MNRAS **000**, 1–14 (2020) Preprint 7 February 2021 Compiled using MNRAS LATEX style file v3.0

Modelling stars with Gaussian Process Regression – I: Augmenting Stellar Model Grid

Tanda Li,[1]★ Guy R. Davies,[1]† Alex Lyttle,[1] Lindsey Carboneau,[1] and A. N. Others[1]
[1] *School of Physics and Astronomy, University of Birmingham, Birmingham, B15 2TT, United Kingdom*

Accepted XXX. Received YYY; in original form ZZZ

ABSTRACT

Grid-based modelling is widely used for estimating stellar parameters. However, stellar model grid is sparse because of the computational cost. This paper demonstrates an application of Gaussian Process (GP) Regression that turns a sparse model grid to a continuous function. We train GP models to map five fundamental inputs (mass, equivalent evolutionary phase, initial metallicity, initial helium fraction, and the mixing-length parameter) to observable outputs (effective temperature, surface gravity, radius, surface metallicity, and stellar age), aiming to augment the stellar grid. We preliminarily test different training approaches with a small subset of stellar model grid and set up the GP models with the most promising methods. To overcome the limitation of training data size of GP framework, we section the whole grid into 10 segment and train each section separately. An off-grid stellar model dataset is then used to test GP models. Mean differences between GP predictions and testing data are approximately zero for all outputs. The standard deviation of global testing errors is $\sim 2K$ for effective temperature, $\sim 0.001dex$ for surface gravity, $\sim 0.002R_{\odot}$ for radius, $\sim 0.001dex$ for surface metallicity, and $\sim 0.02Gyr$ for stellar age. However, we found that local standard deviation is not uniform across the parameter space and it mainly varies with mass, equivalent evolutionary phase, and initial metallicity. We hence train another GP model to describe the systematic uncertainties. We lastly model 100 fake stars to validate the accuracy of GP models. Inferred masses and ages are well consistent with true values within 1 standard deviation. We also note that GP model gives more sensible statistical results than grid-based modelling while estimated stellar parameters because fundamental inputs are continuously sampled.

Key words: Star: Modelling – Machine Learning – keywords

1 INTRODUCTION

Theoretical stellar model has been developed for decades to simulate star structure and evolution. However, modelling of stars are mostly carried out with sparse model grid (e.g. Choi et al. 2016) because massive computation can be time-consuming. Moreover, stellar model contents adjusted input parameters (e.g. the mixing-length parameter). Varying one of these adjusted parameter adds on an input demission and hence exponentially increasing the computational cost. A comprehensive and fine stellar model grid is hence expensive.

When modelling star, a sparse grid is not ideal for statistics. Classical method like interpolation is one option to overcome this disadvantage. For instance, Dotter (2016) developed a method to transform stellar evolution tracks onto a uniform basis and then interpolate to construct stellar isochrones. More recently, Rendle et al. (2019) uses Bayesian statistics and a Markov Chain Monte Carlo approach to find a representative set of interpolated models from a grid. Both works achieve good accuracy for 3-demission gird (inputs are mass, age, and metallicity). However, the interpolation approach become less reliable in higher demissions and it hence limits the variety of input physics. The algorithm is another approach that has been used in stellar codes (e.g. Paxton et al. 2013). It offers an automated likelihood minimisation to search for optimal solutions. This method is statistically sound and works fairly well for modelling individual stars. However, it become much less efficient for modelling a large sample, because the algorithm needs to repeat computing stellar tracks many time for each single star.

Machine learning is being applied to stellar researches recently. Hon et al. (2018) developed a convolutional neural network classifier for solar oscillations in red giants. Wu et al. (2019) determined masses and ages for massive RGB stars from their spectra with a machine-learning method based on kernel principal component analysis. (**Guy: can you add more relevant papers?**)

A machine-learning algorithm that involves a Gaussian process (GP) measures the similarity between data points (the kernel

★ E-mail: t.li.2@bham.ac.uk
† E-mail: G.R.Davies@bham.ac.uk

Table 1. Computation of Stellar model grid.

Primary Grid		
Input Parameter	Range	Increment
M ($M_\odot$)	0.80 – 1.20	0.01
[Fe/H] (dex)	-0.5 – 0.2/0.2 – 0.5	0.1/0.05
$Y_{\rm init}$	0.24 – 0.32	0.02
$\alpha_{\rm MLT}$	1.7 – 2.5	0.2
Additional Grid		
Input Parameter	Range	Increment
M ($M_\odot$)	1.055 – 1.195	0.01
[Fe/H] (dex)	0.25 – 0.45	0.1
$Y_{\rm init}$	0.25 – 0.31	0.02
$\alpha_{\rm MLT}$	1.8– 2.4	0.2
Off-grid Models		
Input Parameters		N
Random M, [Fe/H]$_{\rm init}$ = 0.0, $Y_{\rm init}$ = 0.28, $\alpha_{\rm MLT}$ = 2.1		44
Random M and [Fe/H]$_{\rm init}$, $Y_{\rm init}$ = 0.28, $\alpha_{\rm MLT}$ = 2.1		174
Random M, [Fe/H]$_{\rm init}$, $Y_{\rm init}$, and $\alpha_{\rm MLT}$		4880

function) to predict values for unseen points from training data. **more intro about GP**.

The aim of this paper is training GP models to turn a sparse model grid into a continuous function and augmenting the model grid. The paper is organised as follow. We describe the computation of a representative stellar grid in Section 2. We then introduce the underline theory of GP and set up GP models for training the stellar grid in Section 3. Section 4 demonstrates the results of GP predictions and we analyse the systematic uncertainties of GP models. We subsequently augment the stellar grid and transform it to a set of continuously-sampled models, and we model 100 fake stars with GP-trained models for testing the accuracy of our method. Finally we discuss the limitations of this approach, highlight areas where improvements can be found in the near future, and summary conclusions in Section 6.

2 THEORETICAL MODEL GRID

We compute a stellar model grid as the training dataset. We aim to cover stars with approximate solar mass on the main-sequence and the subgiant phases. Thus, the mass range is set up as 0.8 – 1.2$M_\odot$. The computation of evolutionary tracks starts at the Hayashi line and terminates at the base of red-giant branch (RGB) where $\log g$ = 3.6 dex. Note that we only use models after the zero-age-main-sequence (ZAMS). We define ZAMS as the point where core-hydrogen burning contributes over 99.9% of total luminosity. The stellar model gird considers four independent fundamental inputs which are stellar mass (M), initial helium fraction ($Y_{\rm init}$), initial metallicity ([Fe/H]$_{\rm init}$), and the mixing-length parameter ($\alpha_{\rm MLT}$). We calculated three model grids. The primary grid covers the whole input range. Uniform grid step is applied for M, $Y_{\rm init}$), $\alpha_{\rm MLT}$, and we used tow different grid steps for [Fe/H]$_{\rm init}$. An additional grid is computed in between the primary grid points for $M > 1.05 M_\odot$. This is to increase the resolution for tracks with the 'hook'. Lastly, we computed off-grid models as an independent dataset for validating and testing the results. Details of the computation are listed in Table 1.

We use the stellar code Modules for Experiments in Stellar Astrophysics (MESA, version 12115) for computing the stellar models. MESA is an open-source stellar evolution package which is undergoing active development. Descriptions of input physics and numerical methods can be found in Paxton et al. (2011, 2013, 2015). We adopted the solar chemical mixture [$(Z/X)_\odot$ = 0.0181] provided by Asplund et al. (2009). The initial helium fraction ($Y_{\rm init}$) and initial metallicity ([Fe/H]$_{\rm init}$) are independent inputs. The initial chemical composition is calculated with

$$\log(Z_{\rm init}/X_{\rm init}) = \log(Z/X)_\odot + [{\rm Fe/H}]_{\rm init}. \quad (1)$$

We used the MESA $\rho - T$ tables based on the 2005 update of OPAL EOS tables (Rogers & Nayfonov 2002) and OPAL opacity supplemented by low-temperature opacity (Ferguson et al. 2005). The grey Eddington $T - \tau$ relation is used to determine boundary conditions for modelling the atmosphere. The mixing-length theory is implemented and the convection is adjusted by the mixing-length parameter ($\alpha_{\rm MLT}$). We also apply the MESA predictive mixing scheme (Paxton et al. 2018, 2019), which improves model structures at the convective boundary. Atomic diffusion of helium and heavy elements was also taken into account. MESA calculates particle diffusion and gravitational settling by solving Burger's equations using the method and diffusion coefficients of Thoul et al. (1994). We considered eight elements (^{1}H, ^{3}He, ^{4}He, ^{12}C, ^{14}N, ^{16}O, ^{20}Ne, and ^{24}Mg) for diffusion calculations, and had the charge calculated by the MESA ionization module, which estimates the typical ionic charge as a function of T, ρ, and free electrons per nucleon from Paquette et al. (1986). The MESA inlist used for the computation is available on `https://github.com/litanda/mesa_inlist`.

3 GAUSSIAN PROCESS MODEL

GP can be applied as probabilistic models to regression problems. Here we will use the GP model to generalise a grid of stellar models to a continuous and probabilistic function that maps inputs (i.e., initial mass, chemical composition, etc.) to observable quantities (i.e., effective temperature, surface gravity, radius, etc.). We aim to use the GP model as a non-parametric emulator, that is emulating the comparatively slow calls to models of stellar evolution. We adopt the a tool package named GPyTorch, which is a GP framework developed by Gardner et al. (2018). GPyTorch is a Gaussian process library based on an open source machine learning framework PyTorch (`https://pytorch.org`). This tool package provides significant GPU acceleration, state-of-the-art implementations of the latest algorithmic advances for scalability and flexibility, and easy integration with deep learning frameworks. Source codes and detailed introductions could be found on `https://gpytorch.ai`.

3.1 Gaussian Process Application

We start with a grid of stellar models containing N models with a label we want to learn, for example model effective temperature, which we will denote with the general symbol $\mathbf{y}$, and a set of input labels $\mathbf{X}$ (e.g., mass, age, and metallicity). We can use a GP to make predictions of the effective temperature (or y) for additional input values given by $\mathbf{X}_\star$. The vector $\mathbf{y}$ is arranged $\mathbf{y} = (y_i, ..., y_N)^T$ where the subscript label references the stellar model. The input labels are arranged into a $N \times D$ matrix where D is the number of input dimensions (e.g., D = 3 for mass, age, and metallicity) so that $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N)^T$ where $\mathbf{x_i} = (x_{1,i}, ..., x_{D,i})^T$. The matrix of additional inputs $\mathbf{X}_\star$ has the same form as $\mathbf{X}$ but size $N_\star \times D$.

Williams & Rasmussen (1996), from which our description below is based, define a GP as a collection of random variables, where any finite number of which have a joint Gaussian distribution. In general terms, GP's (**Guy, please check " 's "**) may be written as

$$y(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), \mathbf{\Sigma}\right), \tag{2}$$

where $m(\mathbf{x})$ is some mean function, and $\mathbf{\Sigma}$ is some covariance matrix. The mean function controls the deterministic part of the regression and the covariance controls the stochastic part. The mean function defined here could be any deterministic function and we will label the additional parameters, or hyperparameters, ϕ. Each element of the covariance matrix is defined by the covariance function or *kernel function* k which has hyperparameters θ and is given by,

$$\mathbf{\Sigma}_{n,m} = k(\mathbf{x}_n, \mathbf{x}_m), \tag{3}$$

where the inputs $\mathbf{x}_i$ are D-dimensional vectors and the output is a scalar covariance.

As a GP is a collection of random variables, where any finite number of which have a joint Gaussian distribution, the joint probability of our data $\mathbf{y}$ is

$$p(\mathbf{y}|\mathbf{X}, \phi, \theta) = \mathcal{N}(m(\mathbf{X}), \mathbf{\Sigma}). \tag{4}$$

If we want to obtain predictive distributions for the output $\mathbf{y}_\star$ given the inputs $\mathbf{X}_\star$ the joint probability distribution of $\mathbf{y}$ and $\mathbf{y}_\star$ is Gaussian and given by

$$p\left(\begin{bmatrix}\mathbf{y}\\ \mathbf{y}_\star\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{X}\\ \mathbf{X}_\star\end{bmatrix}, \begin{bmatrix}\mathbf{\Sigma} & \mathbf{K}_\star\\ \mathbf{K}_\star{}^T & \mathbf{K}_{\star\star}\end{bmatrix}\right), \tag{5}$$

where the covariance matrices $\mathbf{\Sigma}$ and $\mathbf{K}$ are computed using the kernel function so that

$$\mathbf{\Sigma}_{n,m} = k(\mathbf{X}_n, \mathbf{X}_m), \tag{6}$$

which is an $N \times N$ matrix.

$$\mathbf{K}_{\star\, n,m} = k(\mathbf{X}_n, \mathbf{X}_{\star\, m}), \tag{7}$$

which is an $N \times N_\star$ matrix, and finally

$$\mathbf{K}_{\star\star\, n,m} = k(\mathbf{X}_{\star\, n}, \mathbf{X}_{\star\, m}), \tag{8}$$

which is an $N_\star \times N_\star$ matrix. The predictions of $\mathbf{y}_\star$ are again a Gaussian distribution so that,

$$\mathbf{y}_\star \sim \mathcal{N}(\hat{\mathbf{y}}_\star, \mathbf{C}), \tag{9}$$

where

$$\hat{\mathbf{y}}_\star = m(\mathbf{X}_\star) + \mathbf{K}_\star^T\, \mathbf{\Sigma}^{-1}\, (\mathbf{y} - m(\mathbf{X})), \tag{10}$$

and

$$\mathbf{C} = \mathbf{K}_{\star\star} - \mathbf{K}_\star^T\, \mathbf{\Sigma}^{-1}\, \mathbf{K}_\star. \tag{11}$$

At point we can make predictions on model properties given a grid of stellar models using equation 9. But these predictions will be poor unless we select sensible values for the form and hyperparameters of the mean function and covariance function. In the following section we detail a number of kernel functions that will be tested against the data. We will then discuss the method for determining the values of the hyperparameters to be used.

3.2 GP Model Inputs and Outputs

We aim to derive stellar parameters based on fundamental inputs of the model grid. As mentioned in Section 2, our model grid has four independent fundamental inputs, i.e., mass, initial metallicity, initial helium fraction, and mixing-length parameter. Moreover, we need another input to describe the evolution and hence the GP model contents five input demissions.

Regarding the evolution index, stellar age is not ideal because its dynamical range varies track by track. The fractional age is a choice. However, we find global parameters (e.g. effective temperature) sharply change as a function of fractional age around the 'hook' and the turn-off point as shown in the left panel of Figure 1. This is because stars evolve in a relatively short time scale at these stages. We hence use an Equivalent Evolutionary Phase (EEP) following Dotter (2016) as the five fundamental input. Note that we define EEP in a different way from Dotter (2016). On each evolutionary track, we compute the displacement between model n and model $n-1$ on the $T_{\rm eff} - \log g$ diagram as

$$\delta d_n = ((T_{\rm eff,n} - T_{\rm eff,n-1})^2 + (\log g_n - \log g_{n-1})^2))^f, \tag{12}$$

and the total displacement of model n from the ZAMS (model 0) can be calculated with

$$d_n = \sum_{i=0}^{i=n} \delta d_i. \tag{13}$$

On the same evolutionary track, d_n is normalised to the 0 –1 range and defined as defined as EEP. For our stellar model grid, EEP equals to 0 at the ZAMS and 1 on the RGB where log = 3.6 dex. The factor f in Eq. 12 is adjustable for modulating EEP to avoid obviours gap in the data space, and we find that $f = 0.18$ gives the best data distribution. In Figure 1, we demonstrate how the effective temperature changes with fractional age and EEP. It can be seen that the usage of EEP significantly smoothes the sharp features at the 'hook' and the turn-off point. We summary GP model inputs and outputs as below.

- GP model inputs and their dynamic ranges:
  Mass (M = 0.8 – 1.2M$_\odot$)
  Equivalent Evolutionary Phase (EEP = 0 – 1)
  Initial metallicity ([Fe/H]$_{\rm init}$ = -0.5 – 0.5 dex)
  Initial helium fraction ($Y_{\rm init}$ = 0.24 – 0.32)
  Mixing-length parameter ($\alpha_{\rm MLT}$ = 1.7 – 2.5)
- GPR model outputs:
  Effective temperature ($T_{\rm eff}$)
  Surface gravity ($\log g$)
  Radius (R)
  Surface metallicity ([Fe/H]$_{\rm surf}$)
  Stellar age (τ)

Thus, the GP model can be described as

$$\text{Outputs} = f(M, EEP, (\text{Fe/H})_{\rm init}, Y_{\rm init}, \alpha_{\rm MLT}). \tag{14}$$

3.3 GP training Procedure

3.3.1 Data Selection

For training GP models properly, we need three types of data for training, validating, and testing GP models. Models in the primary and additional grids (as described in Table 1) are the training data. The off-grid models are divided 50-to-50 as validating and testing

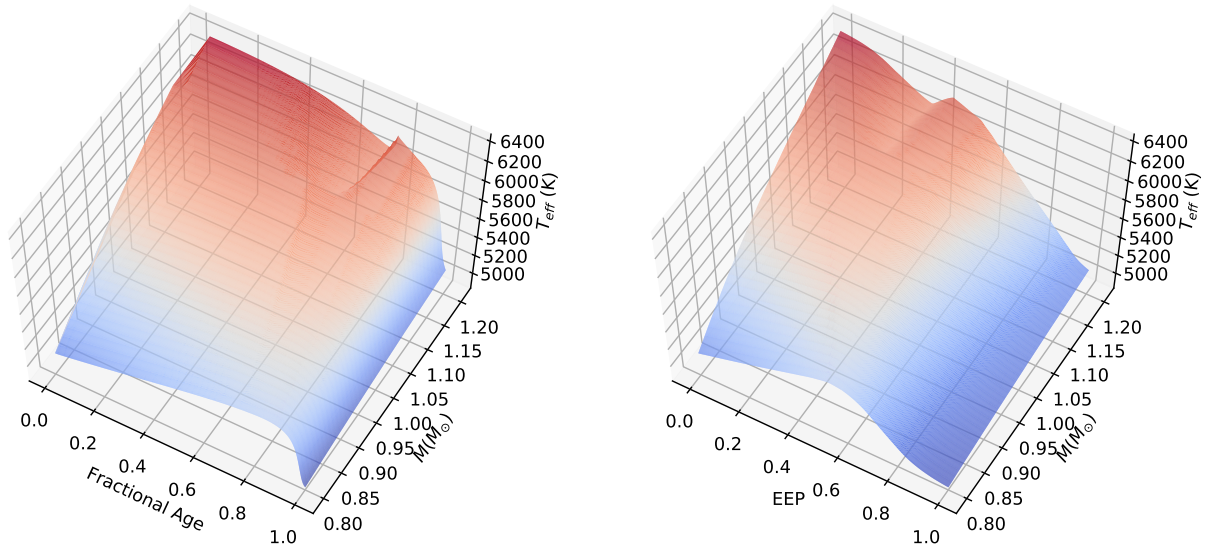

Figure 1. Surface plots of model effective temperature on the mass-fractional age (left) and mass-EEP (right) diagrams. Models in this figure are from the primary grid with fixed initial metallicity ($[\mathrm{Fe/H}]_{\mathrm{init}} = 0.0$), helium fraction ($Y_{\mathrm{init}} = 0.28$) and mixing-length parameter ($\alpha_{\mathrm{MLT}} = 2.1$). It can be seen that the effective temperature changes much smoother on the mass-EEP diagram at the hook and turn-off points.

datasets . Note that validating and testing datasets are not on the same evolutionary tracks so they are independent to each other.

GP has limitation of the data size, because the computational and memory complexity exponentially increase with the number of data points. In practice, typical data size for GP is on an order of 10^4. Given that the grid contents $\sim 10,000,000$ models, only a small subset can be used. The sampling method is hence critical. The evolve step is not the same at different evolutionary stages due to the MESA step-control strategy. For instance, stellar models are dense at the main-sequence and the red-giant phases but quite sparse on the subgiant stage. For this reason, a flat sampling is not appropriate. We test a few methods and find that using the displacement (δd_n) defined in Eq. 12 as the weight of sampling give a relatively uniform data distribution in the parameter space.

3.3.2 Training, Validating, and Testing GP Models

The procedure contents two stages: training and testing. Training is the process that optimises hyperparameters with training dataset and validates the trained GP model with validating dataset. When training is completed, we test the GP model with testing dataset to quantify its accuracy. In the training process, we train and validate the GP model in every iteration. The model will be saved if it has the best validation result so far. The training process terminates when the validating result does not improve in the last 300 iterations. After training, we test the best GP model with the testing dataset. Details of the set up of training will be described in the following Section.

Here we discuss the method for validating and testing a GP model. We do not use any popular methods, such like Root Mean Square Error (RMSE). Because we find that GP model performances are at similar accuracy level across the whole parameter space due to the differences in evolutionary features. We illustrate this with an example in Figure 2. As it can be seen that, we train a GP model which maps M and EEP to the effective temperature. The kernel function in the area of $M \geq 1.05\mathrm{M}_\odot$ and $EEP \leq 0.7$ is more complex than that for other regions because of the appearance of the 'hook'. This particular area are relatively difficult to learn and hence poorly predicted by the GP model. For other outputs, surface gravity and radius are similar to the case of effective temperature. The surface metallicity is not significantly affected by the hook, but it has a quick raise at the early subgiant phase in high-mass tracks. This is because high-mass tracks maintain shallow convective envelope and hence have strong diffusion effect during the main-sequence stage. Their surface metallicities at then end of the main-sequence are generally much lower than the initial value. When evolving to the early subgiant phase, the quick expansion of the surface convective envelope brings back the settling heavy elements to the surface. This results in a sharp raise of the surface metallicity after the turn-off point and GP models show relatively poor accuracy at this phase.

When there is a subregion where a GP model perform worse than other areas, the error distribution is not Gaussian-like and hence a global estimator like RMSE is not suitable. What we need here is a simple method that reflects the general accuracy as well as the worst case. We examine the error distribution as shown at the bottom of Figure 2. Errors of most data follows a Gaussian distribution but about 10% data form long tails on both sides. For this case, the 68% confidential interval is able to describe the accuracy of most data, and the 95% and 99.7% confidential intervals describe the median value and the length of the tail. Thus, we use the sum of three cumulative values at 68%, 95%, and 99.7% of absolute errors as an error index to qualify GP models. For the case in Figure 2, cumulative values at 68%, 95%, and 99.7% are 1.1, 4.9, and 11.1K, which give an error index equals to 17.1K.

3.4 Preliminary Tests and Settings of Training

For training GP models, we need set up mean function, kernel, likelihood function, loss function, and optimiser. We do numbers of primary tests for comparing different options for the above elements and decide which to use. The primary tests are done on 2-demission (2D) GP models that content two inputs M and EEP. The 2D GP models can be described as Outputs = $f(M, EEP)$. Training data for 2D GP models are selected from the primary grid with fixed $[\mathrm{Fe/H}]_{\mathrm{init}}$ (0.0), Y_{init} (0.28), and α_{MLT} (2.1). The

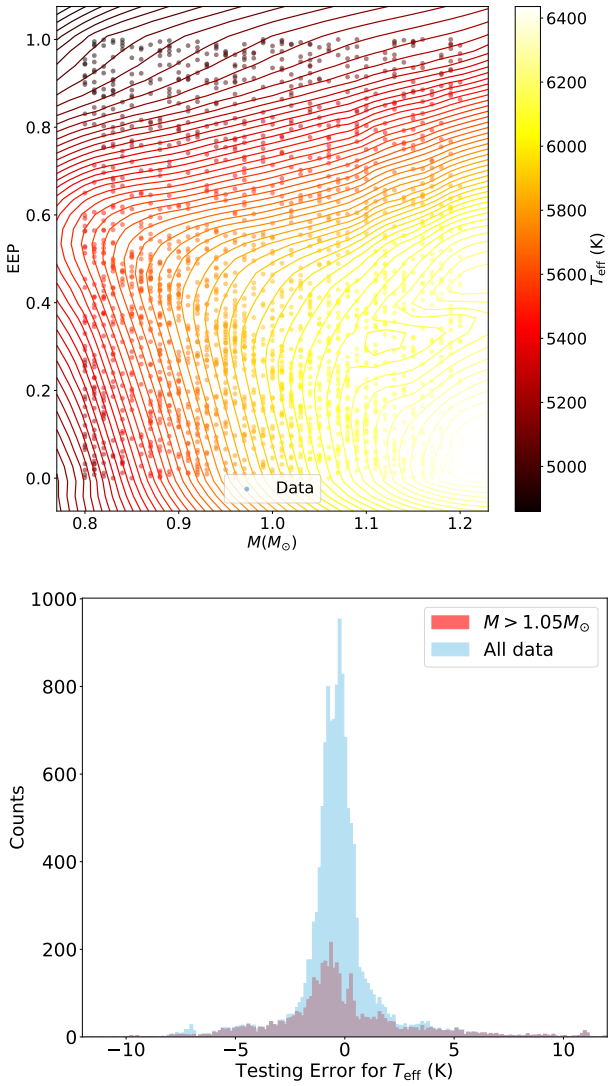

Figure 2. Top: The 2D GP model for $T_{\rm eff}$. Bottom: probability distributions of validating errors of the GP model.

training dataset contents 41 evolutionary track and 24,257 data points. We also computed 44 evolutionary tracks with same input $[{\rm Fe/H}]_{\rm init}$, $Y_{\rm init}$, and $\alpha_{\rm MLT}$ but random M for validating and testing. We use the sampling method mentioned in previous section and selected 20,000 data points for training, 10,000 for validating, and 10,000 for testing. The Exact GP approach is adopted and we test and develop our training training script based on the Simple GP Regression example in the GpyTorch package (https://docs.gpytorch.ai/en/stable/examples/01_Exact_GPs/Simple_GP_Regression.html).

3.4.1 *Mean Function*

We start with the mean function. As discussed above, evolutions of model outputs are not always smooth and can be complex in some areas. Although a GP model does not significantly affected by the choice of the mean function due to the flexibility of kernels, we find that a constant or linear mean function leads to a long time for training and a significant increase of the complexity of kernels. For this reason, we apply a Neural Network mean function which is flexible enough to map smooth as well as curved evolving features. We adopt an architecture includes 6 hidden layers and 128 nodes per layer. All layers apply the linear transformation to the incoming data. We apply the element-wise function (Elu) because it give relatively smooth mean functions. (**may need ref for NN**)

3.4.2 *Likelihood and Loss Function*

We then work on the likelihood function and the loss function. Our training object is a theoretical model grid. There is hence no observed uncertainty for each data point, but a tiny random uncertainty exists due to the approximations in the MESA numerical method. The noise model can be assumed as a Gaussian function with a very small deviation. A likelihood specifies the mapping from latent function values $f(X)$ to observed labels y. We adopt the the standard likelihood for regression which assumes a standard homoskedastic noise model whose conditional distribution is

$$p(y|f(x)) = f + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma^2), \tag{15}$$

where σ is a noise parameter. Given the random uncertainty is small, we use a small and fixed noise parameter and run a few tests. However, we find that it makes the GP models hard to converge and sometimes lead to obvious overfitting. When this noise parameter is set as free, it reduces to a small number anyway in the training progress because it is data-driven. For these reasons, we decide not put strict constraint for or prioritise this noise parameter and let the data determine. In practice, we only set up a loose upper limit ($\sigma < 0.1$) for the noise parameter to speed up the training. One thing should be note that a GP model with a large noise parameter can not be a proper description for the stellar grid even if it gives good validating or testing errors. Because of this, we only adopt GP models with a noise parameter $\lesssim 10^{-4}$. The the loss function is simply the exact marginal likelihood in the logarithmic scale.

3.4.3 *Optimiser*

Afterwards, we run tests to decide the optimiser. We mainly compare two optimisers named SGD and Adam. Here SGD refers to Stochastic Gradient Descent, and Adam is a combination of the advantages of two other extensions of stochastic gradient descent, specifically, Adaptive Gradient Algorithm and Root Mean Square Propagation. The SGD optimiser in the PyTroch package involves the formula given by Sutskever et al. (2013). The formula makes it possible to train using stochastic gradient descent with momentum thanks to a well-designed random initialisation and a particular type of slowly increasing schedule for the momentum parameter. The application of momentum in SGD could improve its efficiency and make it less likely to stuck in local minimums. On the other hand, the Adam optimiser includes the AMSGrad variant developed by Reddi et al. (2018) to improve its weakness in the convergence to an optimal solution. With these new developments, the two optimisers give very similar results. We finally choose Adam because it works relatively efficiently and stable. We adaptive learning rate instead of a fixed value. Our training starts with a learning rate of 0.01 and decreases by a factor of 2 when the loss value does not reduces in the previous 100 iterations.

3.4.4 Early stopping

We set up an early stopping for tow purpose: avoiding overfitting and terminating the training progress. The early stopping is controlled by validating results. When an optimal solution is found, the validating errors stop decreasing and then increase when overfitting happens. In the training process, we track the validation error index (introduced in Section 3.3.2) trigger the early stopping when the validation error index does not decrease in the previous 300 iterations.

3.4.5 GP kernels

With the above set up, we lastly test for the best kernel for this work. We involved four basic kernels in the tests as listed below:

- RBF: Radial Basis Function kernel (also known as squared exponential kernel)
- RQ: Rational Quadratic Kernel (equivalent to adding together many RBF kernels with different lengthscales)
- Mat12: Matern 1/2 kernel (equivalent to the Exponential Kernel)
- Mat32: Matern 3/2 kernel

We applied each basic kernel and a couple combined kernels (RBF + Mat21, RQ + Mat21, Mat32 + Mat21, RBF + Mat32, RQ + Mat32) to train the 2D GP models. Among these kernels, the combined one RBF+Mat21 give the best fit to the training data, however, it does not give the best predictions for the validating and testing data. On the other hand, the GP model with the Mat32 kernel fit the training data reasonably well and its predictions have the best agreement with validating and the testing data. Comparing between the two kernels, the RBF+Mat12 Kernel is a combination of a smooth and a spiky kernel, which offers enough flexibility fit all features in the training data. However, the spiky function do not work well for off-grid regions. The smoothness of Mat32 kernel is somewhere between a spiky (Mat21) and a smooth kernel (RBF). It does not exactly fit some sharp features of the data but performs a good balance between on- and off-grid data. We hence adopted Mat32 kernels in our GP model.

3.5 Strategy for Large sample

The model grid we aim to train contents about 10,000,000 data points, which is much more than the upper limit of data size of the Exact GP (20,000). We hence consider other GP approaches. Our tests for the large-sample strategy are carried out with a 3-demission (3D) GP model, which have three fundamental inputs: M and EEP and $[\mathrm{Fe/H}]_{\mathrm{init}}$. The 3D model can be described as (Outputs = $f(M, EEP, [\mathrm{Fe/H}]_{\mathrm{init}})$). Data for training 3D models (3D data hereafter) are from the primary grid with fixed Y_{init} (0.28), and α_{MLT} (2.1). The training sample include 562 tracks and $\sim$ 300,000 data points. For validating and testing purposes, we compute another 174 evolutionary tracks with the same input Y_{init}, and α_{MLT} but random M and $[\mathrm{Fe/H}]_{\mathrm{init}}$. We firstly train an Exact GP model which includes 20,000 training data as the reference.

We then consider the Stochastic Variational GP (SVGP) with GPyTorch ApproximateGP module. We train our data based on the SVGP example on https://docs.gpytorch.ai/en/v1.1.1/examples/04_Variational_and_Approximate_GPs/SVGP_Regression_CUDA.html. SVGP is an approximate scheme rely on the use of a series of inducing points which can be selected in the parameter space. It trains using minibatches on the training dataset and build up kernels on the inducing numbers. Underline principles and detailed descriptions of this approach can be found in Hensman et al. (2014). The advantage of SVGP is the large capacity of sample size, however, the kernel complexities is still limited by the amount of inducing numbers. Moreover, because more training data is loaded and take off the memory, the SVGP model can only use 10,000 inducing points. This is to say, although more training data is used, the kernel complexity of the SVGP model is even simpler than the Exact GP model. As a result, the SVGP model does not improve the accuracy level. For instance, the testing errors of T_{eff} at 68%, 95%, and 99.7% are 2.0, 5.8, and 15.7 K (error index = 23.5 K) for Exact GP model and 2.2, 6.8, and 15.1 (error index = 24.1 K) for the SVGP model. Here we learn that SVGP is suitable for the case of large data sample but simple kernel complexity.

We investigate another approach designed for the large dataset named Structured Kernel Interpolation (SKI GP). SKI GP was introduced by Wilson & Nickisch (2015). It produces kernel approximations for fast computations through kernel interpolation and is a great way to scale a GP up to very large datasets (100,000+ data points). We follow the example on https://docs.gpytorch.ai/en/stable/examples/02_Scalable_Exact_GPs/KISSGP_Regression.html to develop our script. We run a few tests to train a 3D SKI GP model with 100, 000 training data. Compare with the Exact GP and SVGP, its testing errors of T_{eff} are slightly improved to 2.0, 6.1, and 14.8 K (error index = 22.9 K). However, the further test on the 5-demission data is not ideal: a SKI GP model with 100,000 training data performs much worse than an Exact GP model with 20,000 training data. The poor behaviour of the 5D model consist with what has been discussed in Wilson & Nickisch (2015). The method poorly scale to data with high dimensions, since the cost of creating the grid grows exponentially in the amount of data. We attempt to make some additional approximations with the GPyTorch AdditiveStructureKernel module. It makes the base kernel to act as one-dimension kernels on each data dimension and the final kernel matrix will be a sum of these 1D kernel matrices. However, the testing errors are not significantly improved.

As mentioned above, the GPU memory captivity limits the actual number of data that induce the kernel function. This limit become critical for high-demission cases, because the parameter space exponentially increases with model demission and hence GP model accuracy inevitable declines. Improving GP models requires an increase of data points to induce kernels. A simple way is breaking the grid into sections and train GP models for each section separately. Here we test how this section scenario works for 3D GP models. We divide the dataset into 10 equal segments by EEP. We train one Exact GP model with 20,000 training data for each section. We then use the same testing dataset to quantify GP predictions for the five outputs, and the error index are averagely improved by around 10%. For instance, the error index of T_{eff} decreases from 23.5 to 21.6 (1.7/5.0/14.9 K at 68/95/99.7%).

As expected, the section scenario improves GP models, but there is a major concern about the edge effects at the boundary between sections. If the GP model works significantly poorly at the section boundaries, it will be difficult to map the systematic errors across the whole parameter space. We examine this as illustrated in Figure 3. We inspect absolute testing errors on the $M - EEP$ diagram (the top graph) and do not find obvious edge effects. We also do a statistical comparison between all testing errors and those around section boundaries ($\pm 0.01 EEP$) as demonstrated in the bottom graph. The density distributions of the two datasets are very

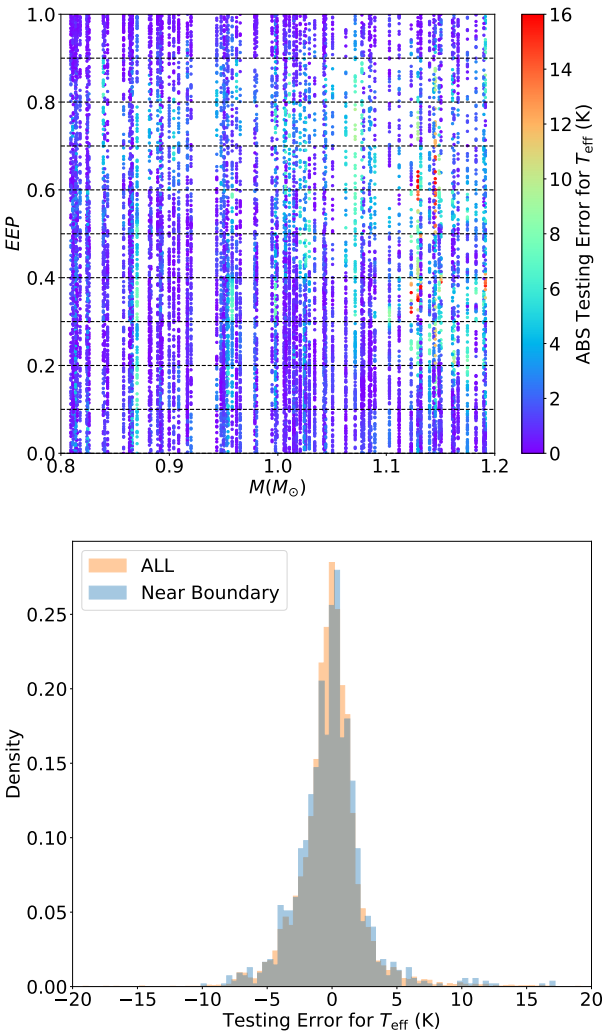

Figure 3. Top: Testing errors of 3D GP model for $T_{\rm eff}$ on the $M - EEP$ diagram. Dashes indicates section boundaries. Bottom: examination of the edge effects of the section scenario. Probability distributions of testing errors of all testing data and those near the boundary (±0.01 EEP) in the upper graph are compared. As it can be seen, testing errors do not raise around the boundary.

similar and we hence conclude that there is no edge effect. The section scenario is adopted in the following study.

Here we summary our set up for the GP model as follow:

- Model Type: Exact GP with the section scenario applied
- Kernel: Mat32 (for all outputs)
- Mean Function: Neural Network with 6 linear layers x 128 nodes and element-wise function (Elu)
- Likelihood Function: Gaussian Likelihood Function
- Loss Function: Exact marginal likelihood
- Optimiser: Adam including AMSGRAD variant
- Early Stoping: controlled by the validating error index

4 GP TRAINING RESULTS

With the GP model described in Section 3, we train GP models for augmenting the MESA grid (5D GP model). The training data includes the primary and the additional girds as mentioned in Table 1. The role of the additional grid is increasing the grid resolution for evolutionary tracks with the 'hook'. The total training data includes ~ 15,000 evolutionary tracks and ~ 10,000,000 data points. For validating and testing the data, we computed another 4880 tracks with random model inputs within the grid ranges. These tracks are split 50-to-50 for validating (in the training progress) and testing (after the training progress) GP models. In the training process, we use 20,000 training and 20,000 validating data points for each section. We do testing when training is completed for sections. The testing dataset contents 50, 000 data points that cover the whole EEP range. Note that we do not use models with $\tau \geq 20.0$ Gyr, $[\mathrm{Fe/H}]_{\rm surf} \leq$ -0.6 dex, or $T_{\rm eff} \geq 7000K$ as testing data.

4.1 Overview of Results

We first train an Exact GP model for the whole grid as a reference. Results of testing errors are listed in Table 2. Compared with 2D and 3D GP models, testing errors of this 5D GP model remarkably increase. The results clearly indicate that GP model accuracy declines when input demission increases. We then train 5D GP models with the section scenario. We gradually increase the number of sections and track changes in testing errors. It is found that testing errors are not significantly improved when the data are divided by more than 10 sections as shown in Table 2. Comparing testing errors for the 10-, 20-, and 100-sections cases, the 68th and 95th testing errors are at same accuracy levels. Although the 99.7th values slightly decrease for more sections, it is more like a random error. It turns out that the 10-sections case is the most efficient strategy. Following analysis are all based on this model.

A overview of the GP model accuracy can be seen in Figure 4, in which we demonstrate rolling medians and standard deviations of testing errors of each output as a function of each input parameter. It can be seen that the median values are approximate along zero in most plots, indicating that there is no obvious systematic offsets. The 68% confidential intervals are generally small and do not significantly vary. However, the 95% confidential intervals change in a in larger dynamic ranges and clearly depends on M, EEP, and $[\mathrm{Fe/H}]_{\rm init}$. This is caused by the tail feature as illustrated in Figure 2. It turns out that the marginal error distributions is not a good way to estimate systematic uncertainties for GP outputs.

4.2 Mapping Systematical Uncertainties across the input parameter space

Systematical uncertainties of GP model are not uniform as shown in Figure 4. They relate to M, EEP, and $[\mathrm{Fe/H}]_{\rm init}$ but not to $Y_{\rm init}$ or $\alpha_{\rm MLT}$. Thus, a comprehensive statistical model for systematical uncertainty can be described as a function of M, EEP, and $[\mathrm{Fe/H}]_{\rm init}$.

We examine local error distributions in the M-EEP-$[\mathrm{Fe/H}]_{\rm init}$ space. We divided this space into equal-size segments. The the choice of the segment size matters. It needs to be small enough for only presenting the local feature, but it can not be very small so that there are not enough data points for proper statistical analysis. To find the appropriate size, we do a statistic test for the data in every segment to test wether the local error distribution has a tail feature. The condition we apply for the test is either the ratio between 68%

Table 2. Training and validating errors for GPR Models

Model Type	Inputs	$N_{\rm Training}$	Sampling rate	Testing Errors (at 68/95/99.7%)				
				$T_{\rm eff}$ (K)	$\log g$ (10^{-3}dex)	R ($10^{-3}R_\odot$)	$[{\rm Fe/H}]_{\rm surf}$ (10^{-3}dex)	τ (10^{-2}Gyr)
Exact GP	2D	20,000 x 1	96%	1/5/11	1/3/8	2/6/14	0.5/2/12	1/3/9
Exact GP	3D	20,000 x 1	5%	2/6/16	1/4/10	3/7/17	2/6/22	2/7/22
Exact GP (10 sections)	3D	20,000 x 10	50%	2/5/15	1/4/11	2/7/17	1/3/20	2/6/19
Exact GP	5D	20,000 x 1	0.2%	3/9/34	2/5/18	4/11/36	2/7/30	3/9/27
Exact GP (3 sections)	5D	20,000 x 3	0.6%	3/8/27	2/5/18	3/7/26	1/4/24	3/7/22
Exact GP (5 sections)	5D	20,000 x 5	1%	2/7/25	1/4/15	3/7/24	1/4/21	2/6/22
Exact GP (10 sections)	5D	20,000 x 10	2%	2/7/27	1/4/14	2/7/26	1/4/20	2/6/21
Exact GP (20 sections)	5D	20,000 x 20	4%	2/7/26	1/4/14	2/7/27	1/3/18	2/6/22
Exact GP (100 sections)	5D	20,000 x 100	20%	2/7/25	1/4/14	2/7/26	1/3/17	2/6/18

confidential interval and 95% confidential interval less than 2.5, or the ratio between 68% confidential interval and 99.7% confidential interval less than 4. After several attempts, we decide to divide the input range into 40 equally spaced segments for M, 50 for EEP, and 20 for $[{\rm Fe/H}]_{\rm init}$. Hence there are 43,911 (41 x 21 x 51) points in the space. Local standard deviation for each point is computed with the data in a 3-segments (1.5 previous and 1.5 after) range. The statistic test shows that $\sim$ 90% points meet the above condition. This infer that the selection of segment size statistically sounds.

We demonstrate local systematic uncertainties of testing errors for $T_{\rm eff}$ ($\sigma T_{\rm eff}$) on the $M-EEP$ diagram in Figure 5. As it can be seen that $\sigma T_{\rm eff}$ values are below $\sim 4K$ in the low-mass regions and raise when mass is greater than $1.05M_\odot$. The distribution is smooth in general but there are many substructures on the diagram. We also do visual inspections for the error distributions of other outputs and all of the show various substructures across the 3D space. It is apparently not convinced to describe the error distribution with any simple mathematical expressions. We hence use another GP model (GP-SYS model) to map three fundamental inputs (M, EEP, and $[{\rm Fe/H}]_{\rm init}$) to local systematic uncertainty of each output.

Here we discuss how we set up the GP-SYS model. There are 43,911 data points and we split them for training and validating. We randomly select 32,000 data points as the training dataset and use the rest 11,311 data points as the validating dataset. Because the training data size exceeds the 20,000 limit for Exact GP, we need other approach for the large data sample. As mentioned in Section 3, the SVGP framework works when the dataset is large and the kernel function is not very complex. It hence suitable for this case. We use 10,000 inducing points which are randomly selected in the 3D space. The training data is divided into 10 random batches for the SVGP training. In each training iteration, the 10 data batches is feeded in by turn to optimises hyperparameters. We use a constant mean function and the RBF kernel for this GP-SYS model because we want a smooth function to describe the systematic uncertainty. The variational evidence lower bound (ELBO) is adopted as the loss function. This approach is designed for when there is too much data for exact inference. The Early Stopping which is controlled by RMSE value of validating data. The training progress terminates when the validating RMSE stops decreasing for 30 iterations. We set up the likelihood function and the optimiser with the same method as training the GP-Grid model. Our set up for the GP-SYS model is listed as follow.

- Model Type: SVGP with 10,000 inducing number.
- Model Inputs: M, EEP, and $[{\rm Fe/H}]_{\rm init}$
- Model Outputs: σ_T, $\sigma_{\log g}$, σ_R, $\sigma_{[{\rm Fe/H}]_{\rm surf}}$, and σ_τ.
- Training dataset: 3200 x 10 data points
- Validating dataset: 11,311 data points
- Kernel: RBF (for all outputs)
- Mean Function: Constant Mean Function
- Likelihood Function: Gaussian Likelihood Function
- Loss Function: The variational evidence lower bound (ELBO)
- Optimiser: Adam including AMSGRAD variant
- Early Stoping: when validating RMSE stops decreasing for 30 iterations

Table 3. Residual of GP models for systematic uncertainty

GP output	Average Residual
$\sigma_{T_{\rm eff}}$ (K)	0.15
$\sigma_{\log g}$ (10^{-3}dex)	0.08
σ_R ($10^{-3}R_\odot$)	0.2
$\sigma_{[{\rm Fe/H}]_{\rm surf}}$ (10^{-3}dex)	0.09
σ_τ (10^{-2} Gyr)	0.2

We train and validate GP-SYS model for each output parameter. In Figure 5, we demonstrate the actual $\sigma_{T_{\rm eff}}$ and the GP-trained $\sigma_{T_{\rm eff}}$ for $[{\rm Fe/H}]_{\rm init}$ on the $M-EEP$ diagram. As it can be seen that, GP well reproduces the $\sigma_{T_{\rm eff}}$ distributions. Validation errors are in a range of $\pm 1K$ and the average residual is only 0.15K. We summary the average residuals for all five output parameter in Table 3.

5 AUGMENTATION OF THE MESA GRID

We have obtained GP-Grid models for predicting observable outputs and GP-SYS models for estimating systematical uncertainty. In Figure 6, we demonstrate GP-trained Kiel diagrams across four input demissions, i.e., M, $[{\rm Fe/H}]_{\rm init}$, $Y_{\rm init}$, and $\alpha_{\rm MLT}$. Compared with the sparse MESA grid, GP models turn it into a non-sparse model set. **some discussions about the FIgure.**

We augment the MESA model grid and turn it into a continuously sampled model set which contents 5,000,000 model data points. We randomly sample GP model inputs with flat distributions, predict observable outputs with GP-Grid models and systematic uncertainties with GP-SYS models. The model set includes five fundamental inputs (M, EEP, $[{\rm Fe/H}]_{\rm init}$, $Y_{\rm init}$, and $\alpha_{\rm MLT}$), five outputs, ($T_{\rm eff}$, $\log g$, R, $[{\rm Fe/H}]_{\rm surf}$, and τ), and five systematic uncertainties ($\sigma_{T_{\rm eff}}$, $\sigma_{\log g}$, σ_R, $\sigma_{[{\rm Fe/H}]_{\rm surf}}$, and σ_τ). This model set

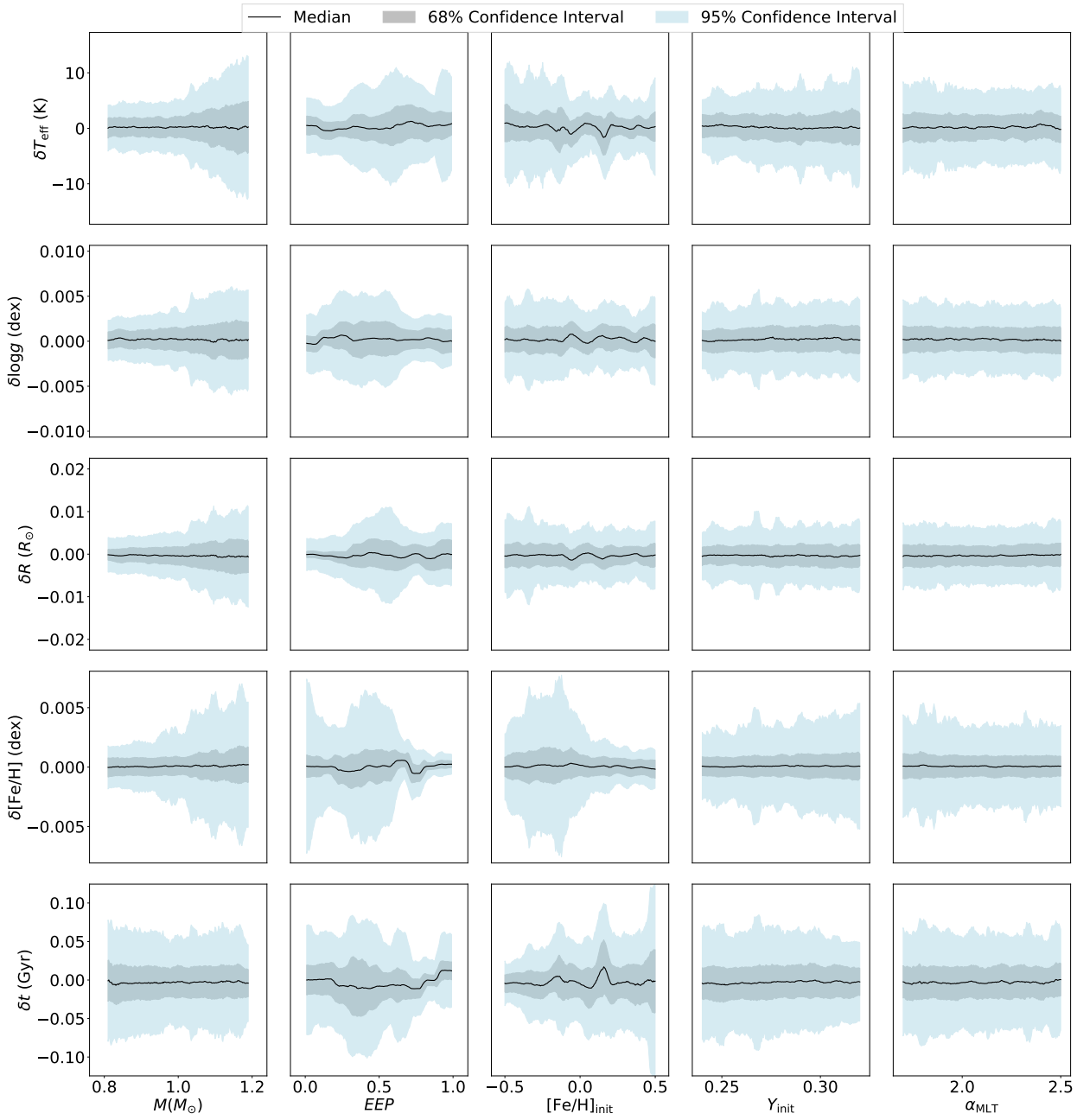

Figure 4. Roll medians and confidential interval of testing errors against GP model inputs. Black solid line indicate the median; grey and blue shadow represent the 68% and 95% confidential interval. Testing errors of $T_{\rm eff}$, $\log g$, and R mainly depend on M and EEP. Metallicity error strongly depends on M, EEP, and $[{\rm Fe/H}]_{\rm init}$, and age error has a significant correlation to EEP and $[{\rm Fe/H}]_{\rm init}$. However, testing errors do not obviously relate to $Y_{\rm init}$ or $\alpha_{\rm MLT}$.

can be downloaded at a-place-for-data. We listed 10 models in Table XXX as examples.

5.1 Modelling fake stars with GP-trained models

We lastly test the general performance of GP models by modelling 100 fake stars. Fake stars are randomly selected from the off-grid MESA models. We use four observables, i.e., $T_{\rm eff}$, $\log g$, R, and $[{\rm Fe/H}]_{\rm surf}$, as input constraints. We applied typical observed uncertainty that is ±50 K for $T_{\rm eff}$ (high-resolution spectroscopy), ±0.005 for $\log g$ (seismology), 3% for R (seismology), and ±0.05 for $[{\rm Fe/H}]_{\rm surf}$ (high-resolution spectroscopy). To avoid edge effect, fakes stars are selected in the parameter ranges of $T_{\rm eff}$ = [4700K, 6800 K], $\log g$ = [3.7, 4.6], $[{\rm Fe/H}]_{\rm surf}$ = [-0.35,0.35], M = [0.85,1.15], EEP = [0.05,0.95], $Y_{\rm init}$ = [0.25,0.31], and $\alpha_{\rm MLT}$ = [1.8,2.4].

Fitting method: 5σ cut - MLE. discuss Figure 7.

Discuss figure 8

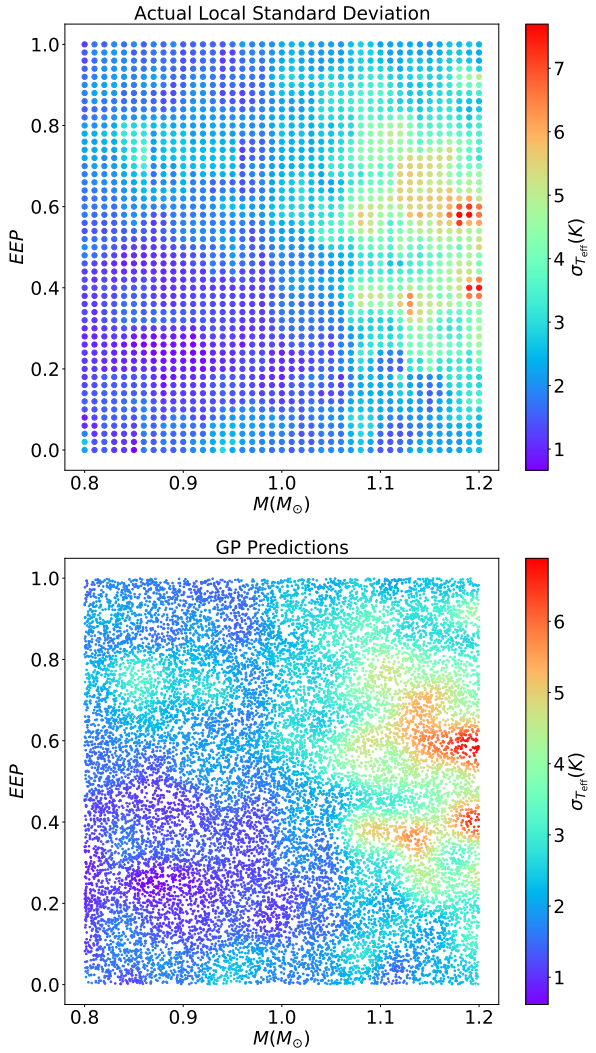

Figure 5. Local standard deviations of testing errors for $T_{\rm eff}$ on the M − −EEP diagram ([Fe/H]$_{\rm init}$ = 0.0).

6 DISCUSSION AND CONCLUSIONS

Discussions: advantages (GP give continuous functions, it is efficient (1 hour/model) , section scenario offers flexibility for large grid); limitations, ... future work: GP individual stars, MCMC etc.

Conclusions: GP can be an accurate and efficient approach for augmenting stellar model grid.

ACKNOWLEDGEMENTS

Development of GPyTorch is supported by funding from the Bill and Melinda Gates Foundation, the National Science Foundation, and SAP.

REFERENCES

Asplund M., Grevesse N., Sauval A. J., Scott P., 2009, Annual Review of Astronomy and Astrophysics, 47, 481
Choi J., Dotter A., Conroy C., Cantiello M., Paxton B., Johnson B. D., 2016, ApJ, 823, 102
Dotter A., 2016, ApJS, 222, 8
Ferguson J. W., Alexander D. R., Allard F., Barman T., Bodnarik J. G., Hauschildt P. H., Heffner-Wong A., Tamanai A., 2005, ApJ, 623, 585
Gardner J. R., Pleiss G., Bindel D., Weinberger K. Q., Wilson A. G., 2018, in Advances in Neural Information Processing Systems.
Hensman J., Matthews A., Ghahramani Z., 2014, arXiv preprint arXiv:1411.2005
Hon M., Stello D., Yu J., 2018, MNRAS, 476, 3233
Paquette C., Pelletier C., Fontaine G., Michaud G., 1986, ApJS, 61, 177
Paxton B., Bildsten L., Dotter A., Herwig F., Lesaffre P., Timmes F., 2011, The Astrophysical Journal Supplement Series, 192, 3
Paxton B., et al., 2013, The Astrophysical Journal Supplement Series, 208, 4
Paxton B., et al., 2015, The Astrophysical Journal Supplement Series, 220, 15
Paxton B., et al., 2018, ApJS, 234, 34
Paxton B., et al., 2019, ApJS, 243, 10
Reddi S., Kale S., Kumar S., 2018, in International Conference on Learning Representations.
Rendle B. M., et al., 2019, MNRAS, 484, 771
Rogers F. J., Nayfonov A., 2002, ApJ, 576, 1064
Sutskever I., Martens J., Dahl G., Hinton G., 2013, in International conference on machine learning. pp 1139–1147
Thoul A. A., Bahcall J. N., Loeb A., 1994, ApJ, 421, 828
Williams C. K., Rasmussen C. E., 1996
Wilson A., Nickisch H., 2015, in International Conference on Machine Learning. pp 1775–1784
Wu Y., et al., 2019, MNRAS, 484, 5315

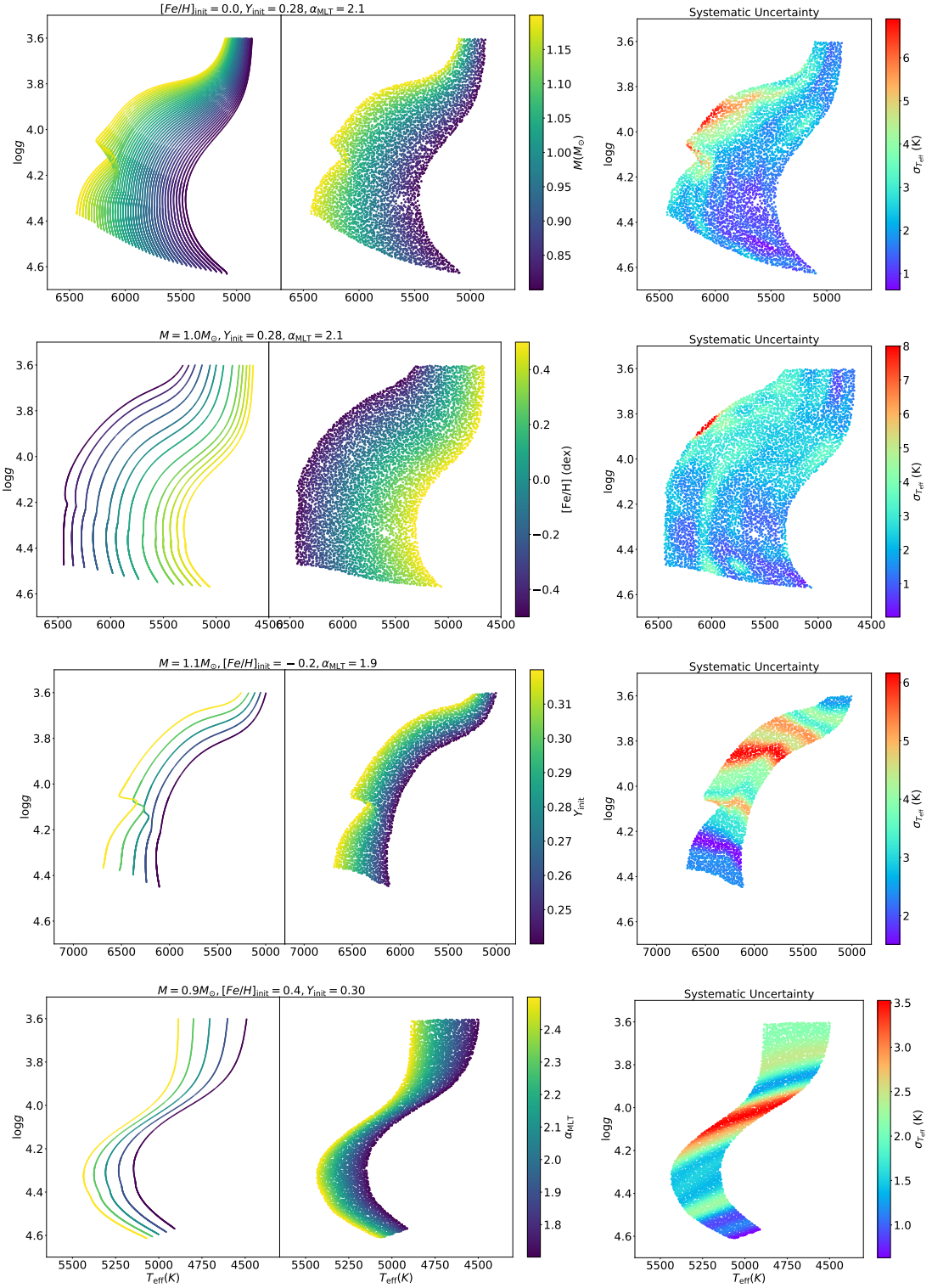

Figure 6. GP-trained models and their systematic uncertainties across four demissions on the Kiel diagram.

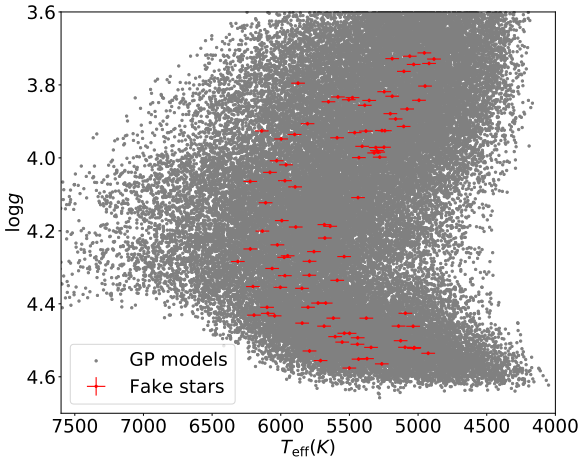

Figure 7. Fake stars.

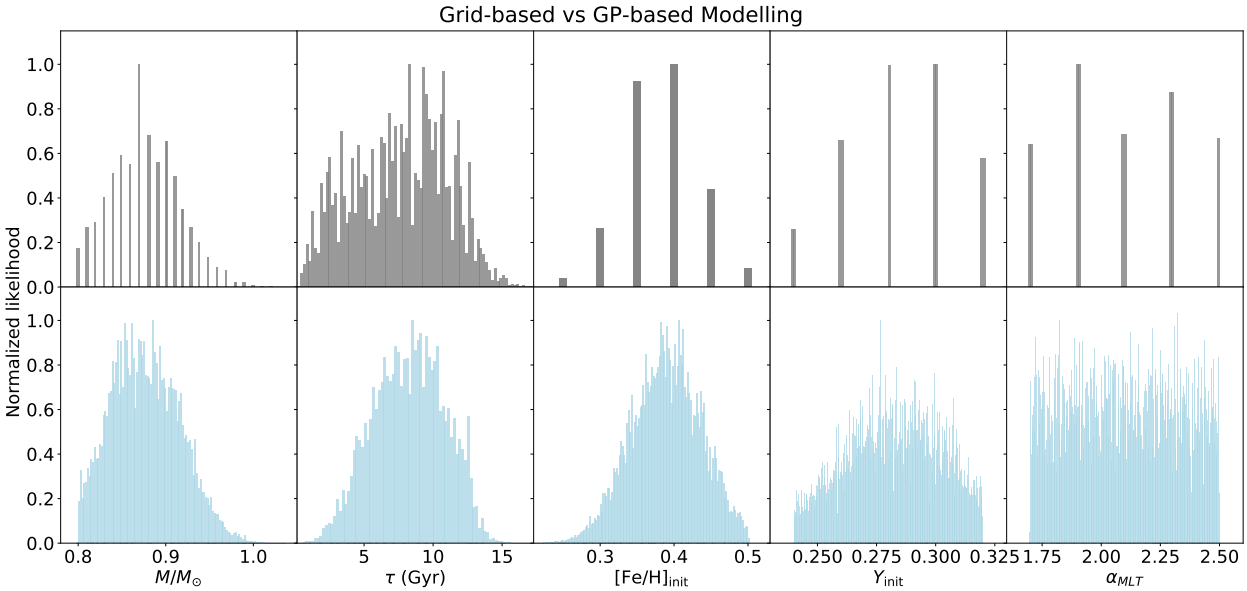

Figure 8. Probability distributions of estimated fundamental parameters from grid-based and GP-based modelling.

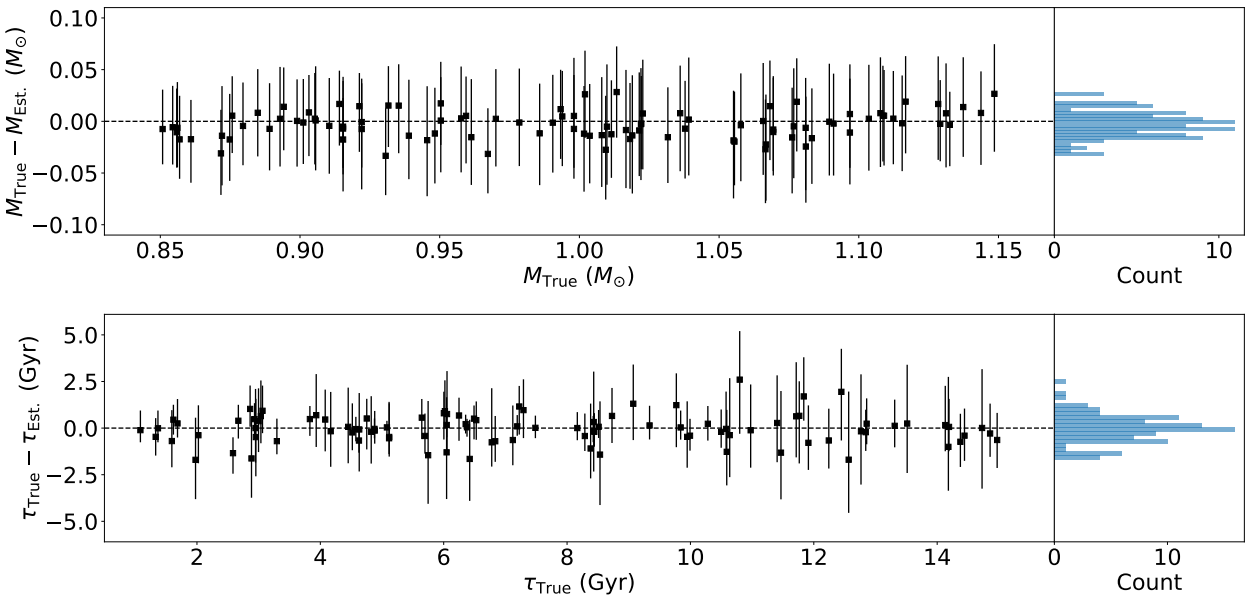

Figure 9. Results of fake stars.

Figure A1. Validations for 5D inputs GPR models before and after chunking.

APPENDIX A: VALIDATION AND PREDICTION OF GPR MODELS

A1 GPR model with 3-D inputs

- Training set
- validation
- GP predictions

A2 GPR model with 5-D inputs

- Training set
- validation
- GP predictions

This paper has been typeset from a TEX/LATEX file prepared by the author.