Bayesian uncertainty calculation in neural network inference of ion and electron temperature profiles at W7-X[a)]

A. Pavone,[1] J. Svensson,[1] A. Langenberg,[1] N. Pablant,[2] U. Hoefel,[1] S. Kwak,[1] R.C. Wolf,[1] and the Wendelstein 7-X Team[1, b)]

[1)] *Max-Planck-Institute for Plasma Physics, Greifswald 17491, Germany*

[2)] *Princenton Plasma Physics Lab, Princeton New Jersey, USA*

(Dated: 14 July 2018)

We make use of a Bayesian description of the neural network (NN) training for the calculation of the uncertainties in the NN prediction. Having uncertainties on the NN prediction allows to have a quantitative measure for trusting the NN outcome and comparing it with other methods. Within the Bayesian framework, the uncertainties can be calculated under different approximations. The NN has been trained with the purpose of inferring ion and electron temperature profile from measurements of a X-ray imaging diagnostic at W7-X. The NN has been trained in such a way that it constitutes an approximation of a full Bayesian model of the diagnostic, implemented within the Minerva framework. The network has been evaluated on measured data and the uncertainties calculated under different approximations have been compared with each other, finding that neglecting the noise on the NN input can lead to an underestimation of the error bar magnitude in the range of 10% to 30%.

I. INTRODUCTION

In nuclear fusion research, neural networks (NNs) have been used for tasks such as prediction of disruption events from plasma parameters[1] and for diagnostic data analysis[2]. A special effort is often put in the development of real time systems[3]. In most of the applications, the output of the NN models are single 'best guess' predictions, obtained with values of the adaptable parameters found minimizing a given cost function. We believe that, in order to have trust-worthy outcomes, uncertainty should be taken into account and delivered as part of the final predictions. In this paper we will describe and make use of a Bayesian framework for the treatment of uncertainties, where the neural network model is seen as a Bayesian model and the training procedure is seen as an inference problem[4,5]. Applications of such framework are scarcely encountered, although it posits a principled picture of neural network modelling. Its implementation relies on the calculation of the second derivative of the neural network's cost function with respect to the network weights, i.e. the Hessian matrix. This is an operation that scales with the square of the number of weights, i.e. as $O(W^2)$, where W is the number of weights. It is therefore a computational expensive calculation. However, the Hessian matrix needs to be calculated only once per training, as it is fixed at evaluation time, when the network is evaluated on the measurements. In Section II we will illustrate the salient points of the Bayesian NN training from a theoretical point of view, describing three different procedures for the calculation of the uncertainties: the first one is derived neglecting noise in the NN input, the second one accounting for it, and the third one making use of a sampling scheme based on a non linear multi-Gaussian approximation. In Section III we will describe the specific application of the method to X-ray imaging crystal spectrometer (XICS) diagnostic data at W7-X, where the NN has been trained for the inference of electron and ion temperature profiles from XICS measurements. In Section IV we will compare the two procedures where we either do take or do not take into account the noise on the NN input, and we will show a single illustrative example of uncertainty calculation with the multi-Gaussian sampling procedure.

II. BAYESIAN NEURAL NETWORKS

We shall describe now the salient points of the Bayesian perspective on NN training which will allow us to calculate uncertainties in the prediction. The notation used here is mostly taken from[5]. The neural network is conceived here as a function f, which maps a generally multidimensional input vector $\mathbf{x}$ to a generally multidimensional output vector $\mathbf{y}$. The function f is also parametrized with a set of free parameters or weights $\mathbf{w}$, whose values are adapted or learned during the training procedure, so that it can be written that $\mathbf{y} = f(\mathbf{x}, \mathbf{w})$. In the specific case of this study, the input vector $\mathbf{x}$ would be an XICS measurement, and $\mathbf{y}$ would be either an electron or ion temperature profile, T_e or T_i respectively. In the analytical treatment that follows, we shall assume a one dimensional output y for the sake of clearer notation. The generalization to multi-dimension output is straightforward. According to the traditional view, the NN training is the procedure employed to find a set of weight values $\mathbf{w}_{\mathrm{MP}}$ that minimizes a given *cost* or *loss* function $L(\mathbf{w})$. In regression problems such function is often chosen to be the sum-of-square error between the

[a)] Published as part of the Proceedings of the 22nd Topical Conference on High-Temperature Plasma Diagnostics, San Diego, California, April, 2018.

[b)] T.S. Pedersen *et al.*, *Nature Communications* **7** (2016)

NN's output y and the target training data t:

$$L(\mathbf{w}) = \sum_{n=1}^{\mathrm{N}} (y^n - t^n)^2 + \nu(\mathbf{w}) \tag{1}$$

where N is the number of training samples, n is an index labelling the n^{th} training sample, and $\nu(\mathbf{w})$ is a *regularizing term* which constrains the weight values to small values. The regularization allows to find a NN function which is smooth in w so that the generalization capabilities of the network are enhanced[5]. The set of weight values found is then used to make predictions at evaluation time. Using this approach, the outcome of the NN function is a single value estimate, given by $f(\mathbf{x}, \mathbf{w}_{\mathrm{MP}})$.

In the Bayesian framework of neural network training, the NN model is conceived as a Bayesian model, where the weights $\mathbf{w}$ are the free parameters, and the target data of the training $\mathbf{t}_n$ are the observed data. According to Bayesian inference rules, a prior distribution $P(\mathbf{w})$ is assigned to the network weights before training, and a likelihood function $P(D|\mathbf{w})$ is assigned to the observed data, where $D \equiv (t_1...t_{\mathrm{N}})$ denotes the target data from the training set. The training procedure is then an inference process on the network's weights. We can then write Bayes formula to express the posterior distribution of the weights $P(\mathbf{w}|D)$ in terms of the prior and the likelihood function:

$$P(\mathbf{w}|D) = \frac{P(D|\mathbf{w})P(\mathbf{w})}{P(D)} \tag{2}$$

where $P(D)$ is a normalization factor, independent of the weights, also known as the *evidence*. We have omitted the conditioning on the training input data $X \equiv (\mathbf{x_1}...\mathbf{x_n})$ in all the terms, for the sake of simpler notation. The full outcome of the training, from the Bayesian point of view, is then not only a single set of values of the network's weights, but the entire posterior distribution $P(\mathbf{w}|D)$. At evaluation time, the spread of the distribution will then correspond to a distribution of output, the *predictive distribution*. We shall see how, under certain assumption and approximations, we can get to an expression for the predictive error bars.

The first step in the application of such method, is the choice of the prior distribution $P(\mathbf{w})$ and the likelihood function $P(D|\mathbf{w})$. We shall assume for both of them Normal distributions. The reason behind this choice is that it allows making the analytical progress required to derive a mathematical expression for the error bars of the network's output. In this way, we will also recover results very well known and established under the traditional view of NN training. In the general case of a multi-layer neural network, we shall choose a prior of the form:

$$P(\mathbf{w}) \propto \exp\left(-\frac{1}{2}\sum_k \alpha_k \|\mathbf{w}\|_k^2\right) \tag{3}$$

where $\alpha_k \equiv 1/\sigma_k^2$ and σ_k^2 denotes the variance of the distribution for the weights at the neural network's layer k. Concerning the likelihood function $P(D|\mathbf{w})$, we shall use an expression of the form:

$$P(D|\mathbf{w}) \propto \exp\left(-\frac{\beta}{2}\sum_{n=1}^{N}(y^n - t^n)^2\right) \tag{4}$$

where $\beta \equiv 1/\sigma_D^2$ and σ_D^2 denotes the variance of the noise in the training target data, i.e. the spread of the distribution of the target variables, for a given, fixed input vector. We can now use Bayes formula to find an expression for the posterior distribution of the weights. If we are interested in a single value solution, we can look for the weight values that maximize the posterior. This is equivalent to minimizing the negative logarithm of Equation 2, $\ln(P(\mathbf{w}|D)) \equiv -S(\mathbf{w})$, which, substituting the expressions in Equation 3 and Equation 4, can be written as:

$$S(\mathbf{w}) = \frac{\beta}{2}\sum_{n=1}^{N}(y^n - t^n)^2 + \frac{1}{2}\sum_{k=1}^{L}\sum_{i=1}^{N_k}\alpha_k w_{k,i}^2 \tag{5}$$

where L is the number of layers in the network, i is an index labelling the weights at layer k and N_k is the number of weights at layer k. Notice that we have omitted terms that do not depend on $\mathbf{w}$, specifically no contribution from the evidence term of Bayes formula appears in this equation, since they would not have any effect in the minimization of $S(\mathbf{w})$ with respect to the weights. The expression in Equation 5 resembles closely the one in Equation 1. Indeed, this is how the Bayesian point of view and the traditional one comes together. The first term on the right-hand side of Equation 1 comes into Equation 5 as the choice of the Gaussian noise model on the target training data, while the second one, the regularizing term, appears here as a consequence of the Gaussian prior on the networks weights. In particular, we notice that the particular choice of the squared norm of the weight vector has led to a regularizing term well known in the neural network field as *L2 regularization* or *weight decay*: it has the effect of constraining the weights to small values with the consequence of improving the generalization of the network mapping, as described in[5].

An analytical expression for the full posterior $P(\mathbf{w}|D)$ can be found taking a Gaussian approximation of it around $\mathbf{w}_{\mathrm{MP}}$[4], where $\mathbf{w}_{\mathrm{MP}}$ is set of weight values found minimizing Equation 5. This approximation is also known as the *Laplace approximation*, and it leads to:

$$P(\mathbf{w}|D) \approx \exp\left(-S(\mathbf{w}_{\mathrm{MP}}) - \frac{1}{2}\Delta\mathbf{w}^{\mathrm{T}}\mathbf{A}\Delta\mathbf{w}\right) \tag{6}$$

where $\Delta\mathbf{w} = \mathbf{w} - \mathbf{w}_{MP}$ and $\mathbf{A} = \nabla\nabla S_{\mathrm{MP}} = \beta\nabla\nabla E^{\mathrm{MP}} + \sum_k \alpha_k \mathbf{I}$ is the Hessian matrix of the error function in

Equation 5, calculated with respect to the weights and evaluated at $\mathbf{w}_{\mathrm{MP}}$, and $E^{\mathrm{MP}} = \frac{1}{2}\sum_{n=1}^{N}(y^n - t^n)^2$ is the sum-of-square errors term evaluated at $\mathbf{w}_{\mathrm{MP}}$, and $\mathbf{I}$ is the identity matrix. We see therefore that $\mathbf{A}$ has two contributions, the first one coming from the choice of the likelihood function, controlled by the parameter β and the second one coming from the choice of the prior distribution of the weights, controlled by the parameters α_k. This allows us now to calculate the distribution of the network outputs, when a new, unseen, input vector $\mathbf{x}$ is provided to the trained network, at evaluation time. It is obtained by marginalization over the network's weights:

$$P(t|\mathbf{x}, D) = \int P(t|\mathbf{x}, \mathbf{w})P(\mathbf{w}|D)\,\mathrm{d}\mathbf{w} \tag{7}$$

where we have now explicitly included in the notation the dependence on the new input vector $\mathbf{x}$. The distribution $P(t|\mathbf{x}, \mathbf{w})$, which is evaluated at fixed value of the weight vector, is given by the noise model on the target data, as in Equation 4. After some manipulation, we get to the final expression:

$$P(t|\mathbf{x}, D) = \frac{1}{(2\pi\sigma_t'^2)^{1/2}}\exp\left(-\frac{(t - y_{\mathrm{MP}})^2}{2\sigma_t'^2}\right) \tag{8}$$

where:

$$\sigma_t'^2 = \frac{1}{\beta} + \mathbf{g}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{g} \tag{9}$$

where $\mathbf{g} \equiv \nabla_{\mathbf{w}} y|_{\mathbf{w}_{\mathrm{MP}}}$. The distribution of the network's output is then given by a Gaussian distribution, centered at the network prediction obtained with weights $\mathbf{w}_{\mathrm{MP}}$ and with standard deviation given by Equation 9. The contribution to the predictive error has two components: one arising from the noise on the target data, controlled by β, and one arising from the posterior width, controlled by $\mathbf{A}$. Equation 9 corresponds to the *first* procedure to calculate uncertainties.

So far, we have neglected uncertainties in the neural network input. This is of course not ideal when the input is a measured quantity with noise, as it is in our application. It can be shown[6] that an expression for the predictive error, which includes noise on the input, is given by:

$$\sigma_t^2 = \sigma_t'^2 + \sigma_x^2\mathbf{h}^{\mathrm{T}}\mathbf{h} \tag{10}$$

where $\mathbf{h} \equiv \nabla_{\mathbf{x}} y|_{\mathbf{x}_{\mathrm{v}}}$, $\mathbf{x}_{\mathrm{v}}$ is the input vector, and σ_x^2 is the variance of the noise of the input vector, here assumed to be Gaussian. Equation 10 corresponds to the *second* procedure to calculate uncertainties. Three main assumptions that have been done to get to Equation 10: the posterior distribution of the weights has been approximated with a Gaussian distribution around $\mathbf{w}_{\mathrm{MP}}$, the network function $y(\mathbf{x}; \mathbf{w})$ has been approximated by its linear expansion around $\mathbf{w}_{\mathrm{MP}}$ and x_{v} in the calculation of σ_t' and σ_t, respectively. Moreover, the Laplace approximation of the weight's posterior is only valid around $\mathbf{w}_{\mathrm{MP}}$. However, several minima of the cost function are likely to exist and they can be found training the network with different initial values of the weights. The single-Gaussian approximation so far described does not take them into account. In order to account for them, it is possible to approximate the posterior of the weights by a sum of Gaussians, each one centered on each of the minima[5]. This can be accomplished by training a *committee* of networks, where each member is trained with different initialization values, and carrying out the Laplace approximation of the posterior for each of them. The overall posterior is then given by:

$$P(\mathbf{w}|D) = \sum_i P(\mathbf{w}|m_i, D)P(m_i|D) \tag{11}$$

where $P(m_i|D)$ is the *a priori* distribution of the minima m_i, and $P(\mathbf{w}|m_i, D)$ is the posterior distribution of the weights corresponding to the local minima m_i, which can be approximated with the Laplace approximation. The predictive distribution can still be written as in Equation 7, where now the second term on the right-hand side is obtained from Equation 11. Assuming $P(m_i|D)$ to be uniform, we can obtain the uncertainties for a multi-Gaussian approximation of the posterior distribution in the following way: (i) we train a number of NNs with different weight initialization, corresponding to the NN functions f_i, (ii) for each of them, we calculate the posterior of the weights under the Laplace approximation, (iii) we obtain samples from the predictive distribution by randomly choosing one member of the committee, say member i, then, sampling a set of weight values, $\mathbf{w}^i_{\mathrm{MP}}$, and an input vector x^*, from the weight posterior and the input noise model respectively, and calculating the corresponding NN output: $y_i = f_i(\mathbf{w}^i_{\mathrm{MP}}, \mathbf{x}^*)$. The whole procedure is repeated a number of times equal to the desired number of samples. The advantage of this sampling procedure to the estimation of the uncertainties is that it doesn't make use of the assumption of linearity of the NN function around $\mathbf{w}_{\mathrm{MP}}$ and the input vector $\mathbf{x}$. It is therefore more accurate. However, it requires large computational time, and it is therefore not suitable in those applications where the execution time is a concern. This is our *third* procedure for calculating uncertainties, and we will refer to it as multi-Gaussian sampling scheme.

III. APPLICATION TO XICS DIAGNOSTIC DATA AT W7-X

A. The XICS diagnostic at W7-X

The XICS diagnostic at W7-X is equipped with a spherical bent crystal to image X-ray emission of Ar impurities. The emission is then collected on a CCD detector. The diagnostic layout and initial measurements during the first operational phase at W7-X have been described in[7–11]. The collected images have spatial resolution along vertical dimension, corresponding to different lines of sight, and wavelength resolution along the horizontal one. The wavelength range is 3.94 - 4.0 Å for He-like Ar spectra. From the measured data is then possible to reconstruct ion and electron temperature profiles. The ion temperature affects the Doppler broadening of the spectral lines, whereas the electron temperature affects the relative intensities. Given the electron density profile n_e, the impurity density profiles can be obtained[8,12]. A forward model of the diagnostic[7] has been developed within the Minerva Bayesian modeling framework[13], and it is used for the inference of the plasma profiles of interest.

B. Neural networks as approximate Bayesian models

In the XICS Bayesian model, a prior distribution is assigned to the free parameters, in this case temperature, electron and impurity density profiles, and likelihood function is assigned to the measured data. A neural network has been trained with the goal to approximate the full model Bayesian inference. The training scheme is described in detail in[14]. The training set is obtained sampling from the joint distribution of the model $P(T, I) = P(I|T)P(T)$: a set of free parameters is sampled from the prior distribution $P(T)$, and subsequently synthetic data are sampled from the likelihood function $P(I|T)$. The distribution $P(I|T)$ represents the noise model on the XICS measurements, which is given by a Gaussian distribution with mean and variance given by the forward model predicted photon counts. When sampling from the priors, all n_e, T_e, T_i, and impurity density profiles were free to vary, but only the T_i and T_e profiles were used as target of the network's training. The set of sampled synthetic images constitutes the network's input during training. Note that such a training set is made exclusively of data synthesized with the Bayesian model. The profiles are expressed with respect to the effective radius, defined as $\rho_{\text{eff}} = \sqrt{\psi/\psi_{\text{LCFS}}}$ where ψ is the magnetic flux and ψ_{LCFS} is the flux at the last closed flux surface. It is worth to emphasize that, because of training on a given fixed model, any systematic deviation introduced by the specific choice of the model would be reflected in the network's inversions.

IV. RESULTS

Two convolutional neural network[14,15] (CNN), each one with two convolutional layers C1 and C2, followed by one hidden fully connected layer M1 and the output layer M2, have been trained on the inference of T_i and T_e profiles respectively. The training has been carried out in the Bayesian scheme described in section II. The values of $\beta = 10$ and $\alpha_k = (\alpha_{\text{C1}}=68.00,\ \alpha_{\text{C2}}=58.33,\ \alpha_{\text{M1}}=576.67,\ \alpha_{\text{M2}}=5.83)$ were used. The Hessian matrix $\mathbf{A}$ has been calculated in the diagonal approximation. The error bars calculated with Equation 9 and Equation 10 have been compared with each other and the results are shown in Figure 1 and Figure 2. In Figure 1 the average relative uncertainty $\langle\sigma_{\text{rel}}\rangle$ calculated accounting and without accounting for the noise on the input, according to $\langle\sigma_{t,\text{rel}}\rangle \equiv \langle\sigma_t(\rho_{\text{eff}})/T(\rho_{\text{eff}})\rangle$ and $\langle\sigma'_{t,\text{rel}}\rangle \equiv \langle\sigma'_t(\rho_{\text{eff}})/T(\rho_{\text{eff}})\rangle$ respectively, is shown for each spatial location of both profiles. The average has been calculated from data collected across 15 plasma shots of the first operational campaign at W7-X. In the case of T_{e} profiles, it is found that $\langle\sigma_{t,\text{rel}}\rangle \approx 0.2$ for $\rho_{\text{eff}} < 0.4$, and $0.3 < \langle\sigma_{t,\text{rel}}\rangle < 0.4$ for $\rho_{\text{eff}} > 0.5$. In the case of T_{i} profiles, instead, the quantity $\langle\sigma_{t,\text{rel}}\rangle$ shows less variation across the different locations, assuming mostly values approximately equal to 0.1. The difference between $\langle\sigma_{t,\text{rel}}\rangle$ and $\langle\sigma'_{t,\text{rel}}\rangle$ reflects what also emerges from Figure 2 and that we will comment in the next paragraph: the two quantities mostly diverge at the positions corresponding to the plasma core and towards the edge. In Figure 2 it is shown the distribution of the contribution of the input noise term relative to the total error bar magnitude, calculated as $\Delta\sigma_{\text{rel}}(\rho_{\text{eff}}) \equiv (\sigma_t(\rho_{\text{eff}}) - \sigma'_t(\rho_{\text{eff}}))/\sigma_t(\rho_{\text{eff}})$, for each spatial location of the T_e and T_i profiles. The distribution has been computed from the same data used for Figure 1. The orange line connects the mean of each distribution. In the case of the T_e profile (left), it is found that for a significant proportion of the analyzed data the input noise contribution accounts for more than 10%, and mostly less than 20%, of the total error bar magnitude, especially in the positions corresponding to the core and towards the edge, $\rho_{\text{eff}} < 0.2$ and $\rho_{\text{eff}} > 0.7$. In the case of the T_i profiles, instead, it is found in a significant number of cases that the same noise source accounts for more than 20% of the total error, again mainly in the positions corresponding to the core and towards the edge of the plasma, $\rho_{\text{eff}} < 0.2$ and $\rho_{\text{eff}} > 0.6$. The input uncertainties are, therefore, in general not negligible. The purpose of Figure 3 is to illustrate the result of the sampling procedure described at the end of Section II. The network's inversion has been applied on a single measured data point for illustrative purposes. The network's input has been obtained averaging over a 500 ms range. The bundle of grey lines is made of 1000 samples obtained sampling from the multi-Gaussian approximation of the network's weights, accounting also for the noise on the input. The orange line shows the network's prediction and corresponding error bar calculated with the single-Gaussian approximation, Equation 10. The error bars show a $2\sigma_t$ deviation. The sampling scheme based on the multi-Gaussian approximation is a more accurate method to calculate uncertainties, which comes at the price of larger computational time: it is therefore suitable in NN applications where execution time is not a concern.

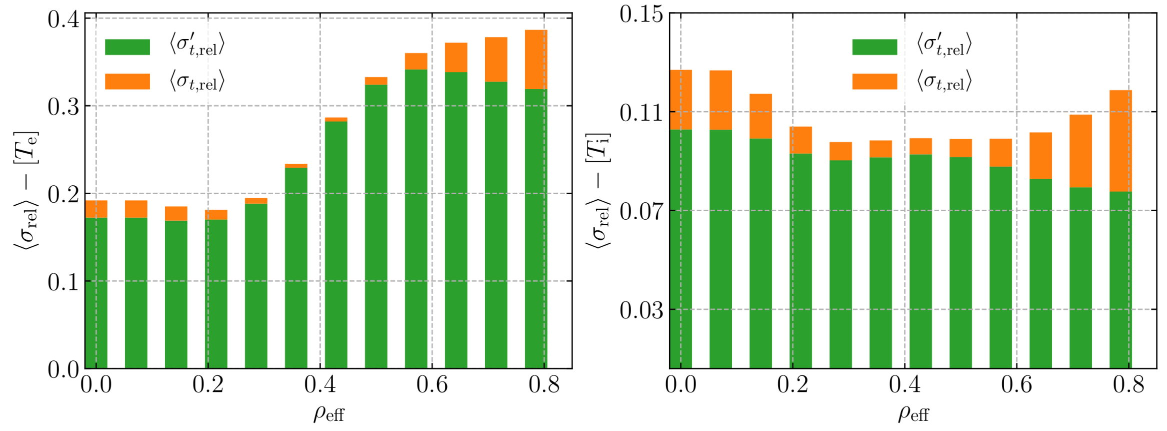

FIG. 1. The average value of the relative uncertainty calculated with (orange bars) and without (green bars) input noise contribution for both T_e (left) and T_i (right) profiles, as found from data collected across different experiments.

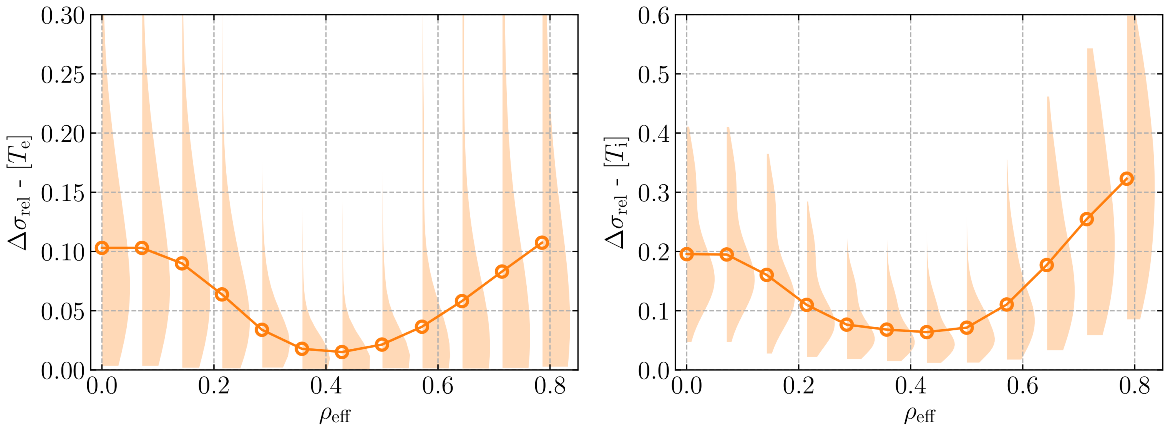

FIG. 2. The distribution of the contribution of the input noise term relative to the total error bar magnitude calculated across data point from different plasma shots for each spatial location in the T_e (left) and T_i (right) profiles. The orange line connects the mean of the distribution at each position.

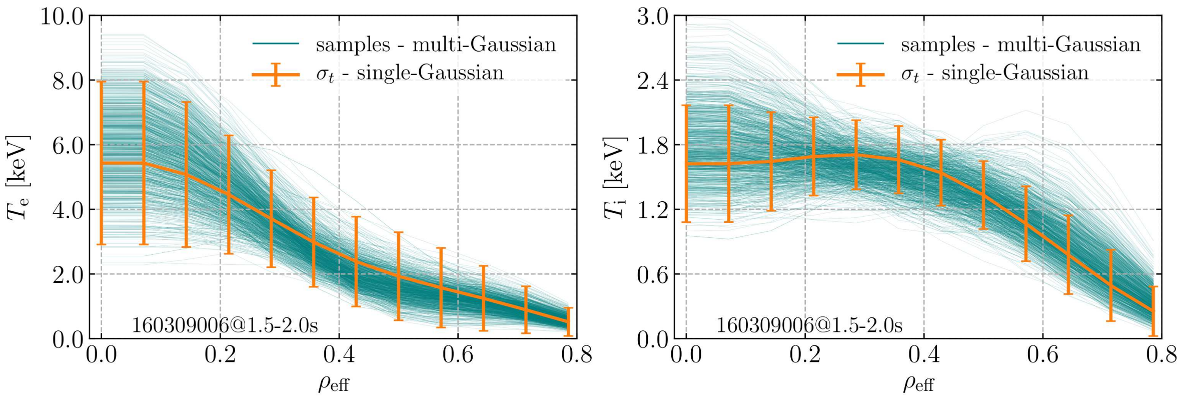

FIG. 3. The neural network prediction and uncertainties calculated in the case of the multi-Gaussian sampling procedure (grey lines), and the single-Gaussian approximation (orange line), for both T_e (left) and T_i (right) profiles.

V. ACKNOWLEDGEMENTS

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

[1]B. Cannas *et al.*, Nuclear Fusion **47** (2007).
[2]J. Svensson *et al.*, Plasma Physics and Controlled Fusion **41** (1999).
[3]J. Svensson *et al.*, ICANN 98. Perspectives in Neural Computing (1998).
[4]D. J. C. Mackay, *Bayesian Methods for Adaptive Models*, Ph.D. thesis, California Institute of Technology (1991).
[5]C. Bishop, *Neural networks for pattern recognition* (Oxford University Press, 1995).
[6]A. Wright, IEEE Transactions On Neural Networks **10** (1999).
[7]A. Langenberg *et al.*, Fusion Science and Technology **69** (2016), 10.13182/FST15-181.
[8]A. Langenberg *et al.*, Nuclear Fusion **57** (2017).
[9]R. König *et al.*, Journal of Instrumentation **10** (2015).
[10]M. Krychowiak *et al.*, Review of Scientific Instruments **87** (2016).
[11]N. A. Pablant *et al.*, 41st EPS Conference on Plasma Physics **38F** (2014).
[12]A. Langenberg *et al.*, 43rd European Physical Society Conference on Plasma Physics, EPS 2016 **40A** (2016).
[13]J. Svensson and A. Werner, IEEE International Symposium on Intelligent Signal Processing (2007).
[14]A. Pavone, (in prep.).
[15]Y. LeCun *et al.*, Proceedings of the IEEE (1998).