The Astrophysical Journal, 916:100 (20pp), 2021 August 1 https://doi.org/10.3847/1538-4357/ac05bd

Analyzing the Galactic Pulsar Distribution with Machine Learning

M. Ronchi[1,2], V. Graber[1,2], A. Garcia-Garcia[1,2,3], N. Rea[1,2], and J. A. Pons[3]
[1] Institute of Space Sciences (ICE, CSIC), Campus UAB, Carrer de Can Magrans s/n, E-08193, Barcelona, Spain; ronchi@ice.csic.es
[2] Institut d'Estudis Espacials de Catalunya (IEEC), Carrer Gran Capità 2–4, E-08034 Barcelona, Spain
[3] Departament de Física Aplicada, Universitat d'Alacant, E-03690 Alicante, Spain
Received 2021 March 22; revised 2021 May 20; accepted 2021 May 25; published 2021 August 3

Abstract

We explore the possibility of inferring the properties of the Galactic population of neutron stars through machine learning. In particular, in this paper we focus on their dynamical characteristics and show that an artificial neural network is able to estimate with high accuracy the parameters that control the current positions of a mock population of pulsars. For this purpose, we implement a simplified population-synthesis framework (where selection biases are neglected at this stage) and concentrate on the natal kick-velocity distribution and the distribution of birth distances from the Galactic plane. By varying these and evolving the pulsar trajectories in time, we generate a series of simulations that are used to train and validate a suitably structured convolutional neural network. We demonstrate that our network is able to recover the parameters governing the distribution of kick velocity and Galactic height with a mean relative error of about 10^{-2}. We discuss the limitations of our idealized approach and study a toy problem to introduce selection effects in a phenomenological way by incorporating the observed proper motions of 216 isolated pulsars. Our analysis highlights that by increasing the sample of pulsars with accurate proper-motion measurements by a factor of $\sim$10, one of the future breakthroughs of the Square Kilometre Array, we might succeed in constraining the birth spatial and kick-velocity distribution of the neutron stars in the Milky Way with high precision through machine learning.

Unified Astronomy Thesaurus concepts: Neutron stars (1108); Pulsars (1306); Convolutional neural networks (1938); Proper motions (1295); Neural networks (1933)

Supporting material: machine-readable table

1. Introduction

Neutron stars have been observed to travel through the Galaxy with typical velocities of around several hundreds of kilometers per second, reaching more than a thousand kilometers per second in some extreme cases (Chatterjee et al. 2005; Hobbs et al. 2005; Hui & Becker 2006; Pavan et al. 2014). Accurate information on neutron star positions and velocities in the Milky Way usually comes from radio timing and interferometric observations (see Chapters 8 and 9 in Lorimer & Kramer 2004; Liu et al. 2020, and references therein) or high-spatial-resolution X-ray observations with, e.g., the Chandra X-ray observatory (Motch et al. 2009). These observations provide measurements of the pulsars' angular positions in the sky and their proper motions projected onto the celestial sphere. In some cases, the radio pulse dispersion measure (DM) or the X-ray absorption density ($N_{\rm H}$) together with models of Galactic free-electron density and hydrogen density (Balucinska-Church & McCammon 1992; Taylor & Cordes 1993; Cordes & Lazio 2002; Yao et al. 2017) can also yield a rough distance estimate. Moreover, in a few cases, a parallax measurement (Deller et al. 2009, 2019; Matthews et al. 2016; Wang et al. 2017) or the presence of a supernova remnant (Yao et al. 2017) might provide better distance measurements.

Such high proper velocities of the neutron star population as a whole exceed those of their progenitors (typically massive OB stars) (see Hansen & Phinney 1997; Lai et al. 2001, and references therein), and cannot be explained by the neutron stars' motion in the Galactic gravitational potential alone. The mechanisms providing such high velocities are still unclear but are likely related to the underlying supernova explosion. One possibility is that the central core of an exploding star receives a kick due to an asymmetric ejection of material from the star's outer layers—a direct result of momentum conservation (Shklovskii 1970; Dewey & Cordes 1987; Mandel & Müller 2020). Additionally, the anisotropic emission of neutrinos has been suggested to impart kicks on compact remnants (Bisnovatyi-Kogan 1993; Fryer & Kusenko 2006; Tamborra et al. 2014; Nagakura et al. 2019).

However, constraining the neutron stars' natal kick-velocity distribution from current observational data is not straightforward. Most pulsars, especially those with very high velocities, have moved far away from their birthplaces, and their proper motions have been modified by the Galactic gravitational potential. Thus, the current velocity of a pulsar may differ substantially from its velocity at birth. Knowing the exact pulsar age and its current 3D spatial velocity, we are in principle able to recover the initial conditions by integrating the pulsar's orbit back in time. However, in general we lack information about the pulsar's line-of-sight velocity, and accurate knowledge about its age, since the characteristic age estimated from the pulsar period and its derivative can differ significantly from the true age (see, e.g., Kaspi et al. 2001; Viganò et al. 2013). Furthermore, estimates of pulsar distances have typically large associated errors due to uncertainties in the underlying density models used to convert pulsar DM or $N_{\rm H}$ into distance estimates (Lorimer et al. 2006; He et al. 2013; Deller et al. 2019).

Reconstruction of the three-dimensional initial position and velocity distribution of pulsars, and comparison with the observed Galactic neutron star population, is therefore a complicated task that requires careful simulations as well as detailed estimates of the observational biases of multiband

surveys. Several studies have performed statistical and population-synthesis analyses to recover the distributions of important neutron star parameters from the observed population (see, e.g., Arzoumanian et al. 2002; Brisken et al. 2003; Hobbs et al. 2005; Faucher-Giguère & Kaspi 2006; Gullón et al. 2014; Verbunt et al. 2017; Cieślar et al. 2020). While these models are broadly able to explain the observational data, high degrees of degeneracy between the different input parameters make it difficult to exactly pin down the distributions that control the pulsars' birth properties, such as their natal kick velocities. Nonetheless, disentangling the birth properties of the isolated neutron star population in our Galaxy is crucial because it has important implications for several lines of research, including formation mechanisms of these compact stars, the evolution of massive stars, as well as extreme events such as gamma-ray bursts, fast radio bursts, and peculiar types of supernovae.

Constraining the birth properties of isolated neutron stars from observational data is the main motivator for our work. Instead of following earlier approaches that have employed standard statistical techniques, we focus on characterizing initial pulsar properties using machine-learning techniques, which have seen increasing interest in the astronomy and astrophysics communities in recent years. This paper, which will be the first in a series, is dedicated to the technical aspects of these efforts and aims to show that a machine-learning (ML) framework can be used to estimate parameters with high accuracy. For this feasibility study, we restrict ourselves to a simplified approach, where selection effects and observational biases are neglected and reduced physical models are sufficient. In particular, we focus on the dynamical properties of the pulsar population and explore the possibility of inferring the parameters that control a given distribution of Galactic pulsar kick velocity and scale height at birth (the two quantities that largely control the spatial distribution of pulsars in the Milky Way) through neural networks. For this purpose, we implement a basic population-synthesis code in `Python` and simulate the dynamical evolution of a synthetic population of isolated neutron stars for a variety of different birth-position and natal-kick distributions. These evolved mock populations are then fed into a suitably structured ML pipeline with the aim of recovering the underlying parameters controlling the distributions. We show that this procedure is successful at estimating birth characteristics. Additionally, we link our framework to the observed sample of pulsars with measured proper motion in a phenomenological way and discuss implications for future pulsar survey efforts, i.e., with the Square Kilometre Array.

Our paper is structured as follows. In Section 2 we describe the methods used to simulate and evolve a mock neutron star population in time. Section 3 contains a description of the ML framework, including the generation of our data sets (Section 3.1), the employed network architectures (Section 3.2), as well as details of the training process (Section 3.3). In Section 4 we present our experiments, which are discussed in detail and connected with observational data in Section 5. Finally, we provide a summary and outlook in Section 6.

2. Population Synthesis

A widely used approach to investigate the properties of the observed neutron star population is through population synthesis (see, e.g., Narayan & Ostriker 1990; Faucher-Giguère & Kaspi 2006; Gonthier et al. 2007; Kiel et al. 2008; Kiel & Hurley 2009; Osłowski et al. 2011; Levin et al. 2013; Bates et al. 2014; Gullón et al. 2014; Cieślar et al. 2020). These frameworks aim to simulate the evolution of a population of neutron stars from birth until today. The resulting mock population is then compared with the real observed population in order to constrain and validate the physical model assumptions that entered the simulation. In particular, the population-synthesis approach relies on assumptions about the distributions of the birth properties of the mock neutron stars, and typically takes advantage of Monte Carlo methods to construct the initial parameters of each simulated star. Starting from these initial conditions the mock population is then evolved over time according to some evolutionary prescriptions, and eventually contrasted with real data. For the development of our population-synthesis framework we largely follow Faucher-Giguère & Kaspi (2006), Gullón et al. (2014), and Cieślar et al. (2020). The necessary ingredients are briefly summarized in the following.

2.1. Age

The age $t_{\rm age}$ of each neutron star is randomly drawn from a uniform probability distribution between 1 and 10^7 yr. By choosing a uniform distribution, we assume that the birth rate of neutron stars is constant in the chosen time range. For all simulations of the synthetic neutron star population we choose an average birth rate of one neutron star per century, compatible with the core-collapse supernova rate in the Galaxy (Rozwadowska et al. 2021). This yields a total of 10^5 simulated neutron stars for each synthetic population, whose evolution we can compute within reasonable timescales.

2.2. Birth Position

To define the initial positions we use both a Cartesian reference frame (x, y, z) and a cylindrical reference frame (r, ϕ, z), whose origins are located at the center of the Galaxy. Here r represents the distance in kiloparsecs from the Galactic center, ϕ is the azimuthal angle in radians, and z is the distance from the Galactic plane. The two coordinate systems are related by the transformation

$$\begin{cases} x = r\cos\phi, \\ y = r\sin\phi, \\ z = z. \end{cases} \tag{1}$$

We assume that the Sun is located at the coordinates $x = 0$ kpc, $y = R_\odot$, $z = z_\odot$, where $R_\odot = 8.3$ kpc and $z_\odot = 0.02$ kpc (see Pichardo et al. 2012, and references therein). We calculate the initial position at birth of each neutron star in both cylindrical and Cartesian galactocentric reference frames. To do so, we execute the following steps.

(i) First, we draw a random distance r from the Galactic center for each neutron star ranging between 10^{-4} and 20 kpc according to a pulsar radial density distribution $P(r)$. In particular, we follow the Milky Way's pulsar surface density $\rho(r)$ defined by Equation (15) in Yusifov & Küçük (2004) to determine the probability density function for the radial distance:

$$P(r) = 2\pi r\rho(r), \tag{2}$$

Table 1
Parameters of the Milky Way Spiral Arm Structure: Winding Constant k, Inner Radius r_0, and Inner Angle ϕ_0

Arm Number	Name	k (rad)	r_0 (kpc)	ϕ_0 (rad)
1	Norma	4.95	3.35	0.77
2	Carina–Sagittarius	5.46	3.56	3.82
3	Perseus	5.77	3.71	2.09
4	Crux–Scutum	5.37	3.67	5.76

Note. Adapted from Table 1 in Yao et al. (2017); see Section 2.2 for more details.

with

$$\rho(r) = A\left(\frac{r + R_1}{R_\odot' + R_1}\right)^a \exp\left[-b\left(\frac{r - R_\odot'}{R_\odot' + R_1}\right)\right], \quad (3)$$

where $A = 37.6\,\mathrm{kpc}^{-2}$, $a = 1.64$, $b = 4.0$, $R_1 = 0.55$ kpc, and $R_\odot' = 8.5$ kpc is the Sun's distance from the Galactic center. Although different from the $R_\odot$ value assumed above, we keep $R_\odot' = 8.5$ kpc in this parameterization in order to be consistent with the results of Yusifov & Küçük (2004). We note that this is the distribution for evolved pulsars rather than that of their progenitors, and Yusifov & Küçük (2004) find small discrepancies between this distribution and that of OB stars. However, Faucher-Giguère & Kaspi (2006) show that the evolved pulsar population is well described by birth positions drawn from Equation (3) and argue that differences fall within the current uncertainties of pulsar distance measurements. Given the lack of a more realistic description, we therefore adopt the above prescription.

(ii) Neutron stars are born mainly within the Galactic spiral arms, as these regions are rich in massive OB stars (Chen et al. 2019). We implement a model for the Galactic spiral structure that includes four arms with a logarithmic shape function that gives the azimuthal coordinate ϕ as a function of the distance from the Galactic center:

$$\phi(r) = k \ln\left(\frac{r}{r_0}\right) + \phi_0. \quad (4)$$

Our values of the model parameters, i.e., the winding constant k, the inner radius r_0, and the inner angle ϕ_0, are reported in Table 1 and evaluated from Table 1 in Yao et al. (2017) in order to match the same functional form as defined in Equation (4). For our analysis, we follow Faucher-Giguère & Kaspi (2006) and do not include the Local arm, whose density is much smaller than that of the four major arms (Cordes & Lazio 2002; Yao et al. 2017). For each star we then randomly select one of the four spiral arms with equal probability, and evaluate the angular coordinate ϕ for its given r according to Equation (4).

The spiral pattern of the Galaxy is not static and as a first approximation can be considered as a rigid structure that rotates with an approximated period $T = 250$ Myr (Vallée 2017; Skowron et al. 2019). Knowing the age of an object and assuming a rotational angular velocity of $\Omega = 2\pi/T$ for the spiral structure, we can derive the angular position at birth of each neutron star. Note that the Galaxy rotates in the clockwise direction, i.e., toward decreasing ϕ angles.

After obtaining the corresponding angular coordinate for each birth position of a neutron star, we add noise to both coordinates r and ϕ to smear out the distribution and avoid artificial features near the Galactic center. For this purpose we add a correction $\phi_{\mathrm{corr}} = \phi_{\mathrm{rand}} \exp(-0.35r\ \mathrm{kpc}^{-1})$ to the ϕ coordinate, where ϕ_{rand} is randomly drawn from a uniform distribution in the interval $[0, 2\pi)$, and to the r coordinate we add a correction r_{corr} randomly drawn from a normal distribution centered at 0 with standard deviation $\sigma = 0.07r$. Although this prescription was introduced by Faucher-Giguère & Kaspi (2006) (see their Section 3.2.1) in a somewhat arbitrary manner, the resulting stellar distribution broadly agrees with that observed for very young high-mass stars as shown in Reid et al. (2019).

Then the birth position in polar coordinates of each neutron star is given by $(r + r_{\mathrm{corr}},\ \phi(r) + \Omega t_{\mathrm{age}} + \phi_{\mathrm{corr}})$ with units [kpc, rad].

(iii) To determine the height z in kiloparsecs from the Galactic plane of each neutron star, we adopt an exponential disk model as given by Wainscoat et al. (1992). It is shaped by the characteristic scale-height parameter h_c:

$$P(z) = \frac{1}{h_c} \exp\left(-\frac{|z|}{h_c}\right). \quad (5)$$

For our ML experiments, we will vary the scale height in the range [0.02, 2] kpc to simulate neutron star populations with different spreads in Galactic height. This range encompasses the value $h_c = 0.18$ kpc, which was adopted by Gullón et al. (2014) to match radio pulsar observations and is also compatible with the population of young massive stars in the Galactic disk (Li et al. 2019). We will consider $h_c = 0.18$ kpc below, whenever a fiducial scale height is required for our synthetic pulsar population. The coordinate z of each neutron star is then randomly drawn according to this height probability distribution in the range from 10^{-4} to 5 kpc. We choose a maximal distance of 5 kpc from the Galactic plane to model a fixed Galactic volume for all of our simulation runs, while also ensuring sufficient resolution for the objects in the Galactic disk for those models with small scale heights. Subsequently, for each star we randomly choose whether z is positive or negative, determining in this way a position above or below the Galactic plane.

2.3. Initial Velocity

We assume that the initial velocity of the neutron stars in the Galaxy is given by two contributions: the progenitor velocity in the Galactic gravitational potential and a kick speed imparted to the neutron stars as a result of the supernova explosion. We consider a circular orbital speed of the progenitor given by the following relation:

$$v_{\mathrm{orb}} = \sqrt{r\frac{\partial \Phi_{\mathrm{MW}}(r, z)}{\partial r}}, \quad (6)$$

where Φ_{MW} is the Milky Way gravitational potential discussed below. We assume that each neutron star has an initial kick velocity $\boldsymbol{v}_{\mathrm{k}}$, whose 3D magnitude v_{k} is randomly drawn from a

Maxwell distribution, shaped by the dispersion parameter σ_k:

$$P(v_k) = \sqrt{\frac{2}{\pi}} \frac{v_k^2}{\sigma_k^3} \exp\left(-\frac{v_k^2}{\sigma_k^2}\right). \quad (7)$$

For our ML purposes, we will vary σ_k in the range [1, 700] km s^{-1} and randomly draw 3D velocity magnitudes from the resulting distribution in the range [0, 2500] km s^{-1}. This spread allows us to easily accommodate the fastest observed neutron stars, whose velocities have been estimated to surpass 1000 km s^{-1} (see for example Chatterjee et al. 2005; Hui & Becker 2006; Pavan et al. 2014). Based on pulsar timing measurements, Hobbs et al. (2005) have suggested that $\sigma_k = 265$ km s^{-1} provides a viable explanation for the proper motions of observed neutron stars. We will use this as a fiducial value below. For a given kick velocity magnitude we then associate a random direction to this velocity in order to evaluate the three components ($v_{k,\,r}$, $v_{k,\,\phi}$, $v_{k,\,z}$) in galactocentric cylindrical coordinates. Therefore, the three components of the total initial velocity of each neutron star in the galactocentric reference frame are computed as ($v_{k,\,r}$, $v_{orb} + v_{k,\,\phi}$, $v_{k,\,z}$).

2.4. Galactic Potential

As is typical for spiral galaxies like the Milky Way, we assume an axisymmetric Galactic potential Φ_{MW} (Carlberg & Innanen 1987; Bovy 2015) that does not incorporate the impact of the spiral arms themselves. We specifically consider a four-component Galactic potential model consisting of a nucleus (Φ_n), a bulge (Φ_b), a disk (Φ_d), and a halo (Φ_h) as discussed in Marchetti et al. (2019):

$$\Phi_{MW} = \Phi_n + \Phi_b + \Phi_d + \Phi_h. \quad (8)$$

The nucleus and the bulge are described by a Hernquist potential (Hernquist 1990):

$$\Phi_n = -\frac{GM_n}{r_n + r}, \quad (9)$$

$$\Phi_b = -\frac{GM_b}{r_b + r}. \quad (10)$$

The disk has a Miyamoto–Nagai disk potential (Miyamoto & Nagai 1975):

$$\Phi_d = -\frac{GM_d}{\sqrt{K^2 + r^2}}, \quad (11)$$

where $K = a_d + \sqrt{z^2 + b_d^2}$ is the shape parameter with a_d as the scale length and b_d the scale height of the disk. The halo has a Navarro–Frenk–White potential (Navarro et al. 1996):

$$\Phi_h = -\frac{GM_h}{r} \ln\left(1 + \frac{r}{r_h}\right). \quad (12)$$

The parameters of this model are reported in Table 2 and were derived by Bovy (2015) through a fit of the mass profile of the Milky Way. We assume that these contributions to the Galactic potential are stationary in time, i.e., they do not evolve over the time span we consider for the dynamical evolution.

Table 2
Parameters of the Milky Way Gravitational Potential Taken from Table 1 in Marchetti et al. (2019); See Also Bovy (2015)

Component	Parameters
Nucleus (n)	$M_n = 1.71 \times 10^9\ M_\odot$
	$r_n = 0.07$ kpc
Bulge (b)	$M_b = 5.0 \times 10^9\ M_\odot$
	$r_b = 1.0$ kpc
Disk (d)	$M_d = 6.8 \times 10^{10}\ M_\odot$
	$a_d = 3.00$ kpc
	$b_d = 0.28$ kpc
Halo (h)	$M_h = 5.4 \times 10^{11}\ M_\odot$
	$r_h = 15.62$ kpc

2.5. Dynamical Evolution

Given the initial conditions defined above, i.e., the initial position, initial velocity, and the Galactic gravitational potential, we can solve the equations of motion to determine the neutron stars' dynamical evolution. The system of dynamical equations that requires solving to determine the orbits of the neutron stars in the Galactic potential is given by the Newtonian equations of motion: $\ddot{\boldsymbol{r}} = -\boldsymbol{\nabla}\Phi_{MW}$. In cylindrical galactocentric coordinates the three components of this vector equation take the form

$$\begin{cases} \ddot{r} - r\dot{\phi}^2 = -\dfrac{\partial \Phi_{MW}}{\partial r}, \\ 2\dot{r}\dot{\phi} + r\ddot{\phi} = -\dfrac{1}{r}\dfrac{\partial \Phi_{MW}}{\partial \phi}, \\ \ddot{z} = -\dfrac{\partial \Phi_{MW}}{\partial z}. \end{cases} \quad (13)$$

For each neutron star we numerically integrate the above equations in time from $t = 0$ yr to $t = t_{age}$ using a discrete time step. We use the Python package `scipy.integrate.odeint`, and to speed up the computational time we also employ the module `jit` ("just in time") from the `Numba` package (https://numba.pydata.org/, Lam et al. 2015).[4] To asses the performance of our integration method we check that both the total energy (i.e., the potential plus kinetic energy) and the z-component of the total angular momentum of all the stars in our simulation are conserved. For simplicity we assume that all pulsars have the same mass and we find that both quantities are conserved with a relative error of $\lesssim 10^{-7}$. The output of the dynamical evolution consists of the position and velocity of each neutron star computed in both galactocentric (GC) and equatorial International Celestial Reference System (ICRS) frames. To transform between different coordinate systems we employed the method `coordinates` from the Python library `Astropy` (Astropy Collaboration et al. 2013, 2018), where we adopted a galactocentric distance of $R_\odot = 8.3$ kpc and Galactic height of $z_\odot = 0.02$ kpc for the Sun.

3. Machine-learning Setup

In the past decade, the accumulation of extensive and heavy data sets has been almost ubiquitous in astronomy and

[4] `Numba` translates Python functions into optimized machine code at runtime, which allows us to achieve a speed-up by about a factor of 6.

astrophysics. In order to take full advantage of these data and perform data-driven science that complements ongoing theoretical modeling efforts, new techniques and analysis pipelines that can handle these large amounts of data (and do so in an automated way) are required. ML has played an important role in developing such new algorithms (Ball & Brunner 2010; Allen et al. 2019; Baron 2019; Fluke & Jacobs 2020). For science related to compact objects, ML algorithms have for example been developed to classify new pulsar candidates (Bethapudi & Desai 2018; Lin et al. 2020; Balakrishnan et al. 2021) as well as transient radio events such as fast radio bursts (Agarwal et al. 2020). Other approaches have aimed at forecasting and analyzing gravitational-wave signals in real time (Cabero et al. 2020; Gerosa et al. 2020; Skliris et al. 2020; Wei & Huerta 2020), interpreting gravitational-wave events in light of population synthesis (Wong & Gerosa 2019), or reconstructing the equation of state of a neutron star from observed quantities (Morawski & Bejger 2020).

For our analysis, we will focus on artificial neural networks (ANNs). ANNs are algorithms inspired by the structure of biological brains that can be thought of as nets of interconnected neurons that exchange information from one to another. When the network receives an input, it is able to process it to produce an output, like a biological brain responds to external stimulation. In ANNs, neurons are usually organized in a stack of layers. Each neuron in a layer receives input signals (typically real numbers) from the neurons in the previous layer and produces an output signal by applying a nonlinear activation function to a linear combination of the input signals according to certain weights and a bias. The output is then passed to the neurons in the following layer, and so on until the final layer is reached and the output is generated. In our particular case, we will focus on supervised learning, where training the neural network consists of making it produce a specific target output when a particular input is passed through it. This is achieved by (i) labeling the input samples in the training data set with a label indicating the property that the network has to learn (the so-called ground truth) and (ii) iteratively adjusting the values of weights and biases, also called network parameters, in order to minimize a specific loss function which measures the distance between the network output prediction and the target ground truth.

Among their numerous applications, ANNs have been employed in regression problems where the network is trained to infer the values of continuous variables for the given input data. This is the kind of problem we are after since we want our network to infer certain parameter values given the evolved neutron star population. In the remainder of this section, we first discuss the simulation data we create for our ML experiments, then focus on the specific network architecture employed, and finally describe the details of our training process.

3.1. Data Set Creation and Processing

The goal of our ML approach is to predict the parameters that control the dynamical properties of an evolved neutron star population. In particular, we focus on predicting the kick-velocity parameter σ_k and the scale-height parameter h_c, which predominantly affect the distribution of pulsars in the Milky Way. To extract these from an evolved population, and follow a supervised learning approach, we first need to train a neural network on a series of simulated populations (created by exploring the ranges for σ_k and h_c). Following the prescription described in Section 2, we perform the following simulation runs:

1. We generate 10 data sets with an increasing number of samples (specifically 4, 8, 16, 32, 64, 128, 256, 512, 1024, and 20,000 simulated populations) by uniformly varying the parameter σ_k of the kick-velocity distribution in the range $[1, 700]\ \mathrm{km\,s^{-1}}$.[5] We also generate a test data set with 1000 samples, each one simulated with σ_k randomly drawn from a uniform distribution in the same range of values. For these simulations, we keep the characteristic scale of the z-distribution fixed to its fiducial value $h_c = 0.18$ kpc.
2. We fix the kick-velocity parameter to its fiducial value $\sigma_k = 265\ \mathrm{km\,s^{-1}}$ and generate a data set of 20,000 samples of simulated populations by uniformly varying the scale-height parameter h_c in the range [0.02, 2] kpc. We also generate a test data set with 1000 samples, each one simulated with h_c randomly drawn from a uniform distribution in the same range of values.
3. We generate six data sets, where we uniformly vary the kick-velocity parameter σ_k in the range $[1, 700]\ \mathrm{km\,s^{-1}}$ *as well as* the characteristic scale of the z-distribution h_c in the range [0.02, 2] kpc. We choose the data set sizes $16 = 4 \times 4$, $64 = 8 \times 8$, $256 = 16 \times 16$, $1024 = 32 \times 32$, $4096 = 64 \times 64$, and $16{,}384 = 128 \times 128$ given by all the combinations of σ_k and h_c values. As an example, the 16 populations in the first set are obtained by combining each of the four values of the σ_k parameter with all four values of the h_c parameter. We also generate a test data set with 1000 samples, each one simulated with both σ_k and h_c randomly drawn from uniform distributions in their respective parameter ranges specified above.

As addressed in detail in Section 4, the smaller simulation data sets will be used to explore the network behavior. The largest data sets containing 20,000 and 16,384 samples, respectively, and the test data sets with 1000 samples will be used to perform the final training experiments and assess the actual network accuracy in generalization scenarios.

After the runs have been performed, we transform the output of the simulation into a representation that can be interpreted by an ML pipeline. Since ANNs require the use of structured data, we represent the position and velocity output of the simulations in the form of 2D binned density and velocity maps in both the galactocentric and ICRS reference frames. The density maps give information about the density of neutron stars in the Galaxy by providing the number count of stars in each spatial bin. On the other hand, velocity maps contain information about the kinematic properties of the neutron stars by providing the average magnitude of the stellar velocity components in each spatial bin. In the galactocentric maps the Galaxy is represented face-on and projected onto the xy-plane of the Cartesian galactocentric frame, extending from -20 kpc to 20 kpc in x and y directions. The ICRS maps instead extend from 0° to 360° in R.A. and from $-90°$ to 90° in decl. To each map we apply a smoothing Gaussian filter (with radius $4.0\sigma + 0.5$ and $\sigma = 1$) in order to add some blurring and avoid sharp features. By doing so, we reduce noisy high-frequency

[5] To generate our simulation data, we partially employ the package `Hydra` (https://hydra.cc/, Yadan 2019), which allows us to easily sweep entire parameter ranges.

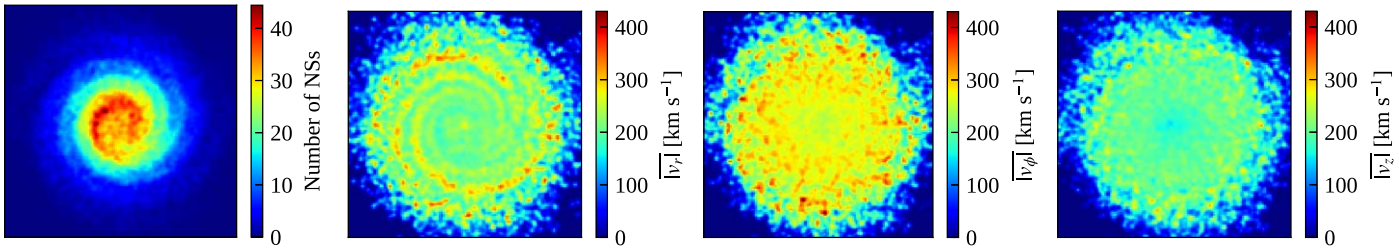

Figure 1. Examples of 128×128 resolution maps in the galactocentric xy-plane extending from -20 to 20 kpc in both x and y directions and showing (in order from left to right) the density of simulated neutron stars and the distribution of average values of the v_r, v_ϕ, and v_z velocity components for a population of neutron stars simulated with $h_c = 0.18$ kpc and $\sigma_k = 265$ km s^{-1}. For visualization purposes, we represent the data using a color map to highlight the resulting structures; red regions are characterized by a higher density of stars or higher average magnitude of the velocity components, while blue areas correspond to lower densities and lower velocity magnitudes. We note that the spiral-arm pattern is still recognizable in the position–density map although high kick velocities tend to blur and disperse the stellar density distribution. In the v_r map, the interarm regions are visible as high-velocity areas, because during the dynamic evolution the space between the spiral arms is progressively filled with high-velocity stars that have escaped from their birthplaces. The other two velocity components exhibit smoother behavior because the spiral-arm structure is smeared out.

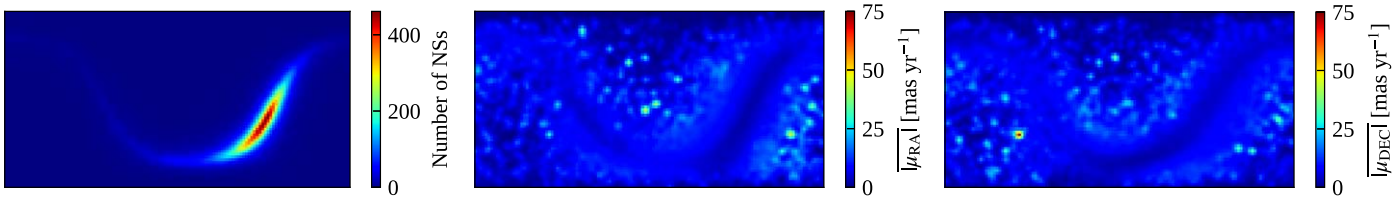

Figure 2. Examples of 128×64 resolution maps in the equatorial ICRS frame extending from 0° to 360° in R.A. and from $-90°$ to 90° in decl. and showing (in order from left to right) the density of simulated neutron stars and the distribution of average values for the $\mu_{R.A.}$ and $\mu_{decl.}$ proper-motion components for a population of neutron stars simulated with $h_c = 0.18$ kpc and $\sigma_k = 265$ km s^{-1}. For visualization purposes, we represent the data using a color map to highlight the resulting structures; red regions are characterized by a higher density of stars or a higher average magnitude of the velocity components, while blue areas correspond to lower densities and lower velocity magnitudes. Note that the Galactic silhouette is visible as a *stream* in the position map with an enhanced stellar density close to the Galactic center. Due to low-number statistics, the regions outside the Galactic stream in the proper-motion maps are dominated by statistical fluctuations, i.e., the corresponding high-velocity regions are attributed to a small number of high proper-motion stars that have escaped the disk. As a result, the disk itself is dominated by stars with lower proper motion.

features and thus make the training more stable, presumably resulting in better generalization capabilities. Therefore, for each simulated population we have

1. one density map in the galactocentric frame;
2. three velocity maps, one for each component of the velocity in cylindrical galactocentric coordinates v_r, v_ϕ, and v_z in km s^{-1};
3. one density map in ICRS coordinates;
4. two proper-motion maps, one for each component of the angular proper motion projected on the celestial sphere $\mu_{R.A.}$ and $\mu_{decl.}$ in mas yr^{-1}.

This set of maps for each population is labeled with the corresponding values of the parameters σ_k and h_c used to simulate that specific population.

We will test the ML performance on three different map resolutions. Thus, we generate each of the above data sets with resolutions 32×32, 128×128, and 512×512 square bins. Note that in the ICRS maps the range of the decl. coordinate axis is half that of the R.A. coordinate axis. Hence, these maps have half as many bins along the decl. coordinate and their resolutions are 32×16, 128×64, and 512×256 square bins. For brevity hereafter we will refer to the three resolutions as 32, 128, and 512 resolutions for both galactocentric and ICRS maps. An example of the maps with resolution 128 for a simulation with fiducial values $h_c = 0.18$ kpc and $\sigma_k = 265$ km s^{-1} is shown in Figures 1 and 2.

Before loading the maps into our ML pipeline, they are normalized so that each bin contains a continuous value between 0 and 1. The same applies to the related labels so that their values range continuously between 0 and 1. The aim of normalization is to speed up the training process and make convergence easier since all inputs will provide signals of similar magnitude to the loss-function minimization. This is useful especially for multiparameter and multichannel training, that is when we train a network to predict more than one parameter or use channels that have different absolute magnitudes. In these cases, training without normalization might lead to slower, worse, or even no convergence at all. Apart from the blurring and normalization described above, we do not apply any additional preprocessing steps to our input data.

3.2. Network Architecture

For the implementation of our ML pipeline we use `PyTorch` (Paszke et al. 2019), an optimized tensor library for deep learning using GPUs and CPUs, which is written in `Python`. The simplest neural network that can be used for this task is a fully connected neural network also referred to as a multilayer perceptron (MLP) with only two layers of neurons, which are referred to as the input and output layers. As this is the starting point to develop models with more complicated and advanced architecture, we first test how this simple configuration behaves. The number of neurons for the input layer is equal to the total number of input features, i.e., $C \times W \times H$. Here, C is the number of input channels (corresponding to the total number of maps used), while W and H are the number of bins in width and height, respectively. The number of neurons in the output layer is equal to the number of regression parameters that we would like to predict, i.e., one or two in our experiments. For the activation function we use the rectified

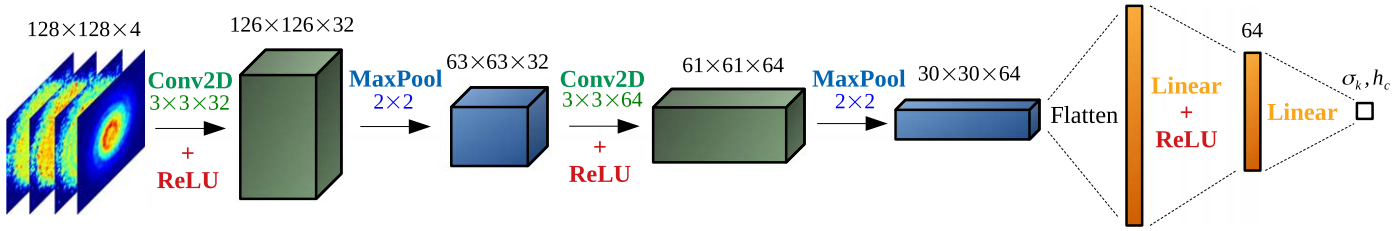

Figure 3. Schematic representation of our CNN architecture for an input of galactocentric maps with resolution 128 × 128 and four channels (one density map plus three velocity maps). A module formed by two blocks that each contain a convolution layer and max pooling layer is followed by a fully connected linear network with one hidden layer. The final network output is either a single parameter or two parameters depending on the experiment specifics.

Table 3
CNN Architecture

Layer	Input	Output
Conv2d + ReLU	$C \times W \times H$	$32 \times (W-2) \times (H-2)$
MaxPool2d	$32 \times (W-2) \times (H-2)$	$32 \times \left(\frac{W}{2}-1\right) \times \left(\frac{H}{2}-1\right)$
Conv2d + ReLU	$32 \times \left(\frac{W}{2}-1\right) \times \left(\frac{H}{2}-1\right)$	$64 \times \left(\frac{W}{2}-3\right) \times \left(\frac{H}{2}-3\right)$
MaxPool2d	$64 \times \left(\frac{W}{2}-3\right) \times \left(\frac{H}{2}-3\right)$	$64 \times \left(\frac{W}{4}-\frac{3}{2}\right) \times \left(\frac{H}{4}-\frac{3}{2}\right)$
Linear + ReLU	$64 \times \left(\frac{W}{4}-\frac{3}{2}\right) \times \left(\frac{H}{4}-\frac{3}{2}\right)$	64
Linear	64	1 (2)

Note. The total number of input and output features is reported. C is the number of channels used, while W and H represent the numbers of bins (i.e., the resolution) in width and height of the density and velocity maps, respectively. Input and output feature numbers have been rounded down to the lower integer.

linear unit (ReLU) defined as $\mathrm{ReLU}(x) = \max(0, x)$. To obtain the output values, the ReLU activation function is applied to a linear combination of the input features with weights and a bias.

A more sophisticated model architecture is represented by a convolutional neural network (CNN). CNNs are a particular type of deep neural network that have proven to be very successful in regression and classification tasks when applied to structured and matrix-like 2D inputs (see Rawat & Wang 2017, for a review). The basic structure of CNNs consists of convolutional, pooling, and fully connected layers. Convolutional layers are multichannel filters that slide along the 2D input maps and are able to extract feature maps. The role of the pooling layer is to downsample the output of a convolutional layer. This inevitably causes a loss of information but in general helps to improve the training efficiency by increasing the size of the receptive field (i.e., the region of the input that produces the feature for each neuron) and reducing the number of trainable parameters. The fully connected layers collect all the output features from the convolution layers into a 1D input and return the final output prediction.

The detailed structure of the CNN we built for our case study can be found in Table 3. A schematic representation of its structure for a four-channel input with galactocentric maps is also shown in Figure 3 as an example. It consists of the following layers:

1. a 2D convolution layer with kernel size 3×3, C input channels, 32 output channels, stride 1, and no padding;
2. a 2D max pooling layer of size 2×2 with stride 2 and no padding;
3. a 2D convolution filter with kernel size 3×3, 32 input channels, 64 output channels, stride 1, and no padding;
4. a 2D max pooling layer of size 2×2 with stride 2 and no padding;
5. a fully connected linear layer with flattened input from the output of the convolutional modules and 64 output neurons;
6. a fully connected linear layer with 64 input neurons and one or two output neurons (depending on the number of parameters we would like to predict).

For the convolutional and pooling layers the stride parameter regulates the amount of displacement in bins that the filter moves over the map at each step. Padding adds one or more bins at the border of the 2D maps, so that the filters can move and cover the whole map without leaving any bins out. We use a padding of 0 because the borders of the maps do not contain relevant information.

The choice of this architecture was found by trial-and-error experiments where we started from a very simple structure and progressively increased the complexity, adding more and more layers to acquire the desired accuracy in predicting the input parameters.

3.3. Training Process

For the training of the network, we use the rms error (RMSE) for both the loss function and validation metric, i.e., to compute the distance between the network predictions and the ground truths of the h_c and σ_k parameters. In general, validation occurs at the same time as training and consists of testing the network over a data set different from the training set. This is needed to assess the ability of the network to generalize what it is learning to an unknown data set. The minimization of the loss function occurs through gradient descent and backpropagation (Kelley 1960; Ruder 2017), i.e., computation of the loss-function gradients with respect to all network parameters (weights and biases). These gradients taken with a negative sign indicate the directions toward which the network parameters should be updated so that the loss is reduced, and hence the network predictions move closer to the true, expected labels. In this regard, a crucial aspect to ensure the best performance of a neural network is to properly initialize the weights and biases. For this purpose we use the Kaiming initialization method (He et al. 2015) in order to avoid exploding or vanishing gradients during the training.

The training process itself is regulated by several hyperparameters. The first one is the learning rate, which is a coefficient for the weight updates. In general, a larger learning rate results in updates of larger magnitude, which could in turn lead to faster convergence, but might also reduce the stability of the training process and thus increase the risk of overshooting the

minima of the loss landscape. A second hyperparameter is the batch size, which defines the number of samples that are packed together and passed through the network before an optimization step is performed. In general, for bigger training data sets, a larger batch size helps to increase the efficiency and stability of the training process, since the gradient-descent steps are averaged over many samples and noise is reduced. For the gradient-descent optimizer we use the adaptive moment estimation (Adam) (Kingma & Ba 2014). As its name suggests, Adam adds an adaptive momentum term to the gradient descent to automatically modify the learning rate and accelerate convergence. When using the Adam optimizer the chosen initial value of the learning rate represents only an upper limit.

We fix the maximum number of learning epochs to 1024. At every epoch the network performs a series of optimization steps by going through the whole training data set once. Then epoch-averaged loss and validation metric values are computed. If the validation metric value has improved with respect to the previous epoch the current status of the optimized network is saved. We set an early stop of 128 epochs, so that if the validation metric does not improve over this epoch span, the training process automatically stops and the weights of the best epoch are stored. This prevents the network from overfitting the training samples, which would reduce its ability to generalize over unknown data.

4. Experiments

As a first step we perform several tests to analyze which configuration of the input feature maps provides the best training experience and which proposed neural network architecture, either the MLP or the CNN, behaves better. Regarding the input configuration, we would like to understand (i) what the best type of map is (galactocentric versus equatorial ICRS); (ii) how many input channels are needed to obtain good results (do density maps provide enough information alone or does the addition of velocity maps improve the results?); (iii) which resolution of the input maps provides the best result. To do so, we first compare the behavior of the MLP and the CNN when trained to predict a single parameter. We explore different types of input signals by varying the resolution of the density and velocity maps as well as the number of input channels. Once we find the optimal configuration for the input maps and the best performing network, we proceed to test its generalization power. A similar strategy is then followed for the two-parameter prediction.

4.1. Single-parameter Predictions

4.1.1. Comparison of Data Representation and Architecture

We focus on predicting the parameter σ_k of the Maxwell kick-velocity function and employ the density and velocity maps generated from simulation run S1. We keep aside the data set with 20,000 samples because it will be used to assess the generalization power of the best performing network (see Section 4.1.2). Thus, we have training sets with an increasing number of samples (from 4 to 1024) and increasing map resolution (32, 128, 512). For each of these training sets we compare the performance on four different kinds of input information:

1. galactocentric position information: one density map in galactocentric coordinates (one channel);
2. galactocentric position and velocity information: one density map plus three velocity maps in galactocentric coordinates (four channels);
3. ICRS position information: one density map in equatorial ICRS coordinates (one channel);
4. ICRS position and velocity information: one density map plus two proper-motion maps in equatorial ICRS coordinates (three channels).

We fix the network architectures and the training setup as described in Section 3.2 and Section 3.3. For these initial experiments we do not incorporate validation but only focus on the training results for different types of data sets, because we are not yet interested in evaluating the generalization power of the networks. To assess the convergence of the training runs we set a threshold for the σ_k RMSE training loss to $10\,\mathrm{km\,s^{-1}}$. If the final RMSE value is higher than this threshold the training is repeated up to a maximum of eight times. If convergence is not reached after eight trials we take the trained model with the lowest final RMSE. For each training experiment we also monitor the computational time needed to perform a single optimization step on a single data batch (see Appendix B for more details).

Initially we perform a search for the starting value of the learning rate that provides the best results. In particular for the MLP we find that, to ensure a decaying RMSE value during training, the initial learning rate needs to be decreased as the input-map resolution increases. Therefore, after several tests we set the initial learning rate to 10^{-4}, 10^{-5}, and 10^{-6}, respectively, for the 32, 128, and 512 resolution maps. The CNN, in contrast, is more flexible and stable and all three initial learning rates are suitable for every resolution. In this case we set it to 10^{-4} to obtain training convergence in the smallest number of epochs.

Next, we tune the batch size to make the learning process more stable and efficient as the number of training samples increases. In general, we keep the batch size to 1 for the data set sizes 4, 8, 16, and 32 and progressively increase it to 4, 8, 16, 32, and 64 as the data set size increases to 64, 128, 256, 512, and 1024, respectively. Only when testing the performance of the MLP on the T2 input configuration do we need to further fine-tune the batch size in order to reach acceptable values of the RMSE. In all other cases the batch sizes mentioned above work well.

The results of our experiments using the MLP and the CNN on the training data sets from S1 with configuration T1, T2, T3, and T4 are shown in Figures 4 and 5, respectively (see Appendix B for the timing results), highlighting the best converged RMSE values as a function of the training data set size and the resolution.

First of all, we note that for both model architectures, ICRS maps allow us to obtain slightly better results in terms of the best RMSE values. An explanation for this could be that only the ICRS maps contain 3D information on the Galaxy, i.e., they encode the stellar height distribution with respect to the Galactic disk, which correlates with the kick-velocity magnitude. In fact the higher the kick velocity of newborn neutron stars the more spread out their distribution in Galactic height z at the end of the dynamical evolution. On the other hand galactocentric maps show the Galaxy represented face-on, and the information on the stars' height z with respect to the Galactic plane is hidden. Therefore, it is likely easier for the

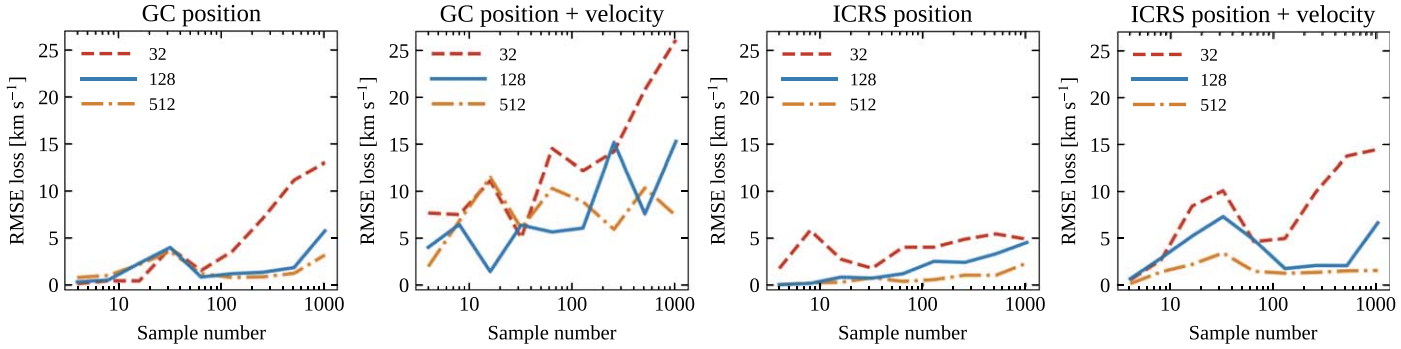

Figure 4. MLP best training RMSE values for the training process on the single parameter σ_k of the Maxwell kick-velocity distribution, as a function of the training data set size and the resolution (red, blue, and orange curves for 32, 128, and 512 respectively) using the four different input configurations T1 (GC position), T2 (GC position + velocity), T3 (ICRS position), and T4 (ICRS position + velocity).

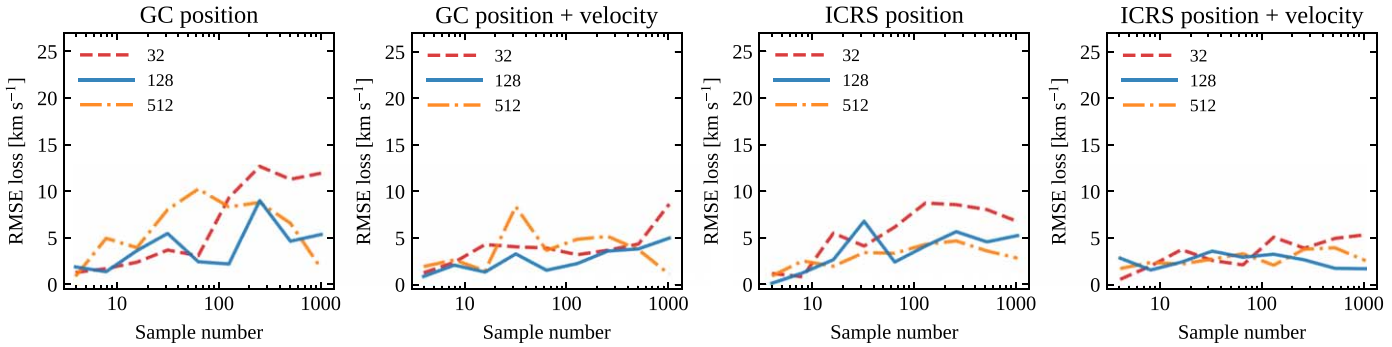

Figure 5. CNN best training RMSE values for the training process on the single parameter σ_k of the Maxwell kick-velocity distribution, as a function of the training data set size and the resolution (red, blue, and orange curves for 32, 128, and 512 respectively) using the four different input configurations T1 (GC position), T2 (GC position + velocity), T3 (ICRS position), and T4 (ICRS position + velocity).

networks to distinguish populations with different σ_k by processing the ICRS maps.

We also note that the addition of the velocity or proper-motion information has opposite effects on the two model architectures. In particular, on one side it worsens the MLP performance, as the best RMSE almost doubles. On the other side, it helps the CNN in reducing the overall RMSE. We interpret this as an indication that, as the complexity of the input data increases, a deeper (with more layers) and more sophisticated model architecture like the CNN is more suitable to process the input data and extract meaningful features to perform the regression task. However, the improvement is not dramatic, which could suggest that the density maps already provide enough information to distinguish, with sufficient precision, populations simulated with different σ_k values.

Concerning the resolution, we observe that higher resolutions allow us to reach slightly lower RMSE values with both the MLP and the CNN but at the expense of longer computation time (see Figures 13 and 14 in Appendix B). We therefore consider the small differences between the best RMSE values obtained with 128 and 512 resolutions not sufficient to justify the choice of the higher resolution. Hence, the use of the 128 resolution appears to be a good compromise to ensure good accuracy and reasonably fast training.

In light of these results for the single-parameter estimation, we conclude that the optimal representation to be used for training is composed of the ICRS density plus proper-motion maps with 128 resolution. Moreover, as the CNN obtains the best results and appears more stable and flexible when compared to the simple MLP (especially for the multichannel input features), we employ our CNN architecture in the following experiments, which should also be less prone to overfitting.

4.1.2. Generalization Results

As the next step we separately train the CNN to predict the σ_k and h_c parameters by using the two big data sets with 20,000 simulations each (see S1 and S2). As input features we use the three-channel ICRS representation with one density map and two proper-motion maps with 128 resolution that ensure the best results as suggested by our earlier experiments. We randomly split both data sets into training and validation subsets with relative percentages of 80%/20%, respectively. Therefore, training is performed over 16,000 simulations, randomly sampled from the entire data sets, while validation is performed over the remaining 4000 simulations. We adopt an initial learning rate of 10^{-4}, a batch size of 64, a total of 1024 learning epochs, and an early stop at 128 epochs to avoid overfitting. The evolution of the training and validation losses is shown in the left panels of Figure 6.

The predictions of the trained network on the validation set for σ_k and h_c are summarized in Figure 6 and Table 4. In the first case, the network is able to predict the value of the kick-velocity parameter σ_k for the simulations in the validation data set with an RMSE uncertainty of 4.4 km s^{-1}, computed over the whole range [1, 700] km s^{-1}. This is indicated by the red dashed lines in the residuals plot (see top central panel of Figure 6), which delimit the 68% uncertainty region. In the second case, the network is able to predict the value of the scale-height parameter h_c for the simulations in the validation data set with an RMSE uncertainty of 0.017 kpc, computed

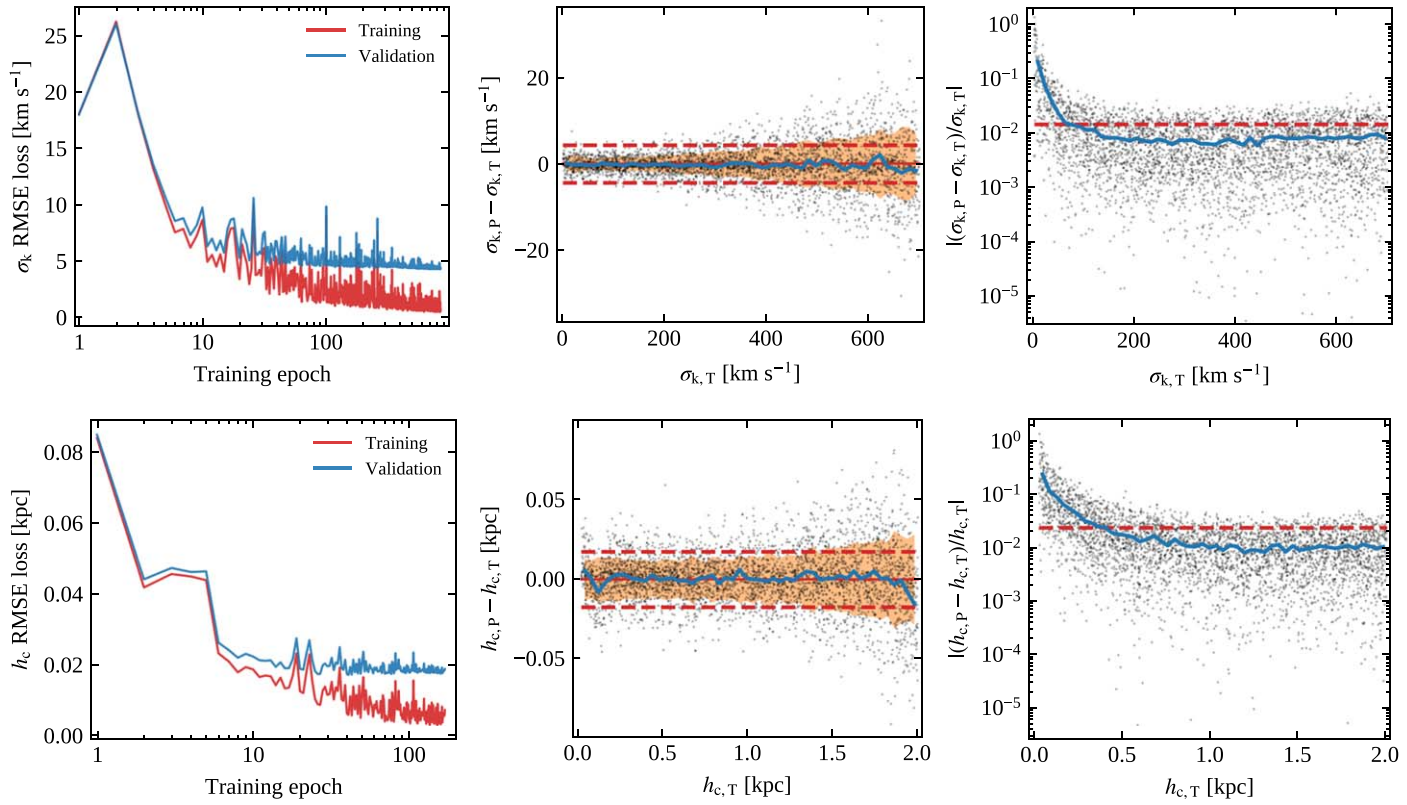

Figure 6. Results of the CNN single-parameter prediction for the kick-velocity parameter σ_k and scale height h_c for the corresponding validation sets. Top (bottom) left panel: evolution of the RMSE training (red) and validation (blue) losses as a function of the training/validation epoch for the σ_k (h_c) parameters. Top (bottom) central panel: residuals of the prediction as a function of the target σ_k (h_c) value; the subscripts P and T refer to the predicted and target values, respectively. The red dashed lines delimit the 68% uncertainty region corresponding to RMSE = 4.4 km s^{-1} (0.017 kpc) computed over the whole range [1, 700] km s^{-1} ([0.02, 2] kpc). The orange region delimits the 68% uncertainty region computed as a running RMSE, for which we have divided the full σ_k (h_c) range into 50 bins of equal size. The blue line shows the trend of the average residuals, which are well centered around the value 0. Top (bottom) right panel: relative error of the prediction as a function of the target σ_k (h_c) value. The red dashed line corresponds to MRE = 0.014 (0.024) computed over the whole range [1, 700] km s^{-1} ([0.02, 2] kpc). The blue line shows the trend of the running MRE computed over 50 bins of equal size. into which we have divided the full σ_k (h_c) range.

Table 4
Summary of the CNN Generalization Results on the Validation and Test (in Parenthesis) Data Sets for the Single-parameter and Two-parameter Cases

	1-par. Generalization		2-par. Generalization	
Parameter	RMSE	MRE	RMSE	MRE
σ_k	4.4 (4.8) km s^{-1}	0.014 (0.018)	8.8 (9.1) km s^{-1}	0.039 (0.033)
h_c	0.017 (0.019) kpc	0.024 (0.029)	0.038 (0.041) kpc	0.061 (0.057)

over the whole range [0.02, 2] kpc (see bottom central panel of Figure 6). Note that in both experiments, the residuals spread out as the target values increase. We visualize this by computing a running RMSE with increasing target values. As is shown by the orange regions in the residuals plots, the running RMSE increases from 1.4 to 7.1 km s^{-1} for σ_k and from 0.013 to 0.028 kpc for h_c. However, the average residuals are consistent with 0 over the entire ranges of target value as marked by the blue line in the residual plots, showing no anomalous trends in the predicted values.

In the right panels of Figure 6 we also show the relative error to highlight the precision with which the network is able to predict the σ_k and h_c parameters for a given target value. We observe that the precision of the predictions improves with increasing σ_k and h_c and eventually stabilizes to a relative error of around 0.01. This is highlighted by the blue lines, which show the trend of the mean relative error (MRE) as a function of the target parameters. The red dashed lines correspond instead to the MRE computed on the whole parameter ranges and are equal to 0.014 and 0.024 for σ_k and h_c, respectively. The fact that the precision of the predictions decreases at the lower end of the target ranges suggests that (for our chosen network setup) the input maps become harder to distinguish as the neutron stars' initial kick velocities and their Galactic birth heights decrease in magnitude.

We then evaluate the generalization capability of the two trained networks on the corresponding test sets with 1000 samples each. We find RMSEs of 4.8 km s^{-1} and 0.019 kpc and MREs of 0.018 and 0.029 for σ_k and h_c, respectively. To further assess the confidence intervals of both estimators, we evaluate the RMSE and the MRE of the parameter values predicted by the two networks over 1000 bootstrapped sets of the related test sets. We find that the RMSE variation is around 3%, while the MRE variation is around 11% for both predicted parameters. These results indicate that the trained networks are able to generalize well over unseen data sets.

4.2. Two-parameter Predictions

4.2.1. Data Representation Comparison

To see how the CNN behaves when two parameters, i.e., the kick-velocity parameter σ_k and the characteristic scale height

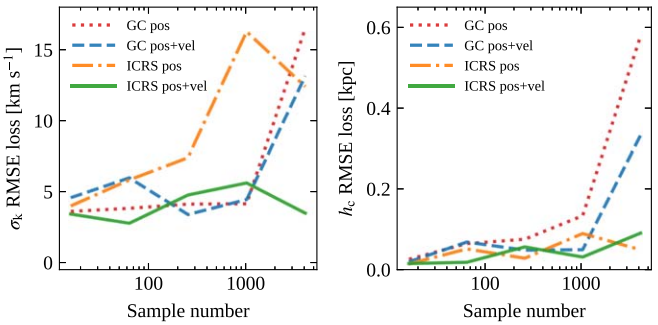

Figure 7. CNN best training RMSE values for the training process on the two parameters σ_k of the Maxwell kick-velocity distribution (left panel) and characteristic scale height h_c (right panel) as a function of the training data set size for the four different input configurations T1 (GC position), T2 (GC position + velocity), T3 (ICRS position), and T4 (ICRS position + velocity).

h_c, are inferred simultaneously, we use the data set of maps generated from simulation run S3 (see Section 3.1). In this case, we have training sets with an increasing number of samples, $16 = 4 \times 4$, $64 = 8 \times 8$, $256 = 16 \times 16$, $1024 = 32 \times 32$, and $4096 = 64 \times 64$. We leave aside the largest data set with $16{,}384 = 128 \times 128$ simulations for our final generalization experiment. Given the results of the single-parameter training experiments, we choose the 128 resolution maps and compare the CNN's performance on the four kinds of input information T1, T2, T3, and T4 as we did for the single-parameter case (see Section 4.1.1). We fix the CNN architecture and the training setup as described in Section 3.2 and Section 3.3, respectively. As for the single-parameter comparison, we only focus on the training behavior for this initial comparison. We keep track of the RMSE training losses for both parameters separately, but training proceeds by minimizing the total loss computed on both parameters. We set the convergence threshold for σ_k to 10 km s^{-1} and that for h_c to 0.5 kpc; if convergence is not reached for both parameters after eight training trials we quote the experiment with the best performance. As before, we also keep track of the computational time for a single optimization step (see Appendix B). The initial learning rate is set to 10^{-4}, while the batch size is changed according to the data set size. In particular, we use batch sizes of 1, 4, 16, 64, and 128 for the data set sizes 16, 64, 256, 1024, and 4096, respectively.

The results of our experiments using the CNN for the two-parameter prediction on the training data sets from S3, with configurations T1, T2, T3, and T4, are summarized in Figure 7 (see Appendix B for the timing results). As in the single-parameter case, we find that the best results are provided by the ICRS maps, which allow us to reach the lowest training RMSE losses for both parameters. In particular, for the ICRS three-channel input, the CNN is able to reach a training RMSE loss $\lesssim$5 km s^{-1} for the σ_k parameter, comparable with the single-parameter case. For h_c, the CNN reaches a training RMSE $\lesssim$ 0.1 kpc. However, we note a drop in accuracy for the σ_k parameter when only the ICRS density maps are used. As already mentioned in Section 4.1.1, information on the stars' z-coordinates is encoded in the ICRS maps. As we simultaneously vary the initial kick velocities and the Galactic heights of the pulsars' birthplaces, the degeneracy between the effects of these two parameters becomes relevant. This makes a distinction of the impact of one parameter over the other more difficult for the network when only ICRS density maps are provided. Adding two extra channels that contain information about the stars' proper motions thus helps to improve the accuracy on the estimation of kick-velocity parameter. The results of these initial explorations are promising and indicate that our simple CNN architecture has good predictive power for both parameters, if provided with all three input channels in the ICRS representation.

4.2.2. Generalization Results

As for the single-parameter analysis, we assess the actual performance of the network when simultaneously predicting σ_k and h_c by training the CNN on the biggest data set with $16{,}384 = 128 \times 128$ simulations (see run S3). As discussed above, we use the three-channel ICRS representation with one density map and two proper-motion maps with 128 resolution as input features. We split the entire data set into training and validation subsets with relative percentages of 80%/20%, respectively, leading to 13,107 samples in the training data set and 3277 samples in the validation one, and use the same configuration as in Section 4.1.2. The evolution of the individual training and validation losses is shown in the left panels of Figure 8.

The results for the trained network's prediction on the validation set for both parameters are shown in Figure 8 and summarized in Table 4. The network is able to predict σ_k and h_c with average RMSE uncertainties of 8.8 km s^{-1} and 0.038 kpc, respectively, which are approximately double those of the single-parameter experiment. These RMSE values are computed over the full target ranges and represented by the red dashed lines in the residuals plots (see central panels of Figure 8). As in the single-parameter case, we observe that the RMSE uncertainties increase with increasing target parameters, as indicated by the orange regions in the residuals plots. The relative errors represented in the right panels of Figure 8 show the same decreasing trend with the target value as for the single-parameter predictions, albeit with larger relative errors. When computing the MRE over the whole range of the two target parameters, we obtain 0.039 and 0.061 for σ_k and h_c, respectively. These values are highlighted by the red dashed lines in the right panels of Figure 8.

We then evaluate the generalization capability of the trained network on the test set with 1000 samples. We find RMSEs of 9.1 km s^{-1} and 0.041 kpc and MREs of 0.041 and 0.057 for σ_k and h_c, respectively. As before, we evaluate the confidence intervals of the two estimators over 1000 bootstrapped sets of the test set and find that the RMSE and MRE variations are around 3% and 8%, respectively, for both predicted parameters. This indicates that the trained network is also stable in the two-parameter prediction and guarantees a good level of generalization power.

However, we find that the CNN trained to simultaneously predict σ_k and h_c is not able to reach the same level of accuracy as in the experiments where a single parameter was predicted at a time. This could be due to one of three distinct causes: (i) our neural network is not sophisticated enough to discern between the effects of the two parameters on the simulation outcomes represented in the ICRS maps, (ii) our choice of ICRS maps as input does not provide sufficient information for the network to distinguish between the two parameters, or (iii) this is a physical (real) degeneracy, and there is a limit to what we can measure. To investigate this issue we train the CNN to predict *only* the parameter h_c using the same two-parameter data set used above where both σ_k and h_c are varied. After predicting on

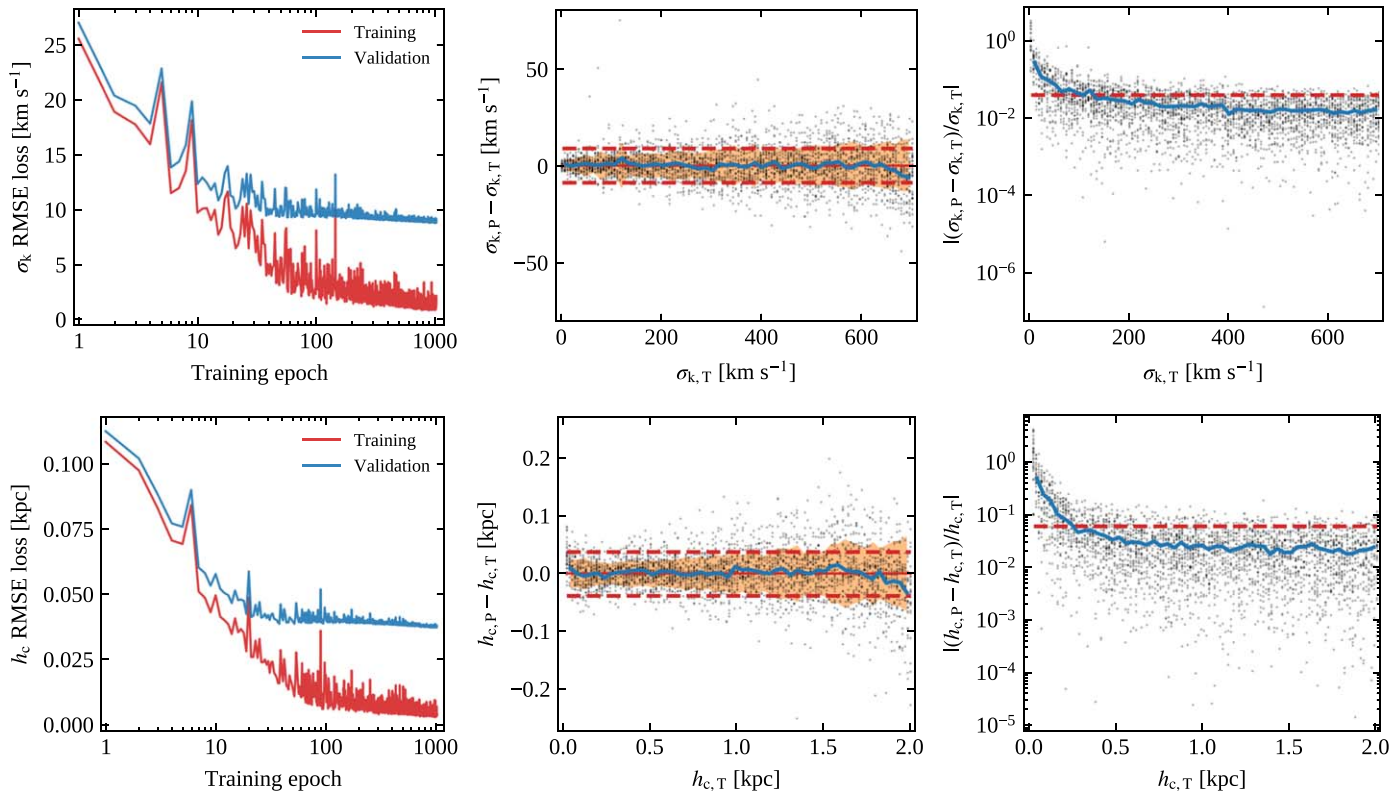

Figure 8. Results of the CNN two-parameter prediction for the kick-velocity parameter σ_k and scale height h_c for the corresponding validation sets. Top (bottom) left panel: evolution of the RMSE training (red) and validation (blue) losses as a function of the training/validation epoch for the σ_k (h_c) parameters. Top (bottom) central panel: residuals of the prediction as a function of the target σ_k (h_c) value; the subscripts P and T refer to the predicted and target values, respectively. The red dashed lines delimit the 68% uncertainty region corresponding to RMSE = 8.8 km s^{-1} (0.038 kpc) computed over the whole range [1, 700] km s^{-1} ([0.02, 2] kpc). The orange region delimits the 68% uncertainty region computed as a running RMSE for which we have divided the full σ_k (h_c) range into 50 bins of equal size. The blue line shows the trend of the average residuals, which are well centered around the value 0. Top (bottom) right panel: relative error of the prediction as a function of the target σ_k (h_c) value. The red dashed line corresponds to MRE = 0.039 (0.061) computed over the whole range [1, 700] km s^{-1} ([0.02, 2] kpc). The blue line shows the trend of the running MRE computed over 50 bins of equal size, into which we have divided the full σ_k (h_c) range.

the validation data set we obtain a RMSE accuracy of 0.038 kpc, which is equal to the result obtained above for the two-parameter prediction. This suggests that the network complexity is suitable to predict either one parameter or two simultaneously. Limitations in performance are therefore due to either an inadequate input representation or a physical degeneracy that imposes a natural accuracy threshold. While we cannot distinguish these two with our current simulation and ML pipeline, we can illustrate the underlying problem in the following way: Figure 9 shows the residuals of the scale-height parameter h_c versus the residuals of the kick-velocity parameter σ_k for the predictions over the validation set. The overall negative slope indicates that the network tends to overpredict h_c in those simulations where σ_k is underestimated and vice versa; i.e., large (small) h_c values have the same overall effect on the phenomenology of the pulsar population as large (small) σ_k values, and the network struggles to distinguish these cases. This highlights the degeneracy between the two parameters already discussed above, which might be broken if the data itself were represented in a different way or additional input information about each neutron star (beyond position and velocity) were provided.

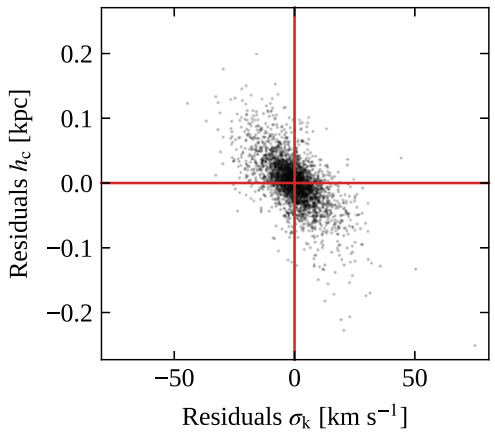

Figure 9. Scatter plot of residuals of the predicted scale-height parameter h_c vs. residuals of the predicted kick-velocity parameter σ_k for the validation data set of our two-parameter generalization experiment. An anticorrelation can be observed.

5. Discussion

In this paper we have studied the potential of an artificial neural network to estimate with high accuracy the dynamical characteristics of a mock population of isolated pulsars. Implementing a simplified population-synthesis framework we focused on the pulsar natal kick-velocity distribution and the distribution of birth distances from the Galactic plane. Taking into account the Galaxy's gravitational potential and evolving the pulsar motions in time, we generate a series of simulations that are used to train and validate a suitably structured CNN.

The generalized results presented in the previous sections are obtained in a very idealized and simplified scenario, implying that caution is required when the uncertainties for the prediction of the kick-velocity dispersion σ_k and birth scale height h_c are taken at face value and conclusions for the real pulsar population are drawn. In particular, our simulations assume that the distribution in Galactic height of neutron star progenitors is represented by the exponential thin-disk model, and that the kick-velocity magnitudes follow a Maxwellian distribution. While the choice of an exponentially thin disk is commonly adopted (Wainscoat et al. 1992; Polido et al. 2013; Li et al. 2019) and can be justified theoretically as the outcome of a self-gravitating isothermal disk (Spitzer 1942), the choice of a Maxwellian model for the kick-velocity distribution is difficult to motivate from a theoretical standpoint. The Maxwellian model has found empirical support because it has been shown to describe the proper motions of observed pulsars well (Hobbs et al. 2005). It is for this reason, and its rather simple mathematical form, that the Maxwell kick-velocity distribution is often adopted in population-synthesis studies of compact stars (Sartore et al. 2010; Cieślar et al. 2020). However, the real functional form of the kick-velocity distribution is still unknown and debated. Several authors have studied the kick-velocity problem and concluded that other models explain observed proper motions of neutron stars equally well. For example, by using maximum-likelihood methods Arzoumanian et al. (2002) found that a bimodal Gaussian distribution with a low-velocity component and a high-velocity one is the preferred model to describe the observed proper motion of a sample of 79 radio pulsars. Faucher-Giguère & Kaspi (2006) studied the variation of the velocity component with Galactic longitude for a sample of 34 pulsars observed with interferometric techniques (Brisken et al. 2002, 2003). After testing a two-component Gaussian model as well as a variety of single-component models, they opted for a single-component description with an exponential shape, although a two-component model could not be ruled out due to the poor statistics of their sample. More recently, Verbunt et al. (2017) and Igoshev (2020) analyzed a sample of isolated young pulsars and found that a two-component Maxwellian model explained the observed sample best.

In general, the presence of a low-velocity component and a high-velocity one could indicate different progenitor properties as well as birth scenarios for the pulsar population. Numerical simulations of supernova explosions, for example, have suggested that neutron stars with lower kick velocities could be generated in the core-collapse supernovae of progenitors with small iron cores or in electron-capture supernovae (Podsiadlowski et al. 2004; Mandel & Müller 2020). While also possible scenarios for isolated systems (Janka 2017), such conditions might generally affect those neutron stars born in binaries (Giacobbo & Mapelli 2020), where mass-loss episodes could strip their progenitors of their hydrogen envelopes. This might favor the formation of small iron cores or accretion-induced electron-capture supernovae, resulting in weaker natal kicks (Schwab et al. 2010; Tauris et al. 2013). Only for the strongest kicks can the binaries be disrupted by the supernova and both companions expelled; otherwise the two stars remain gravitationally bound. Such effects are neglected in our model but could in principle generate an imprint on the observed neutron star population. The ability of ML algorithms to recognize features and patterns in the processed data could help to better constrain the bimodality of the kick-velocity distribution. In future work, we will investigate these aspects in more detail and explore ways to implement an ML framework to reconstruct the shape of the kick-velocity distribution underlying an evolved population of pulsars.

Up to this point, we have not considered any kind of selection effects or observational biases and effectively assumed that all the neutron stars in our simulation are detectable. While this provides direct insight into how various initial conditions affect the evolved population of neutron stars, a direct comparison with observational data in principle requires a careful treatment of biases. For example, due to beaming effects not all Galactic radio pulsars are visible from Earth (Tauris & Manchester 1998; Melrose 2017), while survey sensitivity thresholds and instrumental limitations might hamper the detection of faint or distant sources (Manchester et al. 2001; Johnston et al. 2008; Coenen et al. 2014; Stovall et al. 2014; Good et al. 2020). Additionally, timing noise can significantly limit the sensitivity and precision in the detection of pulsar proper motions via timing analysis techniques (Hobbs et al. 2004; Lentati et al. 2016; Parthasarathy et al. 2019). With the aim of obtaining a rough idea of how selection effects and biases could potentially influence a future comparison with observation, we perform the following experiment. We first collect those neutron stars that have measured proper motions. As the main resource we use the pulsar catalog of the Australia Telescope National Facility (ATNF)[6] (Manchester et al. 2005), but in some cases we provide proper-motion results from more recent analyses (see Appendix C for details). We find a total of 417 neutron stars whose angular positions, proper motions in ICRS coordinates, spin periods, spin-period derivatives, DM values, and distance estimates are reported in Table 5. Out of these objects, we remove those stars that belong to the Magellanic Clouds, are associated with globular clusters, or have a binary companion. We further select only those neutron stars with a spin-period derivative $\dot{P} > 10^{-17}$ to exclude those that have potentially been recycled. Finally, we consider only those for which an estimate of the heliocentric distance $d_\odot$ is available; in the case of radio pulsars we quote values that are derived from their respective DMs using the free-electron density model of Yao et al. (2017) (YMW16 model hereafter). As some neutron stars have a DM that exceeds the maximum Galactic DM allowed by the YMW16 model, these cases are assigned a default distance of 25 kpc. We exclude those cases unless an alternative distance measurement is available. Applying these filters, we obtain a sample of 216 Galactic, likely isolated neutron stars, whose positions and proper motions are illustrated in Figure 10.

In the left panel of Figure 11, the gray histogram shows the distribution of their heliocentric distances. Even if subject to some uncertainties due to imprecisions in the YMW16 model, the distance distribution peaks around 1 kpc and is followed by a sharp exponential decrease. For a realistic pulsar distribution and in the absence of selection effects, we would expect the number of neutron stars to increase with distance due to an increase in the explored volume, until a maximum is reached at a distance of about 10 kpc, which comprises the region around the Galactic center (see the red histogram in the left panel of Figure 11). Thus, the shape of the gray distribution in Figure 11, as expected, indicates that our observed sample of neutron stars with

[6] https://www.atnf.csiro.au/research/pulsar/psrcat/

Table 5
Up-to-date List of 417 Neutron Stars with Measured Proper Motions in R.A. and Decl.

JName	R.A. (h:m:s)	Decl. (°:′:″)	$\mu_{\rm R.A.}$ (mas yr^{-1})	$\mu_{\rm decl.}$ (mas yr^{-1})	Parallax (mas)	Pos. Epoch (MJD)	P (s)	$\dot{P}$ (s s^{-1})	DM (pc cm^{-3})	$d_{\odot}$ (kpc)	Class(Assoc.)	Ref.
J0014+4746	00:14:17.75(4)	+47:46:33.4(3)	19.3 ± 1.8	-19.7 ± 1.5	⋯	49,664	1.24070	5.6446×10^{-16}	30.405 (13)	1.776	PSR	
J0023+0923	00:23:16.87910(2)	+09:23:23.8689(8)	-12.4 ± 0.5	-6.1 ± 0.1	0.4 ± 0.3	56,567	0.00305	1.14234×10^{-20}	14.32810 (4)	1.248	PSR	[6]
J0024-7204ab	00:24:08.1615(5)	−72:04:47.602(2)	4.2 ± 0.6	-2.9 ± 0.5	⋯	51,600	0.00370	9.820×10^{-21}	24.37 (2)	2.54	PSR(GC)	
J0024-7204C	00:23:50.3546(1)	−72:04:31.5048(4)	5.2 ± 0.1	-3.1 ± 0.1	⋯	51,600	0.00576	-4.98503×10^{-20}	24.5955 (8)	2.594	PSR(GC)	
J0024-7204D	00:24:13.88092(6)	−72:04:43.8524(2)	4.24 ± 0.07	-2.24 ± 0.05	⋯	51,600	0.00536	-3.4220×10^{-21}	24.7432 (9)	2.631	PSR(GC)	
J0024-7204E	00:24:11.10528(5)	−72:05:20.1492(2)	6.15 ± 0.03	-2.35 ± 0.06	⋯	51,600	0.00354	9.85103×10^{-20}	24.236 (2)	2.509	PSR(GC)	
J0024-7204F	00:24:03.85547(10)	−72:04:42.8183(2)	4.52 ± 0.08	-2.5 ± 0.05	⋯	51,600	0.00262	6.45029×10^{-20}	24.382 (5)	2.543	PSR(GC)	
J0024-7204G	00:24:07.9603(1)	−72:04:39.7030(5)	4.5 ± 0.1	-2.9 ± 0.1	⋯	51,600	0.00404	-4.21584×10^{-20}	24.436 (4)	2.556	PSR(GC)	
J0024-7204H	00:24:06.7032(2)	−72:04:06.8067(6)	5.1 ± 0.2	-2.8 ± 0.2	⋯	51,600	0.00321	-1.8294×10^{-21}	24.369 (8)	2.541	PSR(GC)	
J0024-7204I	00:24:07.9347(2)	−72:04:39.6815(7)	5 ± 0.2	-2.1 ± 0.2	⋯	51,600	0.00348	-4.5874×10^{-20}	24.43 (1)	2.555	PSR(GC)	
J0024-7204J	00:23:59.4077(1)	−72:03:58.7908(5)	5.27 ± 0.06	-3.59 ± 0.09	⋯	51,600	0.00210	-9.7917×10^{-21}	24.5932 (1)	2.594	PSR(GC)	
J0024-7204L	00:24:03.7721(3)	−72:04:56.923(2)	4.4 ± 0.2	-2.4 ± 0.2	⋯	51,600	0.00435	-1.22045×10^{-19}	24.40 (1)	2.547	PSR(GC)	
J0024-7204M	00:23:54.4899(3)	−72:05:30.756(2)	5 ± 0.3	-2 ± 0.4	⋯	51,600	0.00368	-3.8419×10^{-20}	24.43 (2)	2.553	PSR(GC)	
J0024-7204N	00:24:09.1880(2)	−72:04:28.8907(7)	6.3 ± 0.2	-2.8 ± 0.2	⋯	51,600	0.00305	-2.18570×10^{-20}	24.574 (9)	2.589	PSR(GC)	
J0024-7204O	00:24:04.65254(6)	−72:04:53.7670(2)	5.01 ± 0.05	-2.58 ± 0.08	⋯	51,600	0.00264	3.03493×10^{-20}	24.356 (2)	2.537	PSR(GC)	
J0024-7204Q	00:24:16.4909(1)	−72:04:25.1644(6)	5.2 ± 0.1	-2.6 ± 0.1	⋯	51,600	0.00403	3.4008×10^{-20}	24.265 (4)	2.517	PSR(GC)	
J0024-7204R	00:24:07.5851(2)	−72:04:50.3954(5)	4.8 ± 0.1	-3.3 ± 0.2	⋯	51,600	0.00348	1.48352×10^{-19}	24.361 (7)	2.538	PSR(GC)	
J0024-7204S	00:24:03.9794(1)	−72:04:42.3530(4)	4.5 ± 0.1	-2.5 ± 0.1	⋯	51,600	0.00283	$-1.205413 \times 10^{-19}$	24.376 (4)	2.542	PSR(GC)	
J0024-7204T	00:24:08.5491(5)	−72:04:38.932(3)	5.1 ± 0.6	-2.6 ± 0.7	⋯	51,600	0.00759	2.93805×10^{-19}	24.41 (2)	2.55	PSR(GC)	
J0024-7204U	00:24:09.8366(1)	−72:03:59.6882(4)	4.6 ± 0.2	-3.8 ± 0.1	⋯	51,600	0.00434	9.52279×10^{-20}	24.337 (4)	2.534	PSR(GC)	
J0024-7204W	00:24:06.058(1)	−72:04:49.088(2)	6.1 ± 0.5	-2.6 ± 0.3	⋯	50,000	0.00235	-8.65526×10^{-20}	24.367 (3)	2.539	PSR(GC)	
J0024-7204X	00:24:22.38565(9)	−72:01:17.4414(7)	5.8 ± 0.1	-3.3 ± 0.2	⋯	54,000	0.00477	1.83610×10^{-20}	24.539 (5)	2.587	PSR(GC)	
J0024-7204Y	00:24:01.4026(1)	−72:04:41.8363(4)	4.4 ± 0.1	-3.4 ± 0.1	⋯	51,600	0.00220	-3.51721×10^{-20}	24.468 (4)	2.563	PSR(GC)	
J0024-7204Z	00:24:06.041(2)	−72:05:01.480(6)	4 ± 2	1 ± 2	⋯	51,600	0.00455	-4.54×10^{-21}	24.45 (4)	2.559	PSR(GC)	

Note. Table 5 is published in its entirety in machine-readable format. A portion is shown here for guidance regarding its form and content.

(This table is available in its entirety in machine-readable form.)

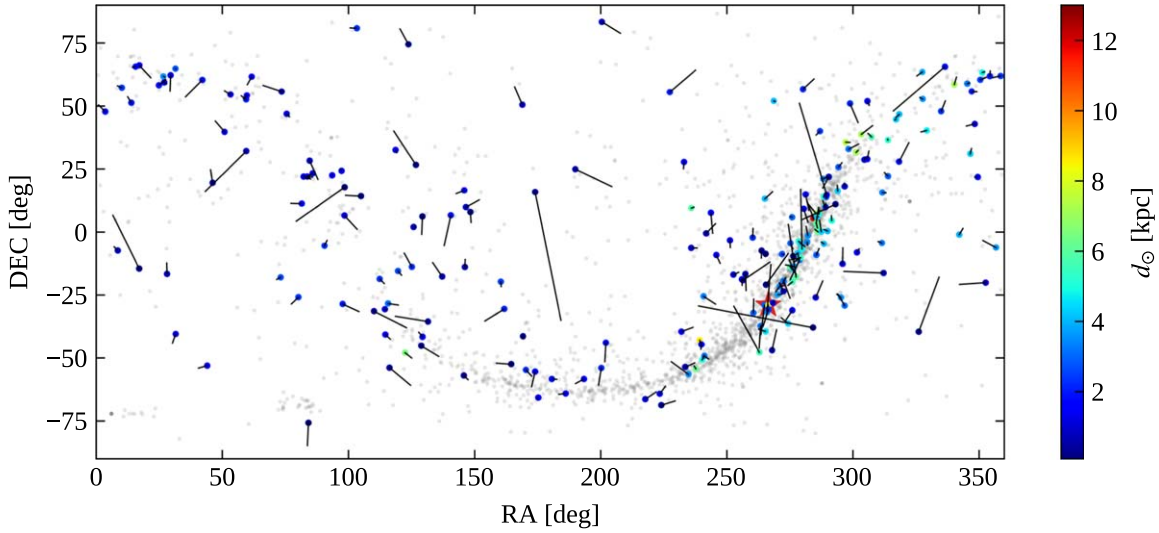

Figure 10. Proper motions of the selected 216 neutron stars in the sky represented in the ICRS reference frame. The current locations of the neutron stars are indicated by the colored circles, whereas the tracks indicate their motion for the past 0.5 Myr, assuming no radial velocity and neglecting the effects of the Galactic potential. The color encodes the heliocentric distance $d_\odot$ of the neutron stars. The corresponding data are provided in Table 5. In the background, we show in gray all non-binary pulsars in the ATNF catalog (those in the Small and Large Magellanic Clouds as well as those in globular clusters are included). The red star highlights the position of the Galactic center.

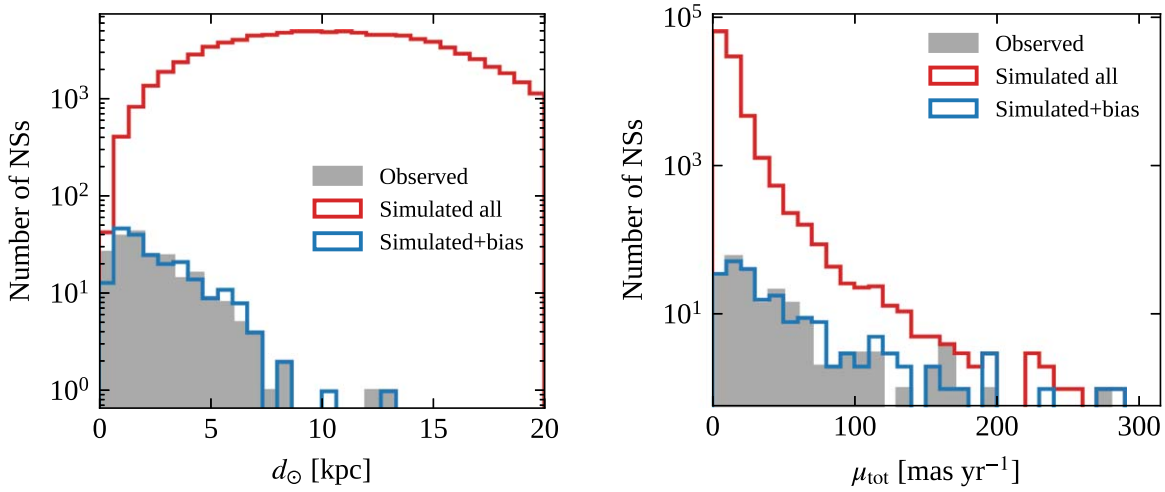

Figure 11. Distribution of heliocentric distances $d_\odot$ (left panel) and proper-motion magnitudes $\mu_{\rm tot}$ (right panel) for the 216 neutron stars with measured proper motion (gray histograms). For comparison, we also show the distances and proper-motion magnitudes for 216 simulated neutron stars (blue histograms) randomly sampled from a mock population generated with the fiducial parameters $\sigma_{\rm k} = 265$ km s^{-1} and $h_{\rm c} = 0.18$ kpc (red histograms). For the weighted sampling, we use the weight function $f(d_\odot) = d_\odot^{-1}\exp(-0.5 d_\odot)$.

measured proper motions is incomplete in distance and subject to selection biases. By looking at the right panel of Figure 11, we also note that a selection bias on distance is also reflected in the distribution of the total proper-motion magnitudes (gray histogram), computed as $\mu_{\rm tot} \equiv \sqrt{\mu^2_{\rm R.A.} + \mu^2_{\rm decl.}}$. Indeed, since the nearest stars are also characterized by larger angular proper motions, there is a higher probability of detecting stars with high proper-motion.

In this first empirical approach, instead of attempting to identify these underlying biases (which will be the subject of a subsequent paper in preparation), we follow a more agnostic approach to introduce a comparable selection effect in our simulated populations. Specifically, we use a weighted random-sampling routine to select n pulsars from our mock populations, where each simulated star is assigned a weight according to a function $f(d_\odot)$ of its heliocentric distance. This weight function has to assign larger weights to closer neutron stars in order to ensure their higher detection probabilities and has to be chosen such that we recover the observed distance distribution with sufficient accuracy. To find $f(d_\odot)$ we simulate a mock population with the fiducial values of the kick velocity and scale height, that is $\sigma_{\rm k} = 265\,{\rm km\,s^{-1}}$ and $h_{\rm c} = 0.18$ kpc, respectively. After using a given $f(d_\odot)$ to weight the simulated neutron stars we sample 216 mock stars and compare their distance and proper-motion distributions (shown as blue histograms in Figure 11) with those of the observed sample by performing two-sample Kolmogorov–Smirnov (KS) tests. After testing various functional forms we find that $f(d_\odot) = d_\odot^{-1}\exp(-0.5 d_\odot)$ is able to reproduce the observed distributions with a good level of accuracy. More precisely, for this choice of $f(d_\odot)$, the KS tests

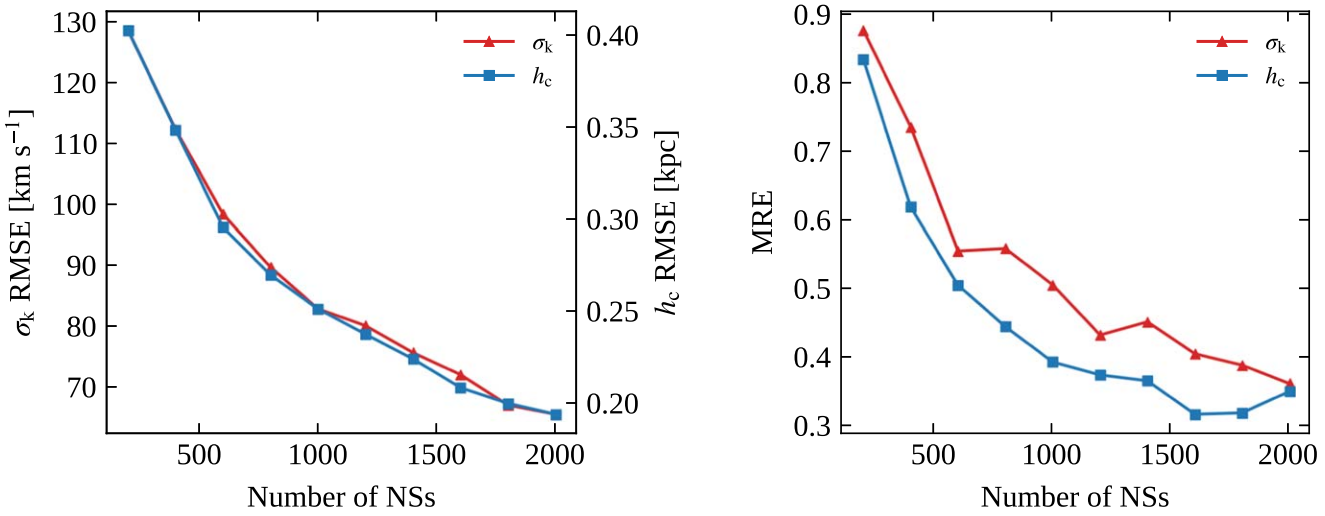

Figure 12. Trend of the RMSE (left panel) and MRE (right panel) uncertainties on the prediction of the two parameters σ_k (blue) and h_c (red) as a function of the number of neutron stars n in the resampled simulations in the validation sets. To train and validate the CNN we use the 16,384 simulated populations from simulation run S3 (with an 80%/20% training/validation split), where both σ_k and h_c are varied, and which are resampled with an increasing number of stars n according to the weight function $f(d_\odot) = d_\odot^{-1}\exp(-0.5d_\odot)$.

performed over 1000 distinct comparisons give average p-values of $\sim$0.3 and $\sim$0.6 for the distance and proper-motion comparisons, respectively. This means that at 95% confidence level, we cannot reject the null hypothesis that the simulated and observed samples are drawn from the same underlying distribution. We have also verified that for this weight function, the KS tests always provide p-values $>$0.05 when comparing the observed sample with the simulated populations for reasonable values of σ_k and h_c. Only in the cases where σ_k and h_c assume extreme values (near the edges of their respective ranges) might the p-values drop below 0.05. However, these cases are associated with simulations with extreme initial conditions that are unlikely to reproduce the observations. For our basic experiment, we further make the simplified assumptions that $f(d_\odot)$ emulates all selection biases and that it is the same for every number n of sampled neutron stars. We stress that for the purpose of this initial analysis we do not aim to accurately constrain the selection function that affects the observed population of neutron stars. Instead we study how the introduction of realistic selection biases will alter the predictive power of our machine learning framework. Although one might intuitively attribute the exponential factor in $f(d_\odot)$ to scattering in the interstellar medium at large distances, the underlying nature and the precise form of the true selection function are certainly more complicated. We expect it to encompass a series of effects due to the physics of the interstellar medium and the pulsar emission itself, as well as selection effects of pulsar surveys and pulsar searches. We postpone a more accurate study disentangling the different effects that contribute to $f(d_\odot)$ to future work.

We then analyze how the predictive power of the CNN evolves as a function of the number n of neutron stars, sampled with the above weight function $f(d_\odot)$. To do so, we vary n from 200 to 2000 in steps of 200, and in each case resample the 16,384 simulated populations from run S3, where both σ_k and h_c are varied. After applying an 80%/20% training-validation split, we retrain the CNN on each of the downsampled simulations. As before, we use the three-channel ICRS input maps (i.e., information on density plus proper motion) but instead opt for a resolution of 32×16 bins to accommodate the smaller number of stars represented in our maps. We have verified that a higher resolution of 128×64 bins does not affect the training results significantly, but it slows down the training process; we therefore choose the lower resolution. We use the same training hyperparameters as in Section 4.1.2, that is an initial learning rate of 10^{-4}, a batch size of 64, and an early stop at 128 epochs. Once trained for each n value, we apply the CNN to the validation sets and compute the RMSE and mean MRE for the σ_k and h_c predictions as a function of n.

In the left panel of Figure 12, we show how the RMSE uncertainties for the predictions of the two parameters σ_k (blue) and h_c (red) diminish with increasing number of neutron stars n sampled from the simulations. We observe that the two curves (with the appropriate rescaling) follow very similar trends. On the right, we show instead how the MREs evolve with n, indicating how the precision of the two-parameter prediction improves with the number of detected neutron stars. This plot shows that, under the assumptions that selection effects are unaltered and the underlying kick-velocity and birth-height distributions have Maxwellian and exponential shapes, respectively, our trained CNN is able to predict σ_k and h_c with a relative error of $\sim$0.35 for a sample of 2000 stars.

These results highlight that the number of neutron stars plays a crucial role in the level of accuracy that the CNN can reach. As expected, a larger number of stars provides more information, which allows the CNN to pinpoint differences between populations evolved from different initial conditions, and also when selection effects are introduced. Future observational efforts aimed at detecting and characterizing new pulsars will thus play an important role in constraining the birth properties of neutron stars. Specifically, the advent of the Square Kilometre Array (SKA) will represent an important step forward in this direction. Due to its high sensitivity (a factor of 10 better than other radio telescopes) and its long baseline (up to 3000 km), SKA has the potential to increase the number of discovered pulsars by a factor of 10 (Smits et al. 2009, 2011). This will allow more precise timing and astrometric measurements of pulsar positions as well as distances and proper motions. A larger and more precise data set could also help to better constrain the shape of the kick-velocity distribution and differentiate between models that try to explain the origin of the natal kicks (Tauris et al. 2015).

6. Summary

In this work we have analyzed the possibility of using ML techniques to reconstruct the dynamical birth properties of an evolved population of isolated neutron stars. For this purpose, we developed a simplistic population-synthesis pipeline to simulate the dynamical evolution of Galactic neutron stars and subsequently used these simulations to train and validate two different neural networks. We specifically focused on their ability to predict two parameters that strongly impact on the phenomenology of the evolved population: the dispersion $\sigma_{\rm k}$ of a Maxwell kick-velocity distribution and the scale height $h_{\rm c}$ of an exponential distribution for the Galactic birth heights. This was achieved by providing the networks with two-dimensional maps of stellar density and velocity in galactocentric and equatorial ICRS coordinate frames. We found that a CNN is able to estimate the physical parameters with high accuracy when multiple input channels, i.e., position and velocity information, are provided. In particular, when simultaneously predicting $\sigma_{\rm k}$ and $h_{\rm c}$ from ICRS maps, the network is able to reach absolute uncertainties lower than 10 km s^{-1} and 0.05 kpc, respectively, corresponding to a relative error of around 10^{-2} for both parameters. Although obtained under simplified assumptions, our feasibility study—the main focus of this paper—has thus demonstrated that ML techniques are indeed suitable to infer information about the pulsar population. Our phenomenological analysis incorporating proper-motion measurements (an attempt at including observational biases in an agnostic way) has further highlighted that increasing the sample of known pulsars and accurately measuring their current characteristics with future telescopes is crucial to tightly constraining the birth properties of the neutron stars in the Milky Way. In particular, our trained CNN is able to predict $\sigma_{\rm k}$ and $h_{\rm c}$ with a relative error of $\sim$0.35 for a sample of 2000 pulsars with measured proper motions.

We also demonstrated that one of the main factors limiting the accuracy of the CNN's predictions in our setup is the degeneracy between $\sigma_{\rm k}$ and $h_{\rm c}$; as they both affect the evolved populations in a similar way, the network struggles to disentangle their effects. This limitation is a direct consequence of our simplified population-synthesis framework. In future works, we will go beyond modeling the dynamical evolution and focus on incorporating additional physics such as magnetothermal and spin-period evolution. We will further model their emission in different electromagnetic bands and study corresponding detectability limits by addressing selection effects as well as observational survey biases. Such additional input information could potentially break the degeneracy between the kick-velocity and Galactic height distributions and provide more accurate model constraints on $\sigma_{\rm k}$ and $h_{\rm c}$ as well as other input parameters.

The ultimate goal will be to use multiwavelength observations of the Galactic neutron star population and take advantage of ML, combined with population synthesis, to recover their birth properties, such as the distribution of natal kick velocity, spin period, or magnetic field. The potential to learn complex patterns from multidimensional data and the power of generalizing to unseen data sets make ML algorithms a powerful tool to tackle such multiparameter optimization studies.

We would like to thank Alice Borghese and Francesco Coti Zelati for their help with the collection of the observed pulsar proper motions, Daniele Viganö for providing comments on the manuscript, and Alessandro Patruno for helpful conversations related to this paper. M.R., V.G., A.G., and N.R. acknowledge support from the H2020 ERC Consolidator Grant "MAGNESIA"' under grant agreement No. 817661 (PI: Rea), and grants SGR2017-1383 and PGC2018-095512-B-I00. J.A.P. acknowledges support by the Generalitat Valenciana grant PROMETEO/2019/071 and by AEI grant PGC2018-095984-B-I00. This work has also been partially supported by the PHAROS COST Action (CA16214). The data production, processing, and analysis tools for this paper have been developed, implemented, and operated in collaboration with the Port d'Informació Científica (PIC) data center. PIC is maintained through a consortium of the Institut de Física d'Altes Energies (IFAE) and the Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (Ciemat). We particularly thank Christian Neissner, Ricard Cruz, Carles Acosta, Gonzalo Merino, and Pau Tallada for their support at PIC.

Software

Astropy (Astropy Collaboration et al. 2013, 2018), Hydra (Yadan 2019), IPython (Perez & Granger 2007), Matplotlib (Hunter 2007), Numba (Lam et al. 2015), NumPy (Oliphant 2006; van der Walt et al. 2011; Harris et al. 2020), Pandas (McKinney 2010), PyTorch (Paszke et al. 2019), and SciPy (Jones et al. 2001; Virtanen et al. 2020).

Appendix A
Hardware and Software

Our test machine features an Intel(R) Xeon(R) Gold 6230R CPU at 2.10 GHz with a single NVIDIA GTX 2080 Ti GPU, 16 GB RAM, and SSD drives. The system is running CentOS Linux release 7.8.2003 (Core) with PyTorch 1.2.0, CUDA toolkit 10.0.130, and GPU driver 455.32.00.

Appendix B
Timing Tests

B.1. Timing for Single-parameter Predictions

We report here the timing benchmarks for the MLP and CNN during the single-parameter training experiments discussed in Section 4.1.1. We run our experiments on the test machine and individually record the forward pass time (the time needed to go through the samples in a batch and compute a prediction) and the backward pass time (the time to compute all the gradients and perform a single optimization step) as a function of the batch size and resolution. Our benchmarks for the training data sets from simulation run S1 using the four training configurations T1, T2, T3, and T4 are shown in Figures 13 (MLP) and 14 (CNN).

The timing benchmark shows that the MLP is slightly faster in performing an optimization step. This is expected because it has fewer trainable parameters than the more complex CNN. We can also see that the forward and backward pass times per sample decrease with increasing batch size for both the MLP and CNN. For a larger batch size, several input samples are transferred from the CPU to the GPU in one step, reducing the overall number of calls between the two. Thus, on average, the processing time for an individual sample reduces when the batch size is increased. Moreover, a higher resolution generally implies an increase in computational cost, albeit being more

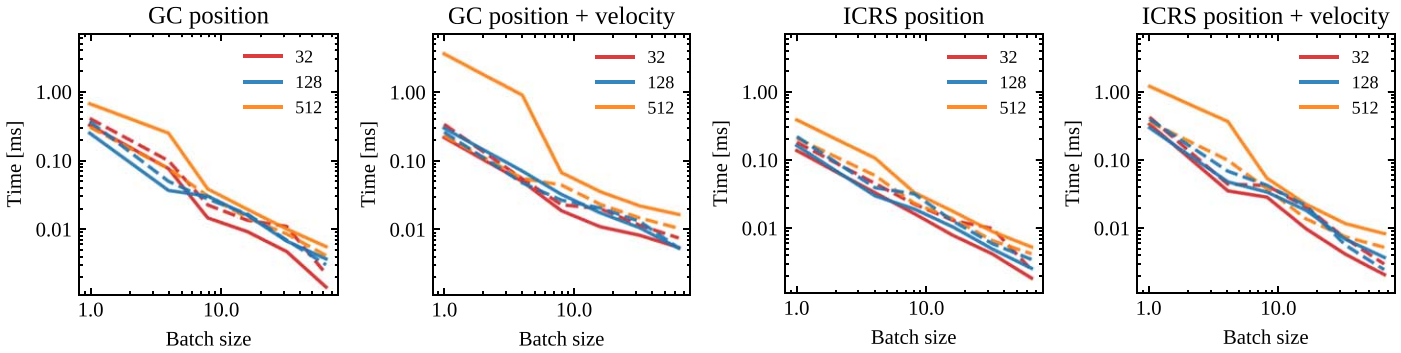

Figure 13. MLP forward (solid lines) and backward (dashed lines) pass times per sample in milliseconds for the training process on the single parameter σ_k of the Maxwell kick-velocity distribution, as a function of the batch size and the resolution (red, blue, and orange curves for 32, 128, and 512 respectively) using the four different input configurations T1 (GC position), T2 (GC position + velocity), T3 (ICRS position), and T4 (ICRS position + velocity).

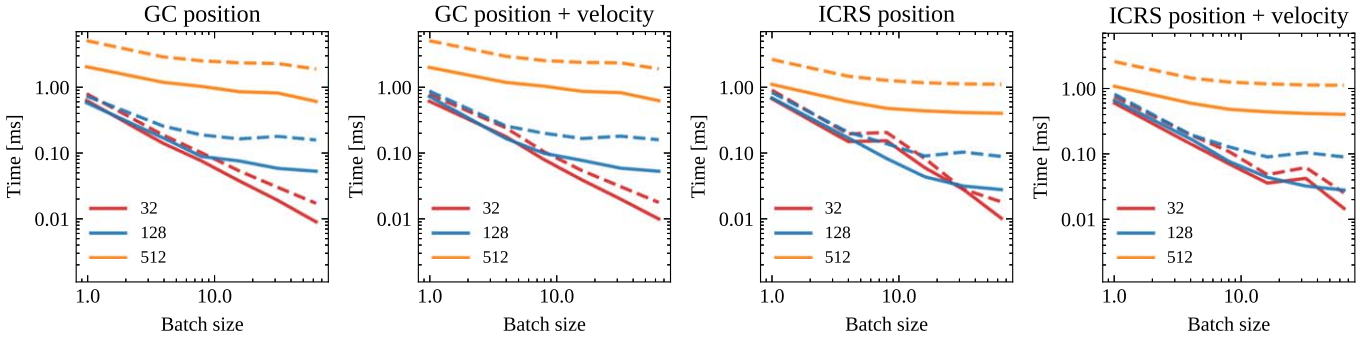

Figure 14. CNN forward (solid lines) and backward (dashed lines) pass times per sample in milliseconds for the training process on the single parameter σ_k of the Maxwell kick-velocity distribution, as a function of the batch size and the resolution (red, blue, and orange curves for 32, 128, and 512 respectively) using the four different input configurations T1 (GC position), T2 (GC position + velocity), T3 (ICRS position), and T4 (ICRS position + velocity).

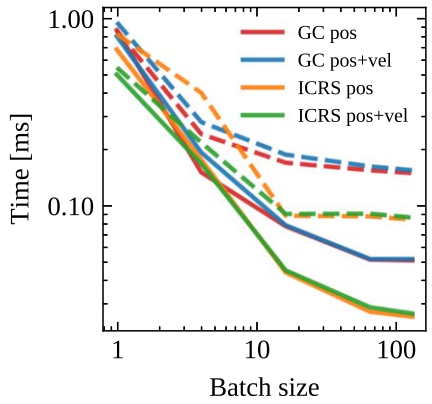

Figure 15. CNN forward (solid line) and backward (dashed line) pass times per sample for the two-parameter experiments as a function of the batch size for the four different input configurations T1 (GC position), T2 (GC position + velocity), T3 (ICRS position), and T4 (ICRS position + velocity).

pronounced in the case of the CNN than the MLP. The number of input channels itself has very little effect on our timings. Finally, we note that ICRS maps are slightly faster to process (in particular for the higher resolutions), because they are smaller than the galactocentric maps (their bins are half as high).

B.2. Timing for Two-parameter Predictions

Following the results for the single-parameter experiments, we restrict our two-parameter predictions to the CNN model only and fix the resolution of the input maps to 128. The results of our timing benchmarks using the training data sets from simulation run S3 with the four configurations T1, T2, T3, and T4 are shown in Figure 15. We again report the timings for the forward and backward passes per sample as a function of the batch size and the type of input channels provided. As for the single-parameter case, we conclude that using ICRS maps ensures the shortest forward and backward pass times.

Appendix C
Neutron Stars with Measured Proper Motion

In Table 5, we report the properties of 417 neutron stars with measured proper motions in R.A. and decl. Data for these neutron stars are primarily collected from the ATNF catalog[7] (Manchester et al. 2005). In some cases, updated estimates are available and those values quoted and the corresponding references specified. Note that in those cases where multiple proper-motion estimates are available, we choose the ones with the lowest absolute error. The columns report in order: (i) the object name based on J2000 coordinates; (ii) the R.A. in hour angle and (iii) decl. in degrees with the last-digit uncertainty given in parentheses; the proper motion in (iv) R.A. and (v) decl. in milliarcseconds per year with corresponding uncertainties; (vi) the parallax measured in milliarcseconds with uncertainty where available; (vii) the position epoch in modified Julian days; (viii) the spin period in seconds; (ix) the spin-period derivative in seconds per second; (x) the dispersion measure in [pc cm^{-3}] with the last-digit uncertainty given in parentheses; (xi) the heliocentric distance derived from the DM using the YMW16 free-electron density model (for

[7] https://www.atnf.csiro.au/research/pulsar/psrcat/

some objects the DM exceeds the maximum Galactic DM allowed by the YMW16 model, which assigns a default value of 25 kpc; when available, we quote other distance estimates; * indicates a distance derived from other techniques, especially for X-ray and gamma-ray sources, which have no measured DM); (xii) the classification of the object, i.e., radio pulsar (PSR), binary pulsar (binary PSR), gamma-/X-ray pulsar (Gamma-/X-ray PSR), magnetar (MAG), X-ray-dim isolated neutron star (XDINS); if the object is associated with a globular cluster (GC) or the Small Magellanic Cloud (SMC) this is reported in brackets; and (xiii) the reference for the proper-motion measurements, indicated only if different from the ATNF catalog, i.e., [1] Motch et al. (2009), [2] Eisenbeiss et al. (2010), [3] Walter et al. (2010), [4] Stovall et al. (2014), [5] Jennings et al. (2018), [6] Perera et al. (2019), [7] Dang et al. (2020), [8] Danilenko et al. (2020).

ORCID iDs

M. Ronchi https://orcid.org/0000-0003-2781-9107
V. Graber https://orcid.org/0000-0002-6558-1681
A. Garcia-Garcia https://orcid.org/0000-0002-9575-6403
N. Rea https://orcid.org/0000-0003-2177-6388
J. A. Pons https://orcid.org/0000-0003-1018-8126

References

Agarwal, D., Aggarwal, K., Burke-Spolaor, S., Lorimer, D. R., & Garver-Daniels, N. 2020, MNRAS, 497, 1661
Allen, G., Andreoni, I., Bachelet, E., et al. 2019, arXiv:1902.00522
Arzoumanian, Z., Chernoff, D. F., & Cordes, J. M. 2002, ApJ, 568, 289
Astropy Collaboration, Price-Whelan, A. M., Sipőcz, B. M., et al. 2018, AJ, 156, 123
Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., et al. 2013, A&A, 558, A33
Balakrishnan, V., Champion, D., & Barr, E. 2021, MNRAS, 505, 1180
Ball, N. M., & Brunner, R. J. 2010, IJMPD, 19, 1049
Balucinska-Church, M., & McCammon, D. 1992, ApJ, 400, 699
Baron, D. 2019, arXiv:1904.07248
Bates, S. D., Lorimer, D. R., Rane, A., & Swiggum, J. 2014, MNRAS, 439, 2893
Bethapudi, S., & Desai, S. 2018, A&C, 23, 15
Bisnovatyi-Kogan, G. S. 1993, A&AT, 3, 287
Bovy, J. 2015, ApJS, 216, 29
Brisken, W. F., Benson, J. M., Goss, W. M., & Thorsett, S. E. 2002, ApJ, 571, 906
Brisken, W. F., Fruchter, A. S., Goss, W. M., Herrnstein, R. M., & Thorsett, S. E. 2003, AJ, 126, 3090
Cabero, M., Mahabal, A., & McIver, J. 2020, ApJL, 904, L9
Carlberg, R. G., & Innanen, K. A. 1987, AJ, 94, 666
Chatterjee, S., Vlemmings, W. H. T., Brisken, W. F., et al. 2005, ApJL, 630, L61
Chen, B. Q., Huang, Y., Hou, L. G., et al. 2019, MNRAS, 487, 1400
Cieślar, M., Bulik, T., & Osłowski, S. 2020, MNRAS, 492, 4043
Coenen, T., van Leeuwen, J., Hessels, J. W. T., et al. 2014, A&A, 570, A60
Cordes, J. M., & Lazio, T. J. W. 2002, arXiv:astro-ph/0207156
Dang, S. J., Yuan, J. P., Manchester, R. N., et al. 2020, ApJ, 896, 140
Danilenko, A., Karpova, A., Ofengeim, D., Shibanov, Y., & Zyuzin, D. 2020, MNRAS, 493, 1874
Deller, A. T., Goss, W. M., Brisken, W. F., et al. 2019, ApJ, 875, 100
Deller, A. T., Tingay, S. J., Bailes, M., & Reynolds, J. E. 2009, ApJ, 701, 1243
Dewey, R. J., & Cordes, J. M. 1987, ApJ, 321, 780
Eisenbeiss, T., Ginski, C., Hohle, M. M., et al. 2010, AN, 331, 243
Faucher-Giguère, C.-A., & Kaspi, V. M. 2006, ApJ, 643, 332
Fluke, C. J., & Jacobs, C. 2020, WIREs Data Mining and Knowledge Discovery, 10, e1349
Fryer, C. L., & Kusenko, A. 2006, ApJS, 163, 335
Gerosa, D., Pratten, G., & Vecchio, A. 2020, PhRvD, 102, 103020
Giacobbo, N., & Mapelli, M. 2020, ApJ, 891, 141
Gonthier, P. L., Story, S. A., Clow, B. D., & Harding, A. K. 2007, Ap&SS, 309, 245
Good, D. C., Andersen, B. C., Chawla, P., et al. 2020, arXiv:2012.02320
Gullón, M., Miralles, J. A., Viganò, D., & Pons, J. A. 2014, MNRAS, 443, 1891
Hansen, B. M. S., & Phinney, E. S. 1997, MNRAS, 291, 569
Harris, C. R., Jarrod Millman, K., van der Walt, S. J., et al. 2020, Natur, 585, 357
He, C., Ng, C. Y., & Kaspi, V. M. 2013, ApJ, 768, 64
He, K., Zhang, X., Ren, S., & Sun, J. 2015, arXiv:1502.01852
Hernquist, L. 1990, ApJ, 356, 359
Hobbs, G., Lorimer, D. R., Lyne, A. G., & Kramer, M. 2005, MNRAS, 360, 974
Hobbs, G., Lyne, A. G., Kramer, M., Martin, C. E., & Jordan, C. 2004, MNRAS, 353, 1311
Hui, C. Y., & Becker, W. 2006, A&A, 457, L33
Hunter, J. D. 2007, CSE, 9, 90
Igoshev, A. P. 2020, MNRAS, 494, 3663
Janka, H.-T. 2017, ApJ, 837, 84
Jennings, R. J., Kaplan, D. L., Chatterjee, S., Cordes, J. M., & Deller, A. T. 2018, ApJ, 864, 26
Johnston, S., Taylor, R., Bailes, M., et al. 2008, ExA, 22, 151
Jones, E., Oliphant, T. E., Peterson, P., et al. 2001, SciPy: Open Source Scientific Tools for Python, http://www.scipy.org/
Kaspi, V. M., Roberts, M. E., Vasisht, G., et al. 2001, ApJ, 560, 371
Kelley, H. J. 1960, ARS J., 30, 947
Kiel, P. D., & Hurley, J. R. 2009, MNRAS, 395, 2326
Kiel, P. D., Hurley, J. R., Bailes, M., & Murray, J. R. 2008, MNRAS, 388, 393
Kingma, D. P., & Ba, J. 2014, arXiv:1412.6980
Lai, D., Chernoff, D. F., & Cordes, J. M. 2001, ApJ, 549, 1111
Lam, S. K., Pitrou, A., & Seibert, S. 2015, Proc. Second Workshop on the LLVM Compiler Infrastructure in HPC, LLVM '15 (New York: Association for Computing Machinery), https://doi.org/10.1145/2833157.2833162
Lentati, L., Shannon, R. M., Coles, W. A., et al. 2016, MNRAS, 458, 2161
Levin, L., Bailes, M., Barsdell, B. R., et al. 2013, MNRAS, 434, 1387
Li, C., Zhao, G., Jia, Y., et al. 2019, ApJ, 871, 208
Lin, H., Li, X., & Luo, Z. 2020, MNRAS, 493, 1842
Liu, J., Yan, Z., Shen, Z.-Q., et al. 2020, PASJ, 72, 70
Lorimer, D. R., Faulkner, A. J., Lyne, A. G., et al. 2006, MNRAS, 372, 777
Lorimer, D. R., & Kramer, M. 2004, Handbook of Pulsar Astronomy (Cambridge: Cambridge Univ. Press)
Manchester, R. N., Hobbs, G. B., Teoh, A., & Hobbs, M. 2005, AJ, 129, 1993
Manchester, R. N., Lyne, A. G., Camilo, F., et al. 2001, MNRAS, 328, 17
Mandel, I., & Müller, B. 2020, MNRAS, 499, 3214
Marchetti, T., Rossi, E. M., & Brown, A. G. A. 2019, MNRAS, 490, 157
Matthews, A. M., Nice, D. J., Fonseca, E., et al. 2016, ApJ, 818, 92
McKinney, W. 2010, in Proc. 9th Python in Science Conference, ed. S. van der Walt & J. Millman, 51–6, http://conference.scipy.org/proceedings/scipy2010/mckinney.html
Melrose, D. B. 2017, RvMPP, 1, 5
Miyamoto, M., & Nagai, R. 1975, PASJ, 27, 533
Morawski, F., & Bejger, M. 2020, A&A, 642, A78
Motch, C., Pires, A. M., Haberl, F., Schwope, A., & Zavlin, V. E. 2009, A&A, 497, 423
Nagakura, H., Sumiyoshi, K., & Yamada, S. 2019, ApJL, 880, L28
Narayan, R., & Ostriker, J. P. 1990, ApJ, 352, 222
Navarro, J. F., Frenk, C. S., & White, S. D. M. 1996, ApJ, 462, 563
Oliphant, T. E. 2006, A Guide to NumPy (USA: Trelgol Publishing), https://web.mit.edu/dvp/Public/numpybook.pdf
Osłowski, S., Bulik, T., Gondek-Rosińska, D., & Belczyński, K. 2011, MNRAS, 413, 461
Parthasarathy, A., Shannon, R. M., Johnston, S., et al. 2019, MNRAS, 489, 3810
Paszke, A., Gross, S., Massa, F., et al. 2019, arXiv:1912.01703
Pavan, L., Bordas, P., Pühlhofer, G., et al. 2014, A&A, 562, A122
Perera, B. B. P., DeCesar, M. E., Demorest, P. B., et al. 2019, MNRAS, 490, 4666
Perez, F., & Granger, B. E. 2007, CSE, 9, 21
Pichardo, B., Moreno, E., Allen, C., et al. 2012, AJ, 143, 73
Podsiadlowski, P., Langer, N., Poelarends, A. J. T., et al. 2004, ApJ, 612, 1044
Polido, P., Jablonski, F., & Lépine, J. R. D. 2013, ApJ, 778, 32
Rawat, W., & Wang, Z. 2017, Neural Computation, 29, 1
Reid, M. J., Menten, K. M., Brunthaler, A., et al. 2019, ApJ, 885, 131
Rozwadowska, K., Vissani, F., & Cappellaro, E. 2021, NewA, 83, 101498
Ruder, S. 2017, arXiv:1609.04747
Sartore, N., Ripamonti, E., Treves, A., & Turolla, R. 2010, A&A, 510, A23
Schwab, J., Podsiadlowski, P., & Rappaport, S. 2010, ApJ, 719, 722

Shklovskii, I. S. 1970, SvA, 13, 562
Skliris, V., Norman, M. R. K., & Sutton, P. J. 2020, arXiv:2009.14611
Skowron, D. M., Skowron, J., Mróz, P., et al. 2019, Sci, 365, 478
Smits, R., Kramer, M., Stappers, B., et al. 2009, A&A, 493, 1161
Smits, R., Tingay, S. J., Wex, N., Kramer, M., & Stappers, B. 2011, A&A, 528, A108
Spitzer, L. J. 1942, ApJ, 95, 329
Stovall, K., Lynch, R. S., Ransom, S. M., et al. 2014, ApJ, 791, 67
Tamborra, I., Hanke, F., Janka, H.-T., et al. 2014, ApJ, 792, 96
Tauris, T. M., Kaspi, V. M., Breton, R. P., et al. 2015, in Proc. of Advancing Astrophysics with the Square Kilometre Array (AASKA14), 39, https://doi.org/10.22323/1.215.0039
Tauris, T. M., Langer, N., Moriya, T. J., et al. 2013, ApJL, 778, L23
Tauris, T. M., & Manchester, R. N. 1998, MNRAS, 298, 625
Taylor, J. H., & Cordes, J. M. 1993, ApJ, 411, 674
Vallée, J. P. 2017, AstRv, 13, 113
van der Walt, S., Colbert, S. C., & Varoquaux, G. 2011, CSE, 13, 22
Verbunt, F., Igoshev, A., & Cator, E. 2017, A&A, 608, A57
Viganò, D., Rea, N., Pons, J. A., et al. 2013, MNRAS, 434, 123
Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, NatMe, 17, 261
Wainscoat, R. J., Cohen, M., Volk, K., Walker, H. J., & Schwartz, D. E. 1992, ApJS, 83, 111
Walter, F. M., Eisenbeiß, T., Lattimer, J. M., et al. 2010, ApJ, 724, 669
Wang, J. B., Coles, W. A., Hobbs, G., et al. 2017, MNRAS, 469, 425
Wei, W., & Huerta, E. A. 2020, arXiv:2010.09751
Wong, K. W. K., & Gerosa, D. 2019, PhRvD, 100, 083015
Yadan, O. 2019, Hydra—A Framework for Elegantly Configuring Complex Applications, Github, https://github.com/facebookresearch/hydra
Yao, J. M., Manchester, R. N., & Wang, N. 2017, ApJ, 835, 29
Yusifov, I., & Küçük, I. 2004, A&A, 422, 545