The Astronomical Journal, 166:88 (15pp), 2023 September https://doi.org/10.3847/1538-3881/acdcfb

Directly Deriving Parameters from SDSS Photometric Images

Fan Wu[1], Yude Bu[1], Mengmeng Zhang[1], Zhenping Yi[2], Meng Liu[2], and Xiaoming Kong[2]

[1] School of Mathematics and Statistics, Shandong University, Weihai, 264209, Shandong, People's Republic of China; buyude@sdu.edu.cn

[2] School of Mechanical, Electrical and Information Engineering, Shandong University, Weihai, 264209, Shandong, People's Republic of China; xmkong@sdu.edu.cn

Received 2023 April 1; revised 2023 June 7; accepted 2023 June 7; published 2023 August 2

Abstract

Stellar atmospheric parameters (effective temperature, surface gravity, and metallicity) are fundamental for understanding the formation and evolution of stars and galaxies. Photometric data can provide a low-cost way to estimate these parameters, but traditional methods based on photometric magnitudes have many limitations. In this paper, we propose a novel model called Bayesian Convit, which combines an approximate Bayesian framework with a deep-learning method, namely Convit, to derive stellar atmospheric parameters from Sloan Digital Sky Survey images of stars and effectively provide corresponding confidence levels for all the predictions. We achieve high accuracy for $T_{\rm eff}$ and [Fe/H], with $\sigma(T_{\rm eff}) = 172.37$ K and $\sigma([{\rm Fe/H}]) = 0.23$ dex. For $\log g$, which is more challenging to estimate from image data, we propose a two-stage approach: (1) classify stars into two categories based on their $\log g$ values (>4 dex or <4 dex) and (2) regress separately these two subsets. We improve the estimation accuracy of stars with $\log g > 4$ dex significantly to $\sigma(\log g > 4) = 0.052$ dex, which are comparable to those based on spectral data. The final joint result is $\sigma(\log g) = 0.41$ dex. Our method can be applied to large photometric surveys like Chinese Space Station Telescope and Large Synoptic Survey Telescope.

Unified Astronomy Thesaurus concepts: Astronomy image processing (2306); Stellar photometry (1620); Stellar abundances (1577); Neural networks (1933); Bayes' Theorem (1924)

1. Introduction

Galaxies are the basic units of structure in the universe. The formation and evolution of galaxies over time, as well as their diversity in morphology, structure and properties, are major topics in modern astrophysics. The atmospheric parameters of stars, such as effective temperature ($T_{\rm eff}$), surface gravity ($\log g$), and metallicity [Fe/H], are essential for studying how galaxies form, structure and evolve chemically, and also help to calculate the distances and ages of stars.

There are two main types of methods for deriving stellar atmospheric parameters: direct and indirect. The direct methods are highly demanding. Like measurement of effective temperature, it requires not only correcting for interstellar extinction, but also obtaining the total flux in each band and angular diameter of the star, and applying the Stefan–Boltzmann law (Halliday & Resnick 1988; Baschek et al. 1991; Roy & Clarke 2003). Besides, it also suffers from distance constraints, as it becomes impractical for distant stars. Another example is using eclipsing binaries (Southworth et al. 2004) to obtain high-precision surface gravity, yet it very difficult to measure single stars. So far, less than 200 stars have been directly measured for their atmospheric parameters, and only a few stars meet the conditions for direct measurement.

Indirect methods can be classified into two categories: spectroscopic and photometric. Spectroscopic methods use spectral data to estimate the parameters, employing various techniques such as traditional statistics, machine learning and deep learning. Ness et al. (2015) used a Bayesian posterior inference at each wavelength to derive the effective temperature ($T_{\rm eff}$), surface gravity ($\log g$), and metallicity [Fe/H] for 550,000 stars. Bu & Pan (2015) proposed a method for estimating stellar atmospheric parameters based on Gaussian process, which has a more easily optimized structure and parameters. Kielty et al. (2018) built a one-dimensional convolutional neural networks to capture the characteristics of spectral data. The spectroscopic method based on high-resolution spectral data is undoubtedly the most accurate method for measuring atmospheric parameters, but it is difficult to apply it on a large scale because of the limited number of high-resolution spectra and its high acquisition cost.

Photometric data are easier to obtain than spectroscopic data. Future surveys will rely more on photometric measurements, thanks to the improved CCD technology and the advanced statistical methods that can handle these lower-resolution data. For example, the Large Synoptic Survey Telescope, which is being built in China, will collect photometric data in six bands and produce a much larger data set than Sloan Digital Sky Survey (SDSS). Although photometric observations are not as accurate as spectroscopic ones for estimating stellar parameters, they have unparalleled advantages over spectral observations, such as covering large or even full sky areas. Therefore, photometric data have enormous potential for applications.

Many studies have explored how to derive stellar parameters from photometric data. For example, Alonso et al. (1996) established an empirical relation between metal abundance and effective temperature using infrared flux methods. Huang et al. (2015) collected a sample of about 200 nearby stars and established empirical relations between effective temperature, stellar color, and metal abundance. Huang et al. (2019) applied a regression polynomial to photometric colors from SMSS to derive atmospheric parameters of red giant stars. Chiti et al. (2021) used SkyMapper v, u, g, i photometry to obtain metallicities of giant stars from SMSS with $[{\rm Fe/H}] < -0.75$ dex. Xu et al. (2022) constructed high-quality samples of giants and dwarfs from Gaia Early Data Release 3 using empirically determined coefficients that describe stellar metallicity loci of colors, and they corrected for magnitudedependent systematic errors and reddening. Huang et al. (2022) combined stellar colors of SMSS DR2 and Gaia EDR3 to infer atmospheric parameters based on a maximum-likelihood approach. Liang et al. (2022) transformed photometric data for all bands (u, v, g, r, i and z) using PCA and estimated atmospheric parameters using LightGBM, a machine-learning algorithm. Fallows & Sanders (2022) employed a neural network method, which used 8-band photometry and parallaxes from Gaia EDR3, 2MASS and WISE surveys to estimate metallicities and uncertainties of red giant stars. Yang et al. (2022) constructed a set of cost-sensitive neural networks (CSNet), which utilized 13-band J-plus photometry to determine stellar parameters, C, N, Mg, Ca, and [α/Fe] abundances. During the training process, they adjusted the model loss according to the density distribution of the samples.

Traditional methods of acquiring magnitudes in each band are not only relatively complicated but also losing a lot of image details due to area integration, which reduces the utilization rate of image information. With the sustained development of deep learning, more and more studies are using photometric image data for various tasks. A deep-learning algorithm (He et al. 2021) was applied to detect sources, and further classify quasars, stars, and galaxies based on detected sources. Shi et al. (2022) constructed a photometry pipeline based on convolutional neural networks (CNNs), including target source detection, target source classification and parameter measurement branch. However, the parameter measurement branch in the paper still relies on traditional methods. At present, relevant work on estimating stellar parameters directly based on photometric images is still in its infancy, with much unexplored territory.

Neural networks have become a powerful tool for various real-world problems, especially visual tasks (He et al. 2015; Dosovitskiy et al. 2021; Dai et al. 2021), in the past decade. However, vanilla neural networks cannot provide uncertainty estimates and their confidence may be inflated, which motivates the development of Bayesian neural networks (BNNs). BNNs differ from vanilla deep neural networks in that their weights are random variables rather than fixed values. The fitted objects are the posterior probability distributions of the predictions, which can provide corresponding confidence levels instead of fixed prediction values. MacKay (1992b) showed how to perform model fitting within the Bayesian framework, which faced the challenge that computing the marginal likelihood is computationally expensive. Subsequently, variational inference (Jordan et al. 1999; Blei et al. 2017) simplified the computation by assuming a specific form for the posterior distribution.

However, modern neural networks require large data sets to achieve powerful inference capabilities. For large data sets, using the complete log-likelihood function as the training objective is impractical. This issue has motivated the development of modern BNNs. Some basic work (Opper & Archambeau 2009; Graves 2011) optimized the model in terms of gradient computation and update methods. Welling & Teh (2011) applied stochastic gradient descent, which uses small batches of data to approximate the likelihood term. A key technique for approximate inference in BNNs, called Bayes by Backprop, uses a reparameterization trick to obtain unbiased estimates of the expected derivatives (Blundell et al. 2015). Shridhar et al. (2019) built Bayesian convolutional neural networks based on Bayes by Backprop and performed well on data sets such as MINIST. The application of Bayesian neural networks will naturally satisfy our need for confidence levels of estimating atmospheric parameters. However, BNNs will have relatively serious convergence difficulty on large-scale networks due to a large number of prior regularizations and parameter multiplications. Moreover, in a real-time scenario, the computational overhead from repeated model inference is huge due to the need for approximation inference from probabilistic models.

In this paper, considering that our model uses image data and our interest in evaluating the uncertainty of the predictions from large models, we propose a novel method for obtaining stellar atmospheric parameters. The main contributions of our work are as follows:

1. We estimate atmospheric parameters directly from star images for the first time, preserving more image details and information than traditional photometric magnitudes.
2. We propose an approximate Bayesian framework that cost-effectively quantifies the model uncertainty and provides theoretical analysis of its validity. This will have more potential in scenarios where confidence is required for downstream tasks based on large pretrained models.
3. To address the challenge of estimating $\log g$ from image data, we introduce a classification stage before regression. This stage effectively eliminates the anomaly in the original regression results and enhances the estimation accuracy.

The paper is organized as follows. Section 2 introduces the approximate Bayesian framework and specific models, namely BNNs and ConViT, to instantiate the framework. Section 3 describes the photometric images and image processing methods. Section 4 presents the main experimental results and discusses them. Finally, Section 5 concludes the paper and outlines potential directions for future research.

2. Method

2.1. Bayesian Neural Networks

BNNs (Buntine & Weigend 1991; MacKay 1992a) are a type of probabilistic models, which can represent the uncertainties of their predictions, unlike vanilla neural networks that use point estimates. In this section, we will briefly introduce how to transform point estimates in neural networks into posterior inference in Bayesian neural networks and apply an efficient backpropagation-compatible algorithm (Blundell et al. 2015) to optimize weights.

Point Estimates of Neural Networks: We first consider the point estimate of the neural networks. Viewing neural networks as probabilistic models, given a set of parameters $\boldsymbol{w} \in \Theta$, and an input $\boldsymbol{x} \in \mathbb{R}^p$, we can obtain the current probability distribution $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w})$ of output $\boldsymbol{y} \in \mathbb{R}^q$. Θ is the space of all possible parameter values that need to be optimized for a probabilistic model. The optimal weights $\boldsymbol{w}^{\mathrm{MAP}}$ can be learned by maximizing the posterior $P(\boldsymbol{w}|\mathcal{D})$, where $P(\boldsymbol{w})$ is an artificially specified prior, and $\mathcal{D}$ is the training data.

$$\begin{aligned}\boldsymbol{w}^{\mathrm{MAP}} &= \arg\max_{\boldsymbol{w}} \log P(\boldsymbol{w}|\mathcal{D}) \\ &= \arg\max_{\boldsymbol{w}} \log P(\mathcal{D}|\boldsymbol{w}) + \log P(\boldsymbol{w}).\end{aligned} \quad (1)$$

In practice, this will be commonly accessed by gradient descent with appropriate loss function (e.g., square error in regression, or cross-entropy in classification).

Being Bayesian: Unlike point estimates that have fixed weights, Bayesian neural networks have stochastic weights that follow $P(\boldsymbol{w}|\mathcal{D})$. Given $\boldsymbol{x}$, we can infer $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w})$ by sampling $\boldsymbol{w}$ from $P(\boldsymbol{w}|\mathcal{D})$ with corresponding probability. Furthermore, by taking into account all the possible configurations of weights, we can obtain an expectation that follows $P(\boldsymbol{y}|\boldsymbol{x}) = \mathbb{E}_{P(\boldsymbol{w}|\mathcal{D})}[P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w})]$. This is equivalent to using an ensemble of infinitely many neural networks for prediction.

However, Bayesian neural networks face the challenge of applying Bayesian inference to determine $P(\boldsymbol{w}|\mathcal{D})$, which is intractable for neural networks of any practical size due to its complexity. To overcome this problem, a variational approximation (Graves 2011) to $P(\boldsymbol{w}|\mathcal{D})$ has been proposed. Variational learning aims to find a simple distribution $q(\boldsymbol{w}|\theta)$ that approximates $P(\boldsymbol{w}|\mathcal{D})$ by minimizing the Kullback–Leibler (KL) divergence between them, as follows:

$$\begin{aligned}\theta^* &= \arg\min_{\theta} \boldsymbol{KL}[q(\boldsymbol{w}|\theta)||P(\boldsymbol{w}|\mathcal{D})]\\ &= \arg\min_{\theta} \int q(\boldsymbol{w}|\theta)\log\frac{q(\boldsymbol{w}\mid\theta)}{P(\boldsymbol{w})P(\mathcal{D}\mid\boldsymbol{w})}d\boldsymbol{w}\\ &= \arg\min_{\theta} \boldsymbol{KL}[q(\boldsymbol{w}|\theta)||P(\boldsymbol{w})] - \mathbb{E}_{q(\boldsymbol{w}|\theta)}[\log P(\mathcal{D}|\boldsymbol{w})], \quad (2)\end{aligned}$$

where θ is the parameter space of the simple distribution. For simplicity, we can denote it as:

$$\mathcal{F}(\mathcal{D}, \theta) = \boldsymbol{KL}[q(\boldsymbol{w}|\theta)||P(\boldsymbol{w})] - \mathbb{E}_{q(\boldsymbol{w}|\theta)}[\log P(\mathcal{D}|\boldsymbol{w})]. \quad (3)$$

This is the objective function of the Bayesian neural networks, which consists of two parts: a prior regularization term and a data-dependent likelihood term.

Although variational inference provides a simpler approach than exact inference, the expectation term that in the objective function is still difficult to compute. Therefore, we use Monte Carlo (MC) sampling to approximate it. In most cases, the KL divergence term has an analytic solution, but we also approximate it by MC sampling to handle more complex scenarios. To do this, we first reformulate $\mathcal{F}(\mathcal{D}, \theta)$ into an expectation-like form and then approximate it:

$$\begin{aligned}\mathcal{F}(\mathcal{D}, \theta) &= \mathbb{E}_{q(\boldsymbol{w}|\theta)}[\log q(\boldsymbol{w}|\theta) - \log P(\boldsymbol{w})P(\mathcal{D}|\boldsymbol{w})]\\ &\approx \sum_{i=1}^{n} \log q(\boldsymbol{w}^{(i)}|\theta) - \log P(\boldsymbol{w}^{(i)})P(\mathcal{D}|\boldsymbol{w}^{(i)}) \quad (4)\end{aligned}$$

where $\boldsymbol{w}^{(i)}$ denotes the ith Monte Carlo sample drawn from the variational posterior $q(\boldsymbol{w}^{(i)}|\theta)$. However, there is a significant challenge in computing the gradient: let $f(\boldsymbol{w}, \theta) = \log q(\boldsymbol{w}|\theta) - \log P(\boldsymbol{w})P(\mathcal{D}|\boldsymbol{w})$, according to the chain rule, the gradient of θ is: $\Delta_\theta = \frac{\partial f(\boldsymbol{w},\theta)}{\partial \boldsymbol{w}}\frac{\partial \boldsymbol{w}}{\partial \theta} + \frac{\partial f(\boldsymbol{w},\theta)}{\partial \theta}$, but $\frac{\partial \boldsymbol{w}}{\partial \theta}$ cannot be computed because we cannot calculate the gradient of the sampling operation. To overcome this problem, we can apply the reparameterization trick, which is commonly used for latent variable models. There is a generalization (Blundell et al. 2015) of it as follows:

$$\frac{\partial}{\partial\theta}\mathbb{E}_{q(\boldsymbol{w}|\theta)}[f(\boldsymbol{w}, \theta)] = \mathbb{E}_{q(\epsilon)}\left[\frac{\partial f(\boldsymbol{w}, \theta)}{\partial \boldsymbol{w}}\frac{\partial \boldsymbol{w}}{\partial \theta} + \frac{\partial f(\boldsymbol{w}, \theta)}{\partial \theta}\right] \quad (5)$$

where ϵ is a random variable with a probability density determined by $q(\epsilon)$, $\boldsymbol{w} = t(\theta, \epsilon)$ with a prerequisite: $q(\epsilon)d\epsilon = q(\boldsymbol{w}|\theta)d\boldsymbol{w}$, where t is a deterministic mapping. The original $\boldsymbol{w}$ from the variational posterior is transformed into a parameter-free stochastic term ϵ and a parameter term θ, then we can obtain $\frac{\partial \boldsymbol{w}}{\partial \theta}$ separately without considering sampling. Next, we will demonstrate how this transform works in the Gaussian case, which is applied in practice.

Suppose the variational posterior is a Gaussian distribution, then ϵ are derived by sampling from the unit Gaussian distribution, and shifting it by mean μ and standard deviation σ to obtain $\boldsymbol{w}$. To ensure that σ is nonnegative, we set $\sigma = \log(1 + \exp(\rho))$. Thus, we have $\boldsymbol{w} = t(\mu, \rho) = \mu + \log(1 + \exp(\rho))\circ\epsilon$.

The optimization proceeds as follows:

1. Sample $\epsilon \sim \mathcal{N}(0, I)$.
2. Random initialization (μ, ρ) for each node, let $\theta = (\mu, \rho)$.
3. Let $\boldsymbol{w} = \mu + \log(1 + \exp(\rho))\circ\epsilon$.
4. Let $f(\boldsymbol{w}, \theta) = \log q(\boldsymbol{w}|\theta) - \log P(\boldsymbol{w})P(D|\boldsymbol{w})$.
5. Calculate the gradient of the mean:
$$\Delta_\mu = \frac{\partial f(\boldsymbol{w}, \theta)}{\partial \boldsymbol{w}} + \frac{\partial f(\boldsymbol{w}, \theta)}{\partial \mu}. \quad (6)$$
6. Calculate the gradient of the standard deviation ρ:
$$\Delta_\rho = \frac{\partial f(\boldsymbol{w}, \theta)}{\partial \boldsymbol{w}}\frac{\epsilon}{1 + \exp(-\rho)} + \frac{\partial f(\boldsymbol{w}, \theta)}{\partial \rho}. \quad (7)$$
7. Update θ:
$$\begin{aligned}\mu &\leftarrow \mu - \Delta_\mu\\ \rho &\leftarrow \rho - \Delta_\rho. \quad (8)\end{aligned}$$

Actually, the $\frac{\partial f(\boldsymbol{w},\theta)}{\partial \boldsymbol{w}}$ term are shared and can be obtained by normal backpropagation algorithm like $[\frac{\partial f(\boldsymbol{w},\theta)}{\partial \mu}, \frac{\partial f(\boldsymbol{w},\theta)}{\partial \rho}]$. Here, we present the primary equations (Equations (1)–(8)) from the prior work (Blundell et al. 2015). The more implementation details, we refer the reader to the original paper. However, BNNs face convergence challenges when the model parameter size is too large. To address this issue, we propose an approximate Bayesian framework in Section 2.3.

2.2. *ConViT*

ConViT (d'Ascoli et al. 2022) is a deep-learning method, which improved self-attention operators with soft convolutional inductive biases in the computer vision field. In this section, we briefly introduce basis operators and the architecture of the ConViT, which serves as the backbone of our model.

Operators: This section introduces the self-attention (SA) operator and its variants, which are the building blocks of ConViT. The self-attention operator (Vaswani et al. 2017) has been widely used in computer vision. It maps a set of query, key and value vectors to an output by mainly computing an attention matrix, where query, key and value are obtained through separate linear projections. Q, K, V denotes query, key, and value; $W_{\text{query}} \in \mathbb{R}^{D_{\text{emb}}\times D_{\text{out}}}$, $W_{\text{key}} \in \mathbb{R}^{D_{\text{emb}}\times D_{\text{out}}}$, $W_{\text{value}} \in \mathbb{R}^{D_{\text{emb}}\times D_{\text{out}}}$ denotes linear weights, where D_{emb} is the dimensions of input channels; D_{out} is the dimensions of target channels. Hence, for an embedding $X \in \mathbb{R}^{L\times D_{\text{emb}}}$ as an input, we have $Q = XW_{\text{query}}$, $K = XW_{\text{key}}$ and $V = XW_{\text{value}}$, where L is the first dimensions of X and will be set in Section 2.3. Then the attention matrix A and SA operator are defined as follows:

$$A = \text{softmax}\left(\frac{QK^T}{\sqrt{D_{\text{emb}}}}\right)$$
$$\text{SA}(X) = AV \tag{9}$$

where $(\text{softmax}[X])_{ij} = \exp X_{ij}/\sum_k \exp X_{ik}$.

However, the experiment (Dosovitskiy et al. 2021) shows that the SA performs similarly regardless of how the position prior is provided, but when the position prior is missing, the performance drops significantly. Thus, it is crucial to incorporate position information into the model. A common method is positional self attention (PSA), which uses relative position encoding of patches $X_{i,:}$ and $X_{j,:}$ to enhance the attention matrix (Ramachandran et al. 2019):

$$\tilde{A}_{ij} = \text{softmax}(Q_i K_j^T + \boldsymbol{v}_{\text{pos}}^T \boldsymbol{r}_{ij}) \tag{10}$$

where $X_{r,:}$ represent the rth row of X, $\boldsymbol{v}_{\text{pos}}^T \in \mathbb{R}^{D_{\text{pos}}}$ is a trainable embedding, and $\boldsymbol{r}_{ij} \in \mathbb{R}^{D_{\text{pos}}}$ is the position embedding between patches $X_{i,:}$ and $X_{j,:}$, only subject to the relative distance. D_{pos} is the position embedding dimension, which relies on the approach of encoding. Hence, the PSA operator as follows:

$$\text{PSA}(X) = \tilde{A}V. \tag{11}$$

The multihead self-attention (MHSA) uses different self-attention operators to extract various semantic features of an image in parallel (each segment is called a head). They have the dimension of each embedding $D_{\text{emb}} = N_h D_h$, where N_h is the number of self-attention operators in parallel. The output of the MHSA is obtained through the following mechanism:

$$\text{MHSA}(X) = C[\text{SA}_h(X)]W_{\text{out}} + b_{\text{out}} \tag{12}$$

where $W_{\text{out}} \in \mathbb{R}^{D_{\text{emb}} \times D_{\text{emb}}}$, $b_{\text{out}} \in \mathbb{R}^{D_{\text{emb}}}$. The hth self-attention $\text{SA}_h(X)$ follows Equation (9), and the shape of shared weights is $W^h_{\text{query/key/value}} \in \mathbb{R}^{D_{\text{emb}} \times D_h}$. C is an operator to concat each output of various self attention.

Similarly, the positional MHSA (P-MHSA) shares an almost identical structure with MHSA. The only difference is that P-MHSA substitutes $\text{SA}_h(x)$ in MHSA with $\text{PSA}_h(x)$, which is as follows:

$$P - \text{MHSA}(X) = C[\text{PSA}_h(X)]W_{\text{out}} + b_{\text{out}} \tag{13}$$

where the hth positional self-attention $\text{PSA}_h(X)$ follows Equation (11), the definitions of other parameters remain consistent with those mentioned above.

It has been proved that Equation (13) of dimension $D_{\text{pos}} \geqslant 3$ can express any convolutional layer of kernel size $\sqrt{N_h} \times \sqrt{N_h}$ and $\min(D_h, D_{\text{out}})$ output channels (Cordonnier et al. 2020), when following the encoding:

$$\boldsymbol{v}_{\text{pos}}^{(h)} = -\alpha^{(h)}(1, -2\Delta_1^{(h)}, -2\Delta_2^{(h)})$$
$$\boldsymbol{r}_\delta = (\|\delta\|^2, \delta_1, \delta_2)$$
$$W_{\text{qry}} = W_{\text{key}} = \boldsymbol{0};\ W_{\text{value}} = \boldsymbol{I} \tag{14}$$

where trainable parameters $\Delta^{(h)} = (\Delta_1^{(h)}, \Delta_2^{(h)})$ is a coordinate, which is used to mark the center of attention in each head. α is a scale coefficient to indicate locality strengths. $\boldsymbol{r}_\delta = (\delta_1, \delta_2)$ is fixed, which represent the relative shift between query and key pixels in two-dimensional space.

In view of the above convolution-equivalence theorem, we can construct the main operator (GPSA) in ConViT as follows:

$$\text{GPSA}_h(X) = \text{normalize}[A^h]V^h$$
$$A_{ij}^h = (1 - \sigma(\lambda_h))\text{softmax}(Q_i^h K_j^{hT})$$
$$+ \sigma(\lambda_h)\text{softmax}(\boldsymbol{v}_{\text{pos}}^{hT}\boldsymbol{r}_{ij}) \tag{15}$$

where $(\text{normalize}[A])_{ij} = A_{ij}/\Sigma_k A_{ik}$ and $\sigma\colon x \mapsto 1/(1 + \exp^{-x})$ is the sigmoid funciton. $\boldsymbol{v}_{\text{pos}}^h$, $\boldsymbol{r}_{ij}$ is set by Equation (14). A_{ij}^h consists of vanilla SA (left term) and a position embedding (right term). λ is used to control the relative magnitude of self-attention against position embedding. In the light of Equation (14), the position embedding is a convolution-equivalence term, which provides convolutional inductive biases to SA.

Architecture: Stemming from the GPSA, we can present the entire ConViT. The architecture of the ConViT are showed in Figure 1. ConViT propagates the input through 12 blocks, and each block in the first ten consists of a GPSA layer followed by two-layer feed-forward networks with GeLU activation, both equipped with residual connections. The last two blocks recover the GPSA layer by vanilla MHSA layer.

2.3. An Approximate Bayesian Framework and Instantiation

To address the challenge of BNNs' convergence on large-scale models, we propose a new framework, which approximates the probabilistic inference and integrates ConViT for image regression tasks.

Approximate Bayesian Framework: In vanilla probabilistic modeling for neural networks, each weight in the layers of the neural networks is sampled from a given probability distribution, and then the output determined by weights is fed to the next layer. Thus, for the deep probabilistic model, each forward result is equivalent to a drawing from a complex distribution that integrates numerous base distributions (e.g., Gaussian distribution). For an input X, this can be denoted as:

$$\text{Output}_i = \mathcal{H}_i(\mathcal{B}_i(X)) \tag{16}$$

where $\mathcal{B}_i$, $\mathcal{H}_i$ are the backbone and head of the model, which are equipped with ith weights sampled from the posterior distribution $P(\boldsymbol{w}|\mathcal{D})$. (We divide the model into two parts: the backbone, which extracts deep semantic features from the input, and the head, which maps the flattened feature map to the label space.)

Fortunately, in the procedure of inference, we just need to focus on the mean and variance of the output to derive the predictions and corresponding uncertainties, regardless of other details of the complex distribution. In addition, we argue that the most crucial role of variance is mainly reflected in relativity. Specifically, under the same dimensionality, if there exists a set of indicators that can preserve the relative information of variance, then this indicator can also measure the relative size of uncertainty; namely, the relative quality of data. That is, let a random variable X_i denotes $\mathcal{B}_i(X)$. If another random variable Y_i exists that satisfies Equation (17), then the two inference frameworks are equivalent when measuring relative uncertainty

$$\mathbb{E}(\mathcal{H}_i(X_i)) = \mathbb{E}(\mathcal{H}_i(Y_i))$$
$$\mathbb{D}(\mathcal{H}_i(X_i)) = g(\mathbb{D}(\mathcal{H}_i(Y_i))) \tag{17}$$

where $g(\cdot)$ is a strictly increasing function.

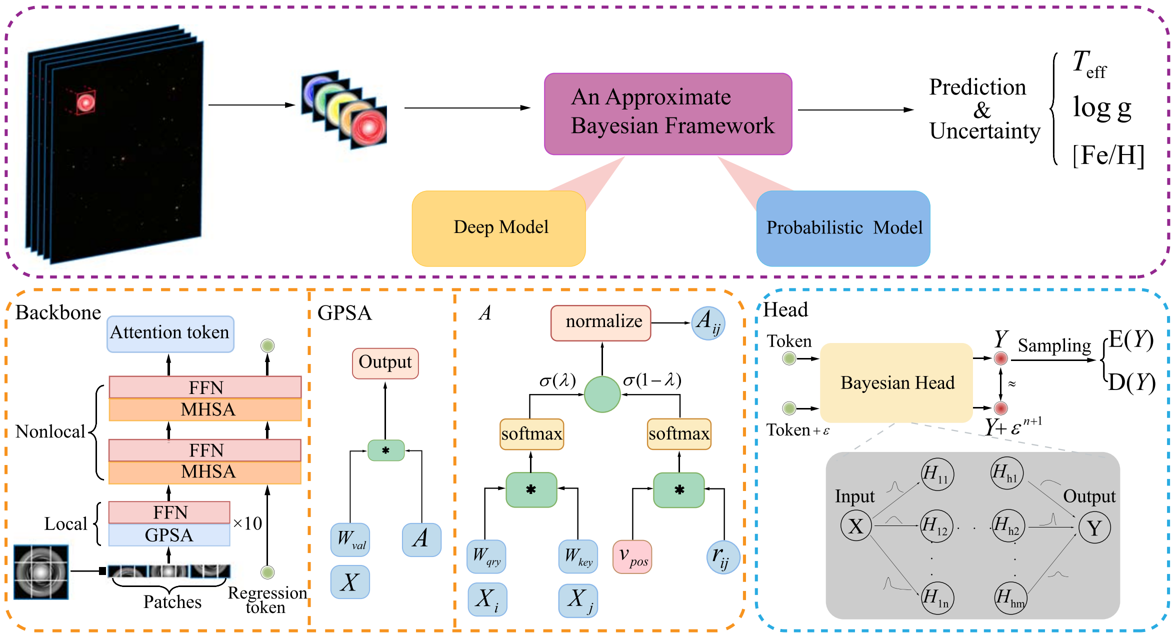


Figure 1. Feed-forward process of our model. Our approximate Bayesian framework consists of a deep model for extracting image features and a probabilistic model for producing outputs and uncertainties. In the deep model, we adopted the backbone of ConViT (bottom left), a version of ViT in that some of the MHSA layers are replaced with gated positional self-attention layers (GPSA; middle and right in orange frame). We use 10 GPSA layers followed by 2 vanilla MHSA layers. FFN: feed-forward network (two linear layers separated by a GeLU activation); $W_{\rm qry}$: query weights; $W_{\rm key}$: key weights; $v_{\rm pos}$: attention center and span embeddings (learned); r_{qk}: relative position encodings; λ: gating parameter; σ: sigmoid function. $E(Y)$ and $D(Y)$ are the sample mean and sample variance, respectively, of the random variable Y. In the probabilistic model, we adopted BNNs (bottom right) to accept the regression token from the backbone. We provide a theoretical analysis that MLP can approximately erode uncertainties (ϵ) from the backbone.

To make the original framework more controllable, we construct such $Y_i^{(n)}$ that $Y_i^{(n)} = 1\big/n\sum_{j=1}^{n} X_j$ (although the overall variance may decrease, the relative characteristics will be preserved), which satisfies Equation (17) (see Appendix A for the proof).

We then conduct probabilistic modeling by replacing X_i with $Y_i^{(n)}$, which ensures that Theorem 1 holds. Theorem 1 suggests that if we replace X_i with $Y_i^{(n)}$ and increase the number of multilayer linear perceptual (MLP) layers in the head, the final perturbation caused by the backbone can be reduced to higher order small terms that become negligible. The proof of Theorem 1 can be found in our supplementary material (Appendices B and C). Therefore, by revisiting the above probabilistic framework with multilayer linear layers in the head, we obtain the following approximation:

$$\zeta(F_b) + \zeta(F_h) \approx F_b + \zeta(F_h) \tag{18}$$

where F_b and F_h are point estimation models, and ζ is a probability factor for transforming point estimation models to probabilistic models. We then employ $F_b + \zeta(F_h)$ to approximate the characterization of relative uncertainties as our framework.

Instantiation: We ran an instance relying on the above framework in a real image task. First, we prepared the input for the model by dividing the image $X \in \mathbb{R}^{H\times W\times C}$ into several segments, which are a sequence of flattened 2D patches $X_p \in \mathbb{R}^{L\times(P^2C)}$. Here, (H, W) is the number of pixels in original image height and width, C is the number of channels. (P, P) represents the number of pixels in patches, where P is the height and width of each patch (16 × 16 pixels patches is put forward; Dosovitskiy et al. 2021), and $L = HW/P^2$ represents the length of sequence. Then we applied a trainable linear projection in X_p to transform the dimension P^2C into $D_{\rm emb}$. This gives us an embedding $X \in \mathbb{R}^{L\times D_{\rm emb}}$ as the input of model.

Second, we will construct an approximate Bayesian framework (shown in Figure 1). Based on the Equation (18), we employed Convit as F_b to extract deep feature. On the other hand, we employed BNNs as $\zeta(F_h)$ to associate labels with feature maps and represent the uncertainty of the entire model.

Finally, we discuss some practical details.

(i) About ConViT: We first trained a backbone based on point estimates. The mean values of the parameters in the last 50 rounds of the training process, before reaching the point of overfitting, were used as the weight expectation of the backbone. We also replaced the Layernorm with the Batchnorm, because we think that the relative sizes between the pixel values of the images in our problem have physical significance and should not be directly erased by the Layernorm.

(ii) About BNNs: We rewrite the BNNs objective function as follows

$$\theta'_* = \arg\min_{\theta} \boldsymbol{KL}[q(\boldsymbol{w_h}|\theta)||P(\boldsymbol{w_h}|\mathcal{B}(\mathcal{D}))] \tag{19}$$

where $\boldsymbol{w_h}$ is parameters of the head and $\mathcal{B}$ represents the backbone. In addition, in the feed-forward process of the head, we replace the output of each layer with the mean value of repeating output. The other details remain consistent with BNNs.

3. Data Introduction

This section describes our data sources and processing methods. We use stellar atmospheric parameters from APOGEE (Majewski et al. 2017), a large-scale spectroscopic survey that operates in the near-infrared (H-band) portion of the electromagnetic spectrum. APOGEE consists of two 300-fiber

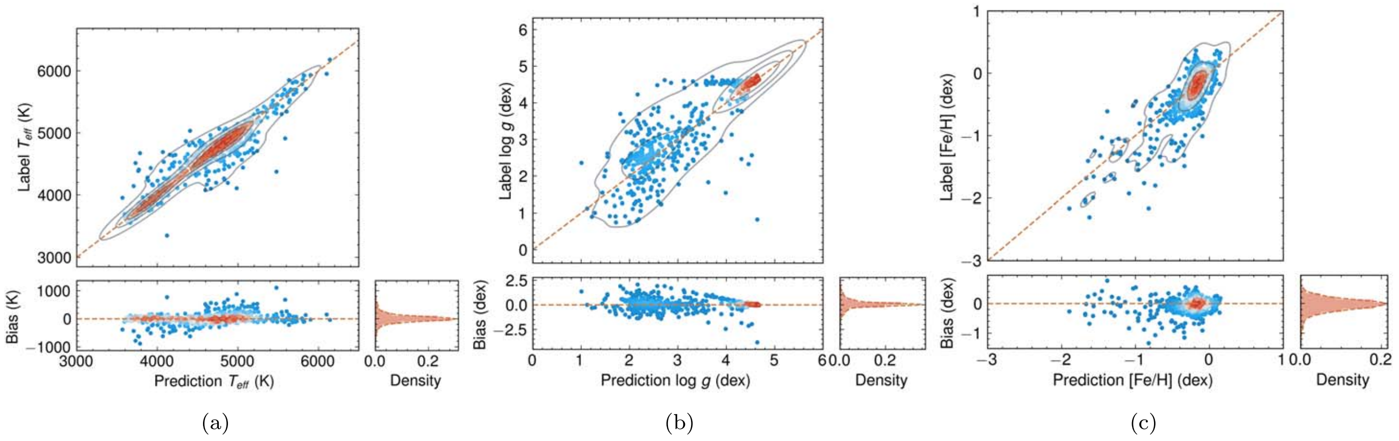


Figure 2. Regression results for three stellar atmospheric parameters. In order to more intuitively demonstrate the overall trend, we employ the kernel density estimation (KDE) method to fit the probability density of the result data and use it as the color gradient in the graph. Here, "Bias" refers to the residuals between the sample labels and the model predictions, while "Density" is the probability density fitted to the Bias using the KDE method (the same setting below). The results are (a) effective temperature ($T_{\rm eff}$), (b) surface gravity (log g), (c) metallicity ([Fe/H]). The figure shows that the $T_{\rm eff}$ and [Fe/H] estimates are good, with $\sigma(T_{\rm eff}) =$ 172.37 K, σ([Fe/H]) = 0.23 dex. But for $\log g$, an anomaly exists. A series of horizontal scatter points is observed near where the label equals 4 dex, and $\sigma(\log g) = 0.49$ dex.

cryogenic spectrographs mounted on two telescopes at Apache Point Observatory in New Mexico: a 2.5 m Sloan Foundation Telescope and a 1 m NMSU Telescope. APOGEE provides high-resolution ($R \sim 22{,}500$) spectra for stars across the Milky Way in the wavelength range of 15140–16940 Å.

We also use photometric images from SDSS (York et al. 2000), the imaging and spectroscopic survey that covers a wide range of wavelengths from 300 to 1000 nm. SDSS uses a dedicated camera with 30 CCDs (Gunn et al. 1998) that operates in Time Delay Integrate mode, also known as drift-scan mode. The camera collects data in five broad bands: *u*, *g*, *r*, *i,* and *z* (Fukugita et al. 1996). The image data undergo calibration for photometry and astrometry, and enables object selection based on unique criteria.

We obtained the photometric images (*u*, *g*, *r*, *i* and *z* bands) from SDSS DR16 and matched them to the APOGEE DR16 spectroscopic parameters ($T_{\rm eff}$, $\log g$, and [Fe/H]) of common stars.

To perform the regression task, we combine all of the bands to ensure that no implicit information is lost. We use CasJobs Server (Li & Thakar 2008) to cross-match the two catalogs, photoobj and apogeeStar, with the following criteria:

1. The image is not saturated (the saturated image contains a large number of missing values);
2. There are no interpolated pixels (due to cosmic rays or bad columns);
3. There are not too many candidate sources in the image (overcrowding can lead to photometric contamination).

The MySQL query[3] is given in Appendix D. We obtained 6074 stars from the database that meet our criteria. To ensure data consistency, we cross-matched the catalogs from CasJobs Server with the file (allStarLite-dr17-synspec.fits[4]) provided by APOGEE DR17 and found 5772 stars. We did not have to worry about version inconsistency, since APOGEE data releases are cumulative and include all the sky coverage of previous releases. We deleted the data with missing values in the catalogs and downloaded the fits file of each field using "run, camcol, and field" as in He et al. (2021), resulting in 4810 stars.

The SDSS survey telescope's CCD camera did not expose all filters simultaneously due to the 71.72 s drift-scan time between neighboring filters, which caused a coordinate shift in each band's images. We used PYTHON's reproject[5] to align the images in different bands to the *r*-band images using the world coordinate system keys in the fits headers.

Each field image contained many other stars, so we cropped our objects using R.A. and decl. and set a fixed size of 48 × 48 pixels. However, some target stars were close to other stars that could contaminate their luminosity. The sample also had some very faint stars that were hard to distinguish by eye, as well as a few broken images. After removing these abnormal images, we obtained 3884 stars for our data set. Finally, we split our data set into training and validation sets in a ratio of 8:2.

4. Experiment Results

We present the optimal result of Bayesian Convit for SDSS image data in this section. To train our model, we first used a vanilla Convit with batch normalization as the backbone. We fixed the average weights from the last 50 training rounds before the point of overfitting (excluding rounds with high volatility in metrics) as the final backbone parameters. Then we trained individual multilayer perception (in the manner of Bayesian neural networks) as the head of the model. We set the number of layers in the head to three in our experiment.

4.1. Precision of Regression

A detailed account of the entire model training process can be found in Appendix E. It suggests that the model performance across all components has nearly converged to the same level on both the training and validation sets. The results of Bayesian Convit for images on the validation set are shown in Figure 2. For SDSS images, we obtain $\sigma(T_{\rm eff}) = 172.37$ K, $\sigma(\log g) = 0.49$ dex and $\sigma([{\rm Fe/H}]) =$ 0.23 dex. We can clearly see that there is a horizontal row of points in the $\log g > 4$ dex part, which means that for the

[3] The query is available at https://skyserver.sdss.org/casjobs/.
[4] https://data.sdss.org/sas/dr17/apogee/spectro/aspcap/dr17/synspec/

[5] https://reproject.readthedocs.io/en/stable/

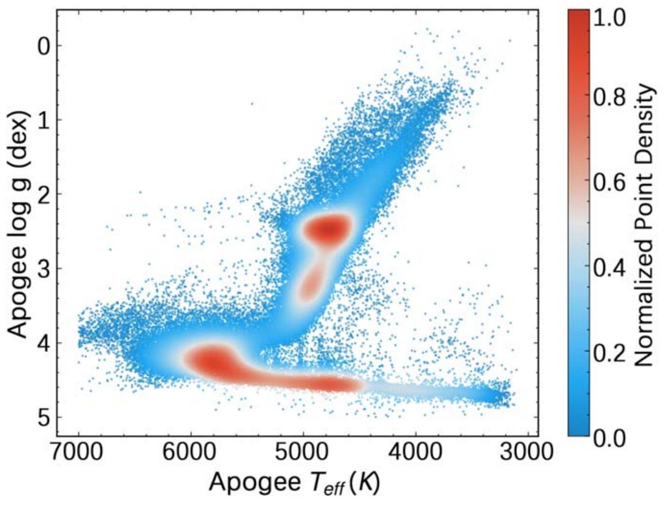


Figure 3. Distribution of APOGEE stars in the $T_{\text{eff}}-\log g$ plane. The colorbar utilizes the same color gradient as the scatter plot to represent point density, computed using the KDE method and normalized to a range from 0 to 1. We can clearly see that for different temperatures, there are two distinct $\log g$ values, mainly located in the regions above and below 4 dex. Moreover, the strong correlation between T_{eff} and photometry implies that there might be a similar relationship between images and $\log g$.

points with labels around 4 dex, the model predicts them as all results less than 4 dex. This indicates that in this part, the model's predictions are highly uncorrelated with the labels. However, this does not only happen in the predictions of this paper, we can likewise find similar phenomena in other articles, such as Liang et al. (2022). Thus, we think this is not an accidental model error, but should be the result of an essential systematic error. Fortunately we got inspiration from the diagram (shown in Figure 3) of T_{eff} and $\log g$.

Figure 3 shows that, for a given effective temperature, there may be two possible values of $\log g$ due to different stellar characteristics (such as dwarf versus giant). We hypothesize that a similar relationship exists between photometric data and $\log g$. However, a single model often struggles with discriminating different types of stars during the process of establishing data associations. When the model has learned the ambiguous, equational-like relationship, this leads to the problem we observe in Figure 2(b). To improve the estimation of $\log g$, which is more challenging from image data, we adopt the following strategy: we create artificial labels based on whether $\log g$ is greater or less than 4 dex and train an individual classification head for the model with the previously trained backbone frozen. Let class_i (0 or 1) denote the predictions. Then, we also freeze the backbone and train two regression heads for the model, one for each subset of $\log g$ (<4 dex or > 4 dex), and let h_1 and h_2 denote the predicted values based on different regression heads. The final output can be expressed as:

$$\text{output}_i = \text{class}_i * h_1 + (1 - \text{class}_i) * h_2. \tag{20}$$

We achieved 98.29% classification accuracy with the regression-based backbone. Figure 4(a) shows the regression results for the subset of $\log g > 4$ dex, where we obtained $\sigma(\log g > 4) = 0.052$ dex. This performance is comparable to that of spectral-based methods (Huang et al. 2020; Olney et al. 2020). Figure 4(b) shows the regression results for the subset of $\log g < 4$ dex, where we obtained $\sigma(\log g < 4) = 0.544$ dex. We speculate that the performance is worse because of the type of stellar characteristics in this subset is more complex. The surface gravity of a star depends on its mass (related to its luminosity) and radius, which are both affected by distance. However, it is hard to capture the distance information from a single star image. Our result suggests that the distance effect is more significant for stars with $\log g < 4$ dex, possibly owing to their larger size and higher luminosity compared with dwarf stars. Because of their larger size and higher luminosity, the upper distance limit of observable giant stars is generally larger than that of dwarf stars, where an increased distance variable space corresponds to higher complexity. We obtained $\sigma(\log g) = 0.36$ dex for the whole validation set based on artificial labels, but this assumes perfect classification accuracy. After obtaining two regression heads and one classifier, we estimated the $\log g$ for the whole validation set using Equation (20), as shown in Figure 4(c). The misclassification errors of the classifier would increase $\sigma(\log g)$ to 0.41 dex.

4.2. Attention of Backbone

We visualized the attention maps of Convit as shown in Figure 5, where warmer colors indicate higher attention and cooler colors indicate lower attention from the model. We find a similar trend in the feature extraction pattern for T_{eff} and $\log g$, i.e., more focus on the edge of the star. However, $\log g$ involves a larger area than T_{eff} in terms of attention. For [Fe/H], the region of model attention almost overlaps with the region of the stellar core. This means that for saturated stellar images, the estimation of stellar metallicity is most affected. Therefore, it is necessary to choose unsaturated images as data set.

4.3. Uncertainty of Regression

Uncertainty Analysis: Our framework provides uncertainty estimates associated with the prediction of stellar atmospheric parameters, as shown in Figure 6. We observe that for effective temperature, the uncertainty has a clear pattern. The network has lower uncertainty in the middle of the value range where the data are more abundant. As the values deviate from the center and approach the edges of the training range, the network's uncertainty increases significantly. The network becomes more cautious when predicting near the boundaries of its training data. For surface gravity, the uncertainty varies depending on whether $\log g$ is greater or less than 4 dex. In the part where $\log g > 4$ dex, there is an improved model fit between $\log g$ and the image. Thus, the model gives a higher confidence level. Similarly, in the part where $\log g < 4$ dex, the model has higher confidence at around $\log g = 2.5$ dex due to a relatively dense sample. However, the overall confidence for this part is not satisfactory because of its complexity, as we discussed in the previous section. For [Fe/H], the uncertainty depends on the metallicity. The model shows higher uncertainty for metal-poor stars due to the lack of samples. These three results are consistent with our intuition.

Perturbation Experiment: To test the robustness of our model, we performed two perturbation experiments. First, we added different levels of noise to the same image (shown in Figure 7) and observed how the model predictions changed with noise intensity. The results are shown in Table 1. We found that for $\log g$ and [Fe/H] estimation, the model was relatively insensitive to noise, as both prediction and uncertainty fluctuated slightly. However, for the T_{eff} estimation,

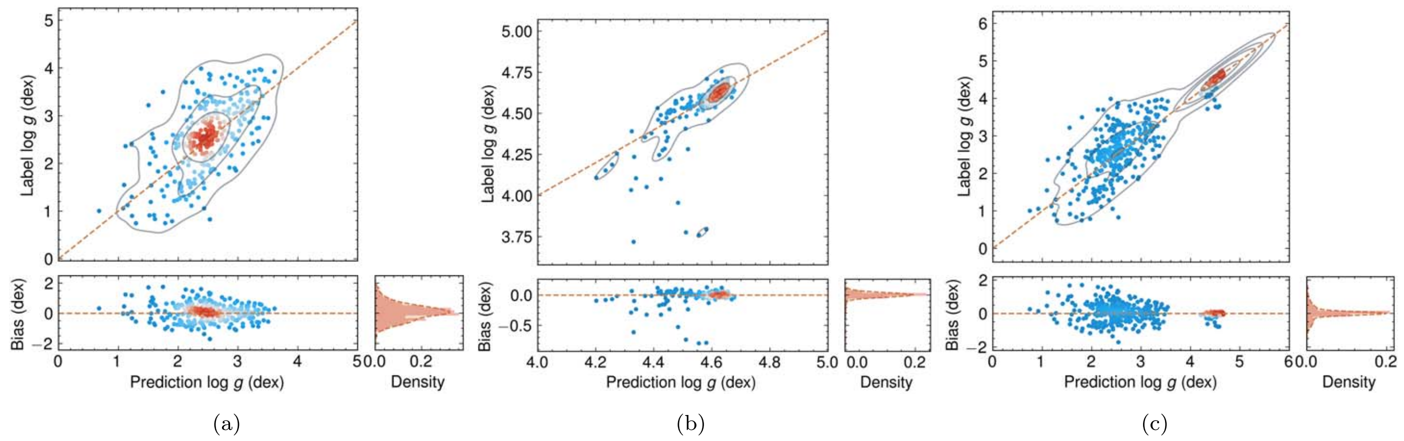


Figure 4. Results after introducing a classification stage. (a) The regression results for the stars with $\log g$ less than 4 dex. (b) The regression results for the stars with $\log g$ greater than 4 dex. (c) The final combined results. The figure shows that dividing the stars by $\log g$ can effectively eliminate the anomaly in the region with $\log g$ greater than 4 dex, resulting in a lower $\sigma(\log g) = 0.41$ dex.

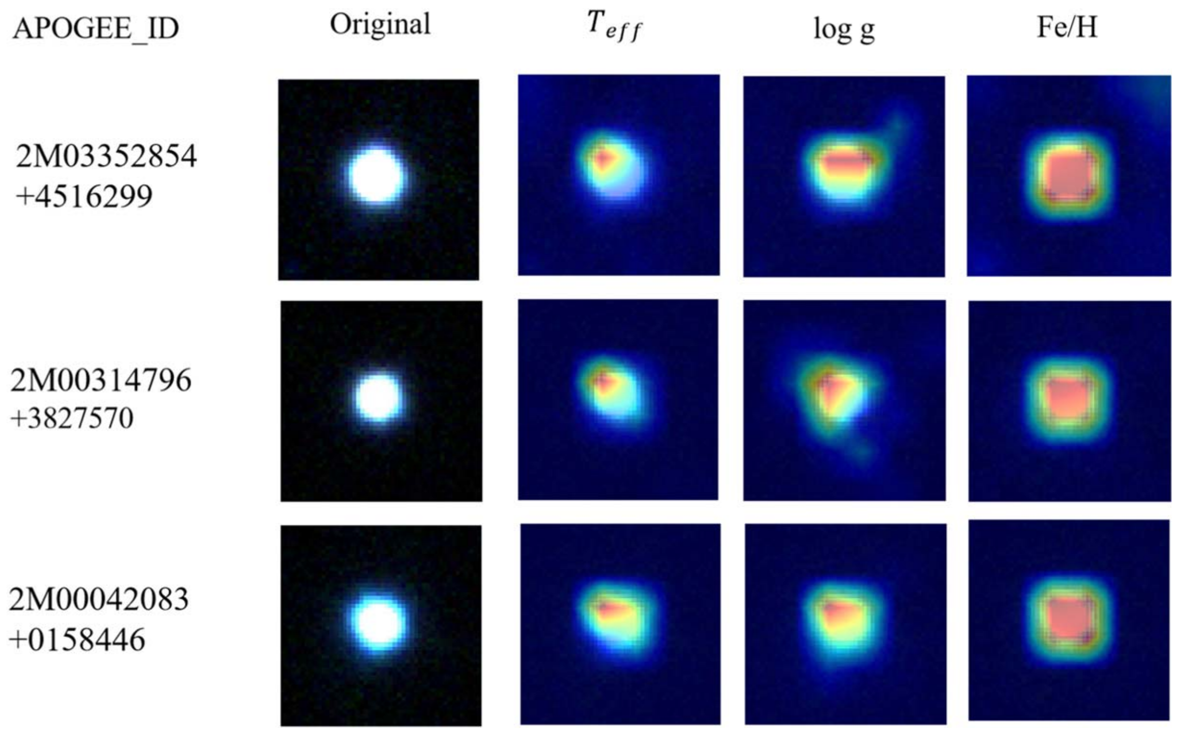
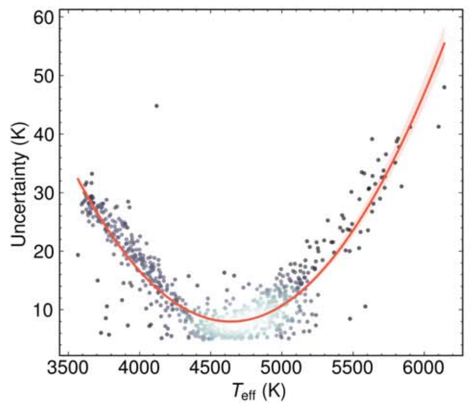


Figure 5. Attention distribution of ConViT over different regions. The warmer the color, the more attention the model pays. The model focuses more on the star's surroundings when estimating $T_{\rm eff}$ and $\log g$, but more on the star's core when estimating [Fe/H].

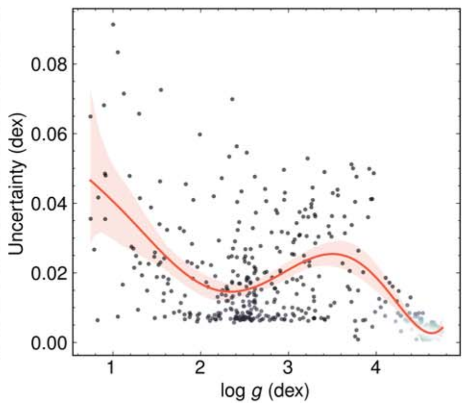
the model was more affected by high noise levels, and we provided uncertainty, although the change in uncertainty here is relatively small. This is because when the model was established, we replaced X with the random variable Y, resulting in a decrease in the magnitude of uncertainty (as we discussed in Section 2.3). However, relativity among uncertainties retains, so we can see that as the noise level increases, the uncertainty increases accordingly. These experiments demonstrate that our model can provide reliable confidence levels for its predictions, which reflect their relative quality. Second, we rotated the images at 30° intervals and fed them into the model. We plotted the model predictions and uncertainties as a function of rotation angle in Figure 8. In our observation, disturbances to model predictions may still occur unavoidably, even when random image rotation, a technique commonly used in data augmentation, is applied to the training data. This arises from the fact that traditional deep-learning frameworks (including CNN- and transformer-based models) fundamentally lack rotational invariance. However, compared to vanilla deep neural network models, our model has an additional function: it can provide a better characterization of these disturbances. That is, our model tended to give lower uncertainties and narrower 95% confidence intervals when the predictions were closer to the true labels. This can be employed to alert users about the current prediction of the model.

4.4. Comparison with Magnitude Provided by SDSS

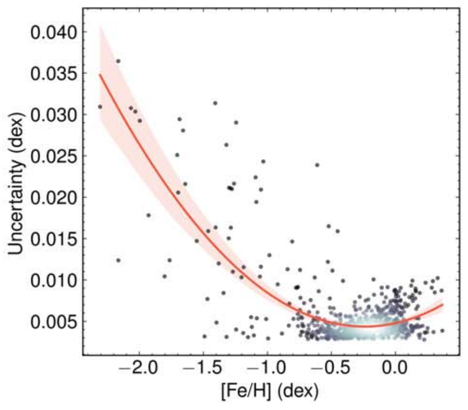
To evaluate whether image data has certain advantages over magnitude data, we obtained the extinction-corrected apparent



(a)

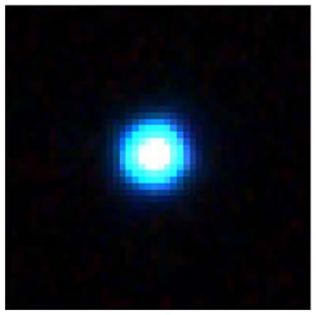


(b)

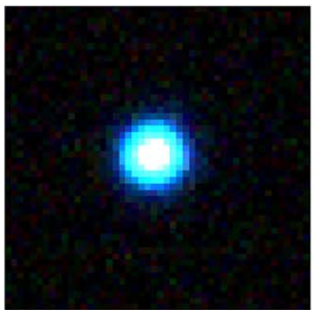


(c)

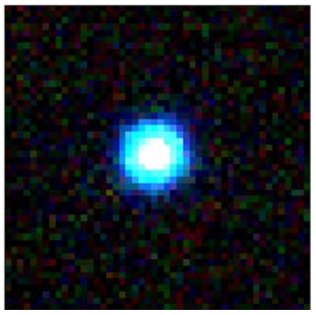
Figure 6. Model uncertainty. (a) Effective temperature T_{eff}, (b) surface gravity $\log g$, and (c) metallicity [Fe/H]. The uncertainty increases when the model predicts values outside the training data range. This trend is also observed when we estimate $\log g$ separately for different categories of stars.



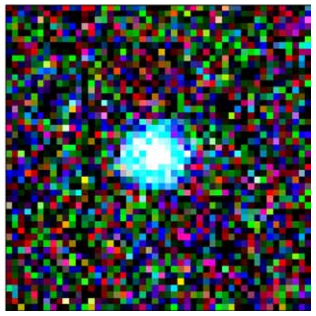
(epsilon=0.0)



(epsilon=0.05)



(epsilon=0.1)



(epsilon=0.5)

Figure 7. Add noise. Gaussian noise is added to the *gri* band, where epsilon indicates the relative magnitude of the noise.

Table 1
Noise Experiment

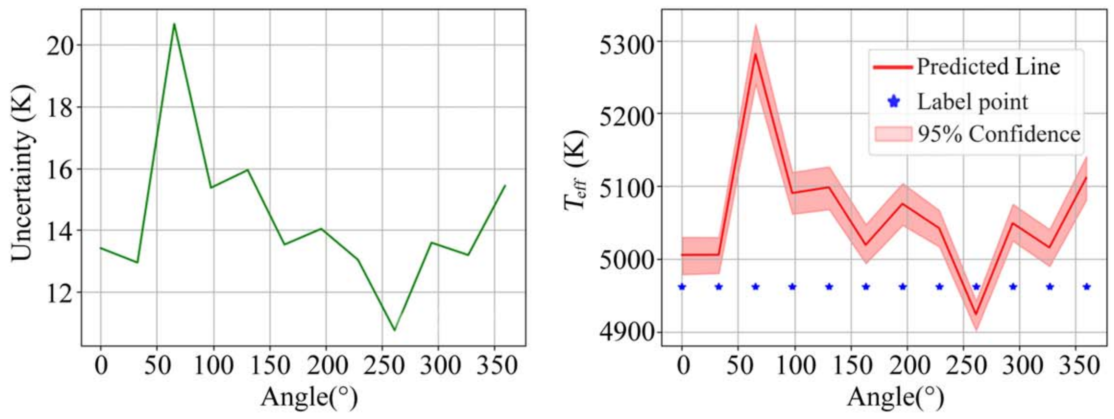
Parameter	epsilon = 0.05			epsilon = 0.1			epsilon = 0.5		
	T_{eff} (K)	$\log g$ (dex)	[Fe/H] (dex)	T_{eff} (K)	$\log g$ (dex)	[Fe/H] (dex)	T_{eff} (K)	$\log g$ (dex)	[Fe/H] (dex)
Prediction	4635.49	4.58	−0.09	4636.95	4.58	−0.08	4683.07	4.58	0.00
Uncertainty	5.24	0.00	0.01	5.41	0.00	0.01	5.89	0.00	0.01
Label	4632.31	4.58	−0.11	4632.31	4.58	−0.11	4632.31	4.58	−0.11

magnitudes for each band (u, g, r, i, z) from the SDSS official website[6] to facilitate comparative experiments. Moreover, considering the potentially stronger relationship between colors and stellar atmospheric parameters (Grady et al. 2021), we constructed the feature vector $f = [u-g, u-r, u-i, u-z, g-r, g-i, g-z, r-i, r-z, i-z]$ based on extinction-corrected magnitudes for an additional set of comparison experiments. We utilized the methods including XGBoost (Chen & Guestrin 2016), lightGBM (Ke et al. 2017), CatBoost (Prokhorenkova et al. 2018), and Random Forest (Breiman 2001) methods to model the data. The results of both sets of experiments are shown in Table 2. All models employed the Bayesian optimization method to determine the hyperparameters of the models. The search space for the hyperparameters, along with the optimal results, can be found in Appendix F. Our method outperforms those based on either magnitudes or colors for each prediction task. This outcome is anticipated, considering that tabular data are challenging to augment effectively and lack the detailed information that image data can provide. This also demonstrates the superiority of image data in determining stellar atmospheric parameters.

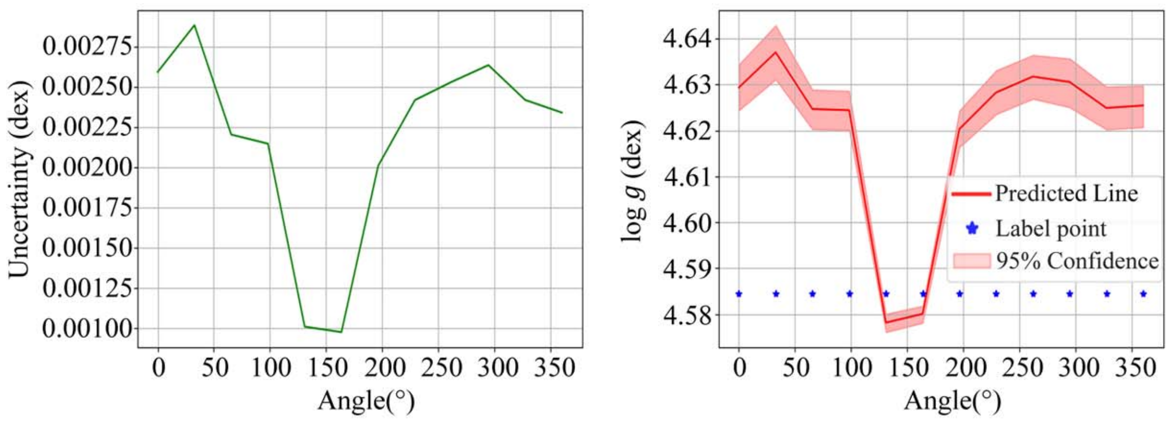
5. Conclusions

We present an approximate Bayesian framework, which improves the convergence of Bayesian neural networks on large models during training, and simplifies the inference process from sampling the whole model to sampling from only the head of the model. This significantly reduces the inference cost, and we provide a theoretical analysis. As pretrained large models (e.g., Du et al. 2022) continue to advance, more projects can fine-tune

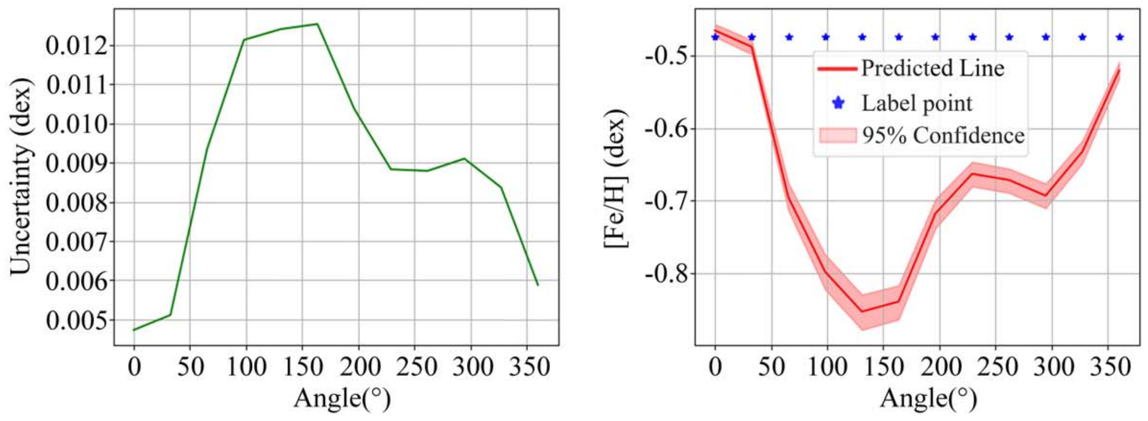
[6] https://www.sdss4.org/dr17/algorithms/magnitudes/#mag_psf



$T_{\rm eff}$



$\log g$



[Fe/H]

Figure 8. Uncertainty in rotating. We rotate the images by 30° in each step for a full circle to examine the effect of rotation on the model estimation and uncertainty. In the figure, the green (left) and red (right) curves represent the uncertainties and predicted values, respectively, as the outputs of the model, while blue points represent the labels. The red shaded area denotes the 95% confidence interval of the predictions; that is, the range from the 2.5% percentile to the 97.5% percentile based on multiple sampling results. The figure shows that the model results are affected by the rotation of the images. However, when the predicted values deviate significantly from the labels, our model tends to give a wider confidence interval with a higher uncertainty, and the transformation trend of the entire uncertainty curve is almost consistent with the deviation trend of the prediction, i.e., our model can account for this interference in a reasonable way.

these models for their specific downstream tasks. Our framework will be very useful and easy to apply when these tasks require confidence in the results.

We apply this method for the first time to directly estimate the three main stellar atmospheric parameters ($T_{\rm eff}$, $\log g$, [Fe/H]) from photometric images. In estimating $\log g$, we found that classifying $\log g$ into two categories (>4 dex or <4 dex) first can effectively improve the accuracy, and $\log g$ results with >4 dex can even be comparable to those based on spectral data. The final results are $\sigma(T_{\rm eff}) = 172.23$ K, $\sigma(\log g) = 0.41$ dex and σ([Fe/H]) = 0.23 dex. We provide confidence levels for all estimation results.

Table 2
Data Comparison Experiment

Methods	$T_{\rm eff}$ (K)	log g (dex)	[Fe/H] (dex)
XGBoost$_{\rm mag}$	265.10	0.81	0.31
LightGBM$_{\rm mag}$	234.98	0.80	0.30
CatBoost$_{\rm mag}$	241.68	0.79	0.29
RF$_{\rm mag}$	239.82	0.80	0.30
XGBoost$_{\rm color}$	189.73	0.66	0.25
LightGBM$_{\rm color}$	191.51	0.66	0.24
CatBoost$_{\rm color}$	189.54	0.65	0.24
RF$_{\rm color}$	188.59	0.65	0.24
Ours	172.37	0.41	0.23

Note. (1) Method$_{\rm mag}$: method based on magnitude data. (2) Method$_{\rm color}$: method based on color data.

For future work, we plan to apply target detection techniques to identify sources directly from field images, which can be integrated with our model to form a pipeline for automated and efficient estimation of stellar atmospheric parameters.

Acknowledgments

We thank the anonymous referee for useful comments that helped us improve the manuscript substantially. We would also like to thank Wei Du for providing us with useful comments. This work is supported by the National Natural Science Foundation of China (NSFC) under grant No. 11873037, the science research grants from the China Manned Space Project with No. CMS-CSST-2021-B05 and CMS-CSST-2021-A08, the Natural Science Foundation of Shandong Province under grant No. ZR2022MA076, and partially supported by the Young Scholars Program of Shandong University, Weihai (2016WHWLJH09) and GHfund A (202202018107). X.K. is supported by the National Natural Science Foundation of China under grant No. 11803016. Z.Y. is supported by the Shandong Province Natural Science Foundation grant No. ZR2022MA089.

Appendix A
Proof of Equivalence

If W is a random variable that represents the parameters of the head (due to the linearity of expectation, we can focus only on the product part), X is the output of each inference in the backbone, and Y is the mean values of n outputs, i.e., $Y^{(n)} = 1/n\sum_{i=1}^{n} X_i$. Then we have $\mathbb{E}(X) = \mathbb{E}(Y)$, and $\mathbb{D}(X) = n\mathbb{D}(Y)$ when n is large enough. We can illustrate this for a one-layer head:

(i) Expectation:

$$\mathbb{E}(\mathcal{H}(X)) = \mathbb{E}(XW) = \mathbb{E}(X)\mathbb{E}(W) = \mathbb{E}(Y)\mathbb{E}(W) = \mathbb{E}(\mathcal{H}(Y)). \tag{A1}$$

(ii) Variance:

$$\begin{aligned}\mathbb{D}(\mathcal{H}(X)) &= \mathbb{D}(XW)\\ &= \mathbb{E}(X^2W^2) - [\mathbb{E}(XW)]^2\\ &= \mathbb{E}(W^2)\mathbb{E}(X^2) - [\mathbb{E}(YW)]^2\\ &= \mathbb{E}(W^2)[\mathbb{D}(X) + \mathbb{E}(X)^2] - [\mathbb{E}(YW)]^2.\end{aligned} \tag{A2}$$

Let $\alpha = \mathbb{E}(W^2)$ and $\beta = \mathbb{E}(W^2)\mathbb{E}(X)^2 - [\mathbb{E}(YW)]^2$, we have:

$$\begin{aligned}\mathbb{D}(\mathcal{H}(X)) &= \alpha\mathbb{D}(X) + \beta\\ \mathbb{D}(\mathcal{H}(Y)) &= \alpha\mathbb{D}(Y) + \beta\\ \mathbb{D}(X) &= n\mathbb{D}(Y).\end{aligned} \tag{A3}$$

Then define $g(x) = nx - (n-1)\beta$, a strictly increasing function, it will meet:

$$\mathbb{D}(\mathcal{H}(X)) = g(\mathbb{D}(\mathcal{H}(Y))). \tag{A4}$$

Finally, the multilayer for linear head will follow the same proof one by one.

Appendix B
Theorem1

Theorem 1. *Suppose* $\mathcal{B}\colon \mathcal{X} \to L(\Omega, \mathcal{F})$, $\mathcal{H}\colon L(\Omega, \mathcal{F}) \to L(\Omega, \mathcal{F})$ *are operators that map samples into random variables, and* $\mathcal{B}_n(x) = 1/n\sum_{i=1}^{n} f_i(x)$, $\mathcal{H}_n(x) = 1/n\sum_{i=1}^{n} g_i(x)$. $f_i(x)$, $g_i(x)$ *are the results from backbone and head with* i*th stochastic weights, separately.* $\mathcal{Y}\colon \mathcal{X} \to L(\Omega, \mathcal{F})$ *maps each input sample to a corresponding point in the output space, which reflects the expected behavior of the whole model.(i.e.,* $\mathcal{H}\circ\mathcal{B}$*).* $\mathcal{Y}_b$ *is the expectation of* $\mathcal{B}$*. If* $P(\boldsymbol{w_h})$ *is a probability distribution with mean 0,* $P(\boldsymbol{w_h}|D)$ *can be limited by regularization trick. Then* $\forall\, \epsilon < 1$*:*

$$|\mathcal{Y} - \mathcal{H}[\mathcal{Y}_b]| < O(\epsilon\,(\epsilon_b)^{(n+1)}) \tag{B1}$$

where $L(\Omega, \mathcal{F})$ *is the random variable space,* $\mathcal{X}$ *is the sample space,* n *is the number of MLP layers in the head,* $\boldsymbol{w_h}$ *is the parameters of the head.*

Appendix C
Proof of Theorem1

Suppose $\mathcal{B}\colon \mathcal{X} \to L(\Omega, \mathcal{F})$, $\mathcal{H}\colon L(\Omega, \mathcal{F}) \to L(\Omega, \mathcal{F})$ are operators that map samples into random variables, and $\mathcal{B}_n(x) = 1/n\sum_{i\,=\,1}^{n} f_i(x)$, $\mathcal{H}_n(x) = 1/n\sum_{i\,=\,1}^{n} 1ng_i(x)$. $f_i(x)$, $g_i(x)$ are the results from backbone and head with ith stochastic weights, separately. $\mathcal{Y}\colon \mathcal{X} \to L(\Omega, \mathcal{F})$ maps each input sample to a corresponding point in the output space. The parameters of head $\boldsymbol{w_h} \sim N(\mu_{\rm Regular}, \sigma^2)$. $\mu_{\rm Regular}$, σ are the expectation and standard deviation of the distribution.

For the output of the backbone, based on the strong law of large numbers, suppose we have some regularization tricks limiting $\mu_{\rm Regular} \leqslant \frac{1}{2}\epsilon_h$, there is the following conclusion: $\forall \epsilon_h < 1$, $\exists N_b$, when $n > N_b$, $\mathcal{B}_n = 1/n\sum_{i=1}^{n} y_{bi}$, y_{bi} is the ith sample from backbone, $\mathcal{Y}_b$ is the expectation of $\mathcal{B}$ then

$$|\mathcal{Y}_b - \mathcal{B}_n| < \epsilon_h. \tag{C1}$$

This means $\exists \epsilon_b < \epsilon_h$, s.t. $\mathcal{Y}_b = \mathcal{B}_n \pm \epsilon_b$, then let $\mathcal{Y} = \mathcal{H}_n[\mathcal{Y}_b \mp \epsilon_b]$. The head part of the model is a linear mapping, and we simply use the following function to express: $g(x) = wx + b$, where w, b is the parameters ($\boldsymbol{w}_h$) of the head. We focus mainly on the multiplicative terms of the neural networks, rather than the additive terms. This is the additive terms are independent of the backbone, all the uncertainty of the additive terms comes from the head part, then we have:

$$\mathcal{Y} = \frac{1}{n}\sum_{i=1}^{N}[\mu_{\text{Regular}} \pm \epsilon_{\sigma i}][\mathcal{Y}_b] \mp \epsilon_b\sum_{i=1}^{N}\frac{1}{n}(\mu_{\text{Regular}} \pm \epsilon_{\sigma i})$$
$$= \mathcal{H}_{e1}[\mathcal{Y}_b] + \epsilon_0 \mp \epsilon_b\sum_{i=1}^{N}\frac{1}{n}(\mu_{\text{Regular}} \pm \epsilon_{\sigma i}) \tag{C2}$$

where $\epsilon_{\sigma i}$ is the ith stochastic bias determined by standard deviation σ in the head, $\mathcal{H}_{e1}[\mathcal{Y}_b]$ represents a multiplicative combination of expectation components in the first term, whereas ϵ_0 denotes a stochastic interaction between a random component and $\mathcal{Y}_b$ in the same term, and it is the uncertainty of head. Then according to the strong law of large numbers we know that, for ϵ_h, $\exists N_h$, when $n > N_h$

$$|\mu_{\text{Regular}} - \frac{1}{n}\sum_{i=1}^{n}(\mu_{\text{Regular}} + \epsilon_{\sigma i})| < \frac{1}{2}\epsilon_h. \tag{C3}$$

Also take $\epsilon'_h < \frac{1}{2}\epsilon_h$, s.t. $1/N_h\sum_{i=1}^{N_h}(\epsilon_{\sigma i}) = \pm\epsilon'_h$. Take $n = \max\{N_b, N_h\}$ then $\mathcal{Y} = \mathcal{H}_{e1}[\mathcal{Y}_b] + \epsilon_0 \pm (\epsilon_b\epsilon'_h + \epsilon_b\mu_{\text{Regular}}) \in [\mathcal{H}_{e1}[\mathcal{Y}_b] - O(\epsilon_h(\epsilon_b)^2) + \epsilon_0, \mathcal{H}_{e1}[\mathcal{Y}_b] + O(\epsilon_h(\epsilon_b)^2) + \epsilon_0]$. One more layer in head, we have:

$$\mathcal{Y} = \frac{1}{n}\sum_{i=1}^{N}[\mu_{\text{Regular}} \pm \epsilon_{\sigma i}][\mathcal{H}_{e1}[\mathcal{Y}_b] + \epsilon_0]$$
$$\pm (\epsilon_b\epsilon'_h + \epsilon_b\mu_{\text{Regular}})\sum_{i=1}^{N}\frac{1}{n}(\mu_{\text{Regular}} \pm \epsilon_{\sigma i})$$
$$\in [\mathcal{H}_{e2}\circ\mathcal{H}_{e1}[\mathcal{Y}_b] + \epsilon_0 - O(\epsilon_h(\epsilon_b)^3),$$
$$\mathcal{H}_{e2}\circ\mathcal{H}_{e1}[\mathcal{Y}_b] + \epsilon_0 + O(\epsilon_h(\epsilon_b)^3)]. \tag{C4}$$

Similarly, repeating the above steps, let u_b denotes the uncertainty involving backbone, then we can obtain: $\mathcal{Y} = \mathcal{H}_n\circ\mathcal{H}_{n-1}\circ \dots \circ\mathcal{H}_1[\mathcal{Y}_b] + u_b \in [\mathcal{H}_n\circ\mathcal{H}_{n-1}\circ \dots \circ\mathcal{H}_1[\mathcal{Y}_b] - O(\epsilon_h(\epsilon_b)^{n+1}), \mathcal{H}_n\circ\mathcal{H}_{n-1}\circ \dots \circ\mathcal{H}_1[\mathcal{Y}_b] + O(\epsilon_h(\epsilon_b)^{n+1})]$; that is

$$|\mathcal{Y} - \mathcal{H}[\mathcal{Y}_b]| < O(\epsilon_h(\epsilon_b)^{(n+1)}). \tag{C5}$$

Finally, to describe the output of the probabilistic model, we employ an approximation involving $\mathcal{H}[\mathcal{Y}_b]$ terms.

Appendix D
SQL Query

```
SELECT TOP 30000
photoobj.run, photoobj.camcol, photoobj.
field, photoobj.objID,
photoobj.ra as photoobj_ra, photoobj.dec as
photoobj_dec,
star.apstar_id, star.ra as star_ra, star.
dec as star_dec,
aspcap.teff into mydb.no_saturated_final
from apogeeStar AS star
CROSS APPLY dbo.fGetNearestObjEq(star.ra,
star.dec, 0.05) AS near
JOIN photoobj ON near.objid=photoobj.objid
JOIN aspcapStar as aspcap ON star.apstar_i-
d=aspcap.apstar_id
WHERE ((flags_u & 262144)=0) and ((flags_g &
262144)=0)
and ((flags_i & 262144)=0)
and ((flags_r & 262144)=0) and ((flags_z &
262144)=0)
and (aspcap.fe_h_flag=0)
and ((flags_u & 131072)=0) and ((flags_g &
131072)=0)
and ((flags_i & 131072)=0)
and ((flags_r & 131072)=0) and ((flags_z &
131072)=0)
and ((flags_u & 2048)=0) and ((flags_g &
2048)=0)
and ((flags_i & 2048)=0)
and ((flags_r & 2048)=0) and ((flags_z &
2048)=0)
```

Appendix E
Process in Training

We provided the learning curves of various components of the entire framework in Figure 9. This was done to evaluate the model's generalization capability.

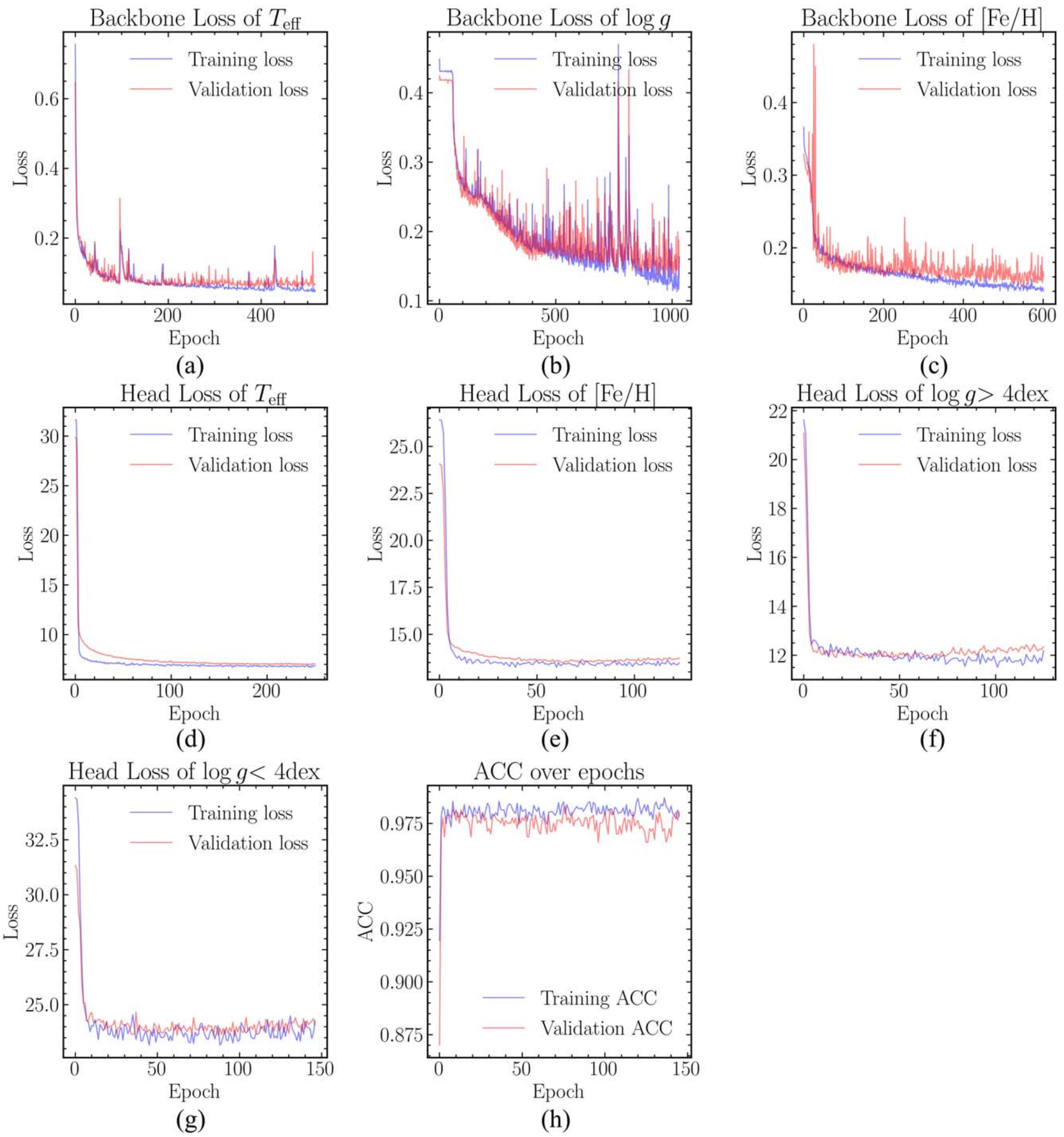


Figure 9. Learning curves. This figure shows the performance of each model component during the training process, as reflected in the training set and validation set. (a)–(c) depict the change in loss during the training of the backbones of each model, where the loss function employed is the smooth L1 loss. (d)–(g) detail the loss alteration during the training of different model head parts, where the loss function, as indicated in Equation (19), is used. (h) shows the changing trend of the accuracy (ACC) regarding the head of the $\log g$ classification during the training process. All training processes adopted data augmentation techniques (random rotation and flipping of images) and early stopping techniques to mitigate overfitting.

Appendix F
Model Configurations

We provided detailed information on determining hyperparameters for different machine learning algorithms based on the Bayesian optimization method, as shown in Tables 3–6.

Table 3
The Model Configurations of XGBoost using Bayesian Optimization

Hyperparameter	XGBoost$_{\text{mag}}$		XGBoost$_{\text{color}}$	
	Range[a]	Optimal[b]	Range[a]	Optimal[b]
colsample_bynode	(0.1, 0.9)	[0.83, 0.77, 0.51]	(0.1, 0.9)	[0.32, 0.78, 0.83]
colsample_bytree	(0.1, 0.9)	[0.68, 0.88, 0.90]	(0.1, 0.9)	[0.79, 0.73, 0.56]
gamma	(0, 1)	[0.57, 0.60, 0.00]	(0, 1)	[0.91, 0.91, 0.29]
learning_rate	(0.001, 0.5)	[0.29, 0.11, 0.10]	(0.001, 0.5)	[0.26, 0.01, 0.25]
max_depth	(1, 10)	[5, 4, 3]	(1, 10)	[9, 6, 3]
min_child_weight	(1, 10)	[8.63, 3.93, 2.42]	(1, 10)	[5.12, 2.23, 2.58]
n_estimators	(100, 1000)	[153, 904, 786]	(100, 1000)	[351, 543, 524]
reg_alpha	(0, 1)	[0.69, 0.88, 0.64]	(0, 1)	[0.25, 0.53, 0.23]
subsample	(0.5, 1)	[0.63, 0.54, 0.81]	(0.5, 1)	[0.59, 0.64, 0.62]

Notes.
[a] Range represents the search space of the hyperparameters (more details about the hyperparameters are available at https://xgboost.readthedocs.io/en/stable/parameter.html).
[b] Optimal consists of the optimal hyperparameters for each model estimating the parameters: T_{eff}, $\log g$, and [Fe/H].

Table 4
The Model Configurations of LightGBM using Bayesian Optimization

Hyperparameter	LightGBM$_{\text{mag}}$		LightGBM$_{\text{color}}$	
	Range[a]	Optimal[b]	Range[a]	Optimal[b]
lambda_l1	(0, 1)	[0.93, 0.99, 0.41]	(0, 1)	[0.80, 0.37, 0.53]
lambda_l2	(0, 1)	[0.83, 0.99, 1.00]	(0, 1)	[0.74, 0.58, 0.56]
learning_rate	(0.001, 0.5)	[0.08, 0.11, 0.10]	(0.001, 0.5)	[0.005, 0.04, 0.01]
max_depth	(1, 10)	[6, 7, 7]	(1, 10)	[5, 9, 5]
min_child_samples	(5, 20)	[6, 5, 8]	(5, 20)	[17, 16, 11]
min_data_in_leaf	(20, 100)	[26, 86, 83]	(20, 100)	[47, 75, 40]
n_estimators	(100, 1000)	[956, 607, 701]	(100, 1000)	[327, 131, 975]
num_leaves	(20, 100)	[35, 53, 86]	(20, 100)	[37, 86, 84]

Notes.
[a] Range represents the search space of the hyperparameters (more details about the hyperparameters are available at https://lightgbm.readthedocs.io/en/v3.3.2/Parameters.html).
[b] Optimal consists of the optimal hyperparameters for each model estimating the parameters: T_{eff}, $\log g$, and [Fe/H].

Table 5
The Model Configurations of CatBoost using Bayesian Optimization

Hyperparameter	CatBoost$_{\text{mag}}$		CatBoost$_{\text{color}}$	
	Range[a]	Optimal[b]	Range[a]	Optimal[b]
border_count	(1, 255)	[244, 254, 254]	(1, 255)	[156, 151, 95]
depth	(1, 10)	[6, 5, 7]	(1, 10)	[5, 9, 9]
l2_leaf_reg	(1, 10)	[2.05, 6.57, 4.98]	(1, 10)	[8.32, 3.39, 3.34]
learning_rate	(0.001, 0.5)	[0.18, 0.24, 0.14]	(0.001, 0.5)	[0.44, 0.18, 0.05]
n_estimators	(100, 1000)	[946, 472, 739]	(100, 1000)	[651, 189, 890]

Notes.
[a] Range represents the search space of the hyperparameters (more details about the hyperparameters are available at https://catboost.ai/en/docs/concepts/parameter-tuning).
[b] Optimal consists of the optimal hyperparameters for each model estimating the parameters: T_{eff}, $\log g$, and [Fe/H].

Table 6
The Model Configurations of RF using Bayesian Optimization

Hyperparameter	RF_{mag}		RF_{color}	
	Range[a]	Optimal[b]	Range[a]	Optimal[b]
max_depth	(1, 10)	[10, 10, 9]	(1, 10)	[4, 10, 9]
max_features	(0.1, 1)	[0.99, 0.99, 0.99]	(0.1, 1)	[0.63, 0.99, 0.84]
min_samples_leaf	(1, 10)	[1, 4, 4]	(1, 10)	[1, 1, 5]
min_samples_split	(2, 10)	[5, 2, 4]	(2, 10)	[4, 2, 9]
n_estimators	(100, 1000)	[183, 454, 970]	(100, 1000)	[405, 442, 950]

Notes.
[a] Range represents the search space of the hyperparameters (more details about the hyperparameters are available at https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html).
[b] Optimal consists of the optimal hyperparameters for each model estimating the parameters: $T_{\rm eff}$, $\log g$, and [Fe/H].

ORCID iDs

Yude Bu https://orcid.org/0000-0002-9474-4734
Zhenping Yi https://orcid.org/0000-0001-8590-4110
Meng Liu https://orcid.org/0000-0003-2442-2841
Xiaoming Kong https://orcid.org/0000-0002-4764-4749

References

Alonso, A., Arribas, S., & Martinez-Roger, C. 1996, A&A, 313, 873
Baschek, B., Scholz, M., & Wehrse, R. 1991, A&A, 246, 374
Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. 2017, JASA, 112, 859
Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. 2015, in Proc. of the 32nd Int. Conf. on Machine Learning (ICML-15) Vol. 30 ed. F. Bach & D. Blei (Lille: JMLR), 1613
Breiman, L. 2001, Mach. Learn., 45, 5
Bu, Y., & Pan, J. 2015, MNRAS, 447, 256
Buntine, W. L., & Weigend, A. S. 1991, Complex Syst., 5, 603, https://wpmedia.wolfram.com/uploads/sites/13/2018/02/05-6-4.pdf
Chen, T., & Guestrin, C. 2016, in Proc. of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, ed. K. Balaji, S. Mohak, & J. S. Alexander (San Francisco, CA: ACM), 785
Chiti, A., Frebel, A., Mardini, M. K., et al. 2021, ApJS, 254, 31
Cordonnier, J.-B., Loukas, A., & Jaggi, M. 2020, in Int. Conf. on Learning Representations (Virtual: OpenReview.net)
Dai, Z., Liu, H., Le, Q. V., & Tan, M. 2021, in Advances in Neural Information Processing Systems Vol. 34 ed. M. Ranzato, A. Beygelzimer, & Y. Dauphin (Red Hook, NY: Curran Associates, Inc.), 3965
d'Ascoli, S., Touvron, H., Leavitt, M., et al. 2022, JSMTE, 2022, 114005
Dosovitskiy, A., Beyer, L., Kolesnikov, A., et al. 2021, in Int. Conf. on Learning Representations (Virtual: OpenReview.net)
Du, Z., Qian, Y., Liu, X., et al. 2022, in Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics Vol. 1 ed. B. Luciana, O. Naoaki, S. Yves et al. (Dublin: ACL), 320
Fallows, C. P., & Sanders, J. L. 2022, MNRAS, 516, 5521
Fukugita, M., Ichikawa, T., Gunn, J. E., et al. 1996, AJ, 111, 1748
Grady, J., Belokurov, V., & Evans, N. W. 2021, ApJ, 909, 150
Graves, A. 2011, in Advances in Neural Information Processing Systems, Vol. 24, ed. J. Shawe-Taylor, R. Zemel, P. Bartlett et al. (Red Hook, NY: Curran Associates, Inc.), https://proceedings.neurips.cc/paper_files/paper/2011/file/7eb3c8be3d411e8ebfab08eba5f49632-Paper.pdf
Gunn, J. E., Carr, M., Rockosi, C., et al. 1998, AJ, 116, 3040
Halliday, D., Resnick, R., & Walker, J. 1988, Fundamentals of Physics (New York: Wiley), https://elearn.daffodilvarsity.edu.bd/pluginfile.php/987150/mod_label/intro/fundamentals-of-physics-textbook.pdf
He, K., Zhang, X., Ren, S., & Sun, J. 2016, in 2016 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (New York: IEEE), 770
He, Z., Qiu, B., Luo, A. L., et al. 2021, MNRAS, 508, 2039
Huang, Y., Beers, T. C., Wolf, C., et al. 2022, ApJ, 925, 164
Huang, Y., Chen, B. Q., Yuan, H. B., et al. 2019, ApJS, 243, 7
Huang, Y., Liu, X. W., Yuan, H. B., et al. 2015, MNRAS, 454, 2863
Huang, Y.-q., Zhong, J., & Hou, J.-l 2020, ChA&A, 44, 413
Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., & Saul, L. K. 1999, Machine Learning, 37, 183
Ke, G., Meng, Q., Finley, T., et al. 2017, in Advances in Neural Information Processing Systems, Vol. 30, ed. I. Guyon, U. Von Luxburg, S. Bengio et al. (Red Hook, NY: Curran Associates, Inc.), https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf
Kielty, C., Bialek, S., Fabbro, S., et al. 2018, AAS Meeting Abstracts, 232, 223.09
Li, N., & Thakar, A. R. 2008, CSE, 10, 18
Liang, J., Bu, Y., Tan, K., et al. 2022, AJ, 163, 153
MacKay, D. J. 1992a, Neural Computation, 4, 448
MacKay, D. J. C. 1992b, Neural Computation, 4, 415
Majewski, S. R., Schiavon, R. P., Frinchaboy, P. M., et al. 2017, AJ, 154, 94
Ness, M., Hogg, D. W., Rix, H. W., Ho, A. Y. Q., & Zasowski, G. 2015, ApJ, 808, 16
Olney, R., Kounkel, M., Schillinger, C., et al. 2020, AJ, 159, 182
Opper, M., & Archambeau, C. 2009, Neural Computation, 21, 786
Prokhorenkova, L., Gusev, G., Vorobev, A., Veronika Dorogush, A., & Gulin, A. 2018, in Advances in Neural Information Processing Systems Vol. 31 ed. S. Bengio, H. Wallach, H. Larochelle et al. (Red Hook, NY: Curran Associates, Inc.)
Ramachandran, P., Parmar, N., Vaswani, A., et al. 2019, in Advances in Neural Information Processing Systems Vol. 32 ed. H. Wallach, H. Larochelle, & A. Beygelzimer (Red Hook, NY: Curran Associates, Inc.)
Roy, A., & Clarke, D. 2003, Astronomy: Principles and Practice, (PBK) (Boca Raton, FL: CRC Press), https://www.academia.edu/25865657/Astronomy_Principles_and_Practice_4th_edition_A_E_Roy_D_Clarke
Shi, J.-H., Qiu, B., Luo, A. L., et al. 2022, MNRAS, 516, 264
Shridhar, K., Laumann, F., & Liwicki, M. 2019, arXiv:1901.02731
Southworth, J., Maxted, P. F. L., & Smalley, B. 2004, MNRAS, 351, 1277
Vaswani, A., Shazeer, N., Parmar, N., et al. 2017, in Advances in Neural Information Processing Systems Vol. 30 ed. I. Guyon, U. Von Luxburg, S. Bengio et al. (Red Hook, NY: Curran Associates, Inc.)
Welling, M., & Teh, Y. W. 2011, in Proc. of the 28th Int. Conf. on Machine Learning (ICML-11), ed. G. Lise & S. Tobias (New York: ACM), 681, http://www.icml-2011.org/papers/398_icmlpaper.pdf
Xu, S., Yuan, H., Niu, Z., et al. 2022, ApJS, 258, 44
Yang, L., Yuan, H., Xiang, M., et al. 2022, A&A, 659, A181
York, D. G., Adelman, J., Anderson, J. E. J., et al. 2000, AJ, 120, 1579