Monthly Notices
of the
ROYAL ASTRONOMICAL SOCIETY

MNRAS **483,** 3255–3277 (2019)
Advance Access publication 2018 November 28

doi:10.1093/mnras/sty3217

Deep learning of multi-element abundances from high-resolution spectroscopic data

Henry W. Leung[1]★ and Jo Bovy[1,2]†

[1]*Department of Astronomy and Astrophysics, University of Toronto, 50 St. George Street, Toronto, Ontario M5S 3H4, Canada*
[2]*Dunlap Institute for Astronomy and Astrophysics, University of Toronto, 50 St. George Street, Toronto, Ontario M5S 3H4, Canada*

Accepted 2018 November 8. Received 2018 November 5; in original form 2018 August 13

ABSTRACT

Deep learning with artificial neural networks is increasingly gaining attention because of its potential for data-driven astronomy. However, this methodology usually does not provide uncertainties and does not deal with incompleteness and noise in the training data. In this work, we design a neural network for high-resolution spectroscopic analysis using APO Galactic Evolution Experiment (APOGEE) data that mimics the methodology of standard spectroscopic analyses: stellar parameters are determined using the full wavelength range, but individual element abundances use censored portions of the spectrum. We train this network with a customized objective function that deals with incomplete and noisy training data and apply dropout variational inference to derive uncertainties on our predictions. We determine parameters and abundances for 18 individual elements at the ≈0.03 dex level, even at low signal-to-noise ratio. We demonstrate that the uncertainties returned by our method are a realistic estimate of the precision and they automatically blow up when inputs or outputs outside of the training set are encountered, thus shielding users from unwanted extrapolation. By using standard deep-learning tools for GPU acceleration, our method is extremely fast, allowing analysis of the entire APOGEE data set of ≈250 000 spectra in 10 min on a single, low-cost GPU. We release the stellar parameters and 18 individual-element abundances with associated uncertainty for the entire APOGEE DR14 data set. Simultaneously, we release `astroNN`, a well-tested, open-source `PYTHON` package developed for this work, but that is also designed to be a general package for deep learning in astronomy. `astroNN` is available at https://github.com/henrysky/astroNN with extensive documentation at http://astroNN.readthedocs.io.

Key words: methods: data analysis – techniques: spectroscopic – stars: abundances – stars: fundamental parameters.

1 INTRODUCTION

With astronomy becoming increasingly characterized by large surveys and big data sets, machine-learning techniques have become staples of astronomical data analysis that are used for low-level data processing, classification, interpolation, pattern recognition, and parameter inference. Among modern machine-learning techniques, deep learning using artificial neural networks (ANNs) is getting increasing attention from astronomers, because of its great potential for data-driven astronomy and its recent successes in many fields such as computer vision, voice recognition, machine translation, etc. While ANNs have been around for decades, they have only become one of the dominant machine-learning techniques in the last few years. The reasons behind this recent progress are the combination of big data, cheap availability of fast computational hardware, advances in the methodology of ANNs, and the availability and accessibility of software platforms implementing this technology (Stoica et al. 2017). In astronomy, the big data era is now fully upon us: large photometric (e.g. SDSS; Aihara et al. 2011), spectroscopic (e.g. APOGEE; Majewski et al. 2017), astrometric (Gaia; Gaia Collaboration 2016), and time-domain (e.g. PTF; Law et al. 2009) data sets already exist and will grow exponentially in terms of quality and quantity with upcoming projects like the Large Synoptic Survey Telescope (LSST; LSST Science Collaboration 2009), Euclid (Laureijs et al. 2011), and projects like the Maunakea Spectroscopic Explorer (MSE; McConnachie et al. 2016). ANNs are poised to play an outsized role in this data-rich future.

★ E-mail: henrysky.leung@mail.utoronto.ca
† Alfred P. Sloan Fellow

The hardware and software landscape for machine learning has vastly changed in the last decade. In particular, the availability of cheap graphics processing units (GPUs) is driven by the development and demand of high-performance low-cost personal gaming, but modern machine learning methods are ideally suited to be run on GPUs. For example, the code and analysis that we describe in this paper is entirely run on a personal desktop computer with an ≈$500 consumer GPU accelerated by the NVIDIA CUDA Deep Neural Network library (Chetlur et al. 2014). The accessibility of software technology for machine learning is supported by open source communities. For example, the open source PYTHON deep learning libraries used in this work – `Tensorflow` (Abadi et al. 2016), `keras` (Chollett et al 2015), and the package developed by us and described in this paper `astroNN` [1] (see the Appendix). The combination of these factors allows astronomers to easily exploit deep learning in astronomical big data analysis without the high cost of development in human resources, time, hardware, and software.

In this work we investigate the application of deep learning to the analysis of high-resolution spectroscopic data. Such data contain a wealth of information about the overall physical state of stars and about the abundances of different elements in their photospheres (Gray 2005). This information is traditionally extracted using tools such as the curve of growth, equivalent widths, or forward modelling with synthetic spectra in what is often a laborious and tedious effort. Here we demonstrate that, as long as a small – thousands of stars – training set of data analysed with more traditional means is available, ANNs can process high-resolution spectra faster and more reliably than other methods.

ANNs have been used for spectroscopic analysis before. Because of the paucity of high-resolution spectra until recently, early use of ANNs was mostly limited to training the network on libraries of synthetic spectra (e.g. Bailer-Jones et al. 1997; Bailer-Jones 2000; Yang & Li 2015). The large spectroscopic data bases provided by the SEGUE (Yanny et al. 2009) and APO Galactic Evolution Experiment (APOGEE) surveys allowed ANNs to be trained directly on observed stellar spectra and map them onto stellar parameters (SEGUE: Re Fiorentin et al. 2007, Lee et al. 2008; APOGEE: Fabbro et al. 2018). Uncertainty estimation has been explored using generative ANNs for the *Gaia* RVS data by Dafonte et al. (2016).

The availability of the large and rich APOGEE spectroscopic data set has spurred many applications of machine-learning methods to these data. Examples of these are the `Cannon 2` (Casey et al. 2016) method for data-driven abundance analysis, dimensionality reduction of the spectral space to determine the dimensionality of abundance space in the Milky Way (Price-Jones & Bovy 2018), and machine-learned outlier detection and similarity directly using the spectra (Reis et al. 2018).

The work most directly related to that described in this paper is the recent `StarNet` ANN, which is trained on spectroscopic data from APOGEE (Fabbro et al. 2018). `StarNet` uses a convolutional NN to infer three labels [$T_{\rm eff}$, $\log g$, and [Fe/H]] from high-resolution spectra and demonstrated that deep learning is an effective way both in terms of performance and of accuracy to do spectroscopic analysis when the number of training data is large.

In this work, we go beyond the `StarNet` method in various way: (i) we present a robust objective function for the NN to learn from incomplete data while taking uncertainty in the training labels into account, (ii) we use a Bayesian NN with dropout variational inference with this objective function to estimate the uncertainties of labels determined by the NN (Gal & Ghahramani 2015), (iii) we simultaneously infer 22 stellar and elemental abundance labels accurately and precisely for both high and low signal-to-noise ratio (SNR) spectra while constraining the model to reflect our physical understanding of stellar spectra, (iv) we implement the method on a GPU using standard tools allowing for more than an order of magnitude speed-up and make these easily accessible (see Appendix A3), and (v) we demonstrate that a large NN can work well with a limited amount of training data (thousands of high SNR stellar spectra). We also present these 22 stellar parameters and elemental abundance predictions with uncertainties for the entire APOGEE DR14 data set.

The outline of this paper is as follows. Section 2 describes the basics of ANNs, of dropout variational inference, and of our robust objective function. Section 3 discusses the data selection and processing from APOGEE DR14 to construct training and test sets. Section 4 describes the performance of our trained NN on unseen individual and combined spectra in the test sets, on stars in open and globular clusters, and we present the results from a sensitivity analysis to understand the NN . Section 5 describes variations in NN training such as: training on the full, uncensored spectrum, training with small data sets with only thousands of spectra, and training with a different continuum normalization process. Section 6 discusses the fast performance of the NN and what types of future work this allows, and comparisons to other, similar spectroscopic analysis approaches. Section 7 gives our conclusions. The Appendix describes the `astroNN` python package developed for this work and gives instructions on how to perform variational inference on arbitrary APOGEE spectra.

Code to reproduce all of the plots in this paper as well as the `FITS`[2] data file containing our NN predictions for 22 stellar parameters and abundances for the whole APOGEE DR14 is available at https://github.com/henrysky/astroNN_spectra_paper_figures.

2 BAYESIAN NEURAL NETWORKS WITH DROP-OUT

Deep learning refers to the usage of multilayer ('deep') ANNs to achieve both supervised and unsupervised machine learning. As opposed to task-specific algorithms, ANNs provide a general learning method that can be used on a variety of learning tasks. Bayesian NN refers to the application of Bayesian inference to NNs to obtain a posterior distribution function (PDF) on the weights that characterize the ANN given some input data. This PDF can then be used to propagate training uncertainty into predictions made with the ANN using new input data. Here, we use dropout variational inference as an approximation to Bayesian NNs. Dropout variational inference is a new method proposed by Gal & Ghahramani (2015) that can be applied to a wide variety of NN architectures, is easy to implement, and is computationally cheap. This technique is previously used in Perreault Levasseur, Hezaveh & Wechsler (2017) for strong gravitational lensing parameters estimation with uncertainty

In this section, we give a brief introduction to ANNs, describe the idea of dropout variational inference, and then discuss the loss function that we use to train our ANN using incomplete and noisy training data.

[1] https://github.com/henrysky/astroNN

[2] https://github.com/henrysky/astroNN_spectra_paper_figures/raw/master/astroNN_apogee_dr14_catalog.fits

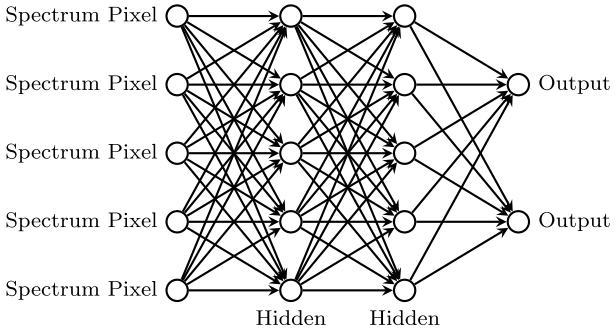

Figure 1. A simple multilayer neural network.

2.1 Artificial neural networks

ANNs were originally inspired by biological systems such as human brains, which consist of numerous neurons interconnected by synapses. The information or stimuli from the external world travel in the form of electrical impulses called action potentials. An ANN mimics this configuration and dynamics by representing a general learning task as a set of layers consisting of neurons that communicate through connections (the 'synapses'). The 'strength' of each connection is given by a simple linear functional form $y = w\,x + b$, which is transformed by each neuron using a non-linear function. The final decision or output of the ANN depends on the input and on the strength of the connections (the weights $\mathbf{W}$). There is no need to pre-program any knowledge in an ANN, i.e. NNs consist of random weights at the initial training stage. In order to achieve learning, error signals representing the agreement between the true output and the ANN output for (a subset of) the training set is back-propagated through the NN and the connection strength of the synapses is adjusted to obtain better agreement between truth and prediction. Fig. 1 shows an example of a typical ANN.

Mathematically, an ANN is a real-valued, smooth function approximation to a general function. Consider data $\{\mathbf{x}, \mathbf{y}\}$ and a NN f parametrized by a set of parameters $\mathbf{W}$ that takes input $\mathbf{x}$ and maps it to $\hat{\mathbf{y}} = f^{\mathbf{W}}(\mathbf{x})$. Each neuron i in an ANN takes an input vector $\mathbf{v}$ (either the actual input $\mathbf{x}$ or the output of a previous layer) and maps it to an output number o as $o = \mathbf{w}_i\,\mathbf{v} + b_i$; this output number is then optionally transformed using a non-linear function before going to the next layer or the output. The parameters $\mathbf{W}$ in our notation represent the total set of $\{\mathbf{w}_i, b_i\}$ of all the neurons.

The quality of the ANN is represented by an objective function $J(\mathbf{y}, \hat{\mathbf{y}})$ that we want to minimize. At each step in the training process, a new set of parameters $\mathbf{W}_{\rm new}$ in ANN optimization can be obtained by descending along the gradient computed using back-propagation (Rumelhart, Hinton & Williams 1986)

$$\mathbf{W}_{\rm new} = \mathbf{W} - \eta \frac{\partial J}{\partial \mathbf{W}}\,, \tag{1}$$

where η is the learning rate. For small η, this update step should lead to $J(\mathbf{y},\, f^{\mathbf{W}_{\rm new}}(\mathbf{x}))$ that is smaller than $J(\mathbf{y},\, f^{\mathbf{W}}(\mathbf{x}))$. To deal with large data sets, the gradients in each step are typically computed using only a small, random subset of the training data set that is different in each update step; this corresponds to a *stochastic gradient descent* algorithm. In our work, we use a more sophisticated version of this type of optimizer, the ADAM optimizer (Kingma & Ba 2014).

'Convolutional neural networks' (CNNs) refer to the method of learning a set of convolution filters as part of the learning process. These convolution filters act as feature extractors, by convolving the input data (or the input data to a given layer in the ANN) with the filter. Useful features are extracted after these convolutional layers and, because they usually are not densely packed in the input data, we can apply a technique called max-pooling to reduce the size of our neural network, thus prevent overfitting. Max-pooling of size n takes the maximum of n pixels in non-overlapping subregions of incoming data, thus reducing the size of the input. Another advantage of max-pooling for spectroscopic analysis is that it induces a degree of translational invariance, making the analysis independent of small errors in the radial velocity correction.

2.2 Dropout variational inference

Variational inference is a general Bayesian inference method where the PDF is obtained not by sampling – as is the case when one uses Markov Chain Monte Carlo methods – but rather by fitting an approximation to the PDF using an objective function obtained by variational calculus. This method of Bayesian inference has the advantage that it can be applied to problems with large numbers of parameters, because optimization is in general faster and easier than sampling. This advantage comes at the expense of accuracy: the obtained PDF is only an approximation to the true PDF.

A Bayesian NN with dropout variational inference works by approximating the true PDF for the weights – which can number in the millions in our application below – as a product of Bernoulli distributions. It can then be shown that a NN trained with dropout applied to every layer except the last one and which has a Gaussian prior on the weights is an approximation to the full Bayesian NN (Gal & Ghahramani 2015). The Gaussian prior is achieved by imposing L2 regularization, parametrized by a regularization constant λ, on the loss function as

$$J_{\rm regularized}(\mathbf{y}, \hat{\mathbf{y}}) = J(\mathbf{y}, \hat{\mathbf{y}}) + \lambda \mathbf{W}^2\,. \tag{2}$$

The hyperparameter λ is determined using the validation set: we set λ to the value that optimizes the NN precision on a validation set. L2 regularization is equivalent to a Gaussian prior in the Bayesian interpretation.

Dropout (Hinton et al. 2012) is a technique that is primarily used to avoid overfitting in NNs, because deep NNs usually have more parameters than data points. Dropout multiplies the hidden neurons – that is, those not in the input or output layer – by a Bernoulli distributed random variable that take the value 1 with a certain probability and zero otherwise. When neurons are multiplied by zero they are effectively dropped; doing this during training prevents neurons from co-adapting to the training data, which would otherwise lead to overfitting. An example of a given instance of dropout for the example ANN in Fig. 1 is shown in Fig. 2.

To use dropout for uncertainty estimation, we run N times Monte Carlo dropout in forward passes through the network; in other words, we keep dropout turned on to make predictions using the ANN. Since dropout drops weights randomly, the NN is probabilistic and has different predictions in every forward pass through the network. The mean value of predictions will be the final prediction and the standard deviation of predictions will be the model uncertainty. In addition to this 'model uncertainty', the NN that we use also gives a 'predictive uncertainty' (see below for how we obtain the predictive uncertainty). The total uncertainty is the sum of model and predictive uncertainty in quadrature (Kendall & Gal 2017).

In more mathematical terms, the Bayesian NN predicts $\{\hat{\mathbf{y}}, \hat{\sigma}^2\} = f^{\widehat{\mathbf{W}}}(x)$ where $\hat{\mathbf{y}}$ is the prediction of the labels, $\hat{\sigma}^2$ is the predictive variance, $f^{\widehat{\mathbf{W}}}$ is the NN with randomly masked weights due to dropout, and $\mathbf{x}$ is the input data. We run the forward pass in the NN

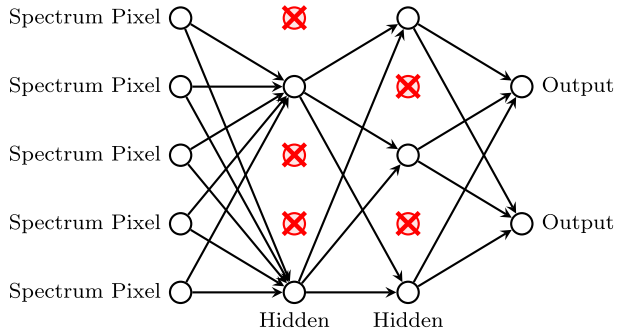

Figure 2. An example of dropout: a certain fraction of neurons are randomly dropped when evaluating the ANN. This is primarily used to prevent overfitting, but can also provide uncertainty estimation when it is used as an approximation to a Bayesian NN .

N times and obtain a set of $\{\hat{\mathbf{y}}_i, \hat{\sigma}_i^2\}_{i=1}^N$. The final prediction $\hat{\mathbf{y}}$ and uncertainty intervals $\hat{\sigma}$ is

$$\hat{\mathbf{y}} \pm \hat{\sigma} = \frac{1}{N}\sum_{i=1}^{N}\hat{\mathbf{y}}_i \pm \sqrt{\frac{1}{N}\sum_{i=1}^{N}\hat{\mathbf{y}}_i^2 - \left(\frac{1}{N}\sum_{i=1}^{N}\hat{\mathbf{y}}_i\right)^2 + \frac{1}{N}\sum_{i=1}^{N}\hat{\sigma}_i^2}\,. \tag{3}$$

2.3 Objective function for incomplete and noisy training data

The objective function $J(\mathbf{y}, \hat{\mathbf{y}})$ is the function that the NN aims to minimize to train the network. Generally, NN s for regression use the mean squared error (MSE), defined as

$$\text{Mean Squared Error} = J_{\text{MSE}}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N}\sum_{i=1}^{N}(\hat{\mathbf{y}}_i - \mathbf{y}_i)^2\,. \tag{4}$$

The MSE is prone to overfitting to outliers due to the squared term and it does not take uncertainty in the training labels into account.

However, astronomical data and observations are often incomplete and noisy. For example, in the case of spectroscopic data, the abundance of an element [X_1/H] may have only a single absorption line in the wavelength range by the detector. But due to reasons such as cosmic rays or if the line's Earth-frame wavelength happens to overlap with a strong sky emission line, the only absorption line [X_1/H] may not be measurable. But other abundances [X_2/H] can still be accurately measured. Previous data-driven approaches on spectroscopic data like the `Cannon 2` (Casey et al. 2016) or `StarNet` (Fabbro et al. 2018) needed to filter such spectra from the training set because of the data incompleteness.

For the Bayesian NN used in this work, we employ the following robust objective to get the loss for label i and assume that unavailable data are labelled using `MAGIC NUM`

$$J(y_i, \hat{y}_i) = \begin{cases} \frac{1}{2}(\hat{y}_i - y_i)^2 e^{-s_i} + \frac{1}{2}(s_i) & \text{for } y_i \neq \texttt{MAGIC NUM} \\ 0 & \text{for } y_i = \texttt{MAGIC NUM} \end{cases} \tag{5}$$

In this expression, $s_i = \ln\left[\sigma^2_{\text{known},i} + \sigma^2_{\text{predictive},i}\right]$, which corresponds to the natural logarithm of the sum of the known uncertainty variance in the labels and an additional predictive variance. This predictive variance is also learned by the NN and forms another output from the NN for each input. The known uncertainty variance is that returned by the reduction pipeline that produces the labels for the training subset. In general, the NN can be trained to give the predictive variance without any known variance in the labels, which is a form of unsupervised training. The predictive variance from the loss function learned by the NN represents any variance in the training set that cannot be explained by the known variance. This predictive variance contributes to the error budget for predictions on new data (see equation 3).

The final loss for the stochastic gradient descent is calculated from a mini-batch partition of the data consisting of N data point and D labels

$$J(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{D}\sum_{i=1}^{D}J(y_i, \hat{y}_i)\right)\mathcal{F}_{\text{correction},i}\,, \tag{6}$$

where $\mathcal{F}_{\text{correction},i}$ is a correction term to correct for the fact that in equation (5) we effectively assume that the NN made no error for missing data. If D is the overall number of labels and D_i is the number of labels not equal to `MAGIC NUM` for data point i, then

$$\mathcal{F}_{\text{correction},i} = \frac{D}{D_i}\,. \tag{7}$$

The objective function in equation (5) acts as a robust version of the conventional MSE objective in equation (4), allowing the NN to take the effect of uncertain and missing labels into account. This makes the model more robust because high uncertainty in a training label will have a smaller effect on the loss, preventing the NN to learn from such labels. Equation (5) also assumes that, without any other information, the prediction from the NN is accurate and thus back-propagates zero loss for incomplete labels. The correction term $\mathcal{F}_{\text{correction},i}$ in equation (6) is factored into the final loss in order to prevent the effective learning rate from decreasing due to the presence of missing labels. $\mathcal{F}_{\text{correction},i}$ equals one in which there are no missing labels, in which case it resembles a conventional loss function.

3 HIGH-RESOLUTION SPECTROSCOPIC DATA FROM APOGEE

The main source of data that we use are spectra and derived labels from the APOGEE Data Release 14 (Abolfathi et al. 2018; Holtzman et al. 2018; Jönsson et al. 2018). APOGEE spectra are obtained with a 300-fibre spectrograph (Wilson et al. 2010) attached to the Sloan Foundation 2.5 m telescope at Apache Point Observatory (Gunn et al. 2006). APOGEE is an infrared (1.5–1.7 μm), high resolution ($R \sim 22,500$), high signal-to-noise ratio (typical SNR > 100) spectroscopic survey. The APOGEE data set contains stellar parameter and chemical abundances obtained using the APOGEE Stellar Parameter and Chemical Abundances Pipeline (ASPCAP; García Pérez et al. 2016). ASPCAP is an automated pipeline for determining the stellar labels from observed spectra by comparing observed spectra to a pre-computed library of theoretical spectra using χ^2 minimization. We describe how we select data from the overall APOGEE DR14 catalogue and how we define our training and tests sets in Section 3.1. In Section 3.2 we discuss the method to process the data in the training and test sets.

3.1 Training, test, and validation data selection from APOGEE DR14

We have created one training and two test sets from the set of APOGEE DR14 spectra. Each data set consists of continuum normalized spectra, 22 ASPCAP labels (T_{eff}, $\log g$, [C/H], [CI/H], [N/H], [O/H], [Na/H], [Mg/H], [Al/H], [Si/H], [P/H], [S/H], [K/H], [Ca/H], [Ti/H], [TiII/H], [V/H], [Cr/H], [Mn/H], [Fe/H], [Co/H],

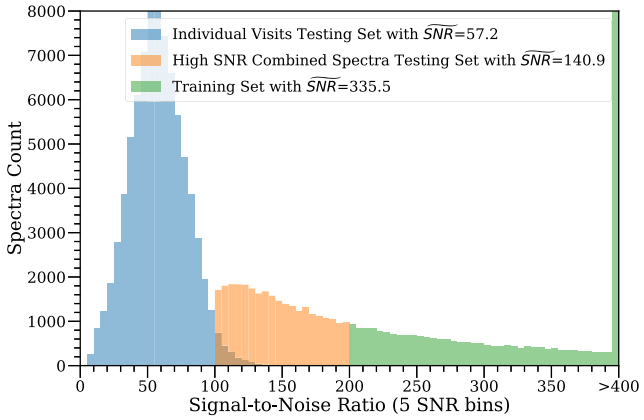

Figure 3. Signal-to-noise (SNR) ratio distribution in the training set and in the two test sets used in this work. $\widetilde{SNR}$ represents median SNR. All spectra with ASPCAP reported SNR > 400 are set to SNR = 400, leading to the peak at high SNR. High SNR combined spectra test set refers to combined spectra with 100 < SNR < 200. Individual visits test set refers to the set of individual visits of stars in the high SNR combined-spectra test set.

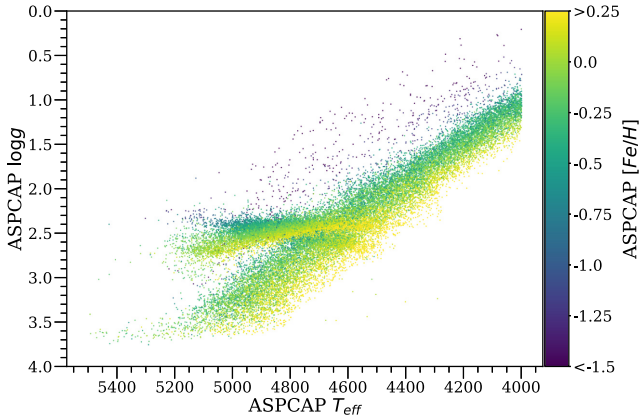

Figure 4. log g versus $T_{\rm eff}$ coloured by [Fe/H] abundances in the training set; all the labels in the training set are determined by ASPCAP. All main-sequence stars in APOGEE DR14 have log g set to MAGIC NUM = −9999 and are not displayed in the plot.

[Ni/H])[3] and their associated ASPCAP uncertainty. ASPCAP determines the abundances of all of these elements using synthetic spectra computed using the line list from Shetrone et al. (2015). The SNR distributions of these subsets are shown in Fig. 3. In Fig. 4 we display the training sample in the space of $T_{\rm eff}$ and log g coloured by [Fe/H]. The training set consists of 33 407 spectra that have SNR > 200. At the start of training, 90 per cent of the training set is randomly selected to train the neural network – that is, used to compute the gradients of the objective function in the training steps – and the remaining 10 per cent constitute a separated validation set used to validate the performance of the NN during the training process. A total of 4.6 per cent of the combination of all training ASPCAP labels are −9999, the value used by ASPCAP to represent highly uncertain or unavailable labels; as discussed in Section 2.3, these spectra are still used in our robust objective function (thus, MAGIC NUM = −9999 for APOGEE in equation 5). Two test sets are used, one consists of spectra with SNR between 100 to 200, which are called the high-SNR test set. These spectra are picked from the set of 'combined' APOGEE spectra, which are the combinations of the individual exposures and it is these combinations that are used by ASPCAP for their analysis (most APOGEE spectra are obtained as a set of at least three individual hour-long exposures to obtain SNR > 100; Holtzman et al. 2015). This set of spectra is entirely separate from the training set. The other test set consists of 81 483 spectra picked from the set of individual visit spectra that go into the combined spectra in the high-SNR test set. These spectra in the individual visit test set have much lower SNR than the spectra in the set of combined training spectra (which all have SNR > 200), see Fig. 3. The advantage of this test set is that we are interested in the performance of our NN for spectra with low SNR, but ASPCAP does not provide labels for individual-visit spectra with low SNR. However, we do have NN or ASPCAP labels for the high-SNR combinations of the individual visit spectra, which we can use to test our method. Because the low SNR test set consists of the same *stars* as the high SNR test set, the labels in the high SNR test set are representative of those in the low SNR test set, even though the *noise* in them is not. Thus, the low SNR test set provides a stringent test of how this method performs at low SNR.

On top of the SNR cut, we perform cuts on the values of the stellar parameters. This is necessary, because all of the knowledge learned by the NN is solely driven by the training data. Therefore, we need to make sure that the training labels are as accurate as possible, because any systematic inaccuracy such as bias at lower SNR will be captured by the NN and propagated to new test data. For this reason, we exclude spectra with surface temperature $T_{\rm eff}$ smaller than 4000 K or higher than 5500 K, because at these temperatures ASPCAP may not be accurate (García Pérez et al. 2016). Additionally, we remove spectra flagged with the APOGEE_ASPCAPFLAG or APOGEE_STARFLAG flags and spectra with a radial velocity scatter larger than 1 km s^{-1}, because these represent potential issues with specific labels, issues with spectra, and potential binary stars, respectively. These cuts ensure the quality of the training and test set.

ASPCAP determines abundances by performing a χ^2 fit on an interpolated, large grid of synthetic spectra computed for the APOGEE wavelength region (García Pérez et al. 2016). After performing the fit, ASPCAP made several calibrations of the stellar parameters and abundances based on the consistency of abundances within open and globular clusters and by comparing to external data. The synthetic spectra are computed under various simplifying assumptions and using line lists that are not 100 per cent complete and this limits the quality of the synthetic grid used by ASPCAP and thus ultimately limits the NN accuracy. Any systematic bias will be propagated from ASPCAP to NN during training. This is a disadvantage of the specific supervised machine-learning method that we are using here (we discuss advantages and disadvantages of our method compared to other methods in more detail in Sections 7.2 and 7.3).

The resulting training set contains both main-sequence dwarfs and red giants. However, ASPCAP does not report calibrated values of log g for main-sequence stars and these are all set to MAGIC NUM = −9999. These log g labels are therefore ignored in our train-

[3][CI/H] and [TiII/H] are measurements of the carbon and titanium abundance using spectral regions that only have neutral atomic carbon and singly ionized atomic titanium features, respectively. The overall [C/H] and [Ti/H] are mostly determined by molecular (for carbon) and neutral atomic (for titanium) features. Because ASPCAP returns these measurements separately, we include them also as part of our label set. The masking windows used can be accessed with the PYTHON function described in https://astronn.readthedocs.io/en/v1.0.0/tools_apogee.html#retrieve-aspcap-elements-window-mask.

ing procedure, although other labels such as $T_{\rm eff}$ for main sequence are used. This means that we cannot hope to determine good $\log g$ for main-sequence stars with our NN . We discuss this further below.

3.2 Data reduction for training

All methods for spectroscopic analysis work with continuum-normalized spectra, because the information about stellar labels is mainly contained in narrow spectral features and the overall continuum is typically not well calibrated. The method used for continuum normalization is important, because we require a method that is invariant with respect to SNR due to the fact that we train the NN on high SNR spectra only but also test it on low SNR spectra. To accomplish this, we use the same method as employed by the `Cannon 2` method (Casey et al. 2016), where continuum normalization is performed by using a set of 'continuum pixels' – identified as pixels that depend little on the stellar labels using their data-driven model for APOGEE spectra – and fitting a continuum to the flux in only these pixels. The spectrum in each APOGEE detector is normalized separately in this method, with a two-degree polynomial that has been demonstrated to be effective by Casey et al. (2016). While we use DR14 spectra, we use a set of continuum pixels obtained using DR12 spectra that is included in the `apogee` software package (Bovy 2016). Spectra in DR14 extend over a slightly wider wavelength range for each detector than those in DR12, which means that our continuum normalization does not perfectly capture the behaviour of the continuum at the edges of the detector. But as we show below, the exact method of continuum normalization does not have a large effect on the performance of our method.

Fig. 5 shows the difference between the continuum normalization used by ASPCAP and that described above for the combined spectrum of 2M19060637 + 4717296 as an example. An ideal continuum normalization would place continuum pixels to have a flux of 1. It is clear that the ASPACP normalization fails to do so (note that this is by design: the DR14 ASPCAP analysis explicitly does not attempt to do this; see Holtzman et al. 2018). Aside from an overall offset, the continuum-normalized spectra are similar for both methods.

After continuum normalization, we check the APOGEE pixel-level mask bits in the `APOGEE_PIXMASK` bitmask and set the flux value of pixels that contain the following bits to 1 (the expected continuum): **0**: bad pixel, **1**: cosmic ray, **2**: saturated, **3**: unfixable, **4**: bad from dark, **5**: bad from flat, **6**: high error, **7**: no sky info, **12**: overlaps a significant sky line.

Besides the continuum normalization of the spectra, we need to standardize the labels and continuum normalized spectra to be able to easily use them with standard NN methods. For the labels, we subtract the mean and divide by the standard deviation such that the training labels all approximately have a mean of 0 and a standard deviation of 1. In our case we have 22 labels, and therefore we calculate 22 means and 22 standard deviations to standardize the labels. Labels that are `MAGIC NUM` $=-9999$ (that is, missing) are not involved in the normalization process, i.e. -9999 values remain constant such that objective function in equation (5) can recognize these missing labels during optimization. For continuum-normalized spectra, we calculate the mean of the flux pixel-by-pixel for the spectra in training set, and subtract the means from all the spectra in the training set. In other words ideally an average spectrum should be a flat straight line at flux $= 0$ after this final normalization step. The reason behind this normalization is that this maps the average spectrum to the average label for a NN with all weights set to zero and we expect that the average spectrum should have stellar labels close to the average of those in the training set. Thus, it will be easier and faster for the NN to converge to a minimum.

While performing variational inference on test sets, it is important to use the same normalization procedure and the same set of means and standard deviations as used for the training set to normalize and denormalize all the data. This is also the reason why we do not scale the training spectra to have standard deviation of 1, because the training and testing spectra have completely different SNR. In other words, training set spectra have high SNR, thus low overall standard deviation and test set spectra have low SNR, thus higher overall standard deviation. Using the parameters to standardize training spectra will not standardize testing spectra so we chose not to scale any spectra.

4 PERFORMANCE ON APOGEE DATA

The main NN that we train and test with APOGEE spectra in this work is the network `ApogeeBCNNCensored()` in `astroNN` (see the Appendix). The NN architecture is shown in Fig. 6. Users only have to provide a continuum-normalized spectrum and are returned a prediction and associated uncertainty. Rather than using a single, simple NN to predict stellar parameters and elemental abundances from an input spectrum, we use a combination of (i) a large NN trained on the full spectral wavelength range to predict [$T_{\rm eff}$, $\log g$, [Fe/H]] (the big grey-coloured network on the right-hand side in Fig. 6) and (ii) 19 mini-neural networks used to predict the 19 [X/H] abundances based on fixed regions of the spectrum that contain known spectral features for each element and the overall [$T_{\rm eff}$, $\log g$, [Fe/H]]. This architecture mimics that of traditional spectroscopic analysis and of ASPCAP, where the overall stellar parameters ($T_{\rm eff}$, $\log g$, [Fe/H], etc.) are determined first and individual abundances are determined afterwards from specific spectral features. However, our method differs from this in a crucial aspect: the network trained to predict [$T_{\rm eff}$, $\log g$, [Fe/H]] from the full spectrum also has a two-neuron connection characterized by two latent variables to the 19 mini-networks used to predict the individual abundances (in addition to feeding [$T_{\rm eff}$, $\log g$, [Fe/H]] to the mini-networks as well). We choose to use two neurons as the connection to mimic ASPCAP, in which [C/Fe] and [α/Fe] are fitted to the full spectrum, because these elements strongly affect the stellar photosphere and thus the formation of all spectral lines. By using two neurons we can learn a latent space similar to these two elements, but we do not require the latent variables to exactly correspond to [C/Fe] and [α/Fe] to give the network the opportunity to learn a better low-dimensional set of latent variables. This allows the mini-networks to use a limited amount of information from the full spectrum that is not captured by [$T_{\rm eff}$, $\log g$, [Fe/H]] in making their predictions. To produce the 19 masked spectra for the 19 mini-networks (one mask per element) we use the windows employed by ASPCAP DR14 to determine individual abundances (García Pérez et al. 2016). These windows were derived by the ASPCAP team using synthetic spectral syntheses to isolate regions of the spectra most sensitive to individual elements. Unlike ASPCAP, we only use the windows as a binary mask – pixels are either in or out – we do not use the weights assigned to pixels within the windows.

The entire combination of the large ANN to predict [$T_{\rm eff}$, $\log g$, [Fe/H]] and the 19 mini-networks, including their two-neuron connection, is trained simultaneously. To achieve this, two unconventional layers are included in the network, `StopGrad` and `MaxNorm`. `StopGrad` is an identity transformation layer with the property that its gradient is always set to 0 during training, but

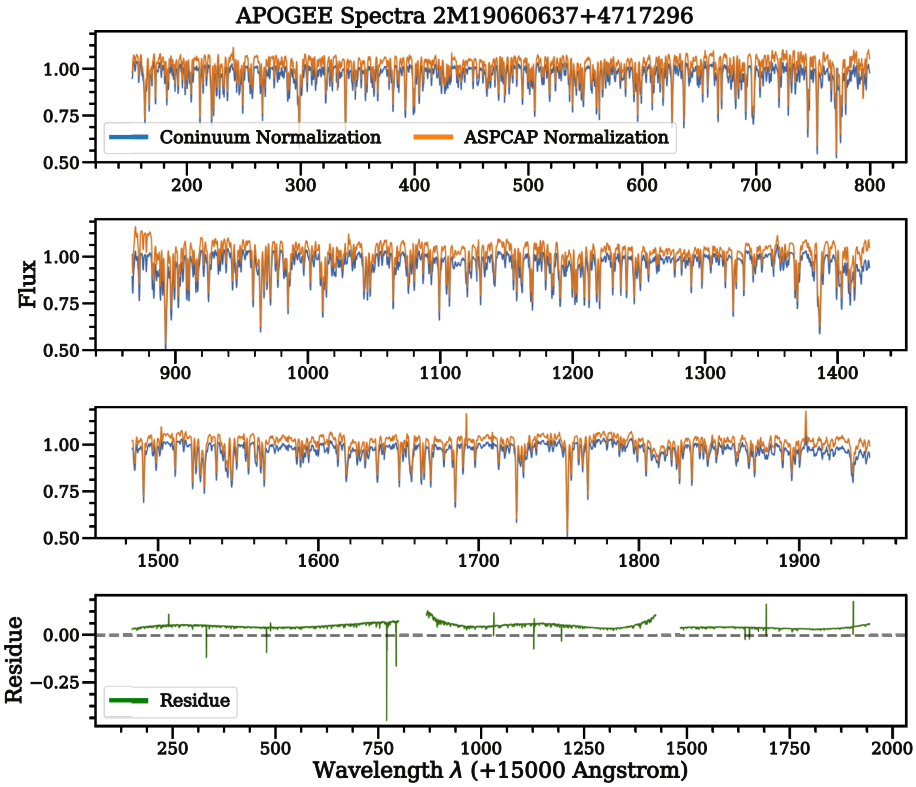

Figure 5. Example of continuum normalization: this figure shows the continuum-normalized combined APOGEE spectrum of 2M19060637 + 4717296, showing the difference between the continuum-normalization performed by ASPCAP spectra and that used in our method. The top three panels display the spectrum in three parts (corresponding to the three detectors used in the APOGEE instrument), while the bottom panels show the difference between the two. The difference between the two normalization methods is a relatively smooth function of wavelength and there is an overall offset due to the fact that ASPCAP does not attempt to trace the actual continuum, while our method does.

otherwise is simply 1 (e.g. when computing the sensitivity of the network output to input; see Section 4.4). This layer prevents the error from an individual abundance [X/H] to be back-propagated during training to the network predicting [$T_{\rm eff}$, $\log g$, [Fe/H]]. That is, we do not allow prediction errors in [X/H] to affect the training of the network that predicts the stellar parameters. `MaxNorm` is a weight-constraint layer that requires $\sqrt{\sum w^2} \leq \delta$, where w are the weights and δ is a constraint constant. δ is determined using the validation data set, such that δ optimizes the performance of NN on the validation set. This layer prevents the mini-neural networks that predict [X/H] from paying too much attention to the full spectrum. The reason why we construct this rather complex network rather than the more straightforward single network that predicts all the labels from the full spectrum is discussed in detail in Section 5.1. Briefly, the reason is that when allowing the full spectrum to inform individual abundances, the ANN predictions in regions of label-space with few training data become highly correlated due to correlations in the training data.

We discuss the performance of the `ApogeeBCNNCensored()` network in detail in the following subsections. We will see that we find that the NN trained on high SNR training data and tested on high SNR testing data displays a fairly high bias, which may result from errors in ASPCAP, but a small amount of scattering. When testing on low SNR individual-visit spectra, the network shows almost no bias and small amounts of scattering (but larger than that at higher SNR). The results are considerably better than any previous work on applying machine-learning techniques to high-resolution spectral analysis. The uncertainty estimation from dropout variational inference that we find makes sense, because it correlates with $1/\sqrt{\rm SNR}$ and is similar to the scatter in the residuals between our method and ASPCAP. For both test sets, we find that the NN performs the best for elements that ASPCAP reports as being their most accurate elements (e.g. [Mg/H] and [Ni/H] in an independent validation of ASPCAP DR13/14 by Jönsson et al. 2018). A sensitivity analysis of how the ANN outputs depend on the input spectra shows that the model depends in a reasonable manner on wavelength. All our results can be reproduced using our online code (see the Appendix), but due to the stochastic nature of dropout and the NN training process, it is impossible to reproduce the exact same results. However, statistically, the results should be very close to those described in this work.

In the following, we define the residual as

$$\text{Residual} = \textit{NN}\text{Prediction} - \text{ASPCAP}\,, \tag{8}$$

where *NN* refers to the neural network. As a robust measurement of the scatter, we use a measure based on the median absolute deviation (MAD): $\sigma^{\rm MAD} = 1.4826\,{\rm MAD}$, where the factor is such that for a Gaussian distribution $\sigma^{\rm MAD}$ equals the Gaussian standard deviation. Thus, for a set of residual R: $[R_1, R_2, ..., R_n]$, $\sigma^{\rm MAD}$ is

$$\sigma^{\rm MAD} = 1.4826\,{\rm median}\,(|R_i - {\rm median}(R)|)\,. \tag{9}$$

In all of these calculations, ASPCAP labels that are equal to −9999 (or, more generally, labels equal to `MAGIC NUM`) are excluded from the calculation.

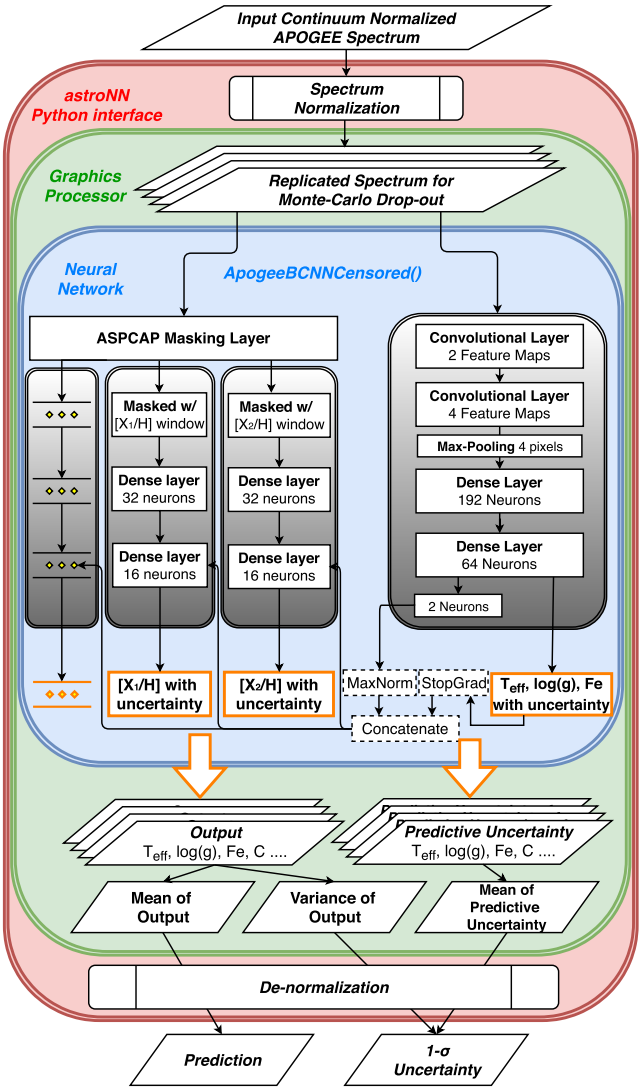

Figure 6. The neural-network architecture mainly used in this work, defined as `ApogeeBCNNCensored()` in `astroNN`.

4.1 Comparison to ASPCAP at high signal-to-noise ratio

We first test the performance of the NN for high SNR spectra, which we define here as spectra with SNR between 100 and 200 (the standard cuts described in Section 3.1 apply as well). In this section, we evaluate the performance of the NN by comparing it to the predictions from ASPCAP for the same stars. Thus, any comparison is affected by biases in the ASPCAP results themselves and we will see that there are reasons to believe that ASPCAP suffers from SNR-dependent biases, even at these high SNRs. These biases inflate the size of the residuals between the NN predictions and ASPCAP. We also do not account for the random uncertainties in the ASPCAP predictions, which also contribute to the size of the residuals. In this sense, the biases and errors derived from comparing to ASPCAP from this section are an upper limit on the size of the biases and errors of the NN predictions. In the next section, we will compare the NN prediction against itself and find much smaller residuals.

A summary of the results is given in Table 1. Fig. 7 displays the predicted $\log g$ versus $T_{\rm eff}$. In this figure, the left-hand panel is colour-coded by [Fe/H] and we overlay PARSEC isochrones

Table 1. Neural-network prediction result on the high SNR test set from comparing NN predictions to ASPCAP.

Label	Median of residual	$\sigma^{\rm MAD}$ of residual
$T_{\rm eff}$	−20 K	30 K
$\log g$	0.012 dex	0.051 dex
[C/H]	0.003 dex	0.040 dex
[CI/H]	0.013 dex	0.058 dex
[N/H]	− 0.004 dex	0.041 dex
[O/H]	− 0.021 dex	0.046 dex
[Na/H]	− 0.01 dex	0.16 dex
[Mg/H]	0.000 dex	0.027 dex
[Al/H]	− 0.038 dex	0.071 dex
[Si/H]	0.000 dex	0.029 dex
[P/H]	− 0.02 dex	0.10 dex
[S/H]	0.006 dex	0.051 dex
[K/H]	− 0.013 dex	0.049 dex
[Ca/H]	− 0.015 dex	0.033 dex
[Ti/H]	− 0.029 dex	0.052 dex
[TiII/H]	0.06 dex	0.17 dex
[V/H]	− 0.009 dex	0.097 dex
[Cr/H]	− 0.002 dex	0.048 dex
[Mn/H]	− 0.018 dex	0.038 dex
[Fe/H]	− 0.004 dex	0.020 dex
[Co/H]	− 0.02 dex	0.14 dex
[Ni/H]	0.003 dex	0.029 dex

(Bressan et al. 2012) at four different metallicities to indicate the expected location of stars in this space. It is clear that the predicted parameters for stars at different metallicities conform well to these expectations. That the predicted $\log g$ is highly precise for giants is clear from the narrowness of the red clump. The right-hand panel of Fig. 7 shows the same predicted $\log g$ versus $T_{\rm eff}$, but now colour-coded by the NN uncertainty in $\log g$. The NN is highly confident in its $\log g$ prediction along the well-populated parts of the giant branch. However, the uncertainty in $\log g$ is large for the group of stars at high $\log g$. This is reasonable, because these are main-sequence dwarfs for which we have no $\log g$ training data (see discussion in Section 3.1 above). The predicted $\log g$ values are clearly wrong, but this is reflected in the high uncertainties for these stars. Similarly, the uncertainty in $\log g$ is high for low metallicity, low-$\log g$ giants, for which training data are sparse (see Fig. 4).

The results in Table 1 show that the NN prediction displays a relatively high bias in all labels when compared to ASPCAP, especially in $T_{\rm eff}$. It is possible that the majority of this bias comes from the ASPCAP prediction on lower SNR spectra instead of the NN. The NN predictions are consistent across a wide range of SNR, while the ASPCAP $T_{\rm eff}$ is probably biased at SNR < 200. Fig. 8 shows the median absolute error (MAE) of ASPCAP−NN and `Cannon`−NN between SNR 50 to 300 for the entire APOGEE DR14 catalogue with good `APOGEE_ASPCAPFLAG` and `APOGEE_STARFLAG` flags, velocity scatter smaller than 1 km s^{-1}, and ASPCAP $4000 < T_{\rm eff} < 5500$. Most labels show strong SNR-dependent trends that are above the empirical precision found for the NN (see discussion below and Fig. 11). The bias with respect to ASPCAP is generally larger than that compared to the `Cannon 2`. These SNR-dependent trends demonstrate that both ASPCAP and the `Cannon 2` are probably biased to a larger degree and at higher SNR than previously thought.

Fig. 9 shows more detailed results on the performance of the NN . In this figure, the blue line represents the MAE between the NN and ASPCAP in bins of ASPCAP labels, the orange error bars represent the median total NN uncertainty in these bins, and the green line gives the contributions of the NN model uncertainty to the total

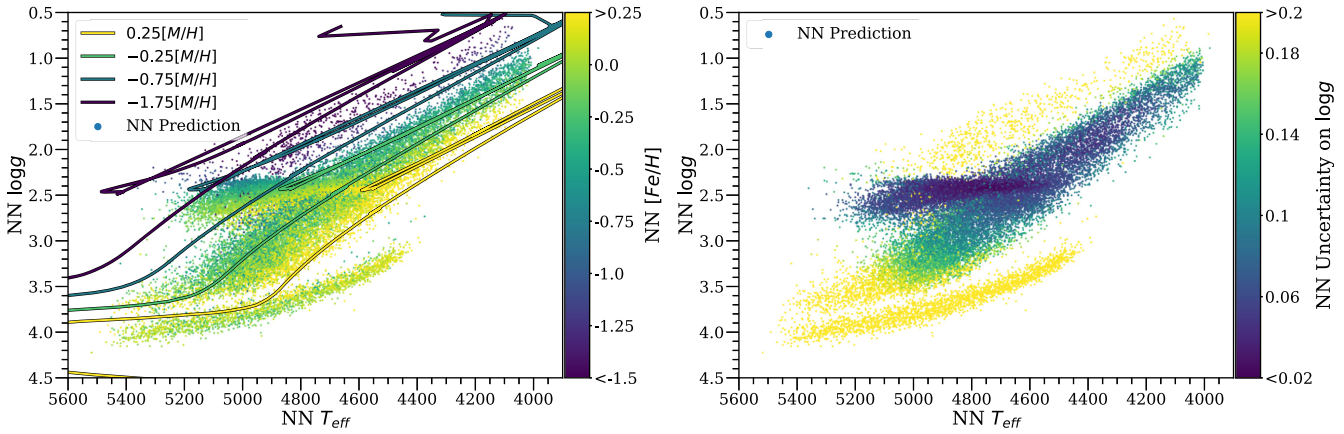

Figure 7. Neural network $T_{\rm eff}$ and $\log g$ prediction colour-coded by [Fe/H] (left-hand panel) and $\log g$ uncertainties (right-hand panel). The group at high $\log g$ is due to 2611 spectra in the high-SNR test set that are dwarfs with `MAGIC NUM` $= -9999$ ASPCAP value. $\log g$ values for these stars are absent in the training data, because ASPCAP in DR14 does not provide the $\log g$ for dwarfs. The NN clearly predicts the wrong values for $\log g$ but this is also reflected in the large uncertainties for these stars in the right-hand panel. For test set objects along the giant branch, the NN returns reasonable parameters (compare to Fig. 4). Similarly, the uncertainty in $\log g$ is high for low metallicity, low $\log g$ giants, for which training data are sparse.

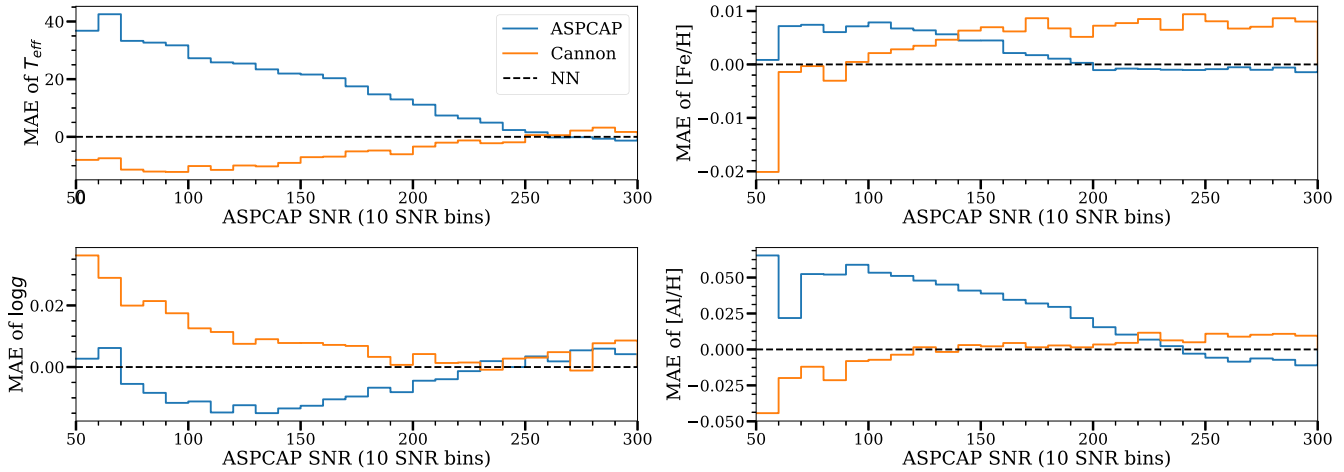

Figure 8. Median offset between our NN method, ASPCAP, and the `Cannon 2`. Each panel shows the median absolute error (MAE) of ASPCAP or the `Cannon 2` assuming that the NN is the ground truth in bins of SNR for the combined APOGEE spectra. The panels show the MAE of $T_{\rm eff}$, $\log g$, and [Fe/H], as well as [Al/H] as a representative element. The MAE bias between the NN and ASPCAP has a strong SNR dependence for $T_{\rm eff}$ and somewhat less strong for $\log g$. This trend is much smaller between the NN and the `Cannon 2`, especially for [Al/H]. Typically, the bias between ASPCAP and the `Cannon 2` (the difference between the blue and orange curves) is higher than that between either of them and the NN. The strong trend with SNR when comparing the NN to the `Cannon 2` or to ASPCAP may be due to a SNR-dependent bias in ASPCAP.

uncertainty. The difference between the orange and green error bar therefore gives a sense of the contribution of the predictive uncertainty (the uncertainties are added in quadrature, so the predictive uncertainty is not simply the difference between the orange and green uncertainties). Both the accuracy and the precision are good for [X/H] $\gtrsim -0.7$, especially around solar metallicity where there is the most training data. The NN has lower accuracy at low [X/H] mainly due to two reasons: first, there are not much training data at low metallicity because [Fe/H] ≈ -0.7 is the lower bound of typical abundances in the Galactic disc and second, spectral features at low metallicity are less defined and therefore less informative about the abundances. The $\log g$ prediction is accurate and precise in the well-populated lower and mid regions of the giant branch, but becomes more uncertain for very luminous, low-$\log g$ giants. By and large, the uncertainties returned by the NN are similar to the typical MAE difference between the NN and ASPCAP. For some individual elements, the uncertainties are smaller than the typical MAE difference at low metallicity, but it is always the case that these uncertainties are quite large ($\gtrsim$0.25 dex), thus signaling that the NN prediction is noisy.

Fig. 10 similarly shows how the accuracy and uncertainties in the individual abundances depend on the surface temperature. Spectral features in stars with higher surface temperatures are weaker, so we expect the accuracy to decrease and the uncertainties to increase. Fig. 10 demonstrates that this is indeed the case, although in general the trend with temperature is quite weak.

4.2 Results at low signal-to-noise ratio

To test the NN at lower SNR, we make use of the set of individual-exposures for the stars in the high-SNR test set, as explained in

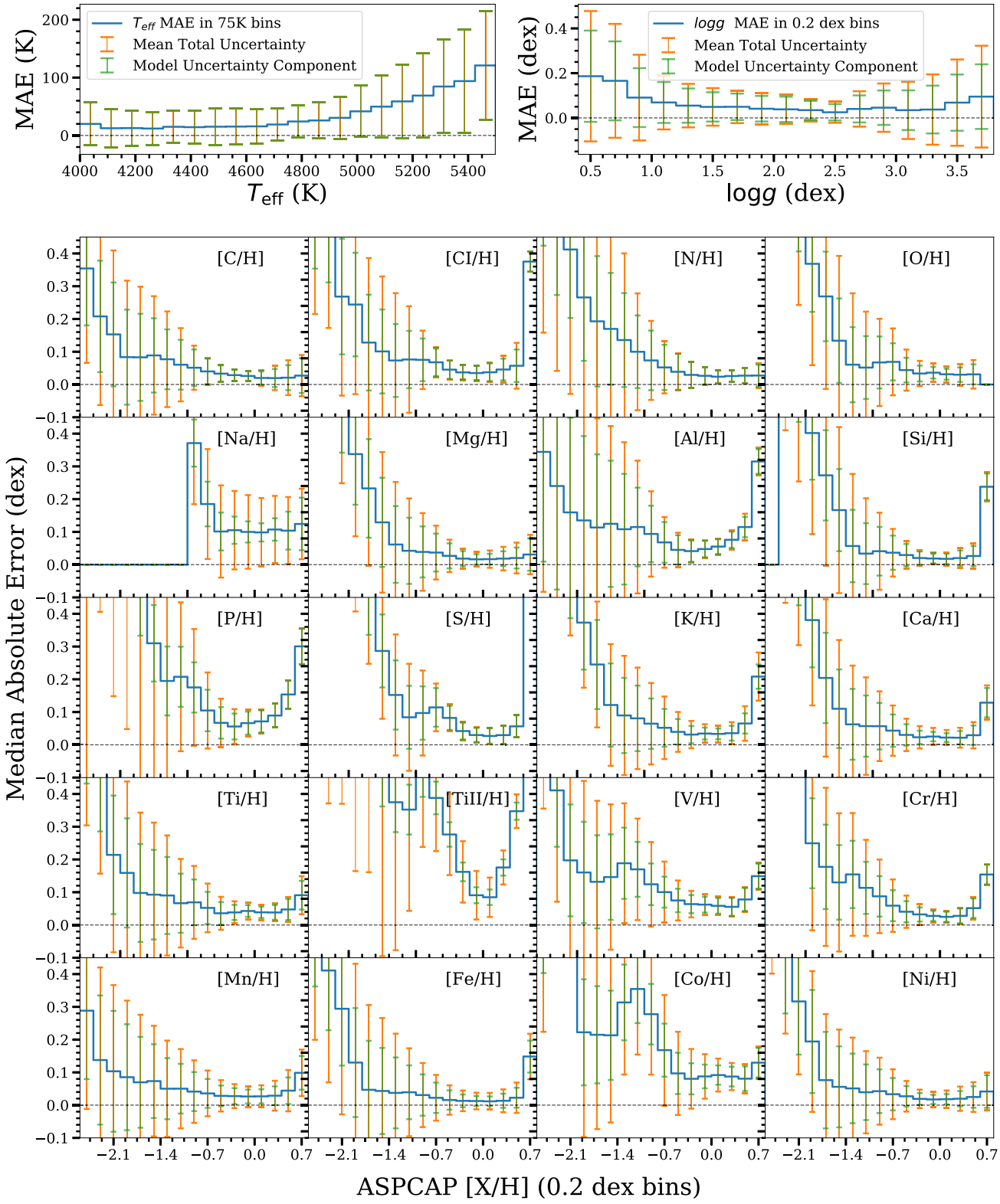

Figure 9. Comparison between NN predictions for $T_{\rm eff}$, $\log g$, and [X/H] and those from ASPCAP at high SNR (SNR between 100 and 200). The blue curve shows the MAE between the NN and ASPCAP in bins of the ASPCAP label, while the green and orange error bars give the NN model and total uncertainty. Overall the MAE is small and the uncertainties are similar to the MAE, but there are bigger residuals at low [X/H], because the training set contains few low-metallicity stars and spectral features are weaker for such stars, leading to worse performance of the NN.

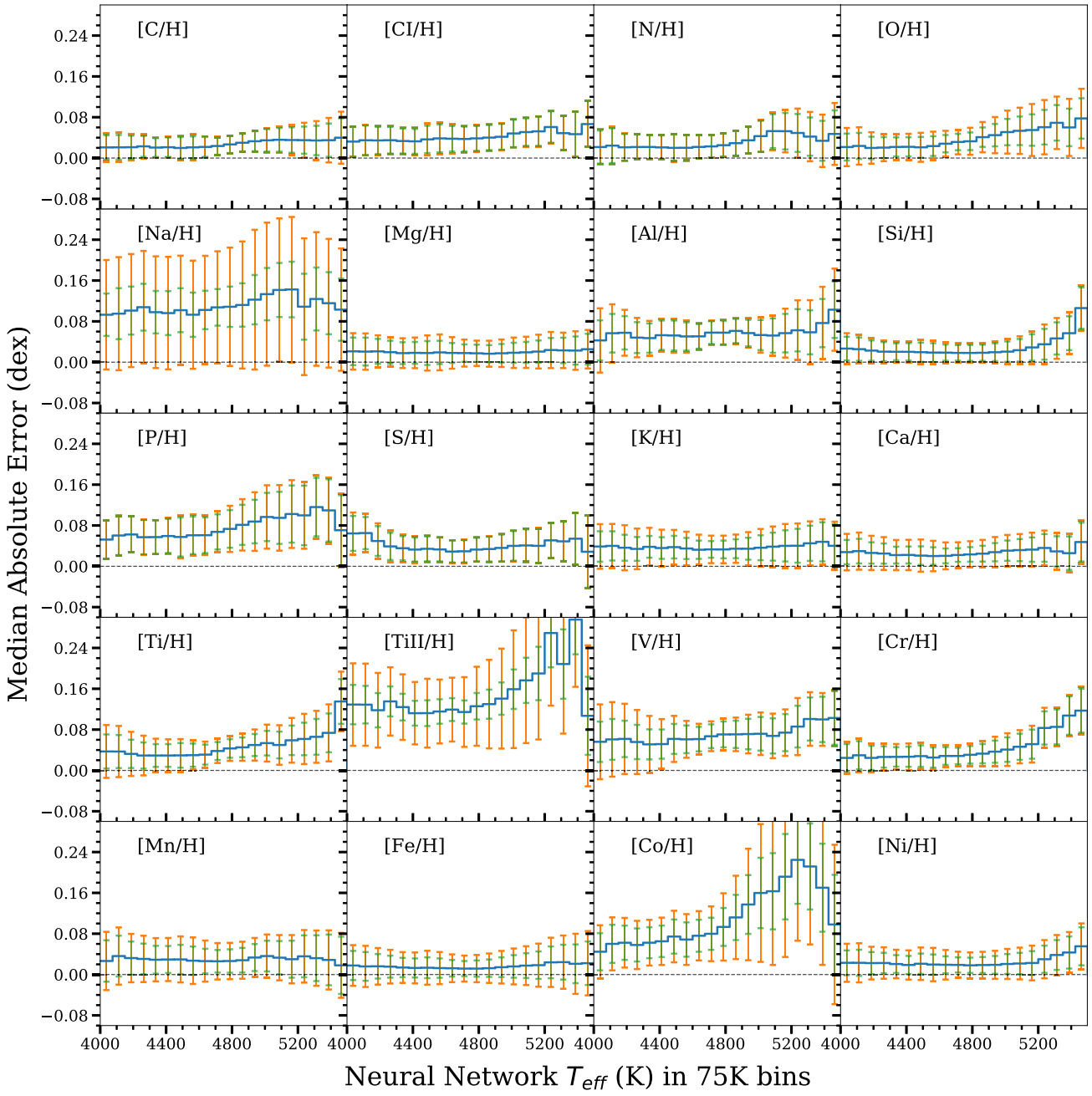

Figure 10. Comparison between NN predictions for [X/H] and the ASPCAP labels for the high SNR test set as a function of $T_{\rm eff}$. Curves and error bars are as in Fig. 9. Spectral features are weaker at higher $T_{\rm eff}$ leading to an increase in the typical residuals that is matched by an increase in the NN uncertainty.

detail in Section 3.1. For the combined spectra in the high-SNR test set, we have results from the NN and we can therefore test the performance of the NN on the low SNR individual exposures by comparing the predictions from the low SNR spectra to the results from the NN on the high SNR combined spectra.

The results from comparing the NN predictions on low SNR spectra to the NN measurements from their counterpart combined spectra are shown in Table 2. The first thing to note is that the bias (median of the residual) with respect to the NN is much smaller than it was in the high SNR comparison in Table 1. The main reason for this is that ASPCAP parameters and abundances are accurate at SNR>200 and the predictions from the NN using the individual exposures are essentially the same parameters as the prediction from their high SNR counterpart. This demonstrates that the NN predictions are robust at low SNR.

Fig. 11 shows the self-consistency of the NN at low SNR in more detail. Here, the MAE is calculated between the predicted NN label for the individual exposure and the predicted NN label from the high-SNR combined counterpart. All abundances show significant errors at SNR < 30. This is expected, because the noisy low-SNR spectra limit the ability of the NN to get information from spectral features. However, the NN still performs well at SNR ≈ 50, with most abundances measured to a few hundredths of a dex. The performance of the NN at SNR ≈ 50 is better than that of the `Cannon 2` (Casey et al. 2016), even though we only use regions of the spectra with known spectral features for the element

Table 2. Neural-network prediction result on the individual visits of low SNR test set spectra from comparing results on the combined spectra of the same test set.

Label	Median of residual	$\sigma^{\rm MAD}$ of residual
$T_{\rm eff}$	−3K	29K
log g	0.000 dex	0.047 dex
[C/H]	− 0.004 dex	0.045 dex
[CI/H]	− 0.003 dex	0.054 dex
[N/H]	− 0.002 dex	0.046 dex
[O/H]	0.000 dex	0.029 dex
[Na/H]	− 0.013 dex	0.060 dex
[Mg/H]	− 0.001 dex	0.023 dex
[Al/H]	− 0.005 dex	0.045 dex
[Si/H]	− 0.001 dex	0.024 dex
[P/H]	0.00 dex	0.10 dex
[S/H]	0.002 dex	0.054 dex
[K/H]	− 0.003 dex	0.030 dex
[Ca/H]	− 0.002 dex	0.021 dex
[Ti/H]	− 0.003 dex	0.026 dex
[TiII/H]	− 0.001 dex	0.043 dex
[V/H]	− 0.003 dex	0.052 dex
[Cr/H]	− 0.006 dex	0.029 dex
[Mn/H]	− 0.005 dex	0.030 dex
[Fe/H]	− 0.004 dex	0.020 dex
[Co/H]	− 0.010 dex	0.086 dex
[Ni/H]	− 0.005 dex	0.024 dex

of interest, while the `Cannon 2` uses the full spectrum for each element. The exact precision at SNR ≈ 50 is shown in each panel, with the best performance of 0.014 dex for [Fe/H] and [Ca/H] to the worst performance of 0.068 dex for [P/H]. Note that we measure *each* element's abundance to better than 0.10 dex, often considered the target uncertainty in large spectroscopic surveys, at SNR ≈ 50. Of course, at lower metallicities the performance is worse, similar to what is seen in Fig. 9.

At high SNR, the residual in the predictions for most labels flattens out to 0.01 to 0.02 dex. This demonstrates that for most of the abundances, the NN can reach a precision of ≈0.01 dex for high SNR spectra.

4.3 Results on open and globular clusters

To further test the performance of the NN predictions for the individual elemental abundances, we apply it to open and globular clusters. Open clusters provide a good testbed for data-driven abundance analyses, because they are a chemically homogeneous population of stars, at least at the level of current abundance precision (e.g. De Silva et al. 2006, 2007; Bovy 2016; Ness et al. 2018). Ideally, the NN predictions for the abundances of stars in open clusters should exhibit a very small spread. Similarly, globular clusters should be homogeneous in all but their lightest elements. For the light elements we expect to see spreads and anticorrelations between pairs of elements (e.g. Mg and Al; Mészáros et al. 2015).

To perform these tests, we select stars from two open clusters that are well populated in the APOGEE data base: M67 and NGC6819. We select members from the catalogue provided by Mészáros et al. (2013) and we exclude spectra with the `APOGEE_STARFLAG` bitmask set. The resulting spreads in the abundances of each element in each cluster are shown in Fig. 12. The overall level of chemical homogeneity in these two clusters found by using the NN predictions is 0.030 ± 0.029 dex, obtained by calculating the mean abundance scatter and the mean uncertainty in the scatter. This is the same level of homogeneity as that reported by using the `Cannon 2` (Ness et al. 2018) and `The Payne` (Ting et al. 2018) by benchmarking on open clusters.

To test the NN 's performance at low metallicity, we use the globular cluster M13, which has many members in the APOGEE catalogue. The spread in the abundances from the NN and from ASPCAP for M13 is displayed in Fig. 13. This figure shows the expected behaviour: the spread in the abundances of heavier elements is small, while that in the abundance of lighter elements is larger, especially for Al. A boutique analysis of the APOGEE spectra for stars in M13 by Mészáros et al. (2015) showed that the spread in Al in M13 is particularly large: ≈0.5 dex. This is similar to the spread in the NN Al abundances in M13, while that in ASPCAP is significantly larger. In Fig. 16 displayed in Section 5.1 below, we show the abundances of both Mg and Al for stars in M13 and these fall roughly along the expected sequence (which in M13 is almost vertical, i.e. there is very little Mg spread in M13; Mészáros et al. 2015).

4.4 Sensitivity analysis

In order to better understand what the NN is doing, we can compute the sensitivity of each label to the input spectrum as this provides a glimpse into how the NN makes its predictions and which regions in wavelength space are crucial for the NN to predict each label. In mathematical terms, every NN maps inputs x to outputs y in a differentiable manner (indeed, this differentiability is crucial in allowing the NN to be optimized by gradient descent). In practice, NN frameworks allow this gradient to be computed analytically by making use of automatic differentiation. This procedure can be applied to compute the derivatives $\frac{\partial \rm Label}{\partial \lambda}$ of each label with respect to every wavelength pixel λ of the input spectrum. This derivative represents the sensitivity of the NN to each pixel for every label. A negative $\frac{\partial \rm Label_i}{\partial \lambda_j}$ indicates that if the flux at the jth wavelength bin λ_j goes up, the value of the ith label decreases and vice versa for a positive value of $\frac{\partial \rm Label_i}{\partial \lambda_j}$.

An example of this type of sensitivity analysis is shown in Fig. 14. This figure displays the derivative $\frac{\partial \rm [Mg/H]}{\partial \lambda_j}$, that is, the sensitivity of the NN for the [Mg/H] abundance. The derivative is averaged over two sets of stars in the high-SNR test set: all metal-poor stars with [Fe/H] < -1.5 and all metal-rich stars with [Fe/H] > 0.4. The green regions in this figure show the ASPCAP windows used to derive the [Mg/H] abundance and the same windows that we use when making the [Mg/H] prediction with the NN. It is clear that for the metal-rich stars that have strong Mg features, the NN mainly pays attention to the regions of the spectrum within the ASPCAP windows and only limited attention to the rest of the spectrum (recall that our NN architecture is such that a limited amount of information about the full spectrum can be used in the [Mg/H] prediction, through the connection between the large NN that predicts the [$T_{\rm eff}$, log g, [Fe/H]] parameters and the mini-network that predicts [Mg/H]). For metal-poor stars, the network still pays much attention to the region of the spectrum within the ASPCAP windows, but it also pays stronger attention to regions outside of the windows (an example is the strong FeI feature in the red part of the spectrum). This is because for metal-poor stars, the spectral Mg features are weaker, and so the NN can improve its predictions by making use of a limited amount of information from the full spectrum. This shows that letting the NN see the whole spectra is essential for it to make sensible predictions in extreme cases.

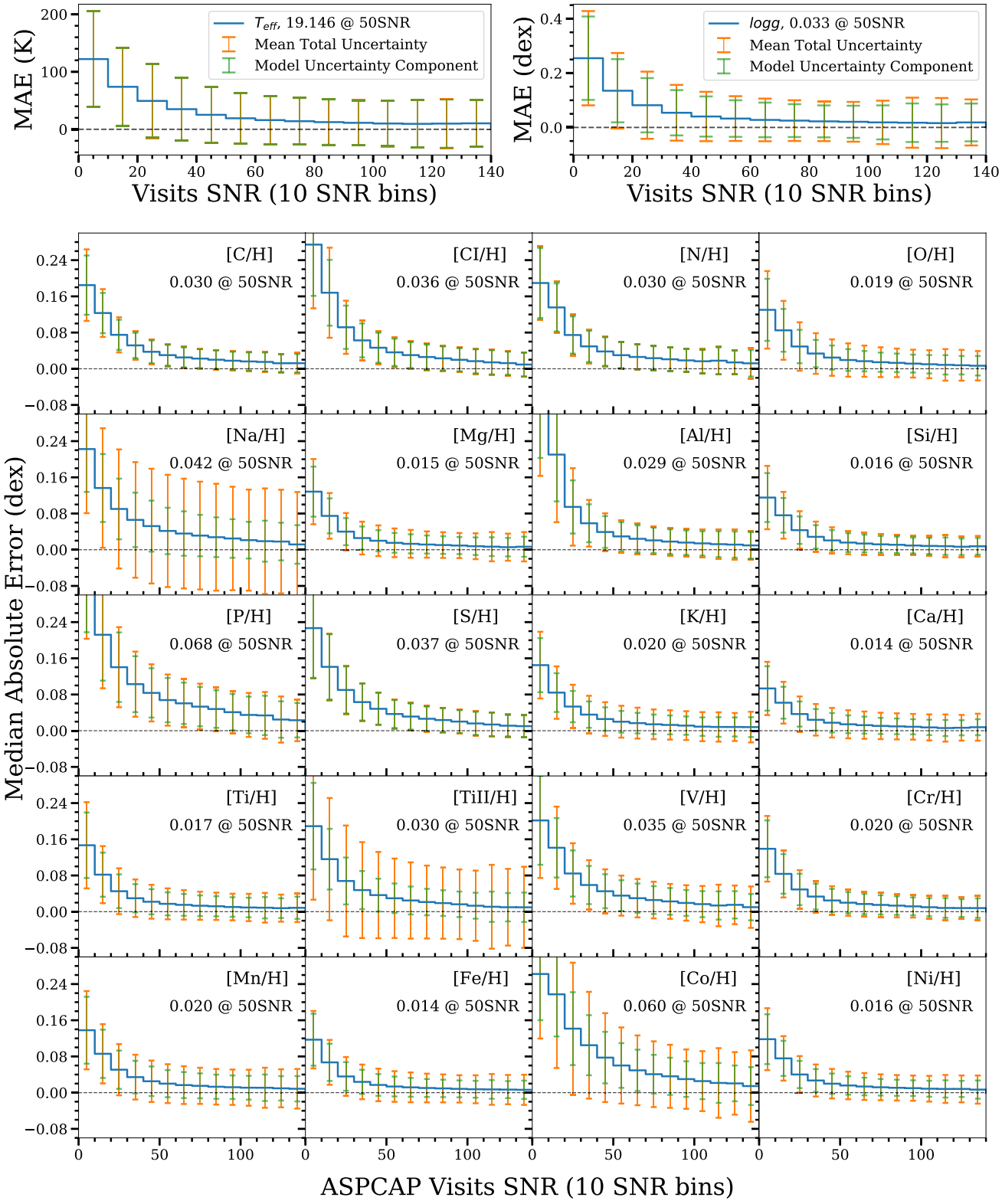

Figure 11. Comparison between NN predictions for $T_{\rm eff}$, $\log g$, and [X/H] from low-SNR, individual-exposure spectra and those from the same NN applied to the combined spectra in high-SNR test set. Curves and error bars are as in Fig. 9. The number beside or below the parameter/abundance label in each plot represents the MAE at SNR = 50. When compared to its own predictions at high SNR, the performance of the NN at low SNR is excellent, with residuals of size 0.01 to 0.02 dex at SNR $\gtrsim$ 100 and residuals that are only slightly bigger even at SNR $\approx$ 50.

The behaviour for other individual elements is similar to that for [Mg/H] shown in Fig. 14. For the $T_{\rm eff}$ and $\log g$ predictions, the NN uses the entire spectral range. For $T_{\rm eff}$ the network mainly gets information from a large number of spectral features. For $\log g$, we

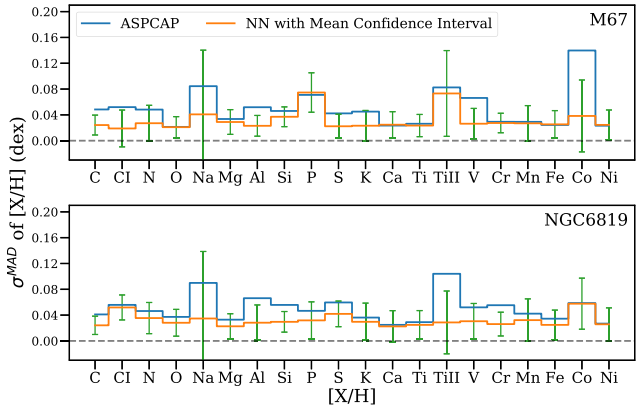

Figure 12. Spread in the predicted NN and ASPCAP labels for stars in the open clusters M67 (14 stars) and NGC6819 (20 stars). The spread in the NN predictions is small for all elements and well matched by the NN uncertainties (the green error bars). The NN has a smaller spread in each element than ASPCAP. The overall level of chemical homogeneity is 0.030 ± 0.029 dex.

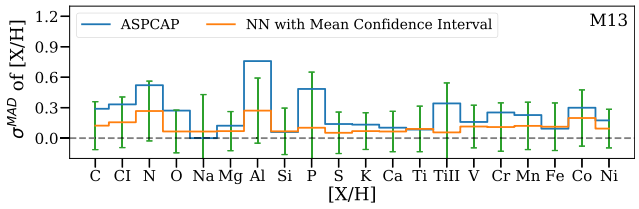

Figure 13. Spread in the predicted NN and ASPCAP labels for stars in the globular cluster M13 (23 stars). As expected, the spread in the abundances of heavier elements is small, while that in lighter elements is larger. This is especially the case for Al, which has a $\approx$0.5 dex spread in M13 (Mészáros et al. 2015) that is well recovered by the NN predictions.

display the derivative $\frac{\partial \log g}{\partial \lambda_j}$ in Fig. 15. While the derivative is non-zero over the full wavelength range, it is especially large near the hydrogen lines (the brackett series). This behaviour makes sense as the strong hydrogen lines are known to be strongly sensitive to $\log g$. Therefore, we see that even when the NN is allowed to use the full spectrum, it uses a physically plausible set of features in the spectrum to predict $\log g$.

5 VARIATIONS

5.1 Training on the full, uncensored spectrum

Before we settled on the `ApogeeBCNNCensored()` NN architecture shown in Fig. 6 that uses censored versions of the spectrum over the full wavelength range when determining the abundances of individual elements, we attempted using a simple multilayered Bayesian convolutional NN with dropout. This network is available as `ApogeeBCNN()` in the `astroNN` python package. Rather than splitting the label determination into a large NN to infer the main stellar parameters [$T_{\rm eff}$, $\log g$, [Fe/H]] and mini-networks to determine individual element abundances, this simple NN is trained on the full spectra to infer all 22 parameters and abundances without any censorship. The performance of this `ApogeeBCNN()` network on both the high SNR and the individual-visits test sets is similar to that of `ApogeeBCNNCensored()` described in Sections 4.1 and 4.2 for all labels. Thus, there is almost no loss in information in using only the censored spectra to determine individual abundances.

However, the `ApogeeBCNN()` network fails to perform well in regions of abundance space that are not well covered by the training set and where the intrinsic abundance trends are different from those for the majority of the sample. This is clearly seen when we apply the `ApogeeBCNN()` network to the M13 globular cluster. As discussed in Section 4.3 above, M13, like many globular clusters, has a wide spread in Al abundances and the Al abundances are anticorrelated with the Mg abundances (although for M13 the actual spread in Mg is very small; Mészáros et al. 2015). In Fig. 16, we show the [Al/H] versus [Mg/H] abundances for stars in M13 for the `ApogeeBCNN()` and `ApogeeBCNNCensored()` networks, as well as those for all stars in the training set. It is clear that for `ApogeeBCNN()`, [Al/H] is very strongly correlated with [Mg/H] in M13. This likely results from the fact that the [Al/H] abundance in M13 is difficult to measure (in large part because for the epoch at which most M13 stars were observed by APOGEE, one of the prominent Al lines in the *H* band overlapped with a strong sky-emission line, rendering it unusable). In the absence of information on [Al/H] from Al lines, `ApogeeBCNN()` falls back on the correlation between [Al/H] and [Mg/H] seen in the training set and therefore, the `ApogeeBCNN()` [Al/H] and [Mg/H] are almost entirely correlated. This happens because `ApogeeBCNN()` does not know which features in the full spectrum belong to [Al/H] and which to [Mg/H].

Because the `ApogeeBCNNCensored()` network uses only regions of the spectrum with Al features to determine [Al/H], it provides [Al/H] measurements that are more in line with the results from Mészáros et al. (2015). As discussed above, the spread in [Al/H] as determined by `ApogeeBCNNCensored()` is about the same as that determined by Mészáros et al. (2015).

Despite the fact that the NN trained on the full spectrum fails in certain regions of parameter space, we do allow a limited amount of information from the full spectrum to be used when determining the individual-element abundances. On the one hand, these are the determinations of the overall stellar parameters [$T_{\rm eff}$, $\log g$, [Fe/H]], but we also include an additional, trainable two-neuron connection in the censored network as shown in Fig. 6. This allows the abundance prediction to depend on the full spectrum in a way that is not pre-determined by us, but is learned from the training set. Such a connection makes physical sense, because the abundance of certain elements has a strong effect on the structure of the stellar photosphere, which in turn affects all parts of the spectrum. This is the case, for example, for carbon and oxygen in the cool stars observed by APOGEE (e.g. Mészáros et al. 2012), but is also the case for other elements that are strong electron donors (e.g. Mg, Si), which therefore may also cause small effects through the spectral region. The two-neuron connection in `ApogeeBCNNCensored()` allows such effects to be determined directly from the training data in a limited manner, without letting abundances of individual elements be determined entirely through correlations with other elements in the training data.

5.2 Training with small data sets

We have trained our NN using high-SNR, high-resolution APOGEE spectra. Such high-quality data are expensive to obtain, because they require a large amount of telescope time. If we want to use a data-driven approach such as the NN trained here to 'transfer' labels from a high-resolution, high-SNR survey (e.g. APOGEE) to a low-resolution, low-SNR survey (e.g. LAMOST, as done by using the `Cannon 2` by Ho et al. (2017), we need to obtain a number of spectra for stars in common between the surveys, which takes away

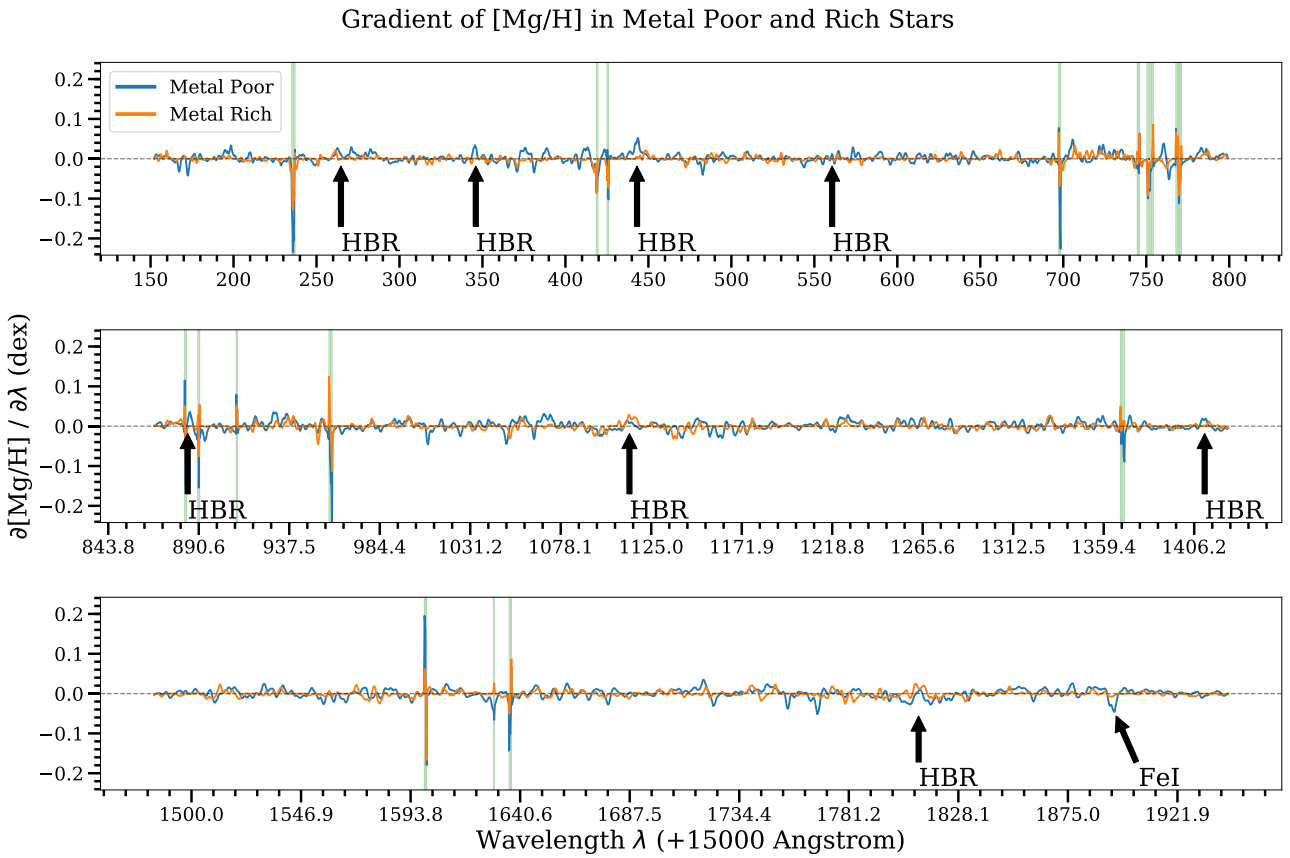

Figure 14. The NN 's [Mg/H] sensitivity, that is mean $\frac{\partial[\mathrm{Mg/H}]}{\partial\lambda_j}$, for metal-poor ([Fe/H] < -1.5; blue curve) and metal-rich stars ([Fe/H] > 0.4; orange curve). The green regions show the ASPCAP [Mg/H] windows. We also label the hydrogen lines and a strong FeI feature. Because of the way the network is structured, information about [Mg/H] in the spectrum is mainly extracted in the ASPCAP windows, but especially for metal-poor stars the network also depends on features outside of the windows.

from the ability to observe new targets. Therefore, we investigate in this section whether we can train the NN with smaller data sets and retain the good performance of the approach.

In traditional machine-learning techniques, the number of parameters is typically kept smaller than the size of the training data, because otherwise the machine-learning method will end up overfitting to the training data and therefore fail to generalize to the test data. However, a modern, medium-sized ANN often has billions of parameters that are optimized with less than a million training data. In our application, the network we use in Section 4 has ≈3.5 million trainable parameters, while the training set has ≈30 000 objects. Therefore, the number of trainable parameters is more than a hundred times the number of training data. However, each spectrum consists of 7514 flux values at different wavelengths, so the total number of training data points is ≈200 million, more than the number of parameters. Therefore, we expect that we may be able to train the network with even fewer training objects.

The key to being able to train a large ANN with smaller training data sets lies in regularization (Zhang et al. 2016). In our case, this regularization is provided by the dropout variational inference method used as a Bayesian approximation. Without this regularization, the network would simply memorize the results for the training set. The dropout procedure reduces the effective capacity of the network, by not allowing this type of memorization to work, and therefore helps the network to generalize to the test data.

We may ask whether we can reduce the number of training data. To do this, we train NN s with exactly the same architecture and parameters as `ApogeeBCNNCensored()` but with training sets that are factors of 2^k smaller, from 2 to 32. The resulting bias and scatter of the residuals for the individual visit test set are displayed in Table 3 for the three main stellar parameters $T_{\rm eff}$, $\log g$, and [Fe/H] (we do not show the results for the individual abundances, but these display similar trends). The bias in $T_{\rm eff}$ stays around 0 K for any size training set, similar to the $T_{\rm eff}$ bias shown in Table 2. Compared to training on the whole training set, the network trained on 4175 spectra (which is 12.5 per cent of the original training set) has larger scatter by 5K, 0.012 and 0.003 dex in $T_{\rm eff}$, $\log g$, and [Fe/H] without introducing too much bias. Since the test set is mostly dominated by solar abundance stars, metrics in Table 3 are also mostly dominated by those stars. In regions of parameter space that are sparsely covered by the original training set, the performance becomes much worse with small training set. With even smaller training sets, the scatter across the whole parameters space as well as NN uncertainty increases significantly. Therefore, what is more important than the overall size of the training set is that it covers a wide range of possible parameter space.

Thus, we can obtain almost the same performance with a network trained on only a few thousand stars. Usage of the NN approach described in this paper for transferring APOGEE labels to surveys like LAMOST therefore look promising.

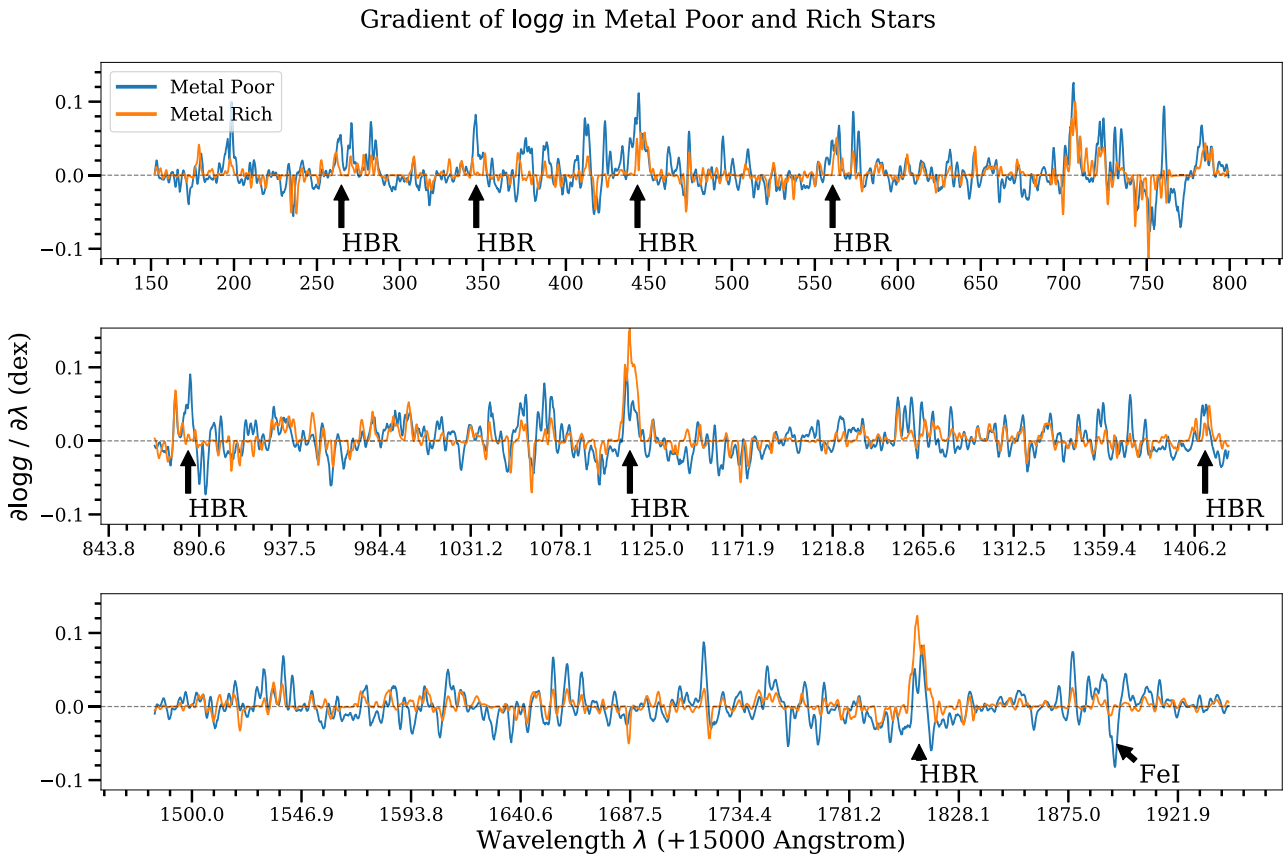

Figure 15. The NN 's $\log g$ sensitivity, that is mean $\frac{\partial \log g}{\partial \lambda_j}$, for metal-poor ([Fe/H] < -1.5; blue curve) and metal-rich stars ([Fe/H] > 0.4; orange curve). We also label the hydrogen lines and a strong FeI feature. Because of the way the network is structured, information about $\log g$ in the spectrum is mainly extracted in the hydrogen lines.

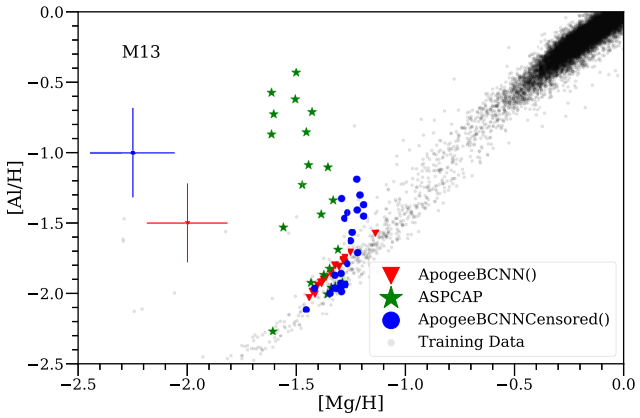

Figure 16. [Al/H] versus [Mg/H] as determined by the `ApogeeBCNNCensored()` NN, which only uses regions of the spectrum containing spectral features for each element to be determined, and by the `ApogeeBCNN()` NN, which uses the full spectrum for each element, for stars in the globular cluster M13. The black points show the distribution of [Al/H] versus [Mg/H] in the training set while the blue and red points with error bars show the median error with `ApogeeBCNNCensored()` and `ApogeeBCNN()`, respectively. Because little information about [Al/H] is available from regions containing Al features for M13 stars, the `ApogeeBCNN()` predictions of [Al/H] follow the correlation with [Mg/H] that is present in the data set. The `ApogeeBCNNCensored()` NN avoids this and displays the correct Al spread in M13 and show a better agreement with the analysis of Mészáros et al. (2013) than ASPCAP does.

Table 3. Training on small data sets: median and $\sigma^{\rm MAD}$ of the $T_{\rm eff}$, $\log g$, and [Fe/H] residuals between the NN prediction and ASPCAP results for the individual test set when the NN is trained with a limited amount of data. Each label column shows median / $\sigma^{\rm MAD}$.

# of objects (7514 pixels each)	$T_{\rm eff}$ (k)	$\log g$ (dex)	[Fe/H] (dex)
33 407	−1 / 24	0.000 / 0.046	− 0.004 / 0.018
16 703	3 / 24	0.010 / 0.055	− 0.005 / 0.019
8351	1 / 27	0.011 / 0.059	− 0.006 / 0.022
4175	1 / 28	0.007 / 0.058	− 0.004 / 0.021
2087	−4 / 35	0.013 / 0.083	− 0.014 / 0.029
1043	9 / 53	0.03 / 0.16	− 0.012 / 0.039

5.3 Importance of continuum normalization

As described in Section 3.2, we have chosen to employ a custom continuum normalization procedure that uses a set of pixels assumed to represent the (pseudo-)continuum on both individual exposures and on the combined spectra, instead of using ASPCAP's pseudo-continuum normalized spectra, which simply fit a polynomial to each detector's spectrum. Both of these procedures are expected to be independent of SNR, but the procedure we use produces continuum-normalized spectra that are more similar to those used in traditional spectroscopic analyses. In this section, we test what happens when we use ASPCAP's procedure instead.

Table 4 shows the result of a `ApogeeBCNNCensored()` NN trained on the same data but with spectra that are continuum nor-

Table 4. Neural-network prediction results on the high-SNR test set from comparing to ASPCAP, when using ASPCAP's procedure for continuum-normalization on the training and testing spectra.

Label	Median of residual	$\sigma^{\rm MAD}$ of residual
$T_{\rm eff}$	−22K	31K
log g	0.010 dex	0.050 dex
[C/H]	− 0.003 dex	0.051 dex
[CI/H]	0.010 dex	0.056 dex
[N/H]	0.008 dex	0.064 dex
[O/H]	− 0.017 dex	0.047 dex
[Na/H]	− 0.01 dex	0.13 dex
[Mg/H]	− 0.000 dex	0.025 dex
[Al/H]	− 0.040 dex	0.070 dex
[Si/H]	− 0.002 dex	0.029 dex
[P/H]	− 0.02 dex	0.11 dex
[S/H]	0.011 dex	0.060 dex
[K/H]	− 0.010 dex	0.046 dex
[Ca/H]	− 0.014 dex	0.030 dex
[Ti/H]	− 0.024 dex	0.050 dex
[TiII/H]	− 0.04 dex	0.11 dex
[V/H]	− 0.007 dex	0.099 dex
[Cr/H]	0.000 dex	0.048 dex
[Mn/H]	− 0.016 dex	0.038 dex
[Fe/H]	− 0.003 dex	0.018 dex
[Co/H]	− 0.02 dex	0.14 dex
[Ni/H]	0.003 dex	0.030 dex

malized using ASPCAP's procedure (we simply use the continuum-normalized spectra provided in the APOGEE data release for this, rather than attempting to reproduce ASPCAP's normalization). We then similarly test on the same testing data as in the high-SNR test set, but with spectra normalized using ASPCAP's procedure. Comparing the results in Table 4 with those in Table 1, we see that the resulting biases are similar, but the scatter in some of the abundance labels (e.g. [C/H] and [N/H]) is considerably larger. This test therefore demonstrates that the continuum normalization that we use does improve the NN 's performance, so there is use in attempting to obtain a (pseudo-)continuum that is close to the true continuum.

6 ABUNDANCE DISTRIBUTIONS IN THE MILKY WAY

To further illustrate the power of the NN approach, we show the [X/Fe] versus [Fe/H] distributions of stars for all elements measured by ASPCAP and compare it to the ASPCAP results for the same stars. We only use a single cut to the full APOGEE DR14 catalogue of spectra: we remove all stars with log g uncertainty larger than 0.2 dex. This cut removes any problematic spectra that result in large label uncertainties (traced by the log g uncertainty) and also cuts out essentially all dwarfs; because the NN log g results for dwarfs are based on extrapolation – the training set contains no log g for dwarfs – the actual NN log g values are very wrong, so it is essential to cut on their uncertainty instead (see Fig. 7). After this cut, we are left with a sample of 157 598 stars.

Fig. 17 displays the abundance distribution for α and odd-Z light elements Na, Al, and K for these stars, both for the NN and for ASPCAP. Fig. 18 shows the distribution of the remaining elements. As above, we show results for CI and TiII separately, as these are measured separately by ASPCAP. It is clear that the NN abundance patterns are tighter than those obtained from ASPCAP. This is especially clear for the α elements (O, Mg, Si, S, Ca, and Ti), which has a distinct bimodal structure at intermediate metallicities ([Fe/H] ≈ -0.2 to -0.5) in *all* α elements. The low-α sequence in all of these distributions is also significantly tighter than in the ASPCAP results. We also see that Al and K essentially behave as α elements, both displaying a clear bimodal structure with the same pattern as the α elements.

7 DISCUSSION

7.1 Performance

Deep learning is a promising tool for the big-data era in astronomy. Besides the better precision and accuracy of the NN compared to other approaches to spectroscopic analysis, the development of modern hardware-accelerated deep-learning technology means that our analysis is also much faster than traditional and other data-driven approaches. The NN described in this work determines 22 labels from ≈100 000 APOGEE spectra consisting of ≈7500 wavelength pixels each, while doing 100 Monte Carlo dropout runs – equivalent to determining 22 parameters from 10000 000 APOGEE spectra without dropout – in ≈300 s or ≈3 ms per star (≈30 μs per star without dropout). This performance is obtained on a Nvidia consumer graphics processor with 4375 GFLOPS at single precision, while the training steps with 40 epochs and a batch size of 64 takes ≈700 s to complete in total.[4] The performance of NN s is expected to get much better with upcoming graphics processors specializing in deep learning, such as Tensor Processing Units (TPUs) in upcoming consumer Nvidia GPUs (Jia et al. 2018).

This fast performance offers great advantages in the era of large spectroscopic surveys. For example, it means that computational needs are far lower when only a small, high-SNR subset of the data (the training set) is analysed with traditional, slow tools, while the majority of the data set can be processed much faster with the NN approach. Currently, the APOGEE ASPCAP pipeline requires a large cluster both to produce the library of synthetic spectra used in the ASPCAP fitting and for performing the fits themselves. Our fast framework allows for a much faster development cycle. This is the case in the narrow sense of providing the opportunity for fast prototyping and exploration of new NN model architectures. But when coupled with development of the tools to produce the small training set, this is also the case in the broader sense of seeing how changes to the input physics used in producing the training set affect the larger test sample.

Our extremely fast analysis coupled with the fact that the dropout procedure produces realistic uncertainties also opens up the possibility of real-time analysis of whether high enough SNR is obtained for a given level of abundance uncertainties. Stellar spectra could be analysed on-the-fly as they are being collected and integration stopped when an acceptable uncertainty in the abundances is reached. With future, large fibre-positioner systems this could be used to run efficient, large spectroscopic surveys. Because our approach automatically assigns large uncertainties for objects far outside the training-set boundaries, such objects, which are likely to be of interest, would automatically obtain high SNR spectra in this approach.

[4] *Batch size* refers to the number of training examples in one forward/backward pass and *epoch* refers one forward pass and one backward pass of all the training examples. For example, with 6400 training data and a batch size of 64 with 5 epochs, each epoch will perform 100 gradient updates, each gradient is an average of 64 training data without replacement, and do it 5 times.

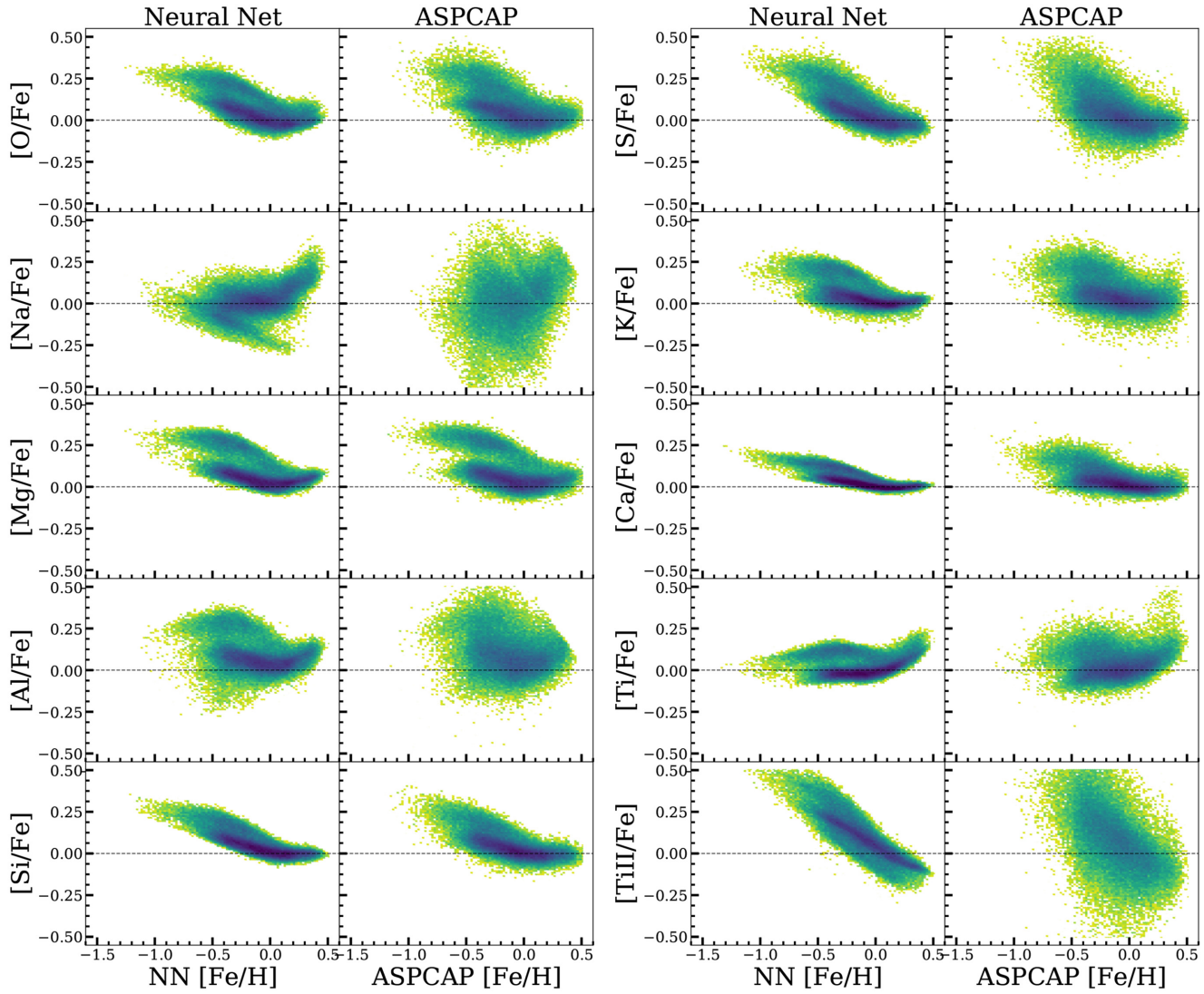

Figure 17. Neural network and ASPCAP predictions on 157 598 APOGEE DR14 stars after a cut on NN log g uncertainty < 0.2 dex as a means to filter out (a) main-sequence stars (as discussed in the right-hand panel of Fig. 7) and (b) problematic spectra that result in a high uncertainty in labels including log g. This figure shows all α and odd-Z light elements Na, Al, and K among our 20 labels prediction. For most abundances, the NN abundances display less scatter and a clearer high/low alpha sequence than ASPCAP.

7.2 Comparison to other data-driven approaches to spectral analysis

Another data-driven approach to high-resolution analysis is provided by the `Cannon` (Casey et al. 2016), which we have already discussed in the text above. The `Cannon`'s approach is fundamentally different from ours, in that the `Cannon` builds a data-driven model of the spectra as a function of stellar labels that is then used to fit labels to observed spectra, while our approach directly determines the mapping from spectra to labels. We therefore make slightly different assumptions about the relation between spectra and labels. To be clear, we list our methods and assumptions (or lack thereof) and how they compare to those made by the `Cannon`:

(i) (a) NNs in this work map spectra directly to labels in a single step (and a sequence of these single steps to determine the uncertainty using dropout). The `Cannon 2` is a generative model that generates realistic spectra from labels and then matches spectra by χ^2 minimization to find abundances.

(ii) We assume that the value of the labels is a continuous, smooth function of the flux. Thus, we assume that similar spectra have similar labels. This assumption is shared by the `Cannon 2` and essentially by all approaches to spectroscopic fitting.

(iii) Despite that, we do not require that spectra with similar labels are similar. This limits our ability to generate a new spectrum for certain set of labels and we cannot generate a set of Δflux for a given spectrum that changes only a single label, while keeping the others constant. The `Cannon 2` assumes that spectra with similar labels are similar.

(iv) The flexibility of the NNs mean that it only performs well on what it is trained and the ability to extrapolate is limited (but note that when extrapolation occurs, the returned uncertainties are very large). This is similar to the `Cannon 2`, although their simpler model may approximate physical models better and thus have better performance when extrapolating.

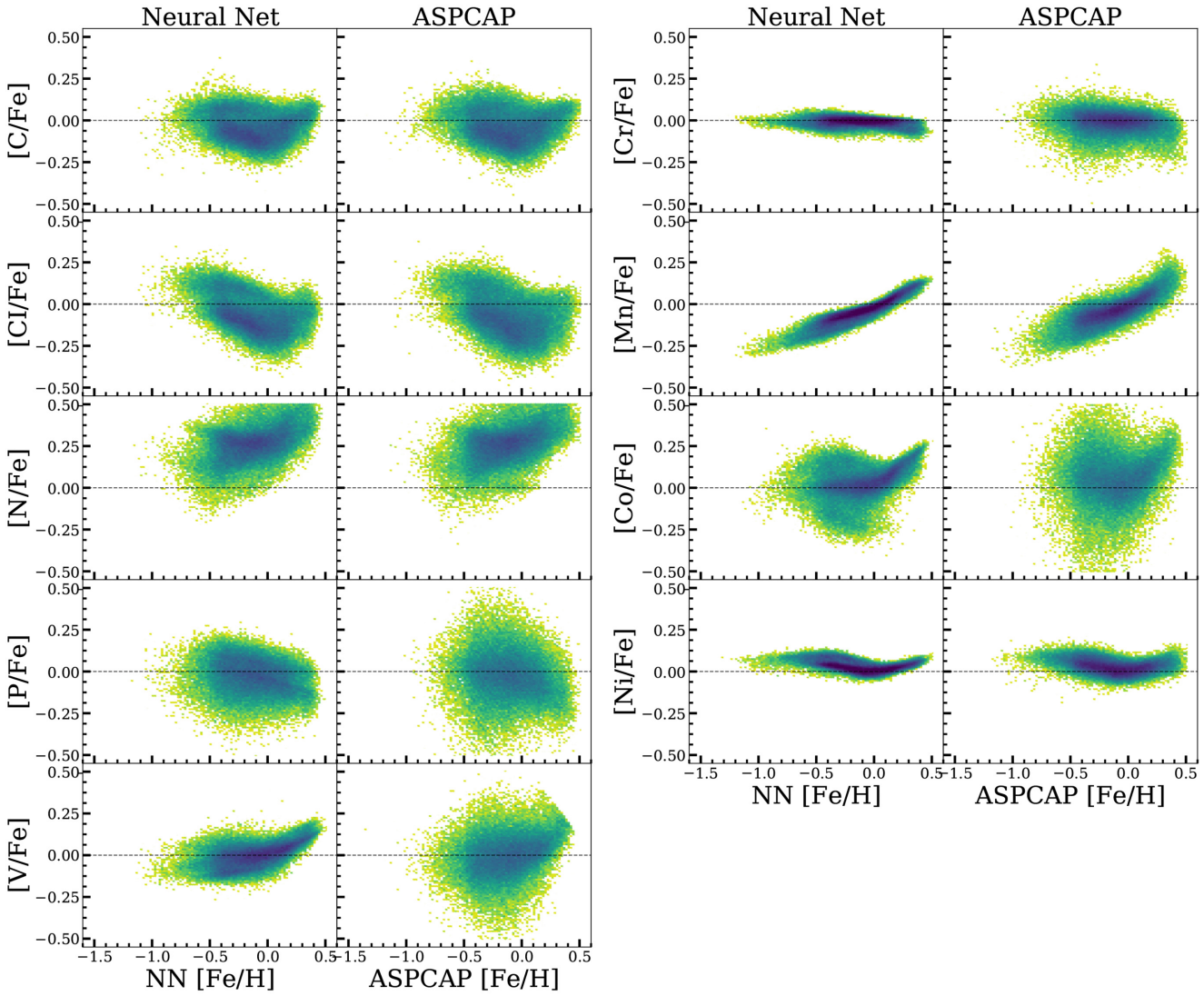

Figure 18. Like Fig. 17, but for the remaining elements. For abundances like [Cr/Fe] and [Ni/Fe], the NN abundances show much tighter scatter at [X/Fe] = 0 as expected.

Our NN approach has some disadvantages with respect to forward-modelling approaches like the `Cannon`:

Interpretability: Because the `Cannon` builds a forward model of the spectra it allows one to generate spectra from the model for a given set of labels. This can be used to inspect the internal functioning of the model and one can ask questions such as: does the behaviour when changing [Al/H] while keeping all other parameters fixed make sense? Because of the reasons given above, our NN cannot answer such questions and therefore does not allow inspection of the model in this sense. It is possible to compute derivatives of each label value with respect to the input spectra (e.g. see Figs 14 and 15), but as these derivatives do not keep other labels fixed, they are less meaningful.

Handling of uncertainties in the input: Our approach ignores uncertainties in the spectra. An approach that forward models the spectra can fit in both the training and test step while taking the flux uncertainties into account. Nevertheless, we have shown that even though we train on high SNR spectra, the NN performs well even at very low SNR. Part of the reason for this is that the flux uncertainties are relatively constant with wavelength. This is not the case for the label uncertainties in the training set (see above) and because it is far easier to only take one of input or output uncertainties into account in any machine-learning technique, it appears more important for the analysis of stellar spectra to take the label uncertainty into account.

On the other hand, the advantages of the NN approach over an approach like the `Cannon` are manifold:

Speed: Because our method learns a mapping from spectra to labels and does not require any fitting when making predictions, it is much faster. As discussed above, we can determine labels for ≈100 000 APOGEE stars in about 3 s on a single ≈$500 GPU (and in 5 min if we also want the uncertainty), while running the `Cannon 2` on the same number of spectra takes about 20 min on a small cluster (Casey et al. 2016).

Realistic uncertainties: The `Cannon 2` can obtain uncertainty estimates from its χ^2 fitting to the spectra and their uncertainties, but these uncertainties are typically much smaller than the scatter in the results obtained from cross-validation, demonstrating that the uncertainties are underpredicted. Our approach returns uncertainties that, at least within the well-populated regions of the training set,

are approximately equal to the scatter from cross-validation or equal to the scatter in open clusters. Thus, our uncertainties can be used in practice to determine whether observed scatter is real or due to noise in the data.

Extrapolation warnings: Data-driven approaches only perform well within the training set and do not perform well when extrapolating, especially approaches as flexible as our NN approach. Data-driven approaches therefore typically need a way to determine first whether or not a test object is within the bounds of the training set. For high-dimensional input and label spaces this boundary can be a complicated, high-dimensional surface that is difficult to determine. In tests, this issue is typically ignored, as most test sets are chosen to represent the training set (e.g. in the standard random partition of the full data set into a training and test set). As we demonstrated above, the uncertainties returned by our NN are very large whenever a spectrum or label outside of the training set is analysed (e.g. the main-sequence stars' $\log g$ in Fig. 7). These large uncertainties can therefore be used as a warning flag for spectra or labels outside of the training set and our method thus provides an automatic check for the training-set boundaries.

Training on noisy and incomplete data: Because we take the uncertainties on the training labels into account and can also use training stars for which not all labels are measured, we can train on an imperfect training set. This is important, because for various reasons many stars in the training set, even at high SNR, do not have complete and high-precision set of labels. The `Cannon 2` in its current implementation does not take uncertainties in the training labels into account and can only train on training objects with a complete set of labels.

The advantage of all data-driven approaches is that they allow properties such as stellar mass, age, or luminosity to be determined directly from stellar spectra. This is difficult to do with theoretical stellar models, because how these properties affect the spectra is not well known. For example, a common way to infer luminosity is by using isochrones based on stellar evolutionary models and stellar parameters and abundances measured from stellar spectra (Santiago et al. 2016), but stellar isochrones are not well calibrated for all types of stars and stages of stellar evolution. Spectra may contain direct indicators of mass (e.g. Ness et al. 2016), age, or luminosity that can be extracted by training on data sets for which these are known quantities (e.g. masses from asteroseismology, e.g. APOKASC: Pinsonneault et al. 2014; or luminosities from *Gaia*).

7.3 Comparison to neural-network approaches to spectral analysis based on synthetic spectra

The training set potentially does not represent all stars in the test set and, as discussed above, NNs trained on an unrepresentative training set will perform poorly for stars outside of the training set. Moreover, abundances determined by ASPCAP contain systematic errors that propagate to the NN during training and we cannot easily quantify these errors or correct them. These systematic errors result from the quality of ASPCAP synthetic grid and errors introduced by the data reduction and calibration steps in ASPCAP. One solution is using the techniques used by `StarNet` or `The Payne` (Ting et al. 2018) by training on theoretical synthetic spectra, because theoretical spectra with different combinations of abundances can be generated and we can choose these combinations to represent the likely set of abundances in the test set. In terms of performance, `StarNet` uses NN s similar to those used here and, when implemented on a GPU, could be as fast to train and evaluate as our method. `The Payne`, however, exhibits far slower performance because it is a forward model of the spectra given the labels similar to the `Cannon`, but with higher complexity. Training `The Payne` takes about 5 CPU min per wavelength pixel or 26 d for all pixels, compared to our 10 min. Evaluating for a test spectrum takes about 1 CPU s per spectrum or about 28 h for 100 000 APOGEE stars, versus our 3 s for 100 000 APOGEE stars (300 s including uncertainties).

Training on synthetic spectra can lead to bad performance when the theoretical spectra do not match the observed spectra well. This can be due to calibration, uncertainties, continuum-normalization issues, or unmodelled physics in the spectral synthesis. For example, without correcting systematic differences from the observed spectra before training, `StarNet` reports that their NN trained on synthetic spectra does not perform well on observed spectra due to differences in the feature distributions between theoretical and real spectra. Of course, because data-driven approaches rely on a training set that is typically analysed using synthetic spectra, they suffer from limitations in the input physics as well. However, through clever use of star clusters, binaries, and similar systems it may be possible to create training sets that are less sensitive to the theoretical modelling of stellar spectra. This is because if we can assume that these systems are chemically homogeneous, we can transfer the chemical labels from well-understood stellar types onto those of poorly understood stellar types, thus creating an empirical training set for the poorly understood types.

Another goal described in `Cannon 2` and the version of `StarNet` that trained on real world spectra is to identify potential unknown abundance lines. The use of a censored NN in this work will render this practically impossible, because we limited the attention of the network to known spectral lines. However, even for a network trained on the full spectrum or on synthetic spectra, it will be questionable whether a previously unknown spectral feature that is identified really belongs to a certain element.

8 CONCLUSIONS

Large spectroscopic surveys are now routinely obtaining high-resolution spectra for hundreds of thousands of stars and upcoming surveys such as WEAVE (Dalton et al. 2014), SDSS-V (Kollmeier et al. 2017), 4MOST (de Jong et al. 2012), and MSE (McConnachie et al. 2016) will soon provide such spectra for millions of stars. Traditionally, such spectra are analysed one-by-one with procedures that are largely manual, which is clearly impractical for large data sets. Fitting spectra with synthetic libraries (e.g. ASPCAP) is the currently favoured approach, but this method is slow and current implementations do not deal well with low SNR spectra. As an alternative approach, we have presented NN models for spectroscopic analysis to infer stellar parameters and chemical abundances with associated uncertainty using Bayesian ANNs with dropout variational inference. We implemented this method in a general, open-source `PYTHON` framework for ANNs called `astroNN` (see the Appendix). We also release a catalogue of abundances for the APOGEE DR14 data set determined by our `ApogeeBCNNCensored()` network (see link in the Introduction).

Our NN method has various special ingredients beyond what would be used in a standard deep learning application. We (i) present a robust objective function for the NN to learn from incomplete data while taking uncertainty in the training labels into account, (ii) use a Bayesian NN with dropout variational inference with this objective function to estimate the uncertainties on the labels determined by the NN , and (iii) combine a large NN to obtain the overall stellar parameters [$T_{\rm eff}$, $\log g$, [Fe/H]] with mini-networks

to determine individual elemental abundances that use versions of the full spectrum censored to only include regions with spectral lines for a given element. With this approach, we simultaneously determine 22 stellar and elemental abundance labels accurately and precisely for both high- and low-SNR spectra. We implemented the method on a GPU using standard tools, which allows speed-ups of more than an order of magnitude and we make these tools easily accessible (see Appendix A3). Our method is extremely fast, allowing stellar parameters and abundances for the entire APOGEE data base to be determined in about 10 min on a single GPU.

We performed detailed tests of our method by comparing to results from the standard APOGEE ASPCAP pipeline and by comparing results on high SNR, combined spectra to those from low SNR, individual exposures. At high SNR, we obtain abundances precise to ≈0.01 to 0.02 dex and even at low SNR (SNR ≈ 50) we get precisions of ≈0.02 to 0.03 dex for most elements. These precisions are confirmed by looking at the scatter in the abundances within open clusters, which we find to be 0.03 ± 0.03 dex. We also recovered the expected abundance trends in globular clusters, but found that the censoring in the network is crucial to obtain this, because training on the full spectrum for all elements causes the NN to depend too strongly on correlations between elements within the training set and therefore to fail when these correlations are absent, like in globular clusters. We also demonstrated that a large NN can work well with a limited amount of training data, finding barely degraded performance for training sets that only consists of thousands of spectra.

The speed and flexibility of NN s mean that they are a highly useful tool for spectroscopic data analysis. They allow the results obtained from a detailed analysis of a small calibration set to be transferred to an entire large data set of millions of spectra in a matter of minutes and can thus be a great aid in the development of the next generation of spectroscopy tools. They could also be of use in earlier stages of the data processing, for example, for homogenizing spectra taken with different instruments (e.g. the northern and southern APOGEE spectrographs) or for removing instrumental systematics such as persistence (e.g. Jahandar et al. 2017). Such applications would be easy to pursue with the `astroNN` software package described in the Appendix.

ACKNOWLEDGEMENTS

It is our pleasure to thank Kim Venn and other members of the `StarNet` group for valuable feedback and for releasing their code publicly. We also thank Natalie Price-Jones for help with the APOGEE data. HL and JB received support from the Natural Sciences and Engineering Research Council of Canada (NSERC; funding reference number RGPIN-2015-05235) and from an Ontario Early Researcher Award (ER16-12-061). JB also received partial support from an Alfred P. Sloan Fellowship.

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.

REFERENCES

Abadi M. et al., 2016, preprint (arXiv:1603.04467)
Abolfathi B. et al., 2018, ApJS, 235, 42
Aihara H. et al., 2011, ApJS, 193, 29
Bailer-Jones C. A. L., 2000, A&A, 357, 197
Bailer-Jones C. A. L., Irwin M., Gilmore G., von Hippel T., 1997, MNRAS, 292, 157
Bovy J., 2016, ApJ, 817, 49
Bressan A., Marigo P., Girardi L., Salasnich B., Dal Cero C., Rubele S., Nanni A., 2012, MNRAS, 427, 127
Casey A. R., Hogg D. W., Ness M., Rix H.-W., Ho A. Q. Y., Gilmore G., 2016 preprint(arXiv:1603.03040)
Chetlur S. et al., 2014, preprint (arXiv:1410.0759)
Chollet F. et al., 2015, Keras Github
Dafonte C., Fustes D., Manteiga M., Garabato D., Álvarez M. A., Ulla A., Allende Prieto C., 2016, A&A, 594, A68
Dalton G. et al., 2014, in Ramsay S. K., McLean I. S., Takami H., eds, Proc. SPIE Conf. Ser. Vol. 9147, Ground-based and Airborne Instrumentation for Astronomy V. SPIE, Bellingham, p. 91470L
de Jong R. S. et al., 2012, in McLean I. S., Ramsay S. K., Takami H., eds, Proc. SPIE Conf. Ser. Vol. 8446, Ground-based and Airborne Instrumentation for Astronomy IV. SPIE, Bellingham, p. 84460T
De Silva G. M., Sneden C., Paulson D. B., Asplund M., Bland-Hawthorn J., Bessell M. S., Freeman K. C., 2006, AJ, 131, 455
De Silva G. M., Freeman K. C., Asplund M., Bland- Hawthorn J., Bessell M. S., Collet R., 2007, AJ, 133, 1161
Fabbro S., Venn K. A., O'Briain T., Bialek S., Kielty C. L., Jahandar F., Monty S., 2018, MNRAS, 475, 2978
Gaia Collaboration 2016, A&A, 595, A1
Gal Y., Ghahramani Z., 2015, preprint (arXiv:1506.02142)
García Pérez A. E. et al., 2016, AJ, 151, 144
Gray D. F., 2005, The Observation and Analysis of Stellar Photospheres. Cambridge Univ. Press, Cambridge
Gunn J. E. et al., 2006, AJ, 131, 2332
Hinton G. E., Srivastava N., Krizhevsky A., Sutskever I., Salakhutdinov R. R., 2012, preprint(arXiv:1207.0580)
Holtzman J. A. et al., 2015, AJ, 150, 148
Holtzman J. A. et al., 2018, AJ, 156, 125
Ho A. Y. Q. et al., 2017, ApJ, 836, 5
Jahandar F. et al., 2017, MNRAS, 470, 4782
Jia Z., Maggioni M., Staiger B., Scarpazza D. P., 2018 preprint(arXiv:1804.06826)
Jönsson H. et al., 2018, AJ, 156, 126
Kendall A., Gal Y., 2017, preprint (arXiv:1703.04977)
Kingma D. P., Ba J., 2014, preprint (arXiv:1412.6980)
Kollmeier J. A. et al., 2017, preprint (arXiv:1711.03234)
Laureijs R. et al., 2011, preprint(arXiv:1110.3193)
Law N. M. et al., 2009, PASP, 121, 1395
Lee Y. S. et al., 2008, AJ, 136, 2022
Lintott C. J. et al., 2008, MNRAS, 389, 1179
LSST Science Collaboration et al., 2009, preprint(arXiv:0912.0201)
Majewski S. R. et al., 2017, AJ, 154, 94
McConnachie A. et al., 2016, preprint(arXiv:1606.00043)
Mészáros S. et al., 2013, AJ, 146, 133
Mészáros S. et al., 2012, AJ, 144, 120
Mészáros S. et al., 2015, AJ, 149, 153
Ness M. et al., 2018, ApJ, 853, 198
Ness M., Hogg D. W., Rix H. W., Martig M., Pinsonneault M. H., Ho A. Y. Q., 2016, ApJ, 823, 114
Perreault Levasseur L., Hezaveh Y. D., Wechsler R. H., 2017, ApJ, 850, L7
Pinsonneault M. H. et al., 2014, ApJS, 215, 19
Price-Jones N., Bovy J., 2018, MNRAS, 475, 1410
Re Fiorentin P., Bailer-Jones C. A. L., Lee Y. S., Beers T. C., Sivarani T., Wilhelm R., Allende Prieto C., Norris J. E., 2007, A&A, 467, 1373
Reis I., Poznanski D., Baron D., Zasowski G., Shahaf S., 2018, MNRAS, 476, 2117
Rumelhart D. E., Hinton G. E., Williams R. J., 1986, Nature, 323, 533
Santiago B. X. et al., 2016, A&A, 585, A42
Shetrone M. et al., 2015, ApJS, 221, 24
Stoica I. et al., 2017, preprint(arXiv:1712.05855)
The Astropy Collaboration et al., 2018, preprint (arXiv:1801.02634)

Ting Y.-S., Conroy C., Rix H.-W., Cargile P., 2018 preprint(arXiv:1804.01530)
Wilson, J. C. et al., 2010, in McLean I. S., Ramsay S. K., Takami H., eds, Proc. SPIE Conf. Ser. Vol. 7735, Ground-based and Airborne Instrumentation for Astronomy III. SPIE, Bellingham, p. 46
Yang T., Li X., 2015, MNRAS, 452, 158
Yanny B. et al., 2009, AJ, 137, 4377
Zhang C., Bengio S., Hardt M., Recht B., Vinyals O., 2016, preprint (arXiv:1611.03530)

APPENDIX: ASTRONNN: A PYTHON LIBRARY FOR DEEP LEARNING IN ASTRONOMY

Data-driven tools are becoming more and more popular in astronomy and deep learning in particular is becoming a popular technique to deal with big data sets. When we started on the research for this paper, we realized that a PYTHON package that (i) has a focus on deep learning, (ii) contains tools and data sets that are relevant to astronomers, (iii) is easy to use and well tested against errors, and (iv) acts as a platform to share astronomy-oriented NN s is needed to advance research in this area.

`astroNN` is a PYTHON package for deep learning in astronomy designed to fulfill the points stated above. `astroNN` relies heavily on `Tensorflow` (Abadi et al. 2016), a standard deep learning PYTHON library, and is easy to set up on all common platforms. `astroNN` employs custom methods for storing and sharing models: models are saved in a folder and consist of the NN itself in `HDF5` format, the training history in `CSV` format, model parameters, and the names of output neurons as well as normalization parameters in `JSON` format. Users simply have to give the reduced data as `numpy` array(s) to `astroNN` and get outputs back as `numpy` array(s); normalization of the inputs and outputs is handled internally.

At this point, `astroNN` as used in this paper consists of about 8300 lines of code in the modules, 1200 line of test code, and 3200 lines of documentation. The test suite covers >90 per cent of the NN -related components of the code. Version control, continuous integration, test statistics, and documentation hosting is provided by `GitHub`, `Travis-CI`, `Coveralls`, and `Read The Docs`, respectively. `astroNN`is available at https://github.com/henrysky/astroNN with extensive documentation at http://astroNN.readthedocs.io.

A1 Modules

The main structure of `astroNN` is as follows:

(i) Three data processing modules that provide basic data reduction and processing for different surveys. `Astropy` (The Astropy Collaboration 2018) is used in these modules to provide unit conversions and `FITS` file reading functionality.

(1) `astroNN.apogee` – Provides basic functionality to read APOGEE DR13/14 data sets and processing of the spectra.
(2) `astroNN.gaia` – Provides basic functionality to read *Gaia* DR1/2 data sets and astrometry-related conversion tools.
(3) `astroNN.lamost` – Provides basic functionality to read LAMOST data sets and spectra processing.

(ii) Two NN related modules that provide different model architectures and infrastructure. All functions are compatible with `Tensorflow` or with `keras` with the `Tensorflow` backend (Chollett et al 2015).

(1) `astroNN.models` – Neural-network architectures and NN classes are defined in this module. Currently, this includes a convolutional NN , a Bayesian NN with dropout variational inference, as well as a variational auto-encoder for unsupervised spectra analysis. Note that the latter is still under development.
(2) `astroNN.nn` – Deep-learning infrastructure such as robust objective functions, objective functions to deal with incomplete data, and customized layers such as Monte Carlo Dropout layer, Gradient Stopping layer, or Boolean Masking layer that are not available in standard tools.

(iii) One data sets module that provides functionality to deal with multiple astronomical data sets.

(1) `astroNN.datasets` – Besides multiple astronomical data sets like APOKASC (APOGEE-Kepler asteroseismology catalogue), this contains 'Galaxy10' which is an alternative to the standard MNIST or Cifar10 NN example training sets that is designed by us using data from Galaxy Zoo (Lintott et al. 2008) and SDSS (Aihara et al. 2011). The data set consists of coloured galaxy images and their morphology and it can be used to introduce astronomy researchers to deep-learning tools. The data set is available at http://astro.utoronto.ca/~bovy/Galaxy10/Galaxy10.h5 with documentation at https://astronn.readthedocs.io/en/v1.0.0/galaxy10.html

A2 Example of using Neural Net to infer parameters and abundances on arbitrary APOGEE spectra

Besides providing general tools for deep learning in astronomy, we also share the actual networks trained and discussed in this paper in a separate `GitHub` repository associated with this paper. Here we give an example of how to use the main `ApogeeBCNNCensored()` network for determining stellar parameters and abundances for a given APOGEE spectrum. First follow the following instructions:

(i) Install `astroNN` by following instructions from https://astronn.readthedocs.io/en/v1.0.0/quick_start.html
(ii) Obtain the repository containing the code to reproduce all figures in this paper at https://github.com/henrysky/astroNN_spectra_paper_figures
(iii) Open a PYTHON terminal under the repository folder but outside the folder `astroNN_0617_run001`

Then copy and paste the following code to do inference with the neural net in this paper on 2M19060637 + 4717296, which is the spectrum shown in Fig. 5

A3 Fast Monte Carlo Inference on GPU

For a probabilistic (i.e. a neural network that has different outputs every time you do inference on the same data) `keras` or `tensorflow.keras` neural network model which has a single input, as opposed to some keras multi-input model (in such case you need to concatenate multiple inputs and unpack the inputs in the model), to be inferred and a single output array which is the concatenation of the prediction and the predictive variance, you can wrap the model using `astroNN`'s `FastMCInference()` to get a new model, which gives you the mean prediction, predictive uncertainty, and model uncertainty with great performance on GPU. An example of this is in the following pseudo-code snippet

```
from astropy.io import fits
from astroNN.apogee import visit_spectra, apogee_continuum
from astroNN.models import load_folder

# the same spectrum as used in figure 5
opened_fits = fits.open(visit_spectra(dr=14, apogee='2M19060637+4717296'))
spectrum = opened_fits[1].data
spectrum_err = opened_fits[2].data
spectrum_bitmask = opened_fits[3].data

# using default continuum and bitmask values to continuum normalize
norm_spec, norm_spec_err = apogee_continuum(spectrum, spectrum_err,
                                            bitmask=spectrum_bitmask, dr=14)

# load neural net
neuralnet = load_folder('astroNN_0617_run001')

# inference, if there are multiple visits, then you should use the globally
# weighted combined spectra (i.e. the second row)
pred, pred_err = neuralnet.test(norm_spec)

print(neuralnet.targetname)  # output neurons representation
print(pred)  # prediction
print(pred_err['total'])  # prediction uncertainty
```

Listing 1: Example of using Neural Net to infer parameters and abundances on APOGEE spectra

```
from astroNN.nn.layers import FastMCInference

# keras_model is your model with 1 output which is a concatenation of all
# label predictions and predictive variance
keras_model = Model(....)

# fast_mc_model is the new keras model capable of fast Monte-Carlo integration on GPU
# n=number of Monte-Carlo run
fast_mc_model = FastMCInference(keras_model, n=100)

# You can just use keras API with the new model such as
result = fast_mc_model.predict(.....)

# here is the result dimension
predictions = result[:, :(result.shape[1] // 2), 0]  # mean prediction
# model uncertainty
mc_dropout_uncertainty = result[:, :(result.shape[1] // 2), 1]
# predictive uncertainty
predictions_var = result[:, (result.shape[1] // 2):, 0]
```

Listing 2: Fast Monte Carlo Inference on GPU

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