RAA 2021 Vol. **21** No. **7**, 169(16pp) doi: 10.1088/1674-4527/21/7/169

http://www.raa-journal.org *http://iopscience.iop.org/raa*

***R**esearch in*
***A**stronomy and*
***A**strophysics*

A novel stellar spectrum denoising method based on deep Bayesian modeling

Xin Kang (康欣)[1], Shi-Yuan He (贺诗源)[1] and Yan-Xia Zhang (张彦霞)[2]

[1] Institute of Statistics and Big Data, Renmin University of China, Beijing 100872, China; *heshiyuan@ruc.edu.cn*
[2] CAS Key Laboratory of Optical Astronomy, National Astronomical Observatories, Beijing 100101, China; *zyx@bao.ac.cn*

Received 2020 December 9; accepted 2021 February 26

Abstract Spectrum denoising is an important procedure for large-scale spectroscopical surveys. This work proposes a novel stellar spectrum denoising method based on deep Bayesian modeling. The construction of our model includes a prior distribution for each stellar subclass, a spectrum generator and a flow-based noise model. Our method takes into account the noise correlation structure, and it is not susceptible to strong sky emission lines and cosmic rays. Moreover, it is able to naturally handle spectra with missing flux values without ad-hoc imputation. The proposed method is evaluated on real stellar spectra from the Sloan Digital Sky Survey (SDSS) with a comprehensive list of common stellar subclasses and compared to the standard denoising auto-encoder. Our denoising method demonstrates a superior performance to the standard denoising auto-encoder, in respect of denoising quality and missing flux imputation. It may be potentially helpful in improving the accuracy of the classification and physical parameter measurement of stars when applying our method during data preprocessing.

Key words: methods: data analysis — methods: numerical — methods: statistical — techniques: spectroscopic

1 INTRODUCTION

With the rapid improvement of astronomical observation technology, modern large-scale sky surveys, such as the Sloan Digital Sky Survey (SDSS; Ahumada et al. 2020) and the Large Sky Area Multi-Object Fiber Spectroscopic Telescope (LAMOST or Guo Shou Jing Telescope; Cui et al. 2012), provide an unprecedented amount of astronomical data and enable us to explore our universe. The immense volume of astronomical data not only offers new opportunities but also brings challenges.

A fundamental data processing task is spectrum denoising when handling spectra. To clean astronomical spectra, a wavelet is a standard tool. The wavelet shrinkage method applies the wavelet transform to noisy observations, shrinks wavelet coefficients by some soft-thresholding or hard-thresholding rules, and takes the inverse wavelet transform to estimate the signal (Donoho 1993; Donoho & Johnstone 1994, 1995). Machado et al. (2013) developed a wavelet-based method for galaxy spectra and estimated their redshifts. An auto-encoder (Hinton et al. 2006; Vincent et al. 2010) is another popular denoising method in machine learning. Its success relies on allowing only limited information to pass through a bottleneck for spectrum reconstruction. Through minimizing the loss objective function, the encoder learns a feed-forward latent representation of its input, and the decoder reconstructs the signal from the latent space.

Despite the success of these algorithms, there are still several unresolved issues. A standard wavelet method and auto-encoder require a complete spectrum as the input. However, some spectral observations have missing flux values due to bad equipment conditions. Existing methods adopt an ad-hoc approach and directly impute the missing observations by some values (e.g. zero). In addition, the spectrum sometimes has a distorted shape and has a wavelength-connection problem, because the full spectrum is combined from the blue and red channels. For some spectra, the two parts are misaligned. Lastly, the flux values sometimes get contaminated by strong night-sky emission lines or cosmic rays in each individual exposure. They induce bias to the existing denoising algorithms.

This work proposes a novel model to address the above problems. Based on deep Bayesian modeling, this paper puts forward a stellar spectrum denoising method, which not only denoises spectra but also recovers the defective spectra. Section 2 presents the description of used data.

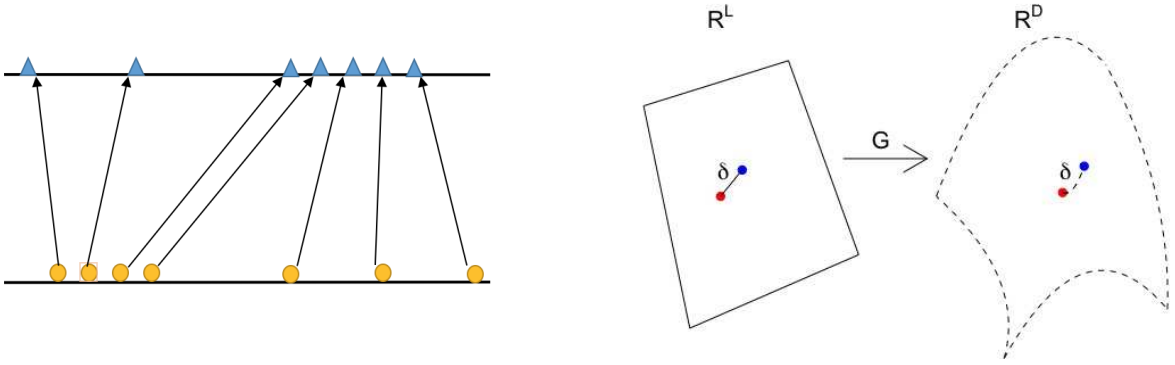

Fig. 1 The left subfigure illustrates a distorted mapping where the density estimation is affected. The right subfigure illustrates a locally isometric mapping where the local distance is preserved. Our generator $\mathcal{G}$ implements this isometry property to avoid distortion in the latent space.

Section 3 introduces our proposed model. The application and experimental results of our model are indicated in Section 4. We summarize and conclude our paper in Section 5.

2 DATA

The Sloan Digital Sky Survey (SDSS; Ahumada et al. 2020) has been the most successful sky survey project in the history, which now provides images, optical spectra, infrared spectra, IFU spectra, stellar library spectra, and catalog data. Current data release is Data Release 16 (DR16). This work is conducted based on the stellar spectral observations from SDSS DR16. In this study, we select a comprehensive list of common stellar subclasses for model training and evaluation: O-type, B-type, A-type, F-type, G-type, K-type, M-type, cataclysmic variables (CVs), carbon class and WD class (including CalciumWD, CarbonWD, WD, WDcooler and WDhotter).

Our proposed model will have several components: one prior distribution for each stellar subclass, a spectrum generator and a NoiseFlow observation model. See Section 3 for more details. For the training dataset of the spectrum generator, the top 200 spectra are selected among each stellar subclass sorted by the r-band signal-to-noise ratio (SNR). Each selected spectrum is normalized to have unit absolute flux summation, and is interpolated to a fixed uniform grid with the length of 2048 over the wavelength ranging from 4000 Å to 9000 Å. Meanwhile, the training dataset of the NoiseFlow observation model is prepared as follows. We extract the multiple-exposure data from the spectra with their SNR ranging from 20 to 30. For every exposure, we connect the red and blue parts of the spectrum, and apply the same interpolation and normalization as processing the training data. Then we subtract this composite spectrum from the average of multiple composite spectra. One subtracted spectrum from an exposure data becomes a training spectrum.

Test datasets also get prepared for model performance evaluation. We select an additional set of spectra with r-band SNR greater than 60. One test dataset consists of up to 300 spectra for each stellar subclass. We also select an additional group of spectra whose r-band SNR is between 10 to 20 to extract new noise from their multiple-exposure data. The final test dataset is constructed by randomly adding these realistic noises to the spectra with high SNR. The goal for the denoising model is to reconstruct the original clean spectra with high SNR.

3 THE DENOISING MODEL

3.1 Deep Bayesian Modeling

We now develop our proposed model for stellar spectrum denoising based on the basic Bayesian model (1)–(2). Suppose the signal spectrum of a star is $\mathbf{s} \in \mathbb{R}^D$. It is a D-dimensional unobserved vector, and we want to recover it from noisy observations. Modern astronomical surveys take multiple exposures to get several noisy observations $\mathbf{y}_1, \cdots, \mathbf{y}_n \in \mathbb{R}^D$ of the clean signal spectrum $\mathbf{s}$. We express the observations in a signal-plus-noise model,

$$\mathbf{y}_i = \mathbf{s} + \boldsymbol{\epsilon}_i, \quad \text{for } i = 1, \cdots, n,$$

where $\boldsymbol{\epsilon}_i$ is a D-dimensional noise vector. The noise could have complex correlation structure across pixels. Most of the time, only a single average spectrum is used, we can directly set $n = 1$ in the above. However, our method is more powerful and can exploit multiple-exposure data where only the red or blue part of the spectrum is recorded in each exposure. Our denoising framework is inspired by

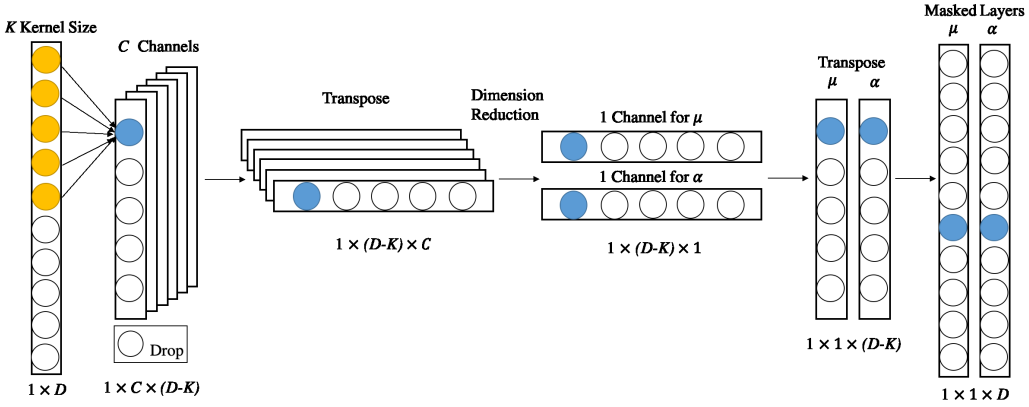

Fig. 2 The neural network architecture of our observation model. The *blue pixel* only depends on the immediately preceding five *yellow pixels* of the input spectrum. This proposed architecture is more parsimonious in parameterization and focuses on extracting local correlation structure of the noise.

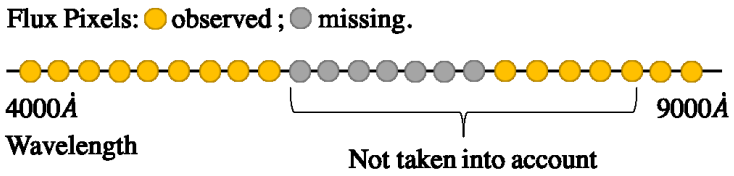

Fig. 3 The masked likelihood for a partially observed spectrum. The *yellow pixels* represent the observed pixel values and the *grey pixels* correspond to the missing ones. The likelihood of a pixel is accounted if and only if the d-th pixel and its immediate K preceding pixels are observed. Our observation model can naturally deal with spectra with missing values.

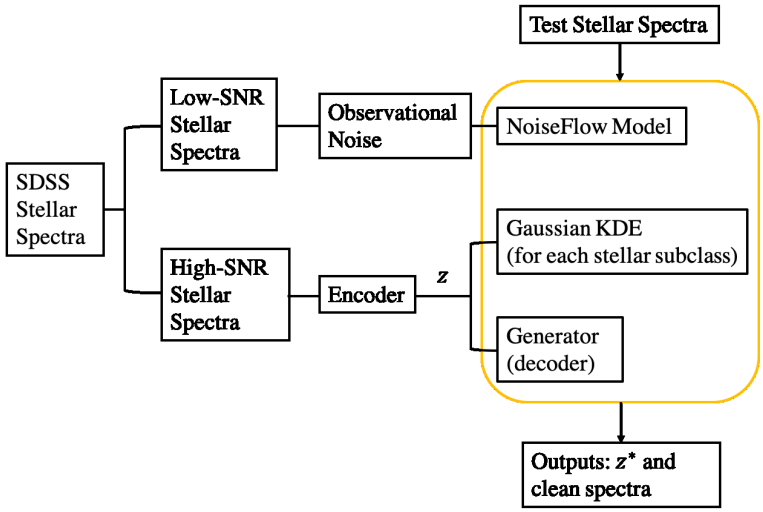

Fig. 4 The modeling workflow of our method. The core Bayesian model in the *yellow box* consists of three parts: one spectrum generator, one Gaussian KDE for each stellar subclass and one NoiseFlow model. Spectra with various levels of SNR are supplied for the training of different model components. With iterative optimization, it outputs the corresponding latent variables and the denoised spectra.

the following fundamental but powerful Bayesian model

$$\mathbf{y}_1, \cdots, \mathbf{y}_n | \mathbf{s} \sim p(\mathbf{y}|\mathbf{s}), \tag{1}$$

$$\mathbf{s} \sim p(\mathbf{s}). \tag{2}$$

In the above, $p(\mathbf{s})$ is a prior density encoding the likelihood of the signal $\mathbf{s}$; $p(\mathbf{y}|\mathbf{s})$ is the probability density of the observed $\mathbf{y}_i$ given the signal vector $\mathbf{s}$. Based on some state-of-the-art deep density estimation methods (see Sect. 3.2), we will construct an expressive prior $p(\mathbf{s})$ and

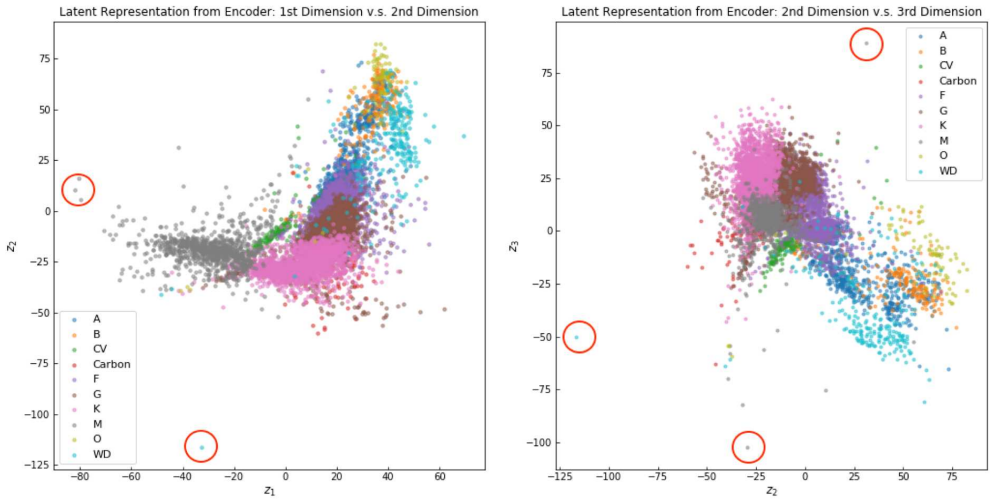

Fig. 5 The three dimensional space for the latent variable $\mathbf{z}$ outputted by the encoder. The horizontal axis in the left subfigure is z_1 and the vertical axis is z_2. The horizontal axis in the right subfigure is z_2 and the vertical axis is z_3. It is evident that observations from the same stellar subclass form a cluster in the latent space. The *red circles* indicate the outlier spectra. This learned latent space informs the latent variable structure for most stellar observations.

an observation model $p(\mathbf{y}|\mathbf{s})$ by neural networks. Given the trained model and the observations $\mathbf{y}_1, \cdots, \mathbf{y}_n$, the true signal vector $\mathbf{s}$ can be inferred from the posterior distribution $p(\mathbf{s}|\mathbf{y}_1, \cdots, \mathbf{y}_n)$.

Several additional adjustments of the model are necessary. It is not straightforward to model a clean spectrum $\mathbf{s} \in \mathbb{R}^D$ in a high dimensional space $\mathbb{R}^D$. To effectively construct a model for the signal spectrum, we exploit that, for a collection of astronomical spectra, the signal vectors $\mathbf{s}$ of various stars typically reside over a low dimensional manifold. This intrinsically low-dimensional structure can be effectively employed for spectrum data analysis. For example, Lawlor et al. (2016) used a local non-linear dimension reduction technique to discover the manifold structure. The low dimensional nature of the data implies that all clean spectra can be represented by a variable in a low dimensional space. Denote this low-dimensional variable by $\mathbf{z} \in \mathbb{R}^L$, and we can construct a mapping $\mathcal{G}$ such that the signal $\mathbf{s} = \mathcal{G}(\mathbf{z})$ is the mapped value of $\mathbf{z}$. Based on this mapping, the prior over the signal $\mathbf{s}$ can be directly expressed as a prior over the latent space $p(\mathbf{z})$. More specifically, we will take the stellar class C into account and construct the prior $p(\mathbf{s}|C)$ conditional on each stellar subclass. The final hierarchical model is summarized as below.

$$\mathbf{y}_1, \cdots, \mathbf{y}_n | \, \mathbf{s} \sim p(\mathbf{y}|\mathbf{s}), \tag{3}$$

$$\mathbf{s} = \mathcal{G}(\mathbf{z}), \tag{4}$$

$$\mathbf{z}|C \sim p(\mathbf{z}|C), \tag{5}$$

$$C \sim p(C). \tag{6}$$

In this model, the class label variable C has a uniform prior distribution over all stellar subclasses. The construction and training of the latent priors $p(\mathbf{z}|C)$ and the generator function $\mathcal{G}$ will be addressed in Section 3.3. The observation model $p(\mathbf{y}|\mathbf{s})$ will be discussed in Section 3.4. The final model training and denoising workflow will be summarized in Section 3.5.

Given our trained model in Equations (3)–(6), the posterior distribution of the latent variable $\mathbf{z}$ and the class label C is proportional to the joint distribution

$$p(\mathbf{z}, C|\mathbf{y}_1, \cdots, \mathbf{y}_n) \propto \prod_{j=1}^{n} p(\mathbf{y}_j|\mathcal{G}(\mathbf{z})) \times p(\mathbf{z}|C) \times p(C).$$

Monte Carlo Markov chain (MCMC) methods can be applied to draw samples from the posterior distribution. However, the MCMC technique is known to be computationally expensive and not scalable to large datasets. For large-scale modern astronomical surveys, we can adopt the faster maximum-a-posterior (MAP) estimation. In this way, the latent variables $\mathbf{z}$ and the class label C are found by

$$\hat{\mathbf{z}}, \hat{C} = \arg\max_{\mathbf{z}, C} \left\{ \sum_{j=1}^{n} \log p(\mathbf{y}_j|\mathcal{G}(\mathbf{z})) + \log p(\mathbf{z}|C) + \log p(C) \right\}. \tag{7}$$

With the computed MAP estimation $\hat{\mathbf{z}}$, the cleaned and denoised spectra are given by $\hat{\mathbf{s}} = \mathcal{G}(\hat{\mathbf{z}})$.

Fig. 6 The three panels correspond to the cases where the missing flux values occur at the left end, middle and right end of each spectrum, respectively. The horizontal axis represents the width of the missing range. The vertical axis is the reconstruction loss of our model. The reconstruction loss increases with the growing number of missing pixels. The model performance is not sensitive to the location of the missing pixels, but missing at the left end (blue end) incurs slightly higher loss as the blue end is feature-rich for most spectra.

3.2 Background on Deep Density Estimation

This subsection reviews some works on deep density estimation. These works serve as the basis for building our deep Bayesian denoising model. Suppose $\mathbf{x} \in \mathbb{R}^D$ represent a D-dimensional observation, for which we want to estimate its distribution. In the deep learning literature, most density estimation methods are based on the intuition that we can transform a simple base density $\pi_{\mathbf{z}}(\mathbf{z})$ into a more complex one $p(\mathbf{x})$ via an invertible differentiable transform $\mathbf{x} = f(\mathbf{z})$. The function f is parameterized by a neural network to achieve flexible and adaptive transformation. By the basic formula of density transformation, it holds that

$$p(\mathbf{x}) = \pi_{\mathbf{z}}(f^{-1}(\mathbf{x})) \left| \det \left(\frac{\partial f^{-1}}{\partial \mathbf{x}} \right) \right|. \tag{8}$$

Several methods have been proposed for deep density estimation. The approach of normalizing flows (Dinh et al. 2014) chooses f as a sequence of composite functions, i.e., $f = f_1 \circ f_2 \circ \cdots \circ f_K$. In this way, $p(\mathbf{x})$ can be regarded as an invertible and differentiable transformation f of a base density $\pi_{\mathbf{z}}(\mathbf{z})$. The base density can be simple multivariate Gaussian distribution. The relationship between the observation data $\mathbf{x}$ and latent variable $\mathbf{z}$ can be represented as $\mathbf{x} \overset{f_1}{\longleftrightarrow} \mathbf{h}_1 \overset{f_2}{\longleftrightarrow} \mathbf{h}_2 \cdots \overset{f_K}{\longleftrightarrow} \mathbf{z}$ via a stack of hidden variables. Under the invertible (or bijective) assumption of f, $\mathbf{z}$ can be calculated as $\mathbf{z} = f^{-1}(\mathbf{x})$ given $\mathbf{x}$. Based on Equation (8), the log probability density can be directly computed as

$$\begin{aligned} \log p(\mathbf{x}) &= \log \pi_{\mathbf{z}}(\mathbf{z}) + \log \left| \det \left(\frac{d\mathbf{z}}{d\mathbf{x}} \right) \right| \\ &= \log \pi_{\mathbf{z}}(\mathbf{z}) + \sum_{i=1}^{K} \log \left| \det \left(\frac{d\mathbf{h}_i}{d\mathbf{h}_{i-1}} \right) \right|, \end{aligned} \tag{9}$$

where the scalar value $\log |\det(d\mathbf{h}_i / d\mathbf{h}_{i-1})|$ is the logarithm of the absolute value of the determinant of the Jacobian matrix $(d\mathbf{h}_i / d\mathbf{h}_{i-1})$, also called the log-determinant. The series of transformations are required to be easily invertible and the log-determinant should be easy to compute. Rezende & Mohamed (2016) devised planar and radial flow as basic blocks for f. NICE (Dinh et al. 2014) adapts additive coupling layers to form a normalizing flow, and its successor Real NVP (Dinh et al. 2017) extends the transformation by stacking affine coupling layers. They all have a tractable triangular Jacobian matrix for the bijective mapping.

Autoregressive flow is another popular and tractable approach to density estimation. It factorizes the joint density as a product of conditional densities $p(\mathbf{x}) = \prod_d p(x_d | \mathbf{x}_{1:d-1})$ via the chain rule of probability (Uria et al. 2016). This factorization makes the Jacobian tractable as the Jacobian becomes a triangular matrix. The inverse autoregressive flow (IAF, Kingma et al. 2017) and the masked autoregressive flow (MAF, Papamakarios et al. 2018) take this approach. Under this approach, the prediction of current value depends on all of its past values, which is referred to as the autoregressive property. They use independent standard Gaussian distributions as the base density. The mean and variance of x_d are functions of the preceding observation vector $\mathbf{x}_{1:d-1}$ or the preceding random numbers. They both use MADE (Germain et al. 2015) as their basic building blocks for the function mapping.

3.3 The Generator and Latent Prior

We first detail our spectrum generator $\mathcal{G}$ and the prior density $p(\mathbf{z}|C)$ for the latent variable. The generator is obtained from the auto-encoder framework, but with an additional local isometry constraint. In the standard auto-encoder framework, an encoder $\mathcal{E}$ maps an observation $\mathbf{y}$ to the latent $\mathbf{z} = \mathcal{E}(\mathbf{y})$, and then maps $\mathbf{z} \in \mathbb{R}^L$ back to the original high-dimensional space $\mathbb{R}^D$ by the generator (decoder) $\mathcal{G}$. The training target is to minimize the reconstruction loss

$$\min_{\mathcal{G},\mathcal{E}} \mathbb{E}_{\mathbf{y}} \|\mathbf{y} - \mathcal{G}(\mathcal{E}(\mathbf{y}))\|^2.$$

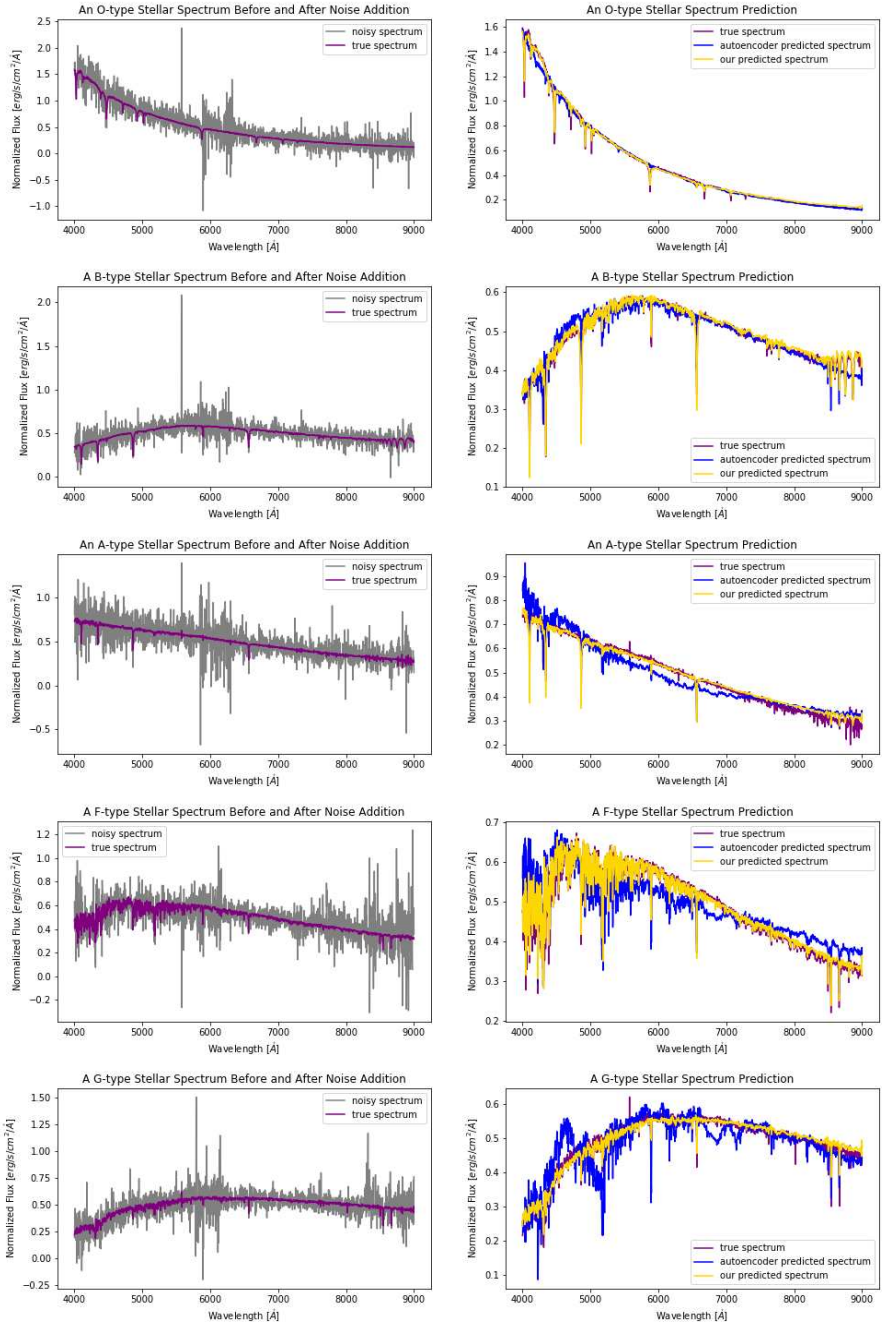

Fig. 7 The five rows show examples of stellar spectra denoised by the convolutional denoising auto-encoder and our method for the stellar subclasses O, B, A, F, G, respectively. The left column shows the true spectrum and its synthetic counterpart with noise. The *purple curve* is the clean spectrum that both denoising algorithms try to recover, and the *grey curve* is the spectrum added with noise. The spectra denoised by both algorithms are compared in the right column. The denoised spectrum from our model (*yellow curve*) is much closer to the purple clean spectrum than the standard auto-encoder result (*blue curve*).

The standard convolutional auto-encoder architecture can be specified for $\mathcal{E}$ and $\mathcal{G}$. However, the generator $\mathcal{G}$ obtained hereby could create distortion in the latent space $\mathbb{R}^L$, affecting density estimation. The left subfigure of Figure 1 shows an example of distorted mapping, where some low-density points (on the bottom right) are mapped to a high-density region (on the top right). To avoid the issue, we construct a *locally isometric* mapping such that $\mathcal{G}$ preserves the distance between samples in the latent space $\mathbb{R}^L$. In other words, it holds that

$$\|\mathcal{G}(\mathbf{z}) - \mathcal{G}(\mathbf{z}')\| \approx \|\mathbf{z} - \mathbf{z}'\|,$$

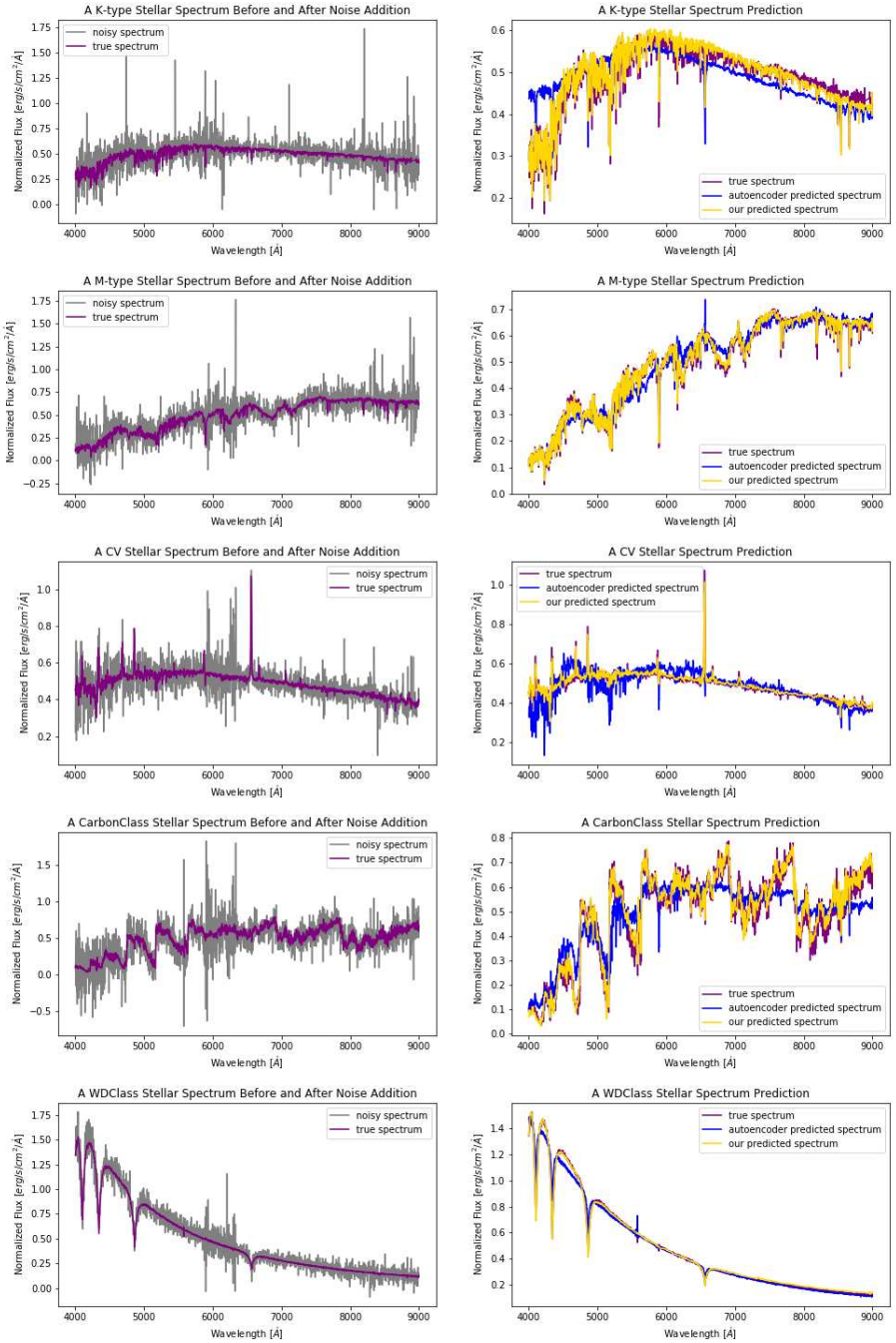

Fig. 8 The five rows show examples of stellar spectra denoised by the convolutional denoising auto-encoder and our method for the stellar subclasses K, M, CV, Carbon, WD, respectively. The left column shows the true spectrum and its synthetic counterpart with noise. The *purple curve* is the clean spectrum that both denoising algorithms try to recover, and the *grey curve* is the spectrum added with noise. The spectra denoised by both algorithms are compared in the right column. The denoised spectrum from our model (*yellow curve*) is much closer to the purple clean spectrum than the standard auto-encoder result (*blue curve*).

where $\mathbf{z}, \mathbf{z}' \in \mathbb{R}^L$ is a pair of latent variables satisfying $\|\mathbf{z} - \mathbf{z}'\| \leq \delta$ for some δ. The local isometry property is illustrated in the right subfigure of Figure 1, where the distance between the red and blue points is preserved under the mapping $\mathcal{G}$. The isometry property allows us to obtain the spectrum density $p(\mathbf{s})$ by directly accessing the low dimensional $\mathbf{z}$. It approximately holds that $p(\mathbf{s}) = p(\mathcal{G}(\mathbf{z})) = p(\mathbf{z})$. This helps us to approach the density estimation and effectively avoid the computation of $\det \frac{\partial f^{-1}}{\partial \mathbf{x}}$ for a complex transformation function f in Equation (8).

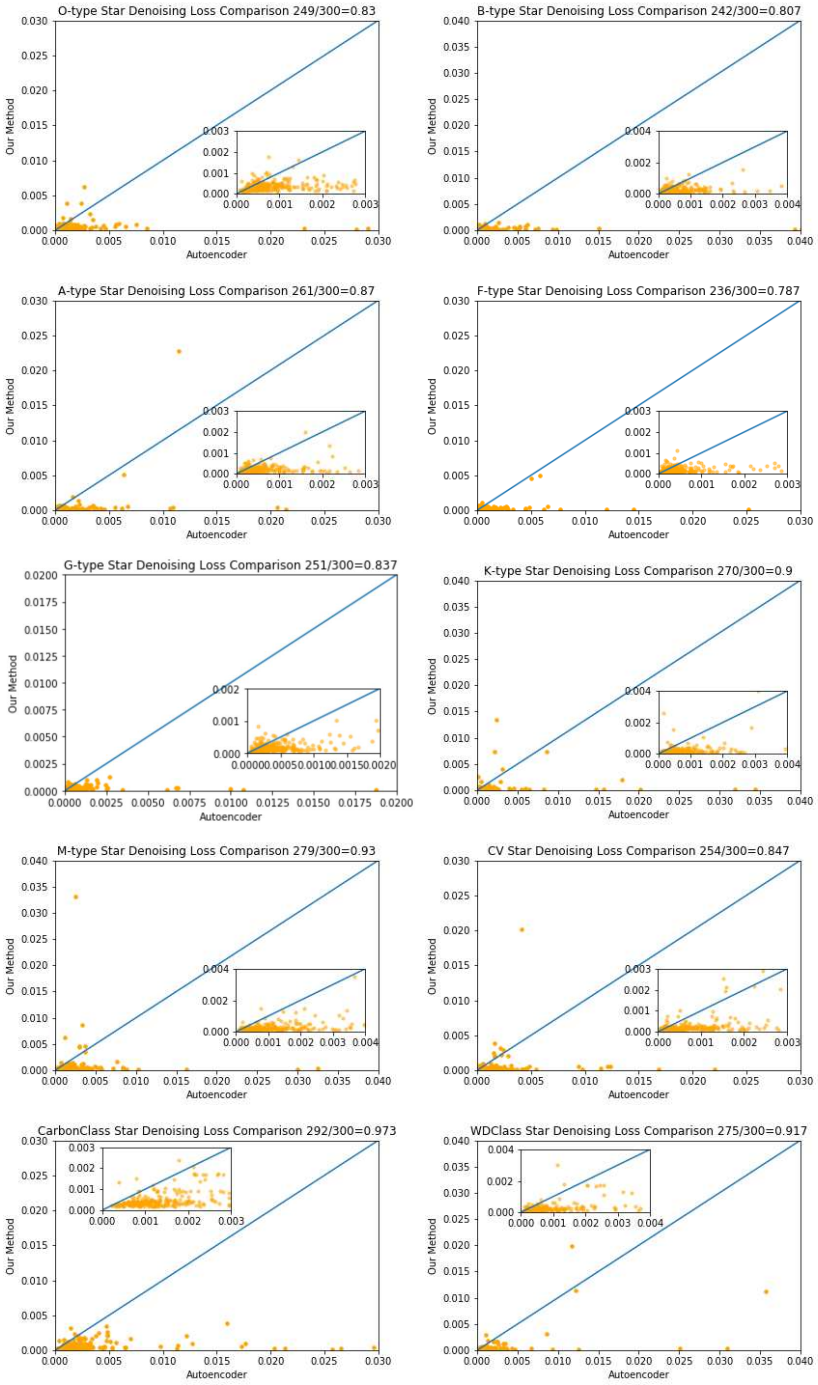

Fig. 9 The loss comparison for the convolutional auto-encoder and our method for the ten stellar subclasses. The horizontal axis is the reconstruction loss for the convolutional denoising auto-encoder, and the vertical axis is the reconstruction loss for our method. Each *yellow point* in a subfigure corresponds to one synthetic noisy spectrum. The *blue line* indicates where the two methods have equal performance. Most points in each subfigure fall below the *blue line*, indicating that our method has smaller reconstruction loss. The title of each subfigure also reports the proportion of spectra for which our method has smaller reconstruction loss. Our method demonstrates improved performance.

With the additional local isometry constraint, the objective function to train the generator becomes

$$\min_{\mathcal{G},\mathcal{E}} \mathbb{E}_{\mathbf{y},\boldsymbol{\delta}} \Big\{ \big[\|\mathcal{G}(\mathbf{z}) - \mathcal{G}(\mathbf{z}+\boldsymbol{\delta})\| - \delta \big]^2 + \|\mathbf{y} - \mathcal{G}(\mathcal{E}(\mathbf{y}))\|^2 \Big\}. \tag{10}$$

In the above, the latent value is $\mathbf{z} = \mathcal{E}(\mathbf{y})$. The perturbation $\boldsymbol{\delta}$ is a random variable, drawn from a uniform distribution over the sphere of radius δ in $\mathbb{R}^L$. Similar loss functions had been employed in the works (Geng et al.

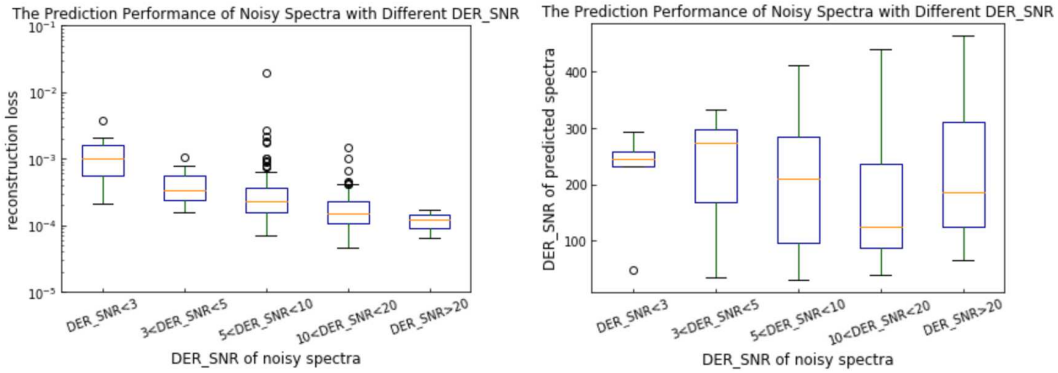

Fig. 10 Model performance under varying levels of signal-to-noise ratio (DER_SNR). The left panel shows how the reconstruction loss (vertical axis) depends on the DER_SNR of the input synthetic spectrum (horizontal axis). The right panel shows how the DER_SNR of the denoised spectrum (vertical axis) varies with the DER_SNR of the input synthetic spectrum (horizontal axis). The reconstruction loss does not increase very quickly as DER_SNR decreases in the left panel, while the DER_SNR of the denoised spectra is stable in the right panel.

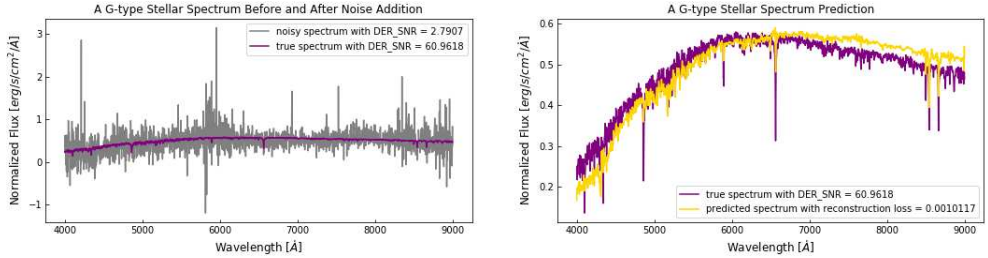

Fig. 11 An illustrative example. In the left panel, the DER_SNR of the synthetic noisy spectrum (*grey curve*) is 2.79 and the DER_SNR of the true spectrum (*purple curve*) is 60.96. In the right panel, the reconstruction loss between the true spectrum (*purple curve*) and the predicted spectrum from our model (*yellow curve*) is 1.0×10^{-3}.

2020; Atzmon et al. 2020) to train the auto-encoder to learn the manifold structure.

After training $\mathcal{E}, \mathcal{G}$ based on Equation (10), we can compute the latent variables $\mathbf{z}$ for all training samples. Then, based on this collection of latent variables, a kernel density estimator (KDE) is deployed over $\mathbf{z}$ for the training samples of each stellar subclass C. This helps us to obtain $p(\mathbf{z}|C)$ for each subclass C.

3.4 The NoiseFlow Observation Model

This subsection constructs the observation model $p(\mathbf{y}|\mathbf{s})$ for $\mathbf{y}_n$ given the true signal $\mathbf{s} = \mathcal{G}(\mathbf{z})$. Recall that we have used the additive noise model $\boldsymbol{\epsilon}_n = \mathbf{y}_n - \mathbf{s}$. Our goal is equivalent to construct a noise density model $p(\boldsymbol{\epsilon}_n)$ and set $p(\mathbf{y}_n|\mathbf{s}) = p(\boldsymbol{\epsilon}_n)$. The density model is built upon the idea of Papamakarios et al. (2018) such that the observation model has an auto-regressive structure. Compared with their model, our model only depends on a local group of pixels within an observation, is more parsimonious in parameterization, and focuses on extracting the local correlation structure of the noise.

Suppose the noise vector is written as $\boldsymbol{\epsilon}_n = (\epsilon_{n1}, \epsilon_{n2}, \cdots, \epsilon_{nK})$. For $d = K+1, \cdots, D$, the noise value ϵ_{nd} for the d-th pixel depends on its immediate K preceding pixels via

$$p(\epsilon_{nd}|\boldsymbol{\epsilon}_{n,(d-K):(d-1)}) = \mathcal{N}(\epsilon_{nd}|\mu_{nd}, (\exp \alpha_{nd})^2)\,, \quad (11)$$

where $\mu_{nd} = f_\mu(\boldsymbol{\epsilon}_{n,(d-K):(d-1)})$, $\alpha_{nd} = f_\alpha(\boldsymbol{\epsilon}_{n,(d-K):(d-1)})$, and f_μ and f_α are two functions expressed by neural networks. The architecture of f_μ and f_α will be specified later. The two functions f_μ and f_α determine the mean and the standard deviation for the noise ϵ_{nd} at the d-th pixel. In particular, we have

$$\epsilon_{nd} = \xi_{nd} \exp(\alpha_{nd}) + \mu_{nd}\,, \quad (12)$$

where $\xi_{nd} \sim \mathcal{N}(0,1)$ follows the standard Gaussian distribution. Equivalently, the random variable can be expressed by the following inverse expression

$$\xi_{nd} = (\epsilon_{nd} - \mu_{nd}) \exp(-\alpha_{nd})\,. \quad (13)$$

The above specifies the local dependence structure for $d = K+1, \cdots, D$. For the first K pixels, there are not enough preceding pixels for us to determine the conditional likelihood (11). Instead, for the first K pixels, the noise ϵ_{nd} is imposed to follow a univariate Gaussian distribution with a fixed mean μ_d and a fixed standard

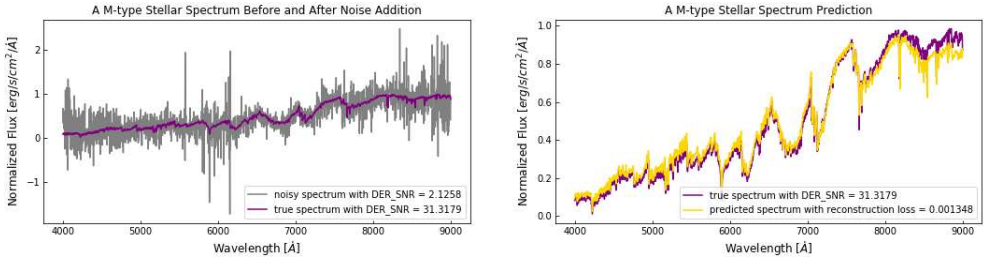

Fig. 12 An illustrative example. In the left panel, the DER_SNR of the synthetic noisy spectrum (*grey curve*) is 2.13 and the DER_SNR of the true spectrum (*purple curve*) is 31.32. In the right panel, the reconstruction loss between the true spectrum (*purple curve*) and the predicted spectrum from our model (*yellow curve*) is 1.3×10^{-3}.

deviation $\sigma_d = \exp(-\alpha_d)$. In summary, our model parameters to be trained include: the scalar values μ_d, α_d for $d = 1, \cdots, K$; and two neural network functions f_μ and f_α.

As μ_{nd} and α_{nd} depend on $\boldsymbol{\epsilon}_{n,(d-K):(d-1)}$ by design, the Jacobian in density transformation Equation (8) is an upper triangular matrix. As a result, the absolute value of the determinant can be easily computed as

$$\log \left| \det \frac{\partial f^{-1}}{\partial \boldsymbol{\epsilon}} \right| = - \sum_{d=K+1}^{D} \alpha_{nd} \,. \tag{14}$$

It follows that the log density of the observation model becomes

$$\begin{aligned} &\log p(\mathbf{y}_n|\mathbf{s}) = p(\boldsymbol{\epsilon}_n) \\ &= \sum_{d=1}^{K} \log p(\epsilon_{nd}) + \sum_{d=K+1}^{D} \log p(\epsilon_{nd}|\boldsymbol{\epsilon}_{n,(d-K):(d-1)}) \\ &= \sum_{d=1}^{K} \Big[-(1/2)\exp(-2\alpha_d)(\epsilon_{nd} - \mu_d)^2 - \alpha_d \Big] \\ &\quad + \sum_{d=K+1}^{D} \Big[-(1/2)\exp(-2\alpha_d)(\epsilon_{nd} - \mu_{nd})^2 \\ &\quad - \alpha_{nd} \Big] + \text{const} \,. \end{aligned} \tag{15}$$

The last equality holds up to some irrelevant constant const.

For our observation model, we develop a neural network architecture for local noise feature extraction, as shown in Figure 2. In the first layer, local features are extracted by a one-dimensional convolution layer. This convolution layer has one input channel and C output channels, kernel size K, one stride and zero padding. Its output $\mathbf{h}$ is of dimension $N \times C \times (D - K + 1)$, where N is the mini-batch sample size. In accordance of the aggressive structure, we drop one redundant column (the first column) in the third dimension of the hidden state $\mathbf{h}^1$, such that its dimension becomes $N \times C \times (D - K)$. We then take a transpose to exchange the second dimension with the third dimension. The resulting $\mathbf{h}$ is of dimension $N \times (D - K) \times C$. In this way, for the d-th pixel ($d = K + 1, \cdots, D$), we get a C-dimensional feature vector for it.

After that, $\mathbf{h}$ is used as the input of the following $L-1$ linear hidden layers with RELU, to sequentially reduce the dimension of hidden variables from $N \times (D - K) \times C$ to $N \times (D - K) \times C'$ for some $C' < C$. In the sequel, there are two separate linear hidden layers mapping from $N \times (D - K) \times C'$ to $N \times (D - K) \times 1$, one is for $\boldsymbol{\mu}$ and the other is for $\boldsymbol{\alpha}$, and we transpose $\boldsymbol{\mu}$ and $\boldsymbol{\alpha}$ back to $N \times 1 \times (D - K)$. At this moment, vector $\boldsymbol{\mu}$ and $\boldsymbol{\alpha}$ contain the mean and standard deviation information for ϵ_{nd} with $d = K + 1, \cdots, D$. As for the first K pixels, their scalar mean and standard deviation values (μ_d, α_d for $d = 1, \cdots, K$) get included via the final masked linear layers.

Moreover, the observation model developed hereby can naturally handle a spectrum with missing flux values due to bad pixels. We can create a mask vector $\mathbf{m}_n$ such that $m_{nd} = 1$ if all of $y_{n,d-K}, \cdots, y_{n,d-1}, y_{n,d}$ are observed, and $m_{nd} = 0$ otherwise. The observation log-likelihood for $\mathbf{y}_n$ becomes

$$\begin{aligned} \log p(\mathbf{y}_n|\mathbf{s}) = &\sum_{d=1}^{K} m_{nd} \log p(\epsilon_{nd}) \\ &+ \sum_{d=K+1}^{D} m_{nd} \log p(\epsilon_{nd}|\boldsymbol{\epsilon}_{n,(d-K):(d-1)}) \,. \end{aligned} \tag{16}$$

In other words, the likelihood of the d-th pixel is taken into account if and only if the d-th pixel and its immediate K preceding pixels are observed, which is shown in Figure 3. The masked likelihood allows us to deal with partially observed spectrum without resorting to ad-hoc missing value imputation.

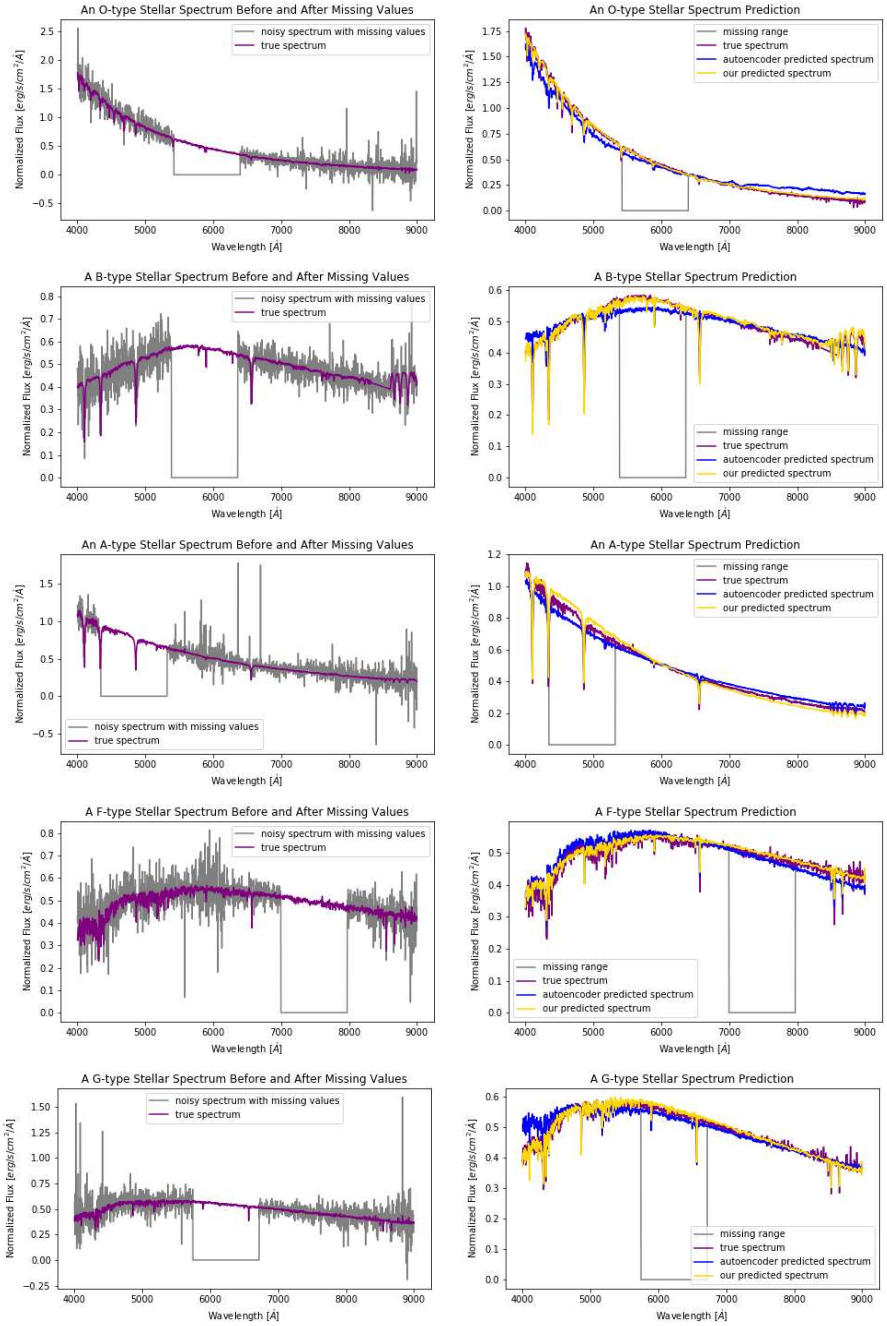

Fig. 13 The five rows show examples of synthetic stellar spectra with missing values and the denoising results for the stellar subclasses O, B, A, F, G, respectively. The left column shows the true spectrum and its synthetic counterpart with noise and missing values. The *purple curve* is the clean spectrum that both denoising algorithms try to recover, and the *grey curve* is the spectrum added with noise and missing values. The spectra denoised by both algorithms are compared in the right column. The denoised spectrum from our model (*yellow curve*) is much closer to the purple clean spectrum than the standard auto-encoder result (*blue curve*).

3.5 Modeling Workflow

The main workflow of our model is summarized in Figure 4. In order to train and apply our proposed model, basically four main steps are involved:

1. Train an encoder $\mathcal{E}$ and a locally isometric generator $\mathcal{G}$ based on a collection of high-SNR optical stellar spectra.
2. Use Gaussian kernel density estimation (Gaussian KDE) to estimate the prior distribution of the latent

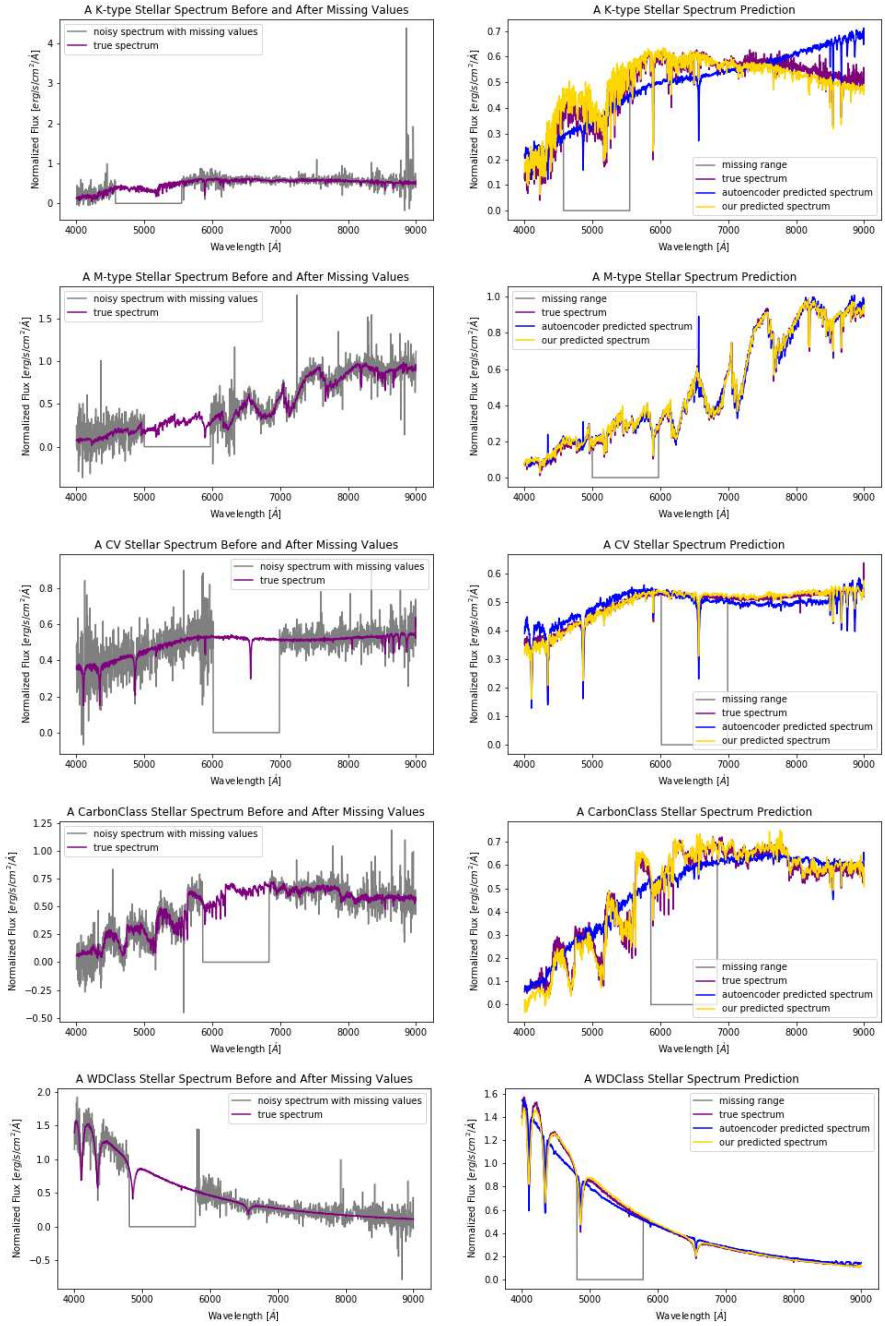

Fig. 14 The five rows show examples of synthetic stellar spectra with missing values and the denoising results for the stellar subclasses K, M, CV, Carbon, WD, respectively. The left column shows the true spectrum and its synthetic counterpart with noise and missing values. The *purple curve* is the clean spectrum that both denoising algorithms try to recover, and the *grey curve* is the spectrum added with noise and missing values. The spectra denoised by both algorithms are compared in the right column. The denoised spectrum from our model (*yellow curve*) is much closer to the purple clean spectrum than the standard auto-encoder result (*blue curve*).

variable $\mathbf{z}$ for each stellar subclass obtained from the encoder trained in the first step.

3. Train a NoiseFlow observation model with the observational noise extracted from low-SNR optical stellar spectra.
4. For each test spectrum, use the Gaussian KDE, the NoiseFlow model and the generator $\mathcal{G}$ trained in the first three steps to obtain its corresponding latent variable and its clean and complete spectrum with iterative optimization of Equation (7).

4 RESULTS

This section compares our method with the convolutional denoising auto-encoder based on two test datasets. In particular, two tasks are created for the purpose of spectrum denoising and missing flux imputation. The experiment is conducted based on the stellar spectral observations introduced in Section 2.

The proposed model is trained over the training dataset of Section 2. Recall that each training spectrum has been interpolated over a grid with a size of $D = 2048$, and we will set the latent space dimension as $L = 3$. This training dataset is supplied to Equation (10) to train the generator $\mathcal{G}$ and the encoder $\mathcal{E}$ with a stochastic gradient algorithm. The encoder learns a latent representation $\mathbf{z} \in \mathbb{R}^3$ for each training spectrum. Figure 5 shows the scatterplot of the latent variables for the training dataset. The points are colored according to the stellar subclasses. It is evident that observations from the same stellar subclass form a cluster. The latent space can also help us to detect outlier spectra, such as those inside the red circles indicated in Figure 5. This learned latent space informs us of the latent variable $\mathbf{z}$ structure for most stellar observations. Therefore, we can use a Gaussian kernel density estimation for each of the ten stellar subclasses to get $p(\mathbf{z}|C)$.

For comparison, the convolutional auto-encoder gets trained based on a larger dataset. The dataset contains spectra both with high SNR and low SNR. Besides, for a fair comparison, the convolution neural network shares the same architecture (e.g., the same hidden layers and the same transposed convolution layers) as our generator $\mathcal{G}$.

4.1 Spectrum Denoising

To test the model performance, we randomly select an independent group of spectra with high SNR for each stellar subclass, as described in Section 2. These selected spectra are regarded as the ground truth that the model endeavors to predict. Then we extract another noisy dataset from an independent group of stellar spectra with SNR ranging from 10 to 20. The noise is randomly sampled and added to the above clean test spectra. These constitute our benchmark test dataset for method comparison. Each stellar subclass is created with 300 test spectra.

Figure 7 and Figure 8 exhibit some examples of denoised spectra for ten stellar subclasses. We choose one representative result from each of the ten stellar subclasses to demonstrate the power of our method. The left column shows the spectrum before and after noise contamination, where the purple curve is the high-SNR spectrum and the grey curve is the clean spectrum with added noise. The noisy spectrum gets cleaned by the standard auto-encoder and our proposed method. The denoised spectrum is shown in the right column of Figure 7 and Figure 8. In each subfigure of the right column, the purple spectrum is the true spectrum that both denoising algorithms try to recover. Though the clean purple spectrum is unknown to the denoising algorithm, our proposed method shows promising ability to recover it from the noisy observation. The predicted spectrum from our model (yellow curve) is much closer to the purple clean spectrum than the standard auto-encoder result (blue curve). Our proposed method also has the capacity to remove strong noisy emission lines (see the second row of Fig. 8) and keeps the signal emission lines (see the third row of Fig. 8).

Figure 9 shows the overall comparison between our method and the convolutional denoising auto-encoder in spectrum denoising for various stellar subclasses. Each yellow point in a subfigure represents one synthetic noisy stellar spectrum. The noisy spectrum gets cleaned and compared with the true high-SNR spectrum, and the reconstruction loss is computed. The reconstruction loss is computed as follows. Suppose that $\mathbf{s}$ is the true signal spectrum and $\hat{\mathbf{s}}$ is the denoised spectrum by a model. Then, the reconstruction loss is $\sum_{d=1}^{D}(s_d - \hat{s}_d)^2/D$. In each subfigure, the horizontal axis is the reconstruction loss for the convolutional denoising auto-encoder, and the vertical axis is the reconstruction loss of our method. The blue line indicates where the two methods have equal performance. We can see that most points in each subfigure fall below the blue line, indicating that our method has smaller reconstruction loss. The title of each subfigure reports the proportion of spectra for which our method has smaller reconstruction loss. Within each stellar subclass, our method produces higher-quality denoised spectra for more than 80% of the testing samples. For subclasses such as carbon class and WD class, our method demonstrates improved performance for almost 100% of the test samples.

We further consider how the signal-to-noise ratio affects our model performance. For the synthetic spectra, we measure their signal-to-noise ratio by the formula given by Stoehr et al. (2008). The computed signal-to-noise ratio is denoted as DER_SNR. To decrease DER_SNR of the synthetic data to a specific level, we add additional gaussian noise with various noise levels to each spectrum. This procedure results in a new group of test dataset. The left panel of Figure 10 shows how the boxplot of the reconstruction loss varies across distinct DER_SNR levels. The reconstruction loss does not increase very quickly as DER_SNR decreases. When the DER_SNR is below 3, the median reconstruction loss is still about 10^{-3}. To illustrate how the spectrum looks like at this level of reconstruction loss, we plot a few examples in Figures 11–12. The grey curve in the left panel of Figure 11

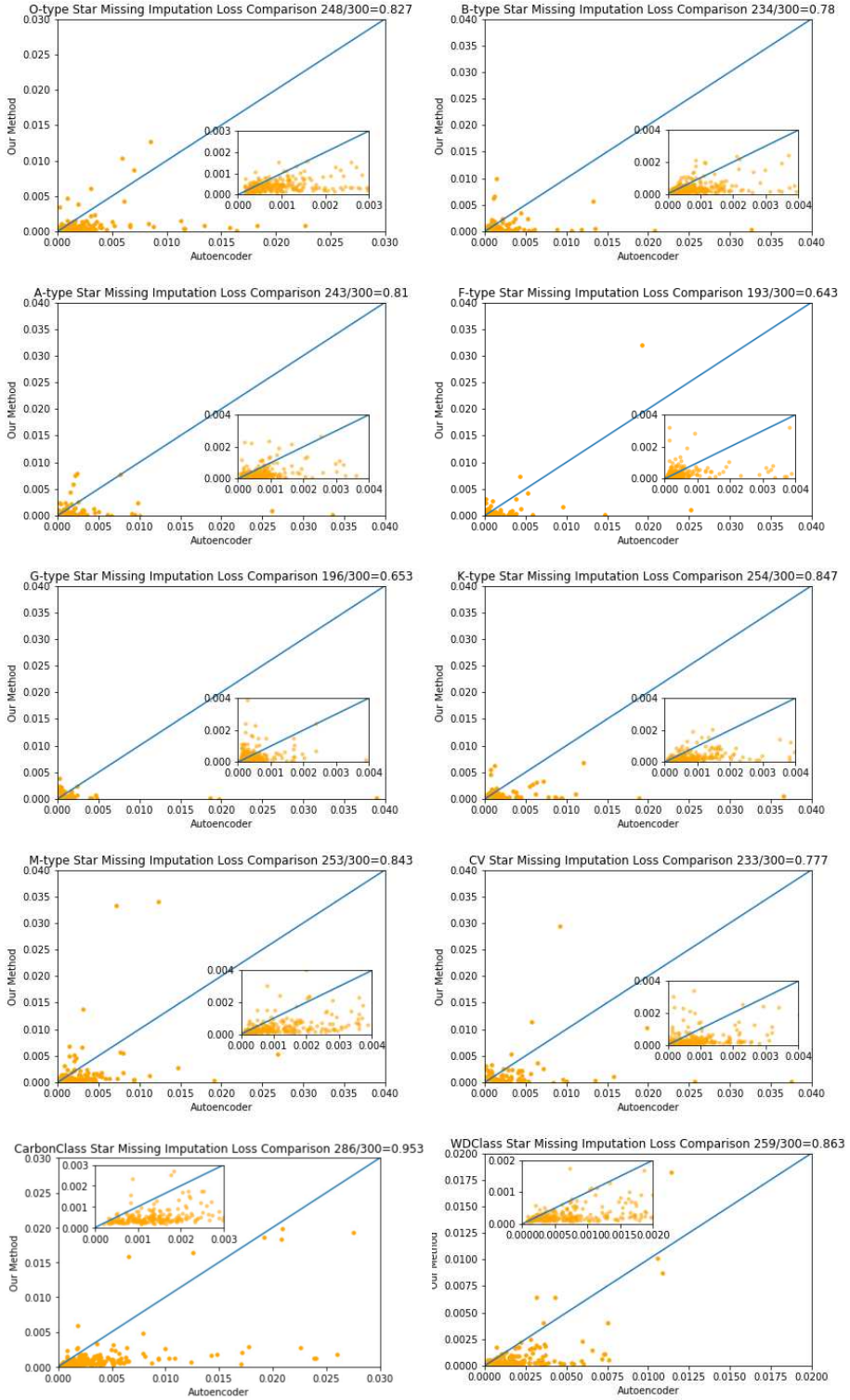

Fig. 15 The loss comparison for the two methods applied to the noisy spectra with missing values. Each panel corresponds to one stellar subclasse. The horizontal axis is the reconstruction loss for the convolutional denoising auto-encoder, and the vertical axis is the reconstruction loss for our method. Each *yellow point* in a subfigure corresponds to one synthetic noisy spectrum with missing values. The *blue line* indicates where the two methods have equal performance. Most points in each subfigure fall below the *blue line*, indicating that our method has smaller reconstruction loss. The title of each subfigure also reports the proportion of spectra for which our method has smaller reconstruction loss. Our method demonstrates improved performance.

is one synthetic spectrum with DER_SNR equaling to 2.79. The reconstruction loss between the true spectrum and the denoised spectrum in the right panel is about 1.0×10^{-3}. Figure 12 shows one more example where the DER_SNR of the synthetic spectrum is 2.12 and the reconstruction loss is about 1.3×10^{-3}. The right panel of Figure 10 compares the DER_SNR before and after denoising for the above dataset. For most spectra, their DER_SNR after denoising is above 100. From the right panel of Figure 10, we can also find that the DER_SNR of our model output is stable regardless of the DER_SNR of the input spectrum. This is because our model prior component and the generator in Equations (4)–(6) are fixed after model training. The denoised spectra, which are generated by the prior and the generator, will always have the same level of SNR of the training dataset.

4.2 Spectrum Denoising with Missing Flux

Besides the above test dataset, another test dataset is created to evaluate model performance for spectrum denoising with missing flux values. To construct this benchmark data, almost the same procedure of Section 4.1 is taken. Realistic noise is added to the clean spectrum with high SNR. In addition, we randomly remove the flux values over an interval range of wavelength. The interval of missing pixels is also randomly selected for each test spectrum.

Based on the idea of masked likelihood Equation (16), our trained model can be directly employed to denoise spectrum with partially missing flux values. In other words, our model does not require re-training to deal with this kind of data. However, the standard denoising convolutional auto-encoder requires re-training to adapt to this dataset. Its original training data also get randomly removed flux values over a random wavelength range. The missing values are imputed by zeros, and the denoising convolutional auto-encoder aims to reconstruct the original full spectrum from the zero-imputed spectrum. The full spectrum is employed for its loss computation and parameter update.

Figure 6 shows the performance of our model depending on different positions and widths of the missing range. The three panels correspond to the cases where the missing flux occurs at the left end (blue end), the middle and the right end (red end) of the spectrum, respectively. In each panel, the horizontal axis is the number of missing pixels out of the total $D = 2048$ pixels. The vertical axis is the reconstruction loss. As expected, the reconstruction loss increases with the growing number of missing pixels. When the length of missing pixels is 800 (i.e., 40% of the whole spectrum), the median reconstruction loss is still below 10^{-3}. Generally, the model performance is not sensitive to the location of the missing pixels, but missing at the left end (blue end) incurs slightly higher loss. This is due to the fact that the blue end is feature-rich and contains more information for spectrum reconstruction for most spectra.

Figure 13 and Figure 14 plot a few examples for the ten stellar subclasses. The left column shows the spectrum before and after noise addition and flux value removal. The purple spectrum is the original spectrum with high SNR. The grey spectrum has noise added, but at the same time, a random interval of flux values is removed. The missing flux is plotted as an interval of zeros. The purple spectrum is the ground truth spectrum that both algorithms try to recover. The denoised and imputed spectrum is shown in the right column. Our predicted spectrum is shown in yellow, and the result of convolutional auto-encoder is shown in blue. Our resulting spectra are much closer to the true spectra in these cases. The overall result for this test data is summarized in Figure 15. The interpretation of the figure is similar to that of Figure 9. Although our model has a moderate lead over the convolutional auto-encoder in F-type and G-type classes, the performance difference margins are wider in the other stellar subclasses.

5 SUMMARY AND CONCLUSIONS

In this paper, we propose a new efficient deep Bayesian model for stellar spectral denoising, defective spectral recovery and sky emission lines or cosmic rays removal. Compared with the existing methods, our model makes a greater usage of available data, exhibits a high robustness and a superior performance in spectral denoising. In summary, our approach has the following advantages:

1. The observation model $p(\mathbf{y}|\mathbf{s})$ takes into account the noise correlation structure. It is able to properly handle the strong sky emissions, cosmic rays, and the background noise of the observational instruments.
2. When some part of the observation is missing due to unpredictable errors (e.g. pipeline handling error, defective spectra), our model only computes the likelihood of the observed pixels, without resorting to ad-hoc missing value imputation.
3. Our prior model $p(\mathbf{s})$ encodes how a true signal should look, making our model less susceptible to defective or distorted observations (due to combining the blue and red channels).
4. Our proposed model can also directly exploit multiple-exposure data, making the posterior inference more reliable than using only one single average data.

The proposed method can be considered as a novel model for large-scale astronomical spectral surveys and

will benefit subsequent astronomical research. In future work, we will continue refining the proposed model and investigating its proper applications in other astronomical spectral analysis tasks. For example, our model will be applied during stellar spectral data preprocessing when performing stellar classification or estimating stellar physical parameters ($T_{\rm eff}$, log$g$, [Fe/H]).

Acknowledgements The authors would like to thank an anonymous reviewer for the helpful comments to improve the work. This paper is funded by the National Natural Science Foundation of China (Grant Nos. 11873066 and U1731109). We acknowledge SDSS databases. Funding for the Sloan Digital Sky Survey IV has been provided by the Alfred P. Sloan Foundation, the U.S. Department of Energy Office of Science, and the Participating Institutions. SDSS-IV acknowledges support and resources from the Center for High-Performance Computing at the University of Utah. The SDSS web site is www.sdss.org. SDSS-IV is managed by the Astrophysical Research Consortium for the Participating Institutions of the SDSS Collaboration including the Brazilian Participation Group, the Carnegie Institution for Science, Carnegie Mellon University, the Chilean Participation Group, the French Participation Group, Harvard-Smithsonian Center for Astrophysics, Instituto de Astrofísica de Canarias, the Johns Hopkins University, Kavli Institute for the Physics and Mathematics of the Universe (IPMU) /University of Tokyo, Lawrence Berkeley National Laboratory, Leibniz Institut für Astrophysik Potsdam (AIP), Max-Planck-Institut für Astronomie (MPIA Heidelberg), Max-Planck-Institut für Astrophysik (MPA Garching), Max-Planck-Institut für Extraterrestrische Physik (MPE), National Astronomical Observatories of China, New Mexico State University, New York University, University of Notre Dame, Observatário Nacional / MCTI, The Ohio State University, Pennsylvania State University, Shanghai Astronomical Observatory, United Kingdom Participation Group, Universidad Nacional Autónoma de México, University of Arizona, University of Colorado Boulder, University of Oxford, University of Portsmouth, University of Utah, University of Virginia, University of Washington, University of Wisconsin, Vanderbilt University, and Yale University.

References

Ahumada, R., Prieto, C. A., Almeida, A., et al. 2020, ApJS, 249, 3

Atzmon, M., Gropp, A., & Lipman, Y. 2020, arXiv preprint arXiv:2006.09289

Cui, X.-Q., Zhao, Y.-H., Chu, Y.-Q., et al. 2012, RAA (Research in Astronomy and Astrophysics), 12, 1197

Dinh, L., Krueger, D., & Bengio, Y. 2014, arXiv preprint arXiv:1410.8516

Dinh, L., Sohl-Dickstein, J., & Bengio, S. 2017, Density Estimation Using Real NVP, arXiv:1605.08803

Donoho, D. L. 1993, in In Proceedings of Symposia in Applied Mathematics (American Mathematical Society), 173

Donoho, D. L., & Johnstone, I. M. 1994, Biometrika, 81, 425

Donoho, D. L., & Johnstone, I. M. 1995, JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION, 1200

Geng, C., Wang, J., Chen, L., et al. 2020, arXiv preprint arXiv:2002.04829

Germain, M., Gregor, K., Murray, I., & Larochelle, H. 2015, MADE: Masked Autoencoder for Distribution Estimation, arXiv:1502.03509

Hinton, G. E., Osindero, S., & Teh, Y. 2006, Neural Computation, 18, 1527

Kingma, D. P., Salimans, T., Jozefowicz, R., et al. 2017, Improving Variational Inference with Inverse Autoregressive Flow, arXiv:1606.04934

Lawlor, D., Budavári, T., & Mahoney, M. W. 2016, ApJ, 833, 26

Machado, D., Leonard, A., Starck, J.-L., Abdalla, F., & Jouvel, S. 2013, A&A, 560, A83

Papamakarios, G., Pavlakou, T., & Murray, I. 2018, Masked Autoregressive Flow for Density Estimation, arXiv:1705.07057

Rezende, D. J., & Mohamed, S. 2016, Variational Inference with Normalizing Flows, arXiv:1505.05770

Stoehr, F., White, R., Smith, M., et al. 2008, in Astronomical Data Analysis Software and Systems XVII, Vol. 394, 505

Uria, B., Côté, M.-A., Gregor, K., Murray, I., & Larochelle, H. 2016, Neural Autoregressive Distribution Estimation, arXiv:1605.02226

Vincent, P., Larochelle, H., Lajoie, I., et al. 2010, Journal of Machine Learning Research, 11