Machine Learning Surrogate Model for Neutron Star Mass and Radius from Piecewise Polytropic Parameters

M. D. Danarianto[1], I. N. Huda[2,1], A. Sulaksono[3], R. Kurniadi[4], S. Viridi[4]
[1] Research Center for Computing, National Research and Innovation Agency, Bogor, Indonesia.
[2] School of Astronomy and Space Science, Key Laboratory of Modern Astronomy and Astrophysics (Ministry of Education), Nanjing University, Nanjing 210023, PR China.
[3] Department of Physics, Faculty of Mathematics and Natural Sciences, Universitas Indonesia, Depok, Indonesia.
[4] Department of Physics, Faculty of Mathematics and Natural Sciences, Institut Teknologi Bandung, Bandung, Indonesia.
m.dio.danarianto@brin.go.id
(Submitted on 17.10.2024; Accepted on 15.12.2024)

Abstract. We have constructed a surrogate model to predict neutron star mass and radius from a three-segment parameterized piecewise polytropic equation of state using an artificial neural network. We have trained the network with the generated data from the fourth-order Runge-Kutta-based solver of the Tolman-Oppenheimer-Volkoff equation. It shows that the neural network predicts the mass-radius with no less than 99% accuracy and significantly reduces the computation time compared to the traditional Runge-Kutta method. However, caution is advised when predicting outside the training data parameter ranges, as the model exhibits poor accuracy in extrapolating data and tends to generate false output values where no stellar solution exists. We have argued that this situation may also occur in other similar neural network-based surrogate models.
Key words: Neutron stars, neural network, surrogate models.

Introduction

Beside black holes, neutron stars (NSs) are one of the most compact objects in the universe. A NS forms in the center of an exploding massive star ($M \gtrsim 8M_\odot$) through gravitational collapse, leaving a dense degenerated star in the core. A NS typically has a mass of about $1.4M_\odot$ with a radius only 12 km, making this object one of the densest in the universe (Camenzind, 2007).

The matter properties in a neutron star determine its overall structure, such as its mass and radius. It can be described by an equation of state (EoS), relating between the (energy) density and pressure of matter. Currently, there are a number of models of the EoS of NSs which usually depend on their constituents. The nuclear-physics-based EoS model is typically derived by modeling the microscopic components and their interactions. However, our knowledge regarding the physics of matter in extreme density objects such as NSs is still very limited (Lattimer and Prakash, 2016). In the case of the NSs there is a possibility that the density is higher than the nuclear saturation density. Hence, it is difficult to test the EoS models using ground-based experiments.

The properties of EoS significantly affect the structural properties of NSs, such as their masses and radii. Previous studies have estimated EoS profiles inside NSs based on the observed mass and radius, such as those studied by Ozel et al. (2015). One approach involves using the Monte Carlo method, where the stellar structure equations are solved multiple times for a given set of EoS models, and the likelihood of each EoS is statistically calculated based on the observational data. This method requires intensive calculations to predict

NS properties from a large number of EoS models in order to estimate the probability distribution of the EoS (Jiang et al., 2022).

One can model the EoS in a phenomenological manner to parameterize EoS models. The most simple and well-known model is the piecewise polytropic (PP) (Read et al., 2008). In this model, realistic EoSs can be approached by four parameters. By parameterizing EoS models, the probability distribution of EoS can be described by the probability distribution of these parameters.

Recently, several approaches have involved machine learning techniques to accelerate the computation of inverse problems. The EoS properties and its NS solution are mapped into an artificial neural network. For example, Fujimoto et al. (2017) used a neural network (NN) method to reconstruct the NS mass-radius from a five-segment piecewise polytropic model with a constant speed of sound. Similarly, Liodis et al. (2023) constructed several NNs on multiple EoSs to estimate NS mass radius in a modified gravity theory. Most of the NN studies in NS are used to deduce EoS properties from the data of observed mass-radius data.

Aside from the EoS, the mass and radius of a NS are determined by the assumed central pressure P_c. Higher central pressure results in a more massive star. Central pressure information also plays a role in determining the radial stability zone, where the $M - \rho_c$ gradient must be positive for a star to be stable (Camenzind, 2007). As a consequence, because the EoS pressure is monotonically growing with energy density, the $M - P_c$ curve must also always rise to ensure stability. However, certain network designs, such as those from Fujimoto et al. (2017) predetermined central pressure (which they express as central enthalpy), into fixed values. This can make it challenging to accurately determine the stability turning point. In contrast, the study from Liodis et al. (2023) includes P_c as an input for their NN. However, their NNs are trained with additional modified gravity parameters, which are handled differently for each EoS, as each NN only represents a single EoS model.

The goal of this paper is to construct a neural network-based model that can predict NS mass and radius accurately from PP parameters given its central pressure. A simple network is proposed to ensure the execution time is shorter than the traditional Runge-Kutta method. The performance of this neural network model is assessed for its accuracy and computing time.

This paper is organized as follows. The next section explains the basic neutron star structure calculations and the piecewise polytropic equation of state. Section 2 describes the basic principles of the neural network used in this study. Section 3 discusses the generation of training data, the design of the neural network, and the training process. Section 4 presents our results and discusses the performance and accuracy of this method. Finally, the last section concludes this work.

1 Neutron star structure from piecewise polytropic EoS

A neutron star has a very compact structure, making it necessary to consider relativistic effects. Therefore, the pressure gradient is derived from the theory of general relativity. The relativistic pressure gradient dP/dr, or the so-called Tolman-Oppenheimer-Volkoff equation (Oppenheimer and Volkoff, 1939), is

represented as

$$\frac{dP}{dr} = -\frac{G\epsilon m}{c^2 r^2}\left(1+\frac{P}{\epsilon}\right)\left(1+\frac{4\pi r^3 P}{mc^2}\right)\left(1-\frac{2Gm}{c^2 r}\right)^{-1}, \tag{1}$$

where P, ϵ, and m are pressure, energy density, and enclosed mass at specific radius r, respectively. The constants, G and c, are the gravitational constant and speed of light. To calculate the stellar structure, Eq. (1) has to be integrated with the mass gradient equation, i.e.,

$$\frac{dm}{dr} = 4\pi r^2 \epsilon, \tag{2}$$

along with the equation of state $\epsilon(P)$ that gives the relation between the pressure and energy density of neutron star matter.

The neutron star equation of state $\epsilon(P)$ can be modeled from the microscopic interaction of matter constituents using several models or theories, such as the relativistic mean field (e.g., Müller and Serot, 1996) or Hatree-Fock (e.g., Sun et al., 2009) based method. There are a number of equation of state candidates that have significantly different NS properties. However, the true equation of state of the neutron star remains uncertain due to limited experimental information.

One of the techniques to model the equation of state is by considering a phenomenological approach, such as parameterized piecewise polytropic (Read et al., 2008). In this approach, the relation between density and pressure $P(\rho)$ is approximated by the collection of polytrope segment $P(\rho) = K_i \rho^{\Gamma_i}$, where K_i and Γ_i are polytropic coefficient and index on a segment between $\rho_{i-1} \leq \rho \leq \rho_i$, respectively. The energy density can be calculated using the first law of thermodynamics (see Read et al. (2008) for the detailed derivation). To make $\rho(P)$ continuous, for $\Gamma \neq 1$, the energy density should satisfy

$$\epsilon(\rho) = (1+a_i)\rho + \frac{K_i}{\Gamma_i - 1}\rho^{\Gamma_i}, \tag{3}$$

where

$$a_i = \frac{\epsilon(\rho_{i-1})}{\rho_{i-1}} - \frac{K_i}{\Gamma_i - 1}\rho_{i-1}^{\Gamma_i - 1} - 1. \tag{4}$$

The polytropic coefficient K_i is calculated by the following equation:

$$K_{i+1} = \frac{p(\rho_i)}{\rho_i^{\Gamma_{i+1}}}. \tag{5}$$

By considering the above equations, the number of polytropic pieces i is modeled as the equation of state. Following the method from Read et al. (2008), the equation of state can be sufficiently approximated by dividing it into three pieces, which is divided by $1.85\rho_{nsat}$ (nuclear saturation density) and $3.70\rho_{nsat}$. The equation of state then is represented by four parameters, $\{p_1, \Gamma_1, \Gamma_2, \Gamma_3\}$. By fitting to specific equation of states, these parameters estimate the physical (microscopic) equation of states with 4% rms error. For example, the SLy equation of state (Douchin and Haensel, 2001) can be parameterized into $\{\log(p_1), \Gamma_1, \Gamma_2, \Gamma_3\} = \{34.384, 3.005, 2.988, 2.851\}$.

2 Neural Network

A neural network is a mathematical structure that can approximate a generic continuous function (Cybenko, 1989; LeCun et al., 2015). A neural network consists of interconnected 'neurons.' Each neuron processes and transforms the input signal to output via, i.e., (Goodfellow et al., 2016)

$$a = f(b + \sum_{i=1}^{n} x_i w_i), \tag{6}$$

where the output signal a processes the input x with information of the bias b of that neuron and weight w via the activation function $f(z)$. A single neuron can process signals from multiple inputs with different x_i and w_i. The activation function can be modeled with different functions. One of the most common is *Sigmoid* function,

$$f(z) = \frac{1}{1 + e^{-z}}. \tag{7}$$

Another useful activation function used in this study is the Sigmoid Linear Unit (SiLU) function, which can be written as

$$f(z) = \frac{z}{1 + e^{-z}}. \tag{8}$$

The choice of activation function is crucial to determining the performance and generality of neural network output.

The output signal from a neuron can be transmitted to the next neuron, and it can also be broadcasted to multiple neurons. These neurons can be arranged to form a layer, where each neuron processes signals from all neurons in the previous layer and broadcasts its output to neurons in the next layer. Passing signals through multiple layers forms a deep neural network. The structure of this network significantly affects its performance. While more complex models may provide more detailed predictions, they also require greater computational resources for training and execution.

To make a neural network usable, the weight and bias of each neuron should be *trained* with data to mimic the function that is able to process input to predict output data. The training process consists of optimizing the values of the weights and biases of the neurons in the NN in such a way that the latter predicts the data correctly. Whether the prediction of the NN is correct is determined by the so-called loss function. This usually utilizes the back-propagation technique to find the best value of parameters. The smaller the loss value (obtained by the loss function) the closer the prediction is to the data. The NN then will be used to predict data using trained values of weights and biases. The training process usually includes iterative training and prediction cycles to monitor the performance of the network while being trained.

Essentially, the trainable parameters (weights and biases) in the context of a neural network used as a surrogate model have no direct physical meaning. In

practice, they act as parameters of a generic function (similar to the polytropic index in piecewise polytropic parameters) to estimate the characteristics of the outputs from given inputs. These parameters capture patterns associated with inputs and outputs and strongly depend on the network architecture. In our case, the weights and biases represent the optimal configuration that allows the model to transform the input to the desired output with minimum error, based on the data it was trained on. The pattern of weights and biases may vary between training runs due to the nature of NNs, which have multiple local minima (i.e., the model identifiability problem; see Goodfellow et al. (2016)). However, the model is designed to capture the complex and non-linear behavior of NS mass and radius as a function of its piecewise polytropic parameters, according to the criteria explained in the next section.

3 Methodology

To create the surrogate model within the neural network framework, we implement a "supervised" learning method by training the feed-forward NN architecture using generated data from numerical integration based on the fourth-order Runge-Kutta. In this section, we describe step-by-step the method for solving the stellar structure equation to generate training data. The preprocess of the data is also explained, followed by the design of NN architecture and the training process.

3.1 Training data generation

To calculate neutron star properties, i.e., stellar mass M and radius R, we integrate Eq. 1 and 2 simultaneously by using piecewise polytropic EoS parameters as the input. We implement the fourth-order Runge-Kutta method with an adaptive step to integrate these equations from the center ($r = 0, m = 0, P = P_c$) to the surface ($r = R, m = M, P = 0$), where both the mass and radius are defined. Each P_c produces a different mass and radius, while a single combination of $\{\log(p_1), \Gamma_1, \Gamma_2, \Gamma_3\}$ from PP EoS gives a single mass-radius curve. Therefore, a unique combination of P_c and $\{\log(p_1), \Gamma_1, \Gamma_2, \Gamma_3\}$ produces a unique M and R pair.

Here, we generated 20,000 randomized EoS with 20 different central pressures with randomized spaces for each EoS, between 5-1200 MeV/fm^3. Therefore, there are 400,000 training data in total. We assumed SLy EoS for the crust (Douchin and Haensel, 2001). The parameter range of data generation is shown in Table 1. Given the uniform distribution of EoS parameters in that range, the distribution of mass and radius data is shown in Fig. 1.

In order to better adapt the parameter range to the neural network effective range, the data is transformed using `MinMaxScaler` function from `sklearn` python package (Pedregosa et al., 2011), ensuring the data ranged between 0 and 1 without changing its relative distribution. The central pressure is then transformed to its logarithmic value, i.e., $P_{c,training} = \log(P_c)$, since mass and radius are significantly altered by P_c in the logarithmic scale.

Table 1: Parameter ranges used in the data generation. The data is generated randomly with a uniform ($\mathcal{U}$) distribution.

Parameter	Range
p_0	$\mathcal{U}(33.9, 34.9)$
Γ_1	$\mathcal{U}(2.2, 4.07)$
Γ_2	$\mathcal{U}(1.2, 3.8)$
Γ_3	$\mathcal{U}(1.3, 3.6)$

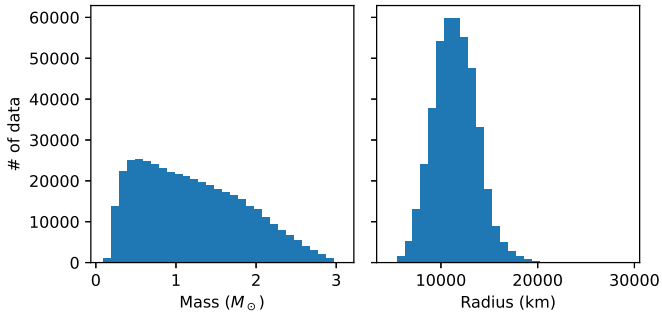


Fig. 1: The histogram of calculated mass-radius data from piecewise polytropic EoS parameters input. Vertical axes represent the number of data.

3.2 NN architecture design

The principle of neural network design for this study is that the neural network should be able to estimate mass-radius with high accuracy, and it has to be simple enough so that the computational cost will be significantly lower than its Runge-Kutta counterpart. We have implemented the neural network architecture using `pytorch` (Paszke et al., 2019). After testing various options of layer size and activation functions, our neural network architecture is shown in Table 2. The NN process $\{\log(P_c), \log(p_1), \Gamma_1, \Gamma_2, \Gamma_3\}$ as the input and $\{M, R\}$ as the output. The sigmoid and sigmoid linear-unit (SiLU) activation functions are chosen to ensure that the prediction (M-R curve) is stable and converges smoothly (e.g., compared to the ReLU function, which tends to give a kinked curve and tends to overfit data).

Table 2: Neural network architecture used in this study. It consists of five layers with sigmoid and sigmoid linear-unit (SiLU) activation functions.

Layer	Description
Input	$\{\log(P_c), \log(p_1), \Gamma_1, \Gamma_2, \Gamma_3\}$
1	Sigmoid(5, 64)
2	SiLU(64, 16)
3	SiLU(16, 16)
4	Sigmoid(16, 2)
Output	$\{M, R\}$

3.3 Training process

The neural network is then trained by utilizing `pytorch`. We divide the training and testing data randomly into a 2:1 ratio, respectively. The training process is then divided again into 256 smaller batches for each step to retain the generality of NN prediction. The loss function is defined as the mean squared error (MSE) of all N data predictions in a batch, i.e.,

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left[(M_{NN,i} - M_{RK,i})^2 + (R_{NN,i} - R_{RK,i})^2\right], \tag{9}$$

where 'NN' and 'RK' index denote mass or radius predicted by a neural network and by the Runge-Kutta method, respectively. The training utilizes the Adam optimizer with a learning rate set to be 0.006 and reduced by the factor of 0.99 for each step after 300 steps. There are in total 1000 steps to train the model until it sufficiently converged.

4 Results and discussions

This section discusses the training and prediction process, as well as the accuracy and performance of the network. It is followed by a demonstration of the network's implementation in solving the inverse problem using Bayesian inference. Finally, the limitations of this model are addressed in the last part of this section.

4.1 Training performance

In order to analyze the NN performance while training generated data, we calculate the MSE error with respect to the training and testing data for each training epoch. Ideally, the loss function for both training and testing data should overlap. The loss between training and testing diverge when there is overfitting or problems with the model.

Our training performance is shown in Fig. 2. In general, both training and testing loss for each epoch are decreasing at a similar rate. The noisy pattern arises from different loss values of 256 batches of training. We stopped the training iteration until the training trend slightly deviated from the testing trend, which indicates overfitting. In our case, this departure occurs at the loss value of 10^{-6}, which is close to the numerical accuracy of our RK4 method.

4.2 Prediction performance

Model accuracy The trained NN is tested to predict the values of mass and radius of well-known EoS models. Here, we take four EoSs as examples, representing different classes of EoS constituents: SLy (Douchin and Haensel, 2001) representing an EoS consisting of neutrons, protons, electrons, and muons; BGN1H1 (Balberg and Gal, 1997) representing an EoS with hyperons, pion and kaon condensates, and quarks; MS1 (Müller and Serot, 1996) representing an EoS with a large maximum mass ($M_{max} > 2.5M_{\odot}$); and ALF1 (Alford

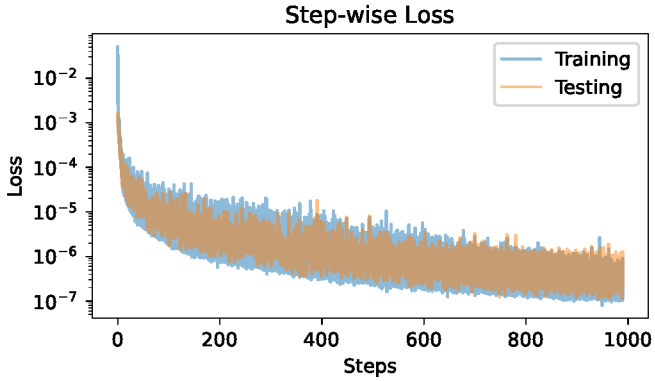


Fig. 2: Prediction performance of the neural network along epoch of training for all 256 batches combined. The blue curve shows training loss and the orange one shows testing loss, both are calculated from mean-squared error, i.e., Eq. (9).

et al., 2004) representing an EoS with hybrid nuclear matter and quark matter. An example of mass and radius prediction is shown in Fig. 3. Figure 4 illustrates mass predictions for different P_c values. The radial stability criterion can be assessed from the turning point of $M - P_c$ curves, where the critical point can be accurately determined by finding the local maxima of these curves. This trained NN is able to precisely determine such critical points.

The accuracy assessment is done by inserting 20 log-equidistant points of P_c the corresponding EoS parameter set to NN, and the output is compared to RK4 calculation. The result is shown in Table. 3. In general, based on assessment using these four EoSs, the error of mass prediction is less than 0.4% and the error in radius is less than 0.6%.

Table 3: Set of parameters used to test the result and their errors with respect to RK4 calculations. The last two columns show the maximum error value of mass and radius of the star, calculated for 20 different P_c.

Model	p_1	Γ_1	Γ_2	Γ_3	M **Error (%, max)**	R **Error (%, max)**
SLY	34.384	3.005	2.988	2.851	0.1517	0.0489
BGN1H1	34.623	3.258	1.472	2.464	0.2673	0.0971
MS1	34.858	3.224	3.033	1.325	0.1709	0.0225
ALF1	34.055	2.013	3.389	2.033	0.3215	0.5295

Computing time The computational performance was assessed by executing 10^5 calculations with the NN method and 10^3 calculations with the RK4 method on a single CPU with a maximum clock speed of 2.3 GHz. The input parameters were randomized within a similar range to those listed in

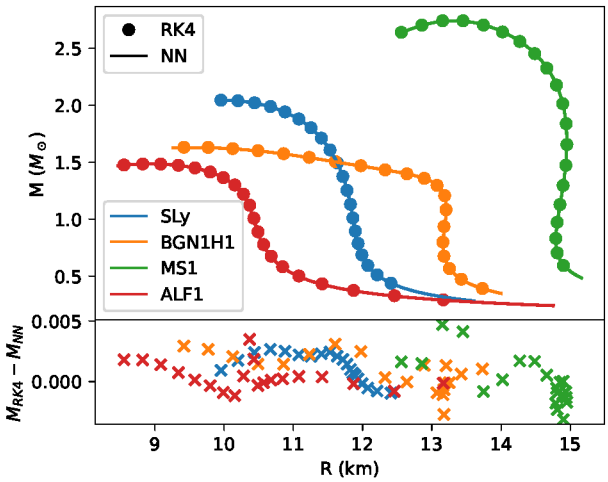


Fig. 3: Prediction of stellar mass and radius from a neural network compared to the RK4 method for different equation of states (SLy, BGN1H1, MS1, and ALF1). The lower box is the residual mass difference between the RK4 method and the NN method in the unit of Solar mass.

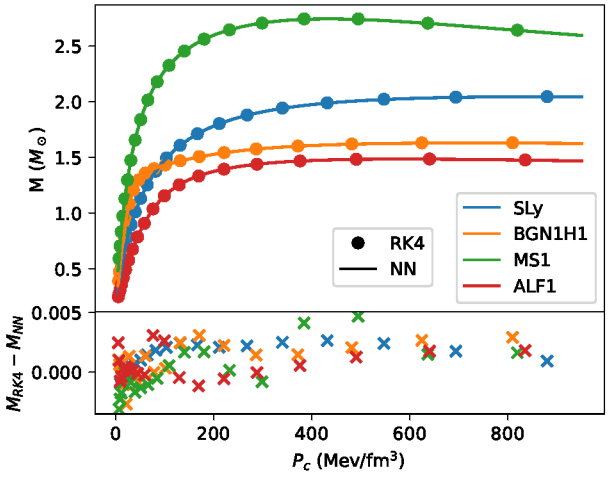


Fig. 4: Prediction of stellar mass along different P_c with residuals similar to Fig. 3. Note that within this range of P_c, the NN predicts the unstable zone in the MS1 EoS ($\gtrapprox$ 400 MeV/fm^3), which is not found in other EoS models listed above.

Table 1. The average time required to compute a single mass-radius curve (20 mass-radius points) is approximately 0.33 ± 0.1 ms for the NN method, compared to 840 ± 180 ms for the RK4 method. This demonstrates that the NN method accelerates the computational time to about 2.5×10^3 times faster than the standard RK4 method. However, it is important to note that the exact computing time strongly depends on the code architecture and hardware capability. For instance, optimizing the code for a specific case or utilizing a GPU for the forward pass could significantly affect performance. For example, the above calculations are done with CPU-optimized code, which may behave differently with other devices such as GPUs. Overall, the NN method used here significantly improves the calculation time.

4.3 Use case example: Inverse problem using nested sampling

As a test case, the neural network is integrated into a statistical inference framework to estimate the probability distribution of piecewise polytropic parameters from observed mass-radius data. The method employs a sampling approach, which involves randomly sampling the prior distributions of input parameters to calculate the likelihood distribution through the network's output. Using Bayesian inference, these steps are combined to estimate the posterior distribution of the parameters, enabling the determination of EoS parameter bounds and calculation of Bayes factor to assess statistical preference between models.

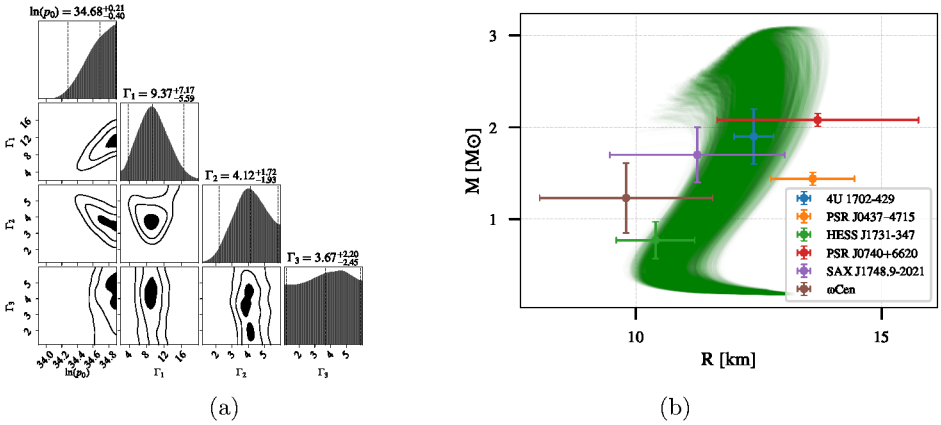
In this case, the prior distribution is set to be uniform, as specified by Table 1. The likelihood $\mathcal{L}$ is modeled as Gaussian with

$$\ln \mathcal{L}_{\mathcal{G}} \propto \frac{(M_{c,i} - M_{obs,i})^2}{\sigma_{M_i}} + \frac{(R_{c,i} - R_{obs,i})^2}{\sigma_{R_i}}, \tag{10}$$

where M_c and R_c represent the closest mass and radius point, respectively, between the mass-radius curve computed by NN and the observed data, M_{obs} and R_{obs}. This approach, commonly referred to as the "closest approach" method, is used to determine the most probable points on the calculated MR curve given multiple M-R data points (Raithel et al., 2017). Additionally, we impose a strict bound to the MR curve, which satisfy

$$\ln \mathcal{L} \propto \begin{cases} -\infty, & \text{if } 2.35 M_\odot \leq M_{TOV} < 3 M_\odot, \\ -\infty, & \text{if } \rho_c > \rho_{c,TOV}, \\ \ln \mathcal{L}_{\mathcal{G}}. & \text{otherwise} \end{cases} \tag{11}$$

The first term ensures that the maximum mass (M_{TOV}) lies between the mass of the PSR J0952-0607 pulsar (Romani et al., 2022) and the lower limit of the canonical mass of stellar-mass black holes ($3M_\odot$). The second term ensures that the mass and radius satisfy the radial stability condition. The posterior is then calculated via the nested sampling method provided by the `dynesty` package (Speagle, 2020). The nested sampling process involves approximately 1.7×10^5 likelihood evaluations (equivalent to the number of NN calls) and takes about 13 minutes to converge using parallel processing on a 10-cores CPU. In comparison, performing the same analysis with the RK4 method



(a) (b)

Fig. 5: Posterior distribution of the piecewise polytropic parameters (a) and posterior predictive checks based on the generated posterior distribution (b). The likelihood is calculated given several observational data from Refs. (Doroshenko et al., 2022; Ozel et al., 2015; Riley et al., 2019). The mass-radius curves shown in the posterior predictive checks represent the samples within 1σ credibility region.

requires several hours to days to converge. The posterior distribution results are shown in Fig. 5a.

It is important to note that our aim is to demonstrate the capability of the NN model in improving recurrent calculations. The case presented here is simplified and should be improved in several aspects. More realistic scenarios typically incorporate other additional observables and criteria, such as the causal limit on the speed of sound in NS matter and other observed astrophysical constraints (Jiang et al., 2022).

4.4 Note on the limitations of the neural network method

The above results show satisfactory performance to use the NN as the surrogate model of the Tolman-Oppenheimer-Volkoff solver based on its accurate and fast calculation. However, Figs. 6 and 7 show that the mass and radius prediction might depart from the RK4 solution in the parameter range outside the training data. This indicates that the NN is inaccurate to extrapolate values outside training data. This situation commonly occurs in neural network-based models, as the NN can only interpolate values between training data (Courtois et al., 2023).

Other than the inaccuracy to predict values outside the parameter range, we should note that in the numerical calculation, it is also possible that the combination of parameters gives no stellar solution. For example, there are some cases where the pressure gradient (i.e., Eq. (1)) yields a positive value, which implies an unstable stellar structure, and therefore mass and radius cannot be defined. In this situation, RK4 will fail to calculate complete mass and radius structure. However, the model is only trained with the generated

data without adding a flag to the failed region. As a consequence, NN prediction will always give a result, whether the complete stellar solution actually exists or not. For example, from Fig. 6 and 7 there are cases where there is no stellar solution for small Γ_3. While the RK4 method could flag the parameter combination in this case as an empty value, the NN cannot. Therefore, in this case, NN prediction is prone to (so-called) hallucination and gives false values for the case of no stellar solution.

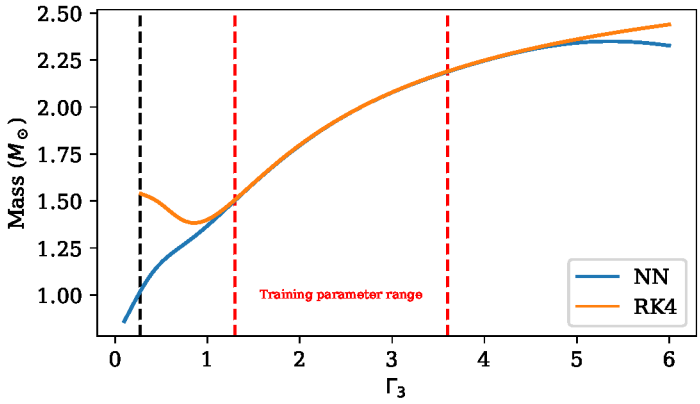


Fig. 6: Values of stellar mass predicted by the NN and the RK4 method on the SLy parameter set with varying Γ_3, showing the accuracy of NN along values of the parameter. The red vertical bar shows the parameter range where the NN is trained, where both the NN and the RK4 overlap with each other (see Table. 1). For the value below the vertical black bar, the RK4 has no stellar solution.

Conclusions

In this paper, we have constructed a neural network-based surrogate model to calculate neutron star mass and radius. The neural network is built with the `pytorch` framework, and consists of four layers, using piecewise polytropic equations of state parameters and their central pressure as the input and mass-radius as the output. Based on the assessment of several PP EoS parameter sets, the NN model achieves more than 99% accuracy in estimating both the NS mass and radius, with errors significantly smaller than the observational precision. This model is also able to reduce the computational time significantly compared to the traditional RK4 method. Hence, it should be useful for application in Monte Carlo calculations, which iteratively compute the same model to determine parameter probability distributions as shown in Sect. 4.3.

However, while the model accurately predicts output within the training parameter range and significantly improves the computational time, several

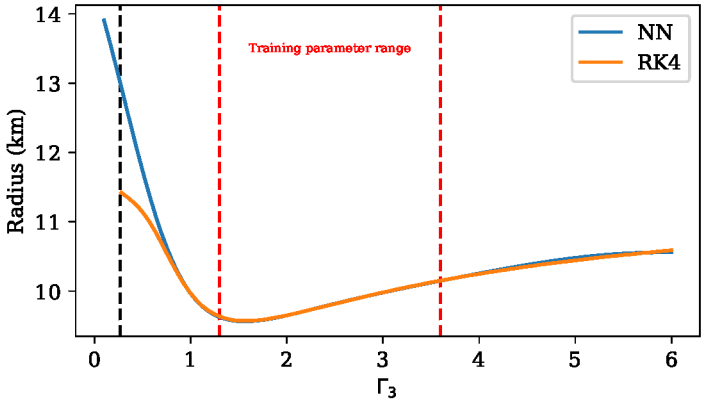


Fig. 7: Similar to Fig. 6 for radius calculation.

precautions have to be taken. Since the model maps TOV solutions based on generated data, predictions outside the training parameter range will yield inaccurate values as the NN is not designed to extrapolate data. Additionally, the model is also prone to hallucinations, where the neural network generates false data in a specific combination of parameters where actually there are no stellar solutions. Therefore, we suggest that such a model should strictly be used to interpolate data within the training parameter range. This issue may also exist in other similar NN models proposed in the literature, where multivariate training data is directly mapped into the NN.

Acknowledgements The computation in this work has been done using the facilities of MAHAMERU BRIN HPC, National Research and Innovation Agency of Indonesia (BRIN).

Bibliography

Alford, M., Braby, M., Paris, M., and Reddy, S. (2004). Hybrid stars that masquerade as neutron stars. *Astrophys.J.629:969-978,2005*, 629(2):969–978.

Balberg, S. and Gal, A. (1997). An effective equation of state for dense matter with strangeness. *Nucl.Phys. A625 (1997) 435-472*, 625(1–2):435–472.

Camenzind, M., editor (2007). *Compact Objects in Astrophysics*. SpringerLink. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg.

Courtois, A., Morel, J.-M., and Arias, P. (2023). Can neural networks extrapolate? discussion of a theorem by pedro domingos. *Revista de la Real Academia de Ciencias Exactas, Físicas y Naturales. Serie A. Matemáticas*, 117(2).

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Math. Control. Signals Syst.*, 2(4):303–314.

Doroshenko, V., Suleimanov, V., Pühlhofer, G., and Santangelo, A. (2022). A strangely light neutron star within a supernova remnant. *Nature Astronomy*, 6(12):1444–1451.

Douchin, F. and Haensel, P. (2001). A unified equation of state of dense matter and neutron star structure. *Astronomy and Astrophysics*, 380(1):151–167.

Fujimoto, Y., Fukushima, K., and Murase, K. (2017). Methodology study of machine learning for the neutron star equation of state. *Phys. Rev. D 98, 023019 (2018)*, 98(2):023019.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Jiang, J.-L., Ecker, C., and Rezzolla, L. (2022). Bayesian analysis of neutron-star properties with parameterized equations of state: the role of the likelihood functions. *The Astrophysical Journal*, 949(1):11.

Lattimer, J. M. and Prakash, M. (2016). The equation of state of hot, dense matter and neutron stars. *Physics Reports*, 621:127–164.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Liodis, I., Smirniotis, E., and Stergioulas, N. (2023). A neural-network-based surrogate model for the properties of neutron stars in 4d einstein-gauss-bonnet gravity.

Müller, H. and Serot, B. D. (1996). Relativistic mean-field theory and the high-density nuclear equation of state. *Nuclear Physics A*, 606(3–4):508–537.

Oppenheimer, J. R. and Volkoff, G. M. (1939). On massive neutron cores. *Physical Review*, 55(4):374–381.

Ozel, F., Psaltis, D., Guver, T., Baym, G., Heinke, C., and Guillot, S. (2015). The dense matter equation of state from neutron star radius and mass measurements. *The Astrophysical Journal*, 820(1):28.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang,

L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Raithel, C. A., Özel, F., and Psaltis, D. (2017). From neutron star observables to the equation of state. ii. bayesian inference of equation of state pressures. *The Astrophysical Journal*, 844(2):156.

Read, J. S., Lackey, B. D., Owen, B. J., and Friedman, J. L. (2008). Constraints on a phenomenologically parameterized neutron-star equation of state. *Phys.Rev.D79:124032,2009*, 79(12):124032.

Riley, T. E., Watts, A. L., Bogdanov, S., Ray, P. S., Ludlam, R. M., Guillot, S., Arzoumanian, Z., Baker, C. L., Bilous, A. V., Chakrabarty, D., Gendreau, K. C., Harding, A. K., Ho, W. C. G., Lattimer, J. M., Morsink, S. M., and Strohmayer, T. E. (2019). A nicer view of psr j0030+0451: Millisecond pulsar parameter estimation. *The Astrophysical Journal Letters*, 887(1):L21.

Romani, R. W., Kandel, D., Filippenko, A. V., Brink, T. G., and Zheng, W. (2022). Psr j0952-0607: The fastest and heaviest known galactic neutron star. *The Astrophysical Journal Letters*, 934(2):L17.

Speagle, J. S. (2020). dynesty: a dynamic nested sampling package for estimating bayesian posteriors and evidences. *Monthly Notices of the Royal Astronomical Society*, 493:3132–3158.

Sun, B. Y., Long, W. H., Meng, J., and Lombardo, U. (2009). Neutron star properties in density-dependent relativistic hartree-fock theory. *Phys.Rev.C78:065805,2008*, 78(6):065805.