar 2024).**

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.

Spectral lines were often only detected in some spectra from the PHOENIX subset. In Figure 2, we show that different grid points **show** differing counts of detected spectral lines.

The **number of detected spectral lines changes as a function of stellar parameter** for different reasons. Perhaps for astrophysical reasons: stellar atmo-

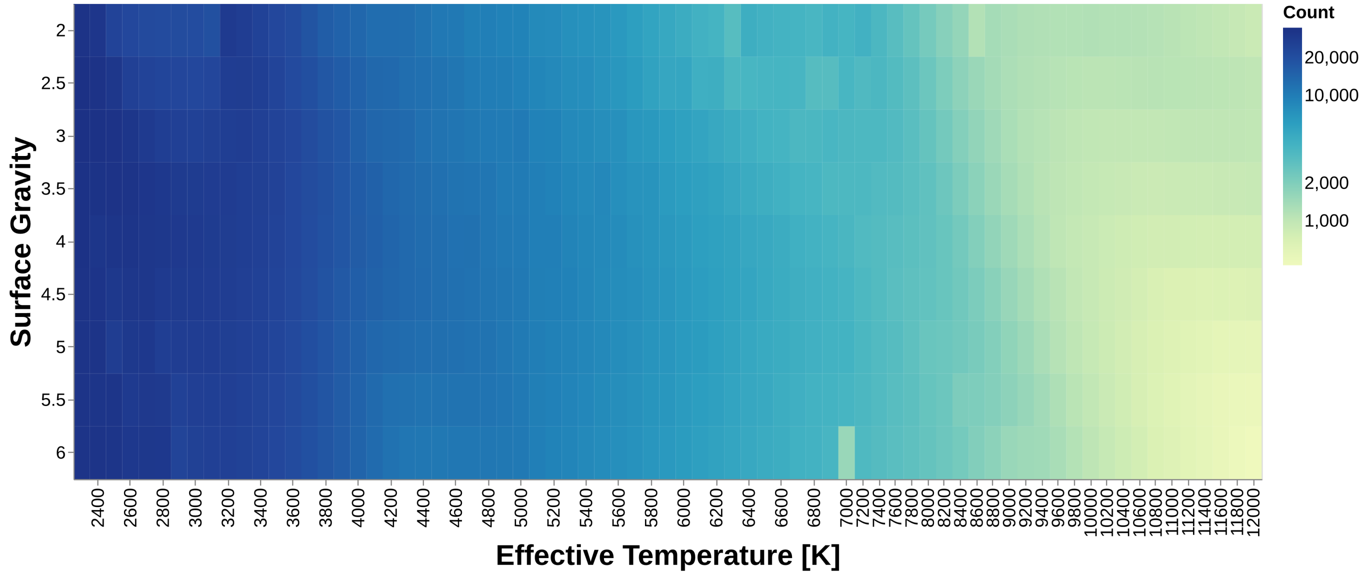

Figure 2. Number of detected spectral lines at each grid point of a slice of the PHOENIX subset at solar metallicity. We can see that the number of detected lines decreases with increasing $T_{\rm eff}$. Also note that from $T_{\rm eff}$ = 7000 K onward, PHOENIX's sampling increment changes from 100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

spheres genuinely do not produce that line at detectable strength at the given temperature and surface gravity. Or alternatively, 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. Examples of the appearance of missing sections in heatmaps where a line does not appear are shown in Figure 3 and Figure 4. 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 **was** interpolated **to map stellar parameters to spectral line properties**.

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 5 and Figure 6, we show the same spectral lines as in Figure 3 and Figure 4, 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 spec-

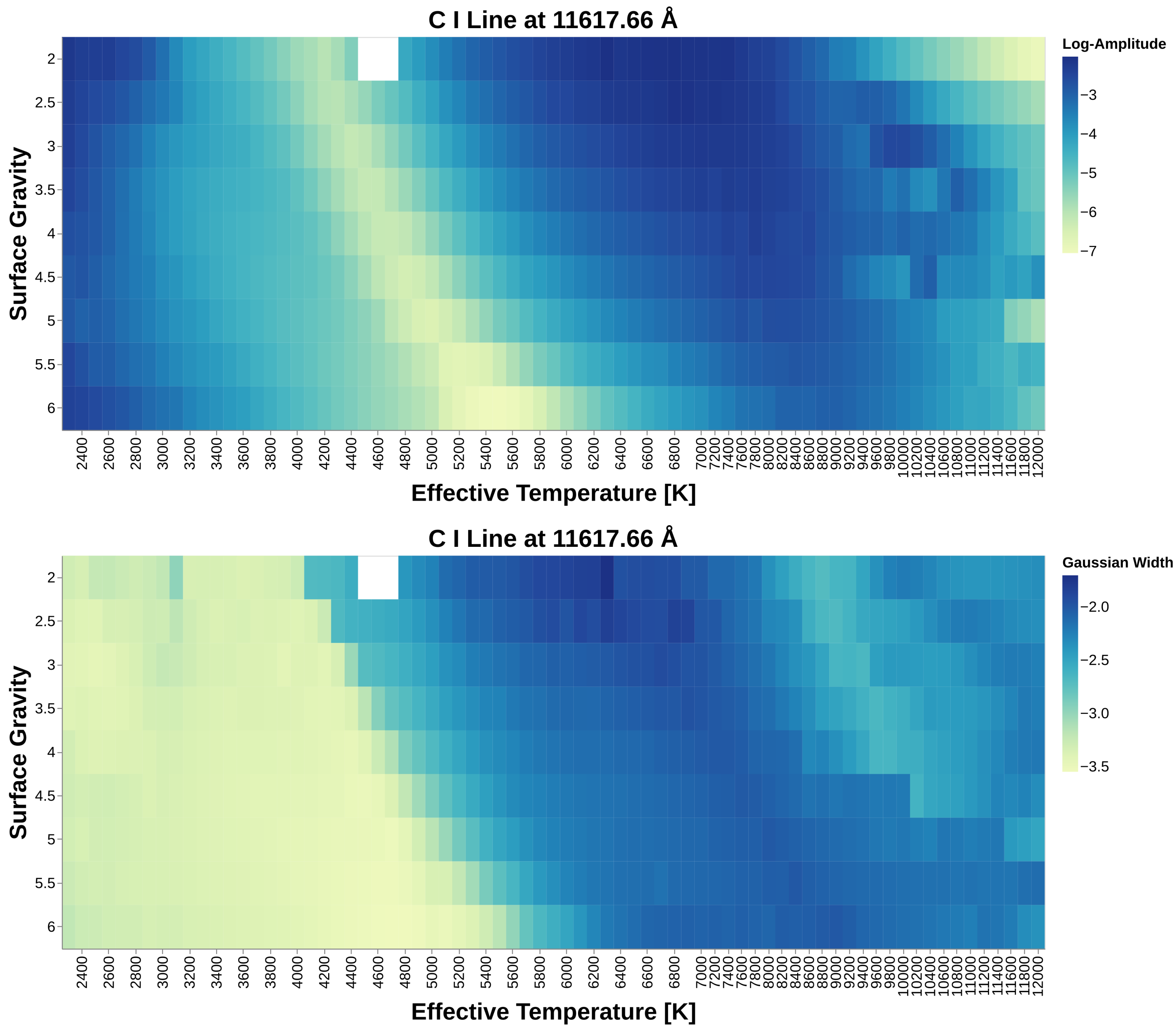

Figure 3. Heatmap showing how $\ln(a)$ and σ vary over the PHOENIX subset slice at solar metallicity of a **C I** spectral line. Notice the missing chunk in the top left of the figure; `blase` did not detect a spectral line here, but we have to artificially populate those points with lines that have $\ln(a) = -1000$. This and all spectral lines shown in this paper were identified using the NIST spectral line database (Kramida et al. 2023).

trum 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. We can observe in Figure 7 that the reconstructed solar spectrum is not simply a pixel interpolation between the nearest grid points. It is interpolating the spectral line properties using hundreds of thousands of manifolds, each representing a nonlinear parameter in the shape of the spectral line.

3.3. *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. `scipy`'s `RegularGridInterpolator` API allows for the passing in of an entire array of input coordinates to be evaluated at once. However using `blase` to reconstruct

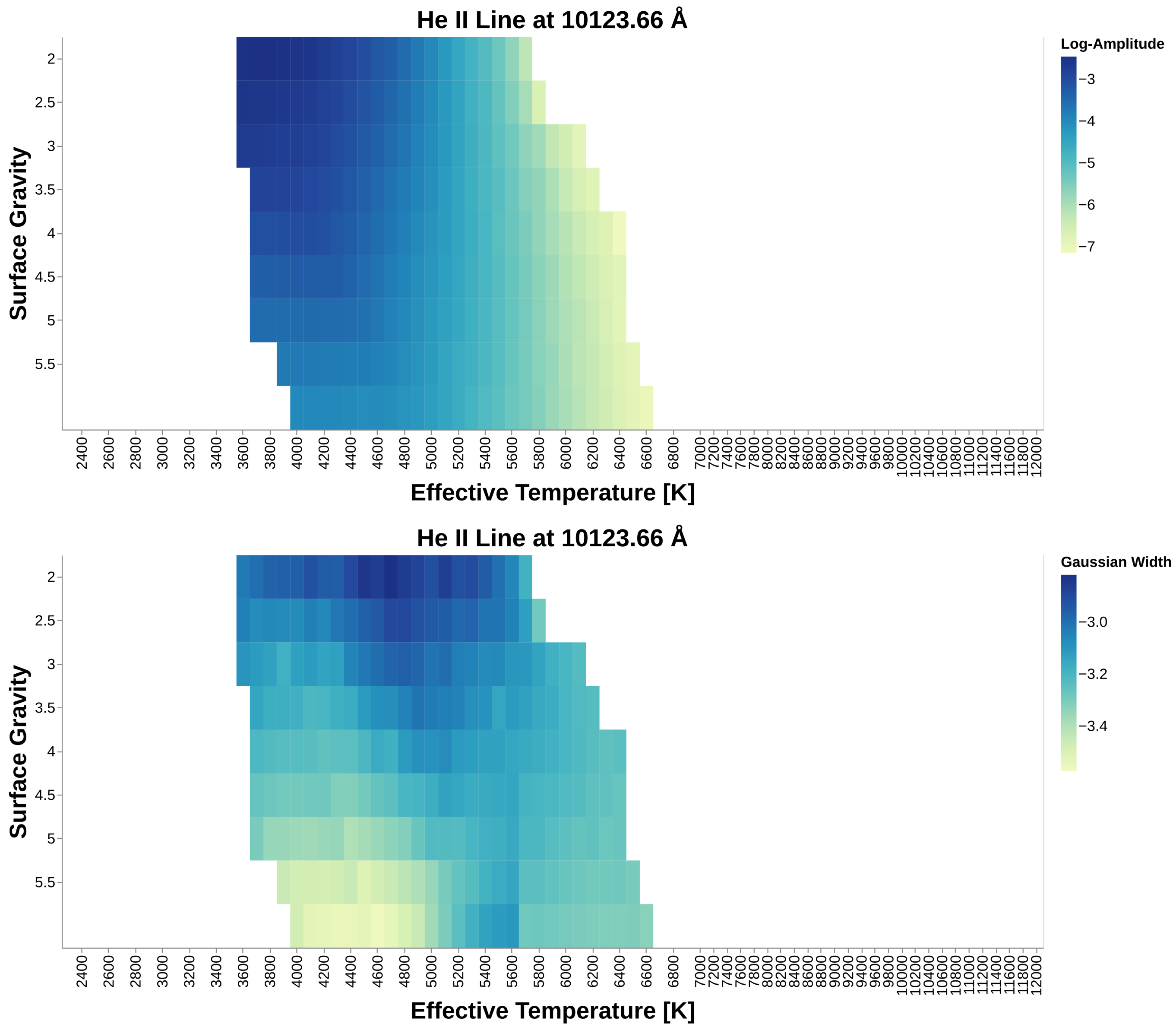

Figure 4. Heatmap showing how $\ln(a)$ and σ vary over the PHOENIX subset slice at solar metallicity of a He II spectral line. We can see that `blase` detects this line at only a select chunk of grid points in PHOENIX, leading to the large amount of missing data for the line.

the spectrum from our interpolated state dictionary is always done serially, leading to what is actually more of a pseudo-parallel evaluation, but extremely performant nonetheless. Performance results are shown in Figure 8, and we can see that the multi-reconstruction is much faster than a series of single reconstructions. **The computer used for this test has the following specifications (note that spectral reconstruction does not currently leverage the GPU):**

CPU	AMD EPYC 7513 (32c/64t, 3.65 GHz boost)
RAM	256 GB
GPU	Nvidia A100 40GB (×2)

Table 2. This machine was used for all computations, but not for generating visualizations.

4. BAYESIAN INFERENCE AND TESTING

4.1. *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

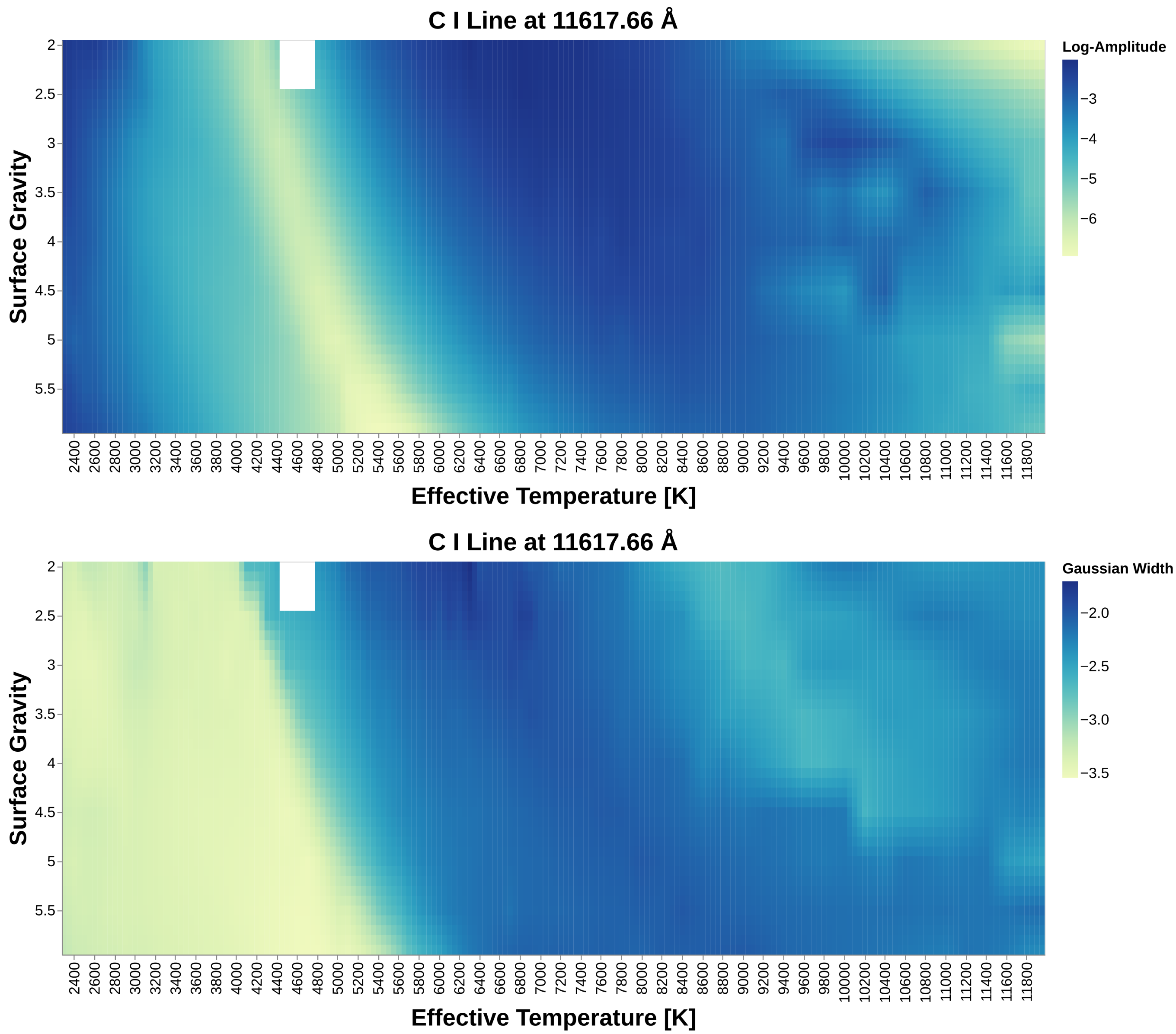

Figure 5. Heatmap showing how $\ln(a)$ and σ of our **C I** line vary over the PHOENIX subset slice at solar metallicity, now supersampled with the PHOENIX generator. Notice that the missing chunk in the top left still exists and does not display any artifacts, as the artificially populated points are 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.

M and the true spectrum D, defined as:

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

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, a large sample to create a fairly detailed surrogate model, then a moderately precise guided evaluation phase, which was deemed sufficient for this study. Fine-tuning these numbers is possible, but simply not warranted for a proof-of concept method. One inference run takes on average just under 7.5 minutes to complete.

4.2. *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 strategy allows us to use the PHOENIX subset as test data in the context of Bayesian optimization. The `gp_minimize` surrogate

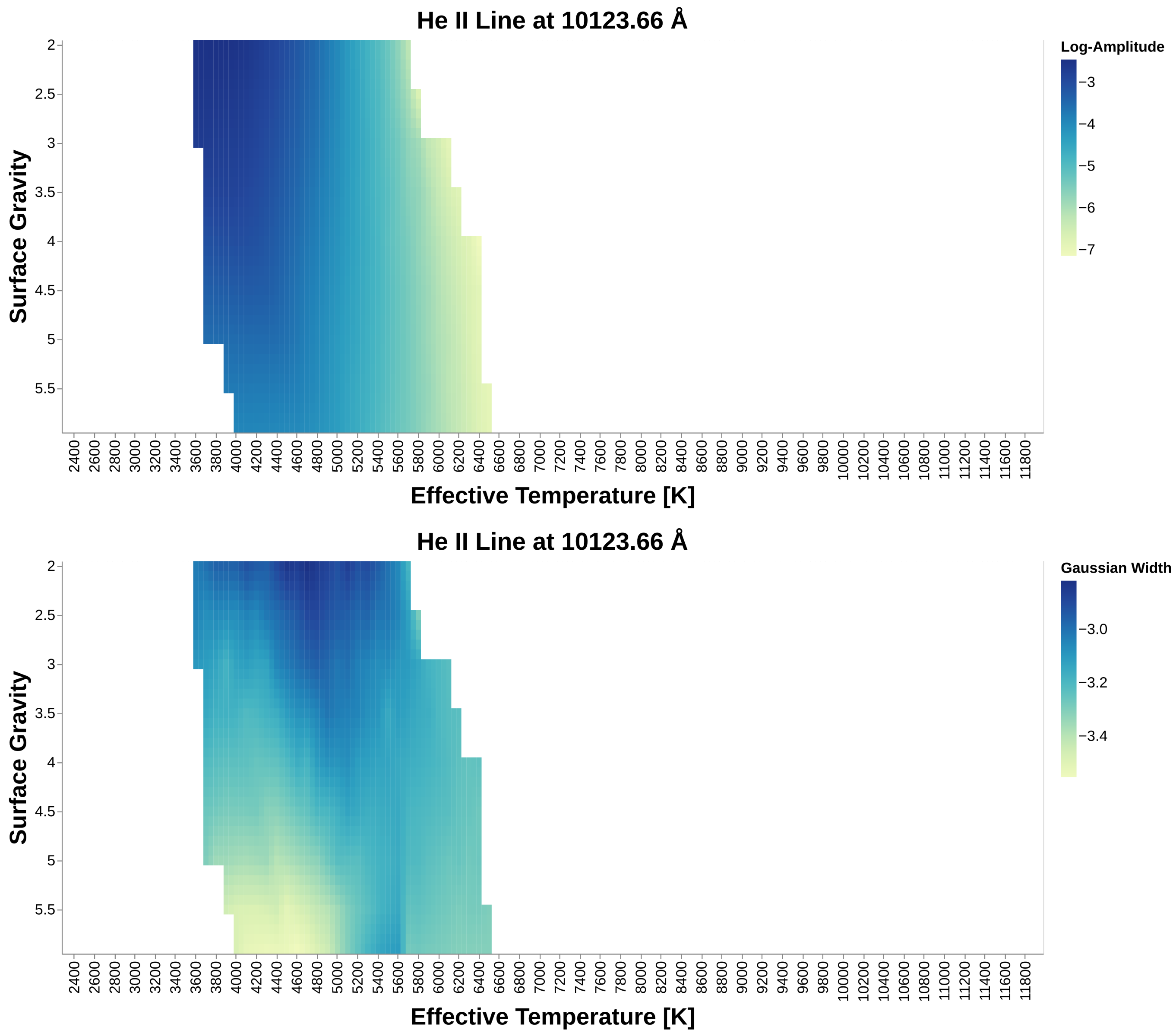

Figure 6. Heatmap showing how $\ln(a)$ and σ of our He II spectral line vary over the PHOENIX subset slice at solar metallicity, now supersampled with the PHOENIX generator.

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 **such as photometric color index**. 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: $T_{\rm eff}$ differed from the true result by an average of 185 K or 2.6%. $\log(g)$ differed by an average of 0.19 or 6.8%. [Fe/H] differed by an average of 0.12 dex, which is 24% of our search range. From this, we can see that $T_{\rm eff}$

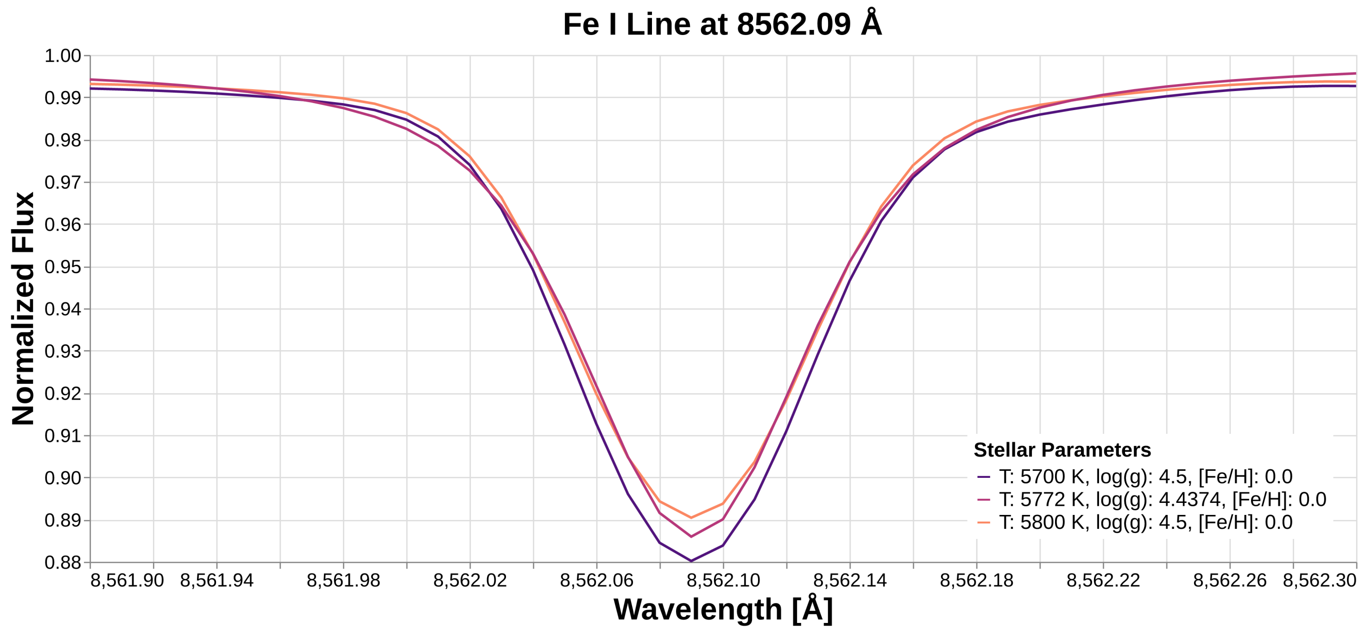

Figure 7. Plot of an Fe I spectral line shown at the closest PHOENIX grid points to the Sun's stellar parameters, as well as a solar spectrum reconstructed with the PHOENIX generator. We can see the spectral line shape be reconstructed mostly between the two native PHOENIX spectra, being closer to the 5800 K spectrum as the Sun's $T_{\rm eff}$ used here is 5772 K.

was the most accurately inferred parameter, followed by $\log(g)$, and then [Fe/H].

5. DISCUSSION

5.1. *Scientific Applications*

This study's paradigm of spectral inference can enable scientists to adopt self-consistent model grids and analyze their spectral line behavior in an interpretable fashion, which has been fairly uncommon practice thus far. Notably, this system tracks the shifting of spectral lines as a function of stellar parameters, which traditionally has been uncommon for algorithms to assess systematically. We balance rigidity and flexibility where we want them while maintaining interpretability.

To summarize, we bring to scientists the ability to understand exactly what spectral properties we consider mathematically and how we expect those properties to behave on an incredibly detailed level (the 4 line parameters of all spectral lines and their corresponding interpolating manifolds), and introduce the concept of postulating one precomputed model grid as the ground truth to base all further analysis on, converting the grid into a generator that can then interface with our inference algorithm.

5.2. *Technical Considerations*

To reiterate, the manifold fitting steps are not end-to-end autodifferentiable. As seen in Figure 1, these steps rely on `scipy`, which is not equipped with autodiff. Without autodifferentiability, the ability to "machine learn" is significantly reduced compared to a hypothetical monolithic JAX or PyTorch system. We see three reasons for for developing a non-autodifferentiable system. First, the familiar `scipy`-based system will serve as an easy entry point for most practitioners who are unfamiliar with autodiff, and may still benefit from and modify the code without expert ML knowledge. Second, this non-autodiff version serves as a**n initial** benchmark against which an inevitable autodifferentiable version may be compared. Finally, the inventory of familiar interpolation algorithms have not yet been ported to PyTorch or JAX, since machine learning or Gaussian Process fitting schemes are generally preferred within the ML community.

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 parameter, 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,

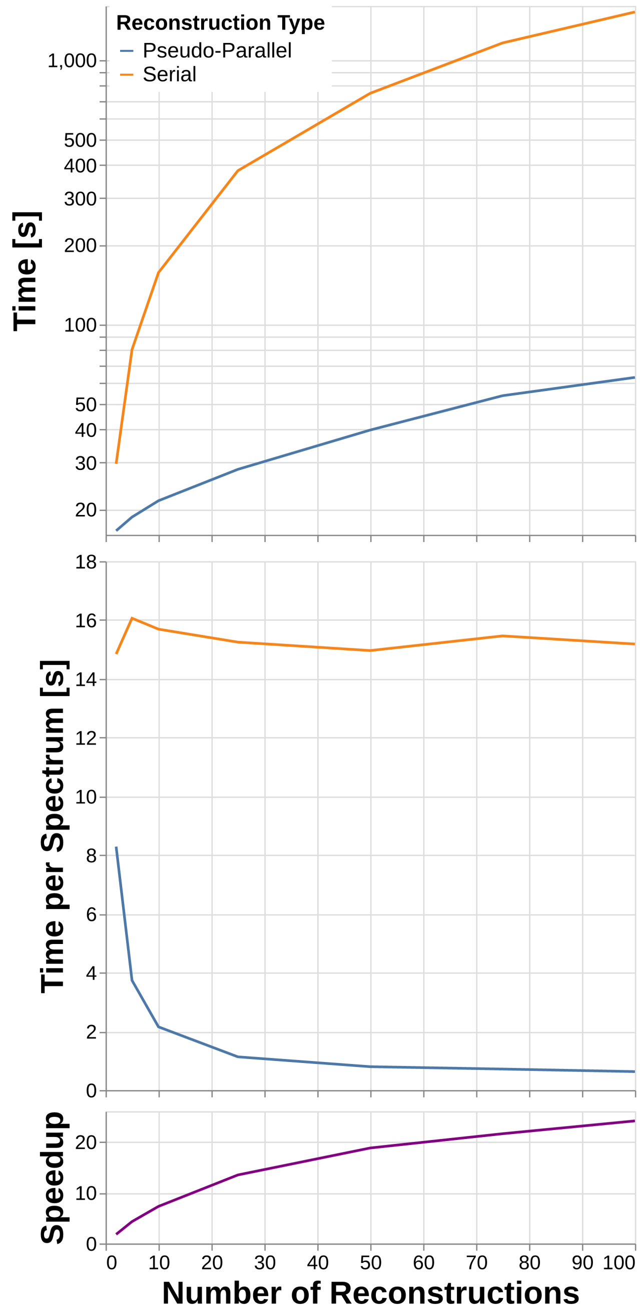

Figure 8. Plots showing the time taken to reconstruct varying numbers of spectra using the PHOENIX generator (lower is better). We can see that the time taken per spectr**um** for the serial implementation hovers around 15 seconds within run-to-run variance, while the pseudo-parallel implementation continually 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.

with the full [500, 55000] Å wavelength range also being left to future work. Users would be able to fit a greater variety of stellar spectra in many different wavelength regimes.

- *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. This takes burden off the users to truncate their data to the PHOENIX generators, making use simpler.

- *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 **(Marley et al. 2021; Karalidi et al. 2021; Morley et al. 2024; Mukherjee et al. 2024)**, ATLAS (Kurucz 2005), and coolTLUSTY (Lacy & Burrows 2023), reaching practitioners studying various types of stars and spectra. **Future `blase` versions will** be able 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 generalizable ML** methods **such as lasso or ridge regression**, which would allow for much better generalization, high speed, 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, enabling users to optimize these frequently-observed extrinsic parameters on top of the base 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. This has the potential to greatly increase the speed of this algorithm, depending on how much overhead is avoided with a bespoke implementation.

- *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 we were able to identify the spectral lines used in our figures, it is not necessarily valuable to try to identify all 128,723 lines that we identify as unique with our algorithm. `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. Having more interpretability would let scientists actually study certain species and their spectral lines.

- *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 continu**a** and develop more advanced methods that can discern continu**a** with greater accuracy and less modeling restrictions. This would increase accuracy for end users.

- *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. This would increase accuracy for end users.

6. CONCLUSION

In this study, we have presented a proof-of-concept algorithm that interpolates a subset of the PHOENIX spectral model grid and then uses GP minimization **to infer stellar paremeters**. We create state dictionaries for all spectra in the PHOENIX subset, lossily distilling the spectra with a data compression factor of around 20. The**y are available on** Zenodo at https://zenodo.org/records/11246174 (Shankar 2024). We create the PHOENIX generator and implement a performant spectral reconstruction algorithm, enabling anyone to create reconstructions of PHOENIX spectra with continuously valued stellar parameters. We introduce and test our inference algorithm, with average absolute deviations from true values of 185 K in $T_{\rm eff}$, 0.19 in $\log(g)$, and 0.12 dex in [Fe/H]. In its current state, our algorithm operates on spectra within the PHOENIX subset parameter ranges in Table 1, requiring that the spectra not contain noticeable Doppler shifting, rotational broadening, or other confounding factors. The methods discussed here represent a step down a road not traveled in spectral inference, and have the potential to become more advanced in the future by fully utilizing the strengths of physics-informed machine learning.

This material is based on work supported by the National Aeronautics and Space Administration under grant No. 80NSSC21K0650 for the NNH20ZDA001N-ADAP:D.2 program. C.V.M. acknowledges support from the Alfred P. Sloan Foundation under grant number FG-2021-16592 and support from the National Science Foundation under grant number 1910969.

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

Horta, D., Price-Whelan, A. M., Hogg, D. W., Ness, M. K., & Casey, A. R. 2025, arXiv e-prints, arXiv:2502.01745. https://arxiv.org/abs/2502.01745

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

Kramida, A., Yu. Ralchenko, Reader, J., & and NIST ASD Team. 2023, NIST Atomic Spectra Database (ver. 5.11), [Online]. Available: `https://physics.nist.gov/asd` [2024, June 12]. National Institute of Standards and Technology, Gaithersburg, MD.

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

Kurucz, R. L. 2005, Memorie della Societa Astronomica Italiana Supplementi, 8, 14

Lacy, B., & Burrows, A. 2023, The Astrophysical Journal, 950, 8, doi: 10.3847/1538-4357/acc8cb

Leung, H. W., & Bovy, J. 2019, MNRAS, 483, 3255, doi: 10.1093/mnras/sty3217

—. 2024, MNRAS, 527, 1494, doi: 10.1093/mnras/stad3015

López-Valdivia, R., Mace, G. N., Han, E., et al. 2023, ApJ, 943, 49, doi: 10.3847/1538-4357/acab04

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

Rains, A. D., Nordlander, T., Monty, S., et al. 2024, MNRAS, 529, 3171, doi: 10.1093/mnras/stae560

Rayner, J. T., Cushing, M. C., & Vacca, W. D. 2009, ApJS, 185, 289, doi: 10.1088/0067-0049/185/2/289

Różański, T., Ting, Y.-S., & Jabłońska, M. 2023, arXiv e-prints, arXiv:2306.15703, doi: 10.48550/arXiv.2306.15703

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

Shankar, S. 2024, blase II: PHOENIX Subset Clone Archive, 0.0.1, Zenodo, doi: 10.5281/zenodo.11246174

Shankar, S., Gully-Santiago, M., Morley, C., et al. 2024, The Journal of Open Source Software, 9, 6601, doi: 10.21105/joss.06601

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

Wheeler, A. J., Abruzzo, M. W., Casey, A. R., & Ness, M. K. 2023, AJ, 165, 68, doi: 10.3847/1538-3881/acaaad