Inverting cosmic ray propagation by Convolutional Neural Networks

Yue-Lin Sming Tsai\textsuperscript{a}, Yi-Lun Chung\textsuperscript{b}, Qiang Yuan\textsuperscript{a,c,d}, Kingman Cheung\textsuperscript{b,e}

\textsuperscript{a} Key Laboratory of Dark Matter and Space Astronomy, Purple Mountain Observatory, Chinese Academy of Sciences, Nanjing 210023, China
\textsuperscript{b} Department of Physics, National Tsing Hua University, Hsinchu 300, Taiwan
\textsuperscript{c} School of Astronomy and Space Science, University of Science and Technology of China, Hefei 230026, China
\textsuperscript{d} Center for High Energy Physics, Peking University, Beijing 100871, China
\textsuperscript{e} Division of Quantum Phases and Devices, School of Physics, Konkuk University, Seoul 143-701, Republic of Korea

Abstract

We propose a machine learning method to investigate the propagation of cosmic rays based on the precisely measured spectra of the primary and secondary cosmic ray nuclei of Li, Be, B, C, and O from AMS-02, ACE, and Voyager-1. We train two convolutional neural networks. One network learns how to infer propagation and source parameters from the energy spectra of cosmic rays, and the other network, which is similar to the former, has the flexibility to learn from the data with added artificial fluctuations. Together with the simulated data generated by \textsc{GALPROP}, we find that both networks can properly invert the propagation process and infer the propagation and source parameters reasonably well. This approach can be much more efficient than the traditional Markov chain Monte Carlo fitting method for deriving the propagation parameters if users choose to update confidence intervals with new experimental data. Both of the trained networks are available at \url{https://github.com/alan200276/CR_ML}.

\textsuperscript{1}smingtsai@pmo.ac.cn
\textsuperscript{2}s107022801@m107.nthu.edu.tw
\textsuperscript{3}yuanq@pmo.ac.cn
\textsuperscript{4}cheung@phys.nthu.edu.tw
1 Introduction

The source spectrum and propagation mechanism of cosmic rays (CRs) remain largely unknown. With the exception of the Voyager spacecraft, which may have travelled to the outside of the heliosphere \(^1\), all other CR data are measured within Earth’s neighbourhood. Based on observational data, the source and propagation parameters of CRs can degenerate each other, which barely renders a direct measure of such parameters. The traditional parameter estimation procedure involves first building a propagation model of CRs, parameterizing the source spectra and propagation processes, and then fitting the data (e.g., \(^2\)). However, traditional procedures are time-consuming from a computational perspective, especially when numerical simulations of CR propagation using simulators such as \textit{GALPROP} \(^3\) or \textit{DRAGON} \(^2\) is employed. Furthermore, in relevant studies with similar issues, such as the search for dark matter annihilation or decay signals, the uncertainty computation of the background models that are often ignored, and only some benchmark settings are investigated due to the difficulties in obtaining full uncertainty bands (see \(^7\), who attempted studies to include the uncertainties described above). Increasing the speed of a large number of repetitive computations is a great challenge for CR physics studies and related particle physics problems.

The general idea of global fitting is to project the probability distribution of the measured CR fluxes to the model parameter space \(^4\). One can use various sampling tools, such as the Markov chain Monte Carlo (MCMC) method or nest sampling, to explore the parameter space. The chi-square $\chi^2$ statistic is built based on fluxes instead of the model parameter space. Hence, too many samples may be used in a trial and error approach in an unwanted region. Moreover, once the likelihood distribution is changed, it is better to restart the whole analysis to obtain good coverage. If there are only mild changes in the likelihood distribution, there are several approaches to efficiently and quickly complete new scans. For example, the old posterior distribution can be used as an updated prior distribution for the new scans or the covariance matrix generated by old scans can be adopted as the initial prior distribution. However, these approaches still require performing the parameter scan again.

To avoid these additional scanning procedures, our goal is to obtain the corresponding propagation and source parameters directly by using spectra of interest from experimental data. The likelihood distribution in CR spectra can be directly taken from experimental measurements without exploring the CR model parameter space. Therefore, we can use the inverse function of the CR propagation equation to calculate the corresponding propagation and source parameters directly from the desired CR spectra. Nevertheless, CR propagation is a complicated partial differential equation, and it is not easy to obtain its inverse function. In this work, we show that machine learning (ML) is a powerful technique that can be used

\(^1\)http://galprop.stanford.edu/
\(^2\)http://www.dragonproject.org/Home.html
\(^3\)The probability distribution can be described by the chi-square $\chi^2$ or likelihood statistics for the Frequentist interpretation, while the Bayesian interpretation is based on the posterior distribution. Note that the Gaussian likelihood probability density can be written as $\sim \exp \left( -\frac{\chi^2}{2} \right)$. 

1
to undertake this task by interpolating a large data set, especially in a multidimensional parameter space.

We can replace the propagation simulations with the ML interpolation function as long as a massive dataset is produced before learning [9]. Therefore, we expect the learning networks to address the time-consuming tasks of global fitting and generating massive simulations. Several groups have made strong efforts in searching for new physics beyond the Standard Model, see for example, Refs. [20–23]. The purpose of this work is to convert experimental data to model parameter distributions directly using ML techniques.

In recent decades, ML methods such as support vector machines [10,11] and boosted decision trees (BDTs) [12–14] show good performance in classification and regression problems. In high-energy particle experiments, BDTs are more popular than support vector models because they are an interpretable model. The performance of BDTs can be examined from input variable distributions such as invariant mass and transverse momentum [15]. More recently, neural network learning has become a leading tool in every field because a neural network can provide a more flexible variety of architectures to obtain hidden information from data. Dense neural networks (DNNs) use several layers consisting of many neurons to learn the principle components in the data. Convolutional neural networks (CNNs) are better than DNNs in handling the correlated information of pixelwise or multiple one-dimensional data. In addition, graph neural networks [16] can build classification rules or regression predictions from point data without a fixed shape. Furthermore, recurrent neural networks and long short-term memory networks [17] aim to cope with time sequence problems. A recently developed architecture called Transformer [18], shows powerful performance in many problems, for instance, the precise identification of top jets [19]. In this work, we adopt a CNN architecture to invert the CR propagation by taking CR spectra and model parameters as inputs and outputs of the network, respectively. A CNN uses a convolution operation and multiple networks with a non-polynomial activation function and is thus a well-designed tool to address not only the correlations between CR spectra (different elements and different energy bins) but also the high-dimensional inverse-propagation problem.

In this work, we build a CNN network to determine a multidimensional inverse function of CR propagation. We first generate simulated data for ML using MCMC scans [4,24]. To demonstrate the capability of updatable measurements, we enlarge the uncertainties of the current CR data by a factor of $\sqrt{10}$ in the MCMC scans. To a certain degree, we are deprived of understanding with respect to new data. We assume that the parameter space allowed by the current CR data, with such enlarged uncertainties, should cover the new measurements, which have smaller error bars.

After building the proposed network, the inverse function of CR propagation (CNN version) allows us to quickly guess the experimental data distribution in the CR model parameter space without performing a new MCMC scan. As a comparison, we introduce two networks: one network learns from the spectra directly, while the other network learns from this data but with some artificial fluctuations. These networks can predict the allowed model parameter space using actual experimental data (with original uncertainties).
The primary goal of this work is to build an inverse function using the machine learning approach, such that the inverse function can readily invert observational datasets into the corresponding model parameter space. The advantage of this method is emphasized as follows. When a new dataset is available or an existing dataset is updated, the proposed networks do not need to be retrained and the whole analysis does not need to be repeated. The trained machines can easily guess the high probability region of the parameter space.

This paper is organized as follows. In Sec. 2 we review the framework of CR propagation. The model parameters used in this work are defined and explained. Next, we briefly introduce the CNN in Sec. 3. Subsequently, we show the architecture of the two CNNs in Sec. 4. The generation of mock data for training and pseudo-data for testing are described in detail. Then, we show the usage of the networks to invert the propagation of CRs. In Sec. 5, the performance of the two networks and a comparison with the traditional global fitting method are presented. Finally, we summarize our work in Sec. 6.

2 Cosmic Ray propagation

Charged CRs diffuse in the random magnetic field of the Milky Way and interact with the interstellar medium (ISM). During propagation, CR particles acquire or lose energy and fragment into secondary particles due to interactions with the ISM and magnetic fields. The general form of the diffusion equation of CRs in the Milky Way with the source function $Q(x, p)$ is

$$\frac{\partial \psi}{\partial t} = Q(x, p) + \nabla \cdot (D_{xx} \nabla \psi - V_c \psi) + \frac{\partial}{\partial p} p^2 D_{pp} \frac{\partial}{\partial p} \frac{1}{p^2} \psi - \frac{\partial}{\partial p} \left[ \dot{p} \psi - \frac{p}{3} (\nabla \cdot V_c \psi) \right] - \frac{\psi}{\tau_f} - \frac{\psi}{\tau_r}, \quad (2.1)$$

where $\psi$ is the CR differential number density per unit momentum interval, $V_c$ is the convection velocity, $\dot{p} \equiv dp/dt$ is the momentum loss rate, and $\tau_f$ and $\tau_r$ are the time scales for fragmentation and radioactive decay, respectively. Considering a particle with momentum $p$ and charge $Ze$, we assume a spatially homogeneous diffusion coefficient $D_{xx}$ as

$$D_{xx} = D_0 \beta^\eta (R/R_0)^\delta, \quad (2.2)$$

where $\beta \equiv v/c$ is the particle velocity at light speed $c$, $R \equiv pc/Ze$ is the rigidity, $\delta$ describes the rigidity-dependence of the diffusion coefficient, and $D_0$ is a normalization parameter. The parameter $\eta$ gives a phenomenological modification of the diffusion coefficient at low energies to better fit the data [27]. The momentum diffusion term describes the reacceleration of CRs during propagation, where the diffusion coefficient $D_{pp}$ relating to $D_{xx}$ as

$$D_{pp} D_{xx} = \frac{4p^2 v_A^2}{3\delta(4 - \delta^2)(4 - \delta)}, \quad (2.3)$$
where \( v_A \) is the Alfvén speed. We assume that CRs are confined within a cylinder with a radius of \( R_h = 20 \text{ kpc} \) and a thickness of \( 2z_h \). CRs outside this region are assumed to escape freely. Ref. \[29\] showed that the reacceleration model can fit the currently available data well. However, propagation with significant convection is disfavoured. Therefore, we restrict the current work to the reacceleration model configuration and the main propagation parameters are \( D_0, \delta, \eta, v_A, \) and \( z_h \).

We can parameterize the rigidity spectrum at the source as

\[
q_s = q_0 \left( \frac{R}{R_{br,1}} \right)^{-\nu_1} \left[ 1 + \left( \frac{R}{R_{br,1}} \right)^2 \right]^{(\nu_1-\nu_2)/2} \left[ 1 + \left( \frac{R}{R_{br,2}} \right)^2 \right]^{(\nu_2-\nu_3)/2}.
\] (2.4)

The parameters \( q_0, \nu_1, \nu_2, \nu_3, R_{br,1}, \) and \( R_{br,2} \) are also treated as free parameters in this work.

Secondary-to-primary ratios, such as B/C and \((\text{Sc+Ti+V})/\text{Fe}\), and the unstable-to-stable ratios of secondary particles, such as \(^{10}\text{Be}/^{9}\text{Be}\) and \(^{26}\text{Al}/^{27}\text{Al}\) are often used to determine the propagation parameters \(2,5,30\). In this work, we use the Li, Be, B, C, and O fluxes measured by AMS-02 \(31,32\), ACE \(33,34\), and Voyager \(35\) to obtain the MCMC samples used for training of the networks. We use the demodulated fluxes given in Ref. \(34\) to avoid addressing the solar modulation effect \(^4\). In Ref. \(34\), a spline interpolation method was employed to parameterize the wide-band spectra of various species and a fit to the Voyager, ACE, and AMS-02 data was performed assuming a force-field solar modulation model. Then, the data were demodulated based on the best-fit modulation parameter and its associated uncertainty. As described in the introduction, the error bars of these data have been artificially enlarged by a factor of \( \sqrt{10} \) to have a wide enough coverage of the parameter space. For C and O, we assume that they share identical primary source spectra, with different normalizations \(q_0^C\) and \(q_0^O\), respectively. The propagation parameters determine the absolute fluxes of secondary nuclei. However, as found in Ref. \(29\), the flux normalizations among different species may differ by up to 20\% due to the uncertainties of the fragmentation cross-sections. Thus, two additional constant factors \(\xi_{Li}\) and \(\xi_{Be}\) are further employed. The total parameter space is 14 dimensional, including five propagation parameters, five source spectral shape parameters, and four normalization parameters. An example input file of the GALPROP simulator, the galdef file, is available on the GitHub website \(^5\).

### 3 Convolutional neural network

To date, the CNN is one of the most widely used deep learning algorithms. Deep learning is a type of ML method. Traditionally, global fittings are employed to scrutinize physical models from observational data. Then, we can learn the correlations between model parameters by analysing the massive datasets generated by global fittings. In contrast, the CNN is a modern technique that can perform this task better than humans because it can mimic the properties

---

\(^4\)This is not crucial in this methodology-based paper.

\(^5\)https://github.com/alan200276/CR-ML/blob/main/MC_setup/galdef_54_crml
of a physical model by training network with massive data. As long as one can provide enough information with well-defined input and output parameters for training, the network can find underlying patterns or correlations based on a set of input parameters and then predict a set of output parameters.

There are three common usages of CNNs: interpolation, fast computation, and inverse functions. The CNN is a good tool for interpolation in a hyperspace of multiple parameters. When the network is trained even with discretized data, the corresponding outputs can be obtained by providing any input parameters. Together with the interpolation function, a trained network can provide predictions very quickly. However, “fast” computations are based on many “slow” calculations to provide data for training. CNN predictions are always faster than the actual computations starting from scratch unless recycling old data is non-beneficial. The CNN can invert a complicated partial differential equation, especially in a high-dimensional parameter space. Once the CNN obtains the inverse function, the experimental likelihood distribution of the physical observables (cosmic ray fluxes) can be translated to model parameters (propagation and source parameters).

The CNN is also capable of distinguishing the features of images, and fits our purpose of determining the correlation between the different energy spectra. Essentially, the CNN structure connects input and output layers with three main layers: a convolution layer, pooling layer, and fully connected layer. These three layers play different roles in quickly capturing the features of the data. We review their functions below.

The convolution layer converts the cosmic ray energy spectrum (an input tensor) into several feature maps via a feature detector (filter). These feature maps form a new tensor with one additional index. Since this new tensor contains more neurons than the input layer, it requires much more computation time and memory. We need a pooling layer after the convolution layer to reduce the tensor dimensions. Moreover, we use max pooling [36], which only takes the maximum value from one kernel. Several convolution and pooling layers can be utilized to achieve better performance. Finally, a fully connected layer is used to flatten the tensors and return an array of outputs.

4 Methodology

The purpose of this work is to build a machine that can invert CR propagation datasets into the allowed model parameters. The major blocks of actions and data processing are depicted in Fig. 1. We will describe the “mock data” in Sec. 4.1, “pseudo-data” in Sec. 4.2 and 4.3, machine training in Sec. 4.4, feedback data in Sec. 4.5, and data processing in Sec. 4.6.

4.1 Mock data

We first generate a massive dataset for machine training to build the correlation between the energy spectra of CRs and model parameters. We use the GALPROP code to simulate the propagation, together with an MCMC parameter scan [37].
Figure 1: Flowchart of our work. The mock/simulated data are generated using GALPROP and used to train two machines. The first machine is trained with the mock data directly. The second machine is trained with pseudo-data (mock data with artificial fluctuations added). The training procedure is (1) → (2) → (3) → (v) for Machine A and (1) → (2) → (3′) → (4′) → (v) for Machine B. The usage of two machines is (i) → (ii) → (iii) → (iv) → (v). Finally, the feedback loop for modifying Machine A is (i) → (ii) → (iii) → (iv) → (v) → (vi) → (1) → (2) → (3) → (v) → (vi). For the feedback loop of Machine B, we can replace (3) with (3′) → (4′) in the feedback loop of Machine A.
Table 1: The propagation and source parameters used in our scans.

Our mock data are a collection of GALPROP CR simulation results, including two sources: one from global fitting and one from feedback data. Ideally, the training data cannot be biased in particular parameter space regions, and the best choice is to use an ideal uniform scan with perfect coverage. Nevertheless, performing an ideal uniform scan is not feasible, especially in a high-dimensional parameter space. Therefore, we utilize the MCMC sampling tool to explore the high-dimensional parameter space, and the collected samples are our initial mock dataset. Our machine can learn and recognize the correlation between CR model parameters and the CR spectra with a mock dataset. Hence, the initial MCMC scan is our first kind of mock dataset.

Because MCMC sampling may not cover the parameter space perfectly, we also implement feedback to the dataset. We collect those feedback data from the validation procedure results and use them as our second type of mock dataset. In the following, we briefly summarize these two different mock datasets. Note that the detailed description for the generation of the feedback dataset will be given in Sec. 4.5.

The first CR mock dataset is collected from MCMC sampling tool together with the GALPROP simulations. We use the Metropolis-Hastings algorithm to generate Markov chains. Based on the latest CR data (from ACE, Voyager, and AMS-02), our global fitting was performed by using the package CosRayMC [4], which engages external GALPROP code (v54) for CR propagation simulation. Considering the coverage deficiency of MCMC scans, our training
data should not strongly depend on the current dataset. Therefore, we perform scans with a $\chi^2$ that is a factor of 10 less than actual data, namely,

$$\chi_{\text{scan}}^2 = \frac{\chi_{\text{tot}}^2}{10},$$  \hspace{1cm} (4.1)

where $\chi_{\text{tot}}^2$ sums all $\chi^2$ values from ACE, Voyager, and AMS-02 data. Moreover, $\chi_{\text{scan}}^2$ is used to drive the direction of the scan. We expect that future experimental data will provide smaller error bars, which will likely update the chi-square value as $\chi_{\text{scan}}^2 \rightarrow \chi_{\text{tot}}^2$. This type of mock dataset is very conservative for this type of study. One may also extend the mock data based on actual past, present, or future data. However, our methodology will not be affected.

In the parameter scan, we vary the 14 model parameters described in Sec. 2. The prior ranges for some of these parameters are given in Table 1. However, the normalization parameters are just overall scaling factors that are irrelevant to the shape of the spectra and may introduce a parameter degeneracy with propagation and source parameters. In addition, it is difficult to distinguish between the scaling factors and the data whitening factors in machine learning. Therefore, we treat these normalization parameters as nuisance parameters that are only determined when computing the $\chi^2$ of the data. The parameters included in the machine learning are

$$\theta = \{D_0, \delta, \eta, v_A, z_h, \nu_1, \nu_2, \nu_3, \log R_{\text{br},1}, \log R_{\text{br},2}\}.$$ \hspace{1cm} (4.2)

We collected $O(2 \times 10^5)$ mock data points for the inputs of the machine learning.

When using the machine trained only with the first CR mock dataset, we find that the machine cannot recognize some CR spectra. We understand that this may be because our MCMC scans do not provide sufficient coverage of the parameter space. Therefore, we attempt to fix this problem by using a learning-predicting-examining loop, called a feedback loop in this paper. We discuss this feedback loop in Sec. 4.5. Finally, we collect approximately $O(50,000)$ data points generated by the feedback loop. These feedback data are our second mock data source, and we combine them with the initial MCMC samples as the training datasets.

In this work, we only prepare the mock data based on the CR model given in Sec. 2, albeit it is the most popular and well-accepted model in the community. Currently, our machines can learn the relationship between this particular CR model and the energy spectra. If one wants to switch to a new CR model, a new MCMC scan has to be prepared. In other words, we must know the form of a function to find its inverse. Therefore, this could be a drawback until the ultimate CR model is built. Fortunately, the layer structure of our trained machines can be more or less fixed so that we can reuse the same machines to learn other CR models.

### 4.2 Pseudo-data and artificial fluctuations

A Gaussian distribution for the standard deviation can describe all the observable distributions, including the systematic and statistical uncertainties. Although the mean value of the total measurements fluctuates, it will eventually stabilize after accumulating enough data. However, as long as the central value and the standard deviation of the measured Gaussian
distribution are given, one can perform random sampling with the same Gaussian distribution to generate the pseudo-data. We can treat the standard deviation as a fluctuation of the mean value. Therefore, the method used to generate pseudo-data is called the pseudo experiment. Note that different experimental data distributions can give different pseudo-data.

When we use the trained machines for prediction, we feed arbitrary CR energy spectra into each machine. The generation of arbitrary CR energy spectra is similar to performing a pseudo-experiment. The only difference is that we randomly add artificial fluctuations to a simulated spectrum instead of using the actual experimental central values and errors. For convenience, the collected simulation data generated by GALPROP are called “mock data”, and the ones produced by pseudo-experiments are called “pseudo-data”, hereafter. Simply speaking, our pseudo-data are formed by adding artificial fluctuations to mock data.

The traditional method to estimate the likelihood of the entire parameter space is to visit all areas of the parameter space with a sampling algorithm and then compute its statistical strength $\chi^2$ accordingly. However, this process utilizes a very large amount of computing resources. In contrast, our machines can determine the propagation parameters by using the inverse function with the energy spectra as the network inputs. If taking the CR spectra directly from pseudo-data, we can obtain the allowed CR model parameters efficiently without using complicated sampling algorithms. Namely, one can treat pseudo-experimental data as a prior distribution rather than spending much time and effort to find the parameter space with the highest likelihood probabilities. In Sec. 4.3, we explicitly show how to obtain the network inputs from the generated pseudo-data.

In summary, “Mock Data” plus artificial fluctuations constitute the “Pseudo-data”.

### 4.3 Preparing for network inputs

This subsection addresses the technical details of the machines. It is recommended to skip this subsection for smoother reading if this is the first time reading the paper.

Note that we input the ratios of the spectra into the network. The reasons for using ratios are as follows: i) to reduce normalization dependence, ii) to reduce systematics, and iii) to more easily understand the correlation between the spectra. More details are given below.

There are still three difficulties in preparing model inputs. The first difficulty is that we have reintroduced four factors: $q_0^C$, $\xi_{Li}$, $\xi_{Be}$, and $q_0^O$, which are only determined by minimizing the chi-square statistics of the CR data. If there is a lack of CR experimental data, these four parameters are difficult to determine. Moreover, their effects can degenerate with some combinatorial changes in propagation parameters. Therefore, our machines cannot learn the features generated by these four normalization factors. The second difficulty is that the experimental CR spectra from ACE, Voyager, and AMS02 in some energy bins are missing. Therefore, it is problematic to directly use experimental bins as the basis of our network inputs when we want to apply our machine to future data. Finally, the third difficulty is that there can be some correlations between different energy bins as well as different CR spectra. CNNs might need a deeper network and spend more time training to capture the features of
spectra. In the following section, we will address solutions to these three difficulties.

Our strategy to solve the first difficulty is to determine the four normalization factors and remove them from the pseudo-data to match our network inputs. When utilizing GALPROP simulated data, which does not involve experimental data, these four normalization factors are somewhat arbitrary and irrelevant. When performing the $\chi^2$ calculation, experimental data can subsequently determine these four factors. If these factors are included in the learning process, we find that the machine has performance difficulties because the whitening procedure often washes out the rescaling effect.

We can fix the second difficulty (missing data in some energy bins of CR spectra) as follows. Instead of generating pseudo-data on the actual experimental energy bins, we generate the pseudo-data based on new energy grids. Specifically, we randomly pick a spectrum $\Phi'(E)$ from $4\sigma$ allowed mock data. We use this selected $\Phi'(E)$ as the new central value, but the averaged ratio of the experimental error bar to the central value is the size of the artificial fluctuation. Considering a total of 398 experimental data points from ACE, Voyager, and AMS02, the averaged uncertainty is approximately 5% of the central value. Hence, we adopt at most 5% as our artificial fluctuation to add to the selected $\Phi'(E)$ and the new spectrum $\Phi_{\text{pseudo}}$ is the actual pseudospectrum used in this study. The new total chi-square measure becomes

$$\chi^2_{\text{tot}} = \sum_{i,j} \left( \frac{\Phi_{\text{pseudo}}^{i,j} - \bar{\Phi}_{i,j}}{\sigma_{i,j}} \right)^2,$$

where $i$ proceeds to use experimental data from AMS02, Voyage, and ACE, but $j$ is the energy bin index. This treatment also prevents miscounting the weight for specific energy bins when combining data from different experiments. In Fig. 2, we show the demodulated kinetic energy spectrum of carbon. The solar modulation parameter $\phi$ is fixed at 696 MV. The green band indicates the $3\sigma$ uncertainty obtained by the latest experimental data from AMS02, Voyage, and ACE. The 84 bins of the pseudospectrum (red dots) are evenly located in the kinetic energy range $10^{-3}$ GeV $< E < 1.1 \times 10^5$ GeV. We can see that the pseudospectrum closely resembles the mock spectrum in the small energy region where experimental error bars are small. However, fluctuations become more significant in the region with no experimental data. We explicitly show the relative difference between the pseudo and mock data in the bottom panel. We refer readers to Appendix A.1 for the kinetic energy spectra of Li, Be, B, and O considered in this work.

For the third difficulty, we can help the machines capture the features of spectra when introducing some known correlations. Implementing the correlation between different energy bins or CR spectra is more complicated. Their correlation will depend on CR physics and cannot be directly observed from the experimental data. Using a spectrum from mock data as the central value of pseudo-data can automatically introduce the correlation between different energy bins. However, the correlation between spectra is nontrivial and depends on

---

6We expect that all future observational data will more or less fall into the range of $\pm 4\sigma$ allowed mock data.
Figure 2: The demodulated kinetic energy spectrum of carbon. The solar modulation parameter $\phi$ is fixed at 696 MV. The experimental error bars of AMS02, Voyage, and ACE are coloured blue, black, and purple, respectively. The red dashed line shows a random mock spectrum. The solid red dots represent a random pseudospectrum. The artificial fluctuations are a maximum of 5%. The green band indicates the $3\sigma$ (99.73% C.L.) combined uncertainty of the latest measurements (Li, Be, B, C, and O) from AMS02, Voyage, and ACE. We show a zoomed-in insert for the small error bars of the AMS02 data. The lower panel presents the relative uncertainty as defined by (Pseudo Data – Mock Data)/Mock Data.
Table 2: The ratio of spectra to include the correlations between different CR elements. We use a range of kinetic energy \( E \) from \( 10^{-3} \) GeV to \( 1.1 \times 10^5 \) GeV with 84 bins on a logarithmic scale. The rows represent the normalized inputs of lithium, beryllium, boron, carbon, and oxygen given by mock data, where \( N_{Li} = \Phi_{Li}(E)/\xi_{Li} \), \( N_{Be} = \Phi_{Be}(E)/\xi_{Be} \) and \( N_{O} = \Phi_{O}(E) \times q_{C0}/q_{O0} \). The energy spectrum of lithium, beryllium, boron, carbon, and oxygen are presented by \( \Phi(E) \). The normalization \( \Phi_{C110} \) is taken from the carbon flux with energy equal to 110 GeV.

| Element  | Ratio Spectra          |  \( N_{Li}/N_{C} \) |  \( N_{Li}/N_{O} \) |  \( N_{Be}/N_{C} \) |  \( N_{Be}/N_{O} \) |  \( N_{O}/\Phi_{C110} \) |
|----------|------------------------|----------------------|----------------------|----------------------|----------------------|----------------------|
| Lithium  | \( \Phi_{Li}(E)/\xi_{Li} \) | \( N_{Li}/N_{C} \) | \( N_{Li}/N_{O} \) | \( N_{Be}/N_{C} \) | \( N_{Be}/N_{O} \) | \( N_{O}/\Phi_{C110} \) |
| Beryllium| \( \Phi_{Be}(E)/\xi_{Be} \) | \( N_{Be}/N_{C} \) | \( N_{Be}/N_{O} \) |                      |                      |                      |
| Boron    | \( \Phi_{Be}(E)/\xi_{Be} \) | \( N_{Be}/N_{C} \) | \( N_{Be}/N_{O} \) |                      |                      |                      |
| Carbon   | \( \Phi_{C}(E)/\Phi_{C110} \) | \( N_{C} \) | \( \Phi_{C}(E)/\Phi_{C110} \) | \( N_{Be}/N_{C} \) | \( N_{Be}/N_{O} \) | \( N_{O}/\Phi_{C110} \) |
| Oxygen   | \( \Phi_{O}(E) \times q_{C0}/q_{O0} \) | \( N_{O}/\Phi_{C110} \) |                      |                      |                      |                      |

the propagation rather than the source parameters. This type of correlation can be simply identified by determining the ratios of the spectra of any two elements. As shown in Table 2, we therefore introduce several ratios where \( N_{Li} = \Phi_{Li}(E)/\xi_{Li} \), \( N_{Be} = \Phi_{Be}(E)/\xi_{Be} \) and \( N_{O} = \Phi_{O}(E) \times q_{C0}/q_{O0} \). We note that these ratios can not only introduce correlations but can also significantly improve the performance of learning. Hence, in this work, we use the ratios of spectra in Table 2 as pseudo-data and inputs of our machines.

Last, we would like to comment on our usage of pseudo-data. We design our networks for taking completed spectra as inputs, with energies from \( 10^{-3} \) GeV to \( 1.1 \times 10^5 \) GeV with 84 bins on a logarithmic scale. In principle, one can insert any shape of spectra, but their energies must match these 84 energy bins. These inputs give us flexibility for the exploration of future CR data, which may contain a broader energy range and better energy resolution. However, given the shortcomings of the current CR data structure as described previously, we have to use pseudo-data, not experimental spectra, as network inputs in this work.

Note that one can freely select some spectra from mock data as the central value of the pseudo-data, which agrees with experimental data at a required level. This technique is similar to the Bayesian updating prior approach, but there are two differences; i) our prior distributions are a function of the energy spectra rather than CR model parameters, and ii) adding random fluctuations allow us to use new spectra, which have never seen in the original mock data. Once future CR measurements cover all energy ranges of interest, we can use the experimentally provided spectra as the network inputs directly.

4.4 Machines for inverting propagation

We utilize two machines to learn tasks:

- **Machine A**: a CNN machine to directly learn the inverse propagation from a set of mock data to a set of model parameters.
• **Machine B**: a CNN machine to directly learn the inverse propagation from a set of pseudo-data to a set of model parameters.

We can see that the difference between Machines A and B is the input data used for training. While Machine A is only able to find the solutions closest to the mock data, we expect that Machine B is more flexible for the general-purpose use with inputs of *any* data, even with fluctuations.

We show the architectures of Machine A and Machine B in Fig. 3. The input layer contains eight spectra for 84 energy bins and the output layer contains 10 propagation and source spectral shape parameters (the 4 normalization parameters, \( q_0^C \), \( q_0^O \), \( \xi_{Li} \), and \( \xi_{Be} \) are not trained). For Machine A, there are five one-dimensional convolutional layers, with 512, 512, 256, 256, and 128 filters, accordingly. The argument `kernel_size` of one-dimensional convolutional filters is set to 1, the size of the maximum pooling layers is `pool_size=2`, and the stride length, `strides=1`. There are a total 5 fully connected dense layers with 128, 128, 64, 64, and 32 neurons, respectively. In the bottom panel of Fig. 3, we can see that the architecture of Machine B is slightly different from that of Machine A. As Machine B is expected to have more tolerance with fluctuating input data (this is always the case for real measurements), the learning spectra of Machine B include artificial fluctuations. Instead of using five one-dimensional convolutional layers, Machine B utilizes six one-dimensional convolutional layers, among which the additional layer contains 128 filters. There are only two dense layers for Machine B, each of which contains 128 neurons. The exponential linear unit (ELU) \(^{38}\) is used as the activation function in both architectures. The last dense layer is fully connected to ten output neurons with a linear activation function in both machines. We use Adam \(^{39}\) as the optimizer, and the loss function is taken as log cosh. The Keras-2.1.6 library is used to train the machines with Tensorflow-1.10.1 \(^{40}\) backend, on NVIDIA Tesla P100-5XM2-16GB.

Both Machine A and Machine B can invert the cosmic-ray propagation to predict the CR model parameters after learning a large amount of data. As shown in Table 2, the ratios of spectra at the Earth are inputs to the machines, but the CRs model (source and propagation) parameters are outputs. Because of the strong correlation among the inputs, we adopt the CNN method\(^7\) to address this type of training, such as recognizing pixels in photographs.

With 84 bins in each spectrum and a total of 8 spectra, as shown in Table 2, the total input degrees of freedom for Machine A and Machine B are \(84 \times 8\). There are ten output dimensions, five propagation parameters and five source parameters, as shown in Table 1. We perform the whitening procedure as follows. Except for \( R_{br,1} \) and \( R_{br,2} \), whose range is wider than two orders of magnitude, we also whiten the other propagation parameters to be between 0 to 1. We take the logarithm of \( R_{br,1} \) and \( R_{br,2} \) before the whitening procedure. After the whitening procedure, we randomly divide our total input data points so that 90% of data is used for training and 10% is used for testing. For better performance, we choose

---

\(^7\)We have compared with a relatively simple Deep Neural Network, but its performance is much worse than the CNN in this case.
Figure 3: The architectures of our Machine A and Machine B. See the text for more details.
Figure 4: Left panels: the learning curve of training. Right panels: the mean square error distribution of the test data. The two upper figures are based on Machine A while the two lower figures on Machine B.
the log-cosh loss function in both machines. Our trained machines can predict the source spectra and propagation parameters with some necessary tuning for the hyperparameters. The explicit setting is given in Fig. 3.

Here, we investigate the effects of adding artificial fluctuations by comparing Machine A and Machine B. Moreover, adding artificial fluctuations is a method of creating spectrum ratios different from those of the collected simulated data (called mock data) because our pseudo-data generation still relies on the mock data. Once artificial fluctuations are included, despite some instances of unexpected spectra, the fluctuation effects might be degenerate with the changes in model parameters. We build Machine B, which learns the pseudo-data (simulated data with artificial fluctuations) as our “control machine”. As a “subject machine”, Machine A is restricted to learning the simulated data without the implementation of artificial fluctuations. Under two different setups, one expects that Machine A is unable to recognize artificial fluctuations but makes predictions based on the most similar spectral shape. In contrast, Machine B has some level of capability to recognize spectra with artificial fluctuations.

In Fig. 4, we present the performance of Machine A (upper two panels) and Machine B (lower two panels). The total loss function (log cosh) decreases to the number of training epochs in both machines. After \( \sim 80 \) epochs, the loss becomes a constant for both the training and test data. We utilize a machine based on 100 epochs of training in our work. The two right panels only show the conventional mean squared error (MSE) distribution for the test data. The MSE of the test data is a well-distributed Gaussian function, which implies neither overtraining nor undertraining of the data. In the left panel, the mean and 95% C.L. lines are given by the vertical solid and dashed lines, respectively. We especially note that the MSE distribution of Machine B has a larger width than Machine A. We found that Machine B may not be able to discriminate the changes in CR parameters from artificial fluctuations.

We want to emphasize that one can reuse the same architectures of these two machines for training different CR models because the purpose (finding inverse functions) is identical. Hence, time can be saved in modifying CNN layers and hyperparameters. Usually, the most time-consuming procedure is not the training but the machine architecture modification. It may be sufficient to only retrain the last few layers in some optimistic cases.

### 4.5 Feedback loop

After Machine A and Machine B are trained, we can use the pseudo-data as inputs to both machines. Moreover, the outputs \( \{\theta\} \) in these two machines can be inserted back into the GALPROP code for the validation of learning as shown by the sequence \((v) \rightarrow (vi) \rightarrow (1)\) in Fig. 1. The explicit path of the feedback loop starts with the process from \((i)\) to \((vi)\) before connecting to the label \((1)\) (GALPROP simulator). The first machine prediction of CR model parameters in label \((v)\) is based on the pseudo-data. The process follows the sequence \((1) \rightarrow (2) \rightarrow (ii) \rightarrow (iii) \rightarrow (iv) \rightarrow (v)\). Once we reach the purple label \((v)\) again, we can obtain the second machine prediction and the feedback loop for modification ends. The
second machine prediction of CR model parameters is based on the recomputed spectra by 
GALPROP, which uses the first machine prediction parameters as inputs. We note that the 
randomly generated pseudo-data can be different from the one computed by using GALPROP, 
even if we use the same CR model parameters.

In this work, for each model parameter $\theta_i$ given in Eq. (4.2), we introduce the systematic 
uncertainty $\tau_i$ of network prediction as

$$
\tau_i = \frac{\theta_i' - \theta_i}{\theta_i}.
$$

Comparing the result returned from GALPROP ($\{\theta_i'\}$) and predicted by randomly generated 
pseudo-data ($\{\theta_i\}$), the systematic uncertainties of each parameter introduced by machine 
learning can be treated as $\{\tau_i\}$. Based on the set $\{\theta_i\}$, we set our stop criteria as follows:
when 95% of the total predictions agree with the condition $\frac{1}{10} \sum |\tau_i| < 0.1$, we terminate the 
repeated feedback loop. Here, we take the averaged error of the 10 model parameters. Although 
this criterion is not extremely stringent, the first machine predicted model parameters 
by using pseudo-data as inputs still give very similar values to the parameters predicted by 
using the spectra generated by GALPROP.

In label (vi) of Fig. 1, if we do not reach the stop criteria, we repeat the training with 
updated mock data, including those generated during the modification process. This approach 
of checking for modification follows the algorithm of "Generative Adversarial Network" [41], 
see also its application to LHC physics [42]. Namely, instead of hiring discriminative networks 
to evaluate our machines, we verify the result by inserting predicted parameters back to the 
GALPROP simulation.

### 4.6 Data processing

In this subsection, we summarize the data processing and all our steps in Fig. 1. This 
blocks show all the steps adopted in our work, and the arrows represent the direction of data 
processing. For example, the sequence (1) $\rightarrow$ (2) $\rightarrow$ (3) indicates the following operations. 
First, we use GALPROP (1) to generate a set of data including CR model parameters and 
spectra. Then, we append these data to other pre-existing datasets (if any) to form a large 
collection called mock data (2). Finally, we send a very large amount of mock data to 
Machine A for training (3). Likewise, the sequence (i) $\rightarrow$ (ii) $\rightarrow$ (iii) $\rightarrow$ (iv) $\rightarrow$ (v) 
can be understood as follows. From any CR experimental data, we can extract the central 
values and errors in (i). With the updated central values and errors, we can recompute the 
new $\chi^2$ for our massive mock data collection (2). Requiring specific statistical criteria, for 
example, $\delta \chi^2 < 33.2$ for spectra less than 3$\sigma$ CI in our study, one can randomly select a set 
of model spectra from this refined 3$\sigma$ bracket. In label (ii), we perform a pseudo-experiment 
by simply adding random artificial fluctuations that do not exceed 5% to this selected model 
spectra. The newly formed spectra are pseudospectra. We then use these pseudospectra to 
perform data preprocessing for calculations of ratio spectra (iii) and whitening transformation.
Eventually, our two trained machines can read their inputs given by (iv) and predict their corresponding CR model parameters as label (v). Note that we exactly compute $\chi^2$ as Eq. (4.3) by using pseudospectra in (ii), not the simulated model spectra in (2).

In this paper, we begin our work from GALPROP simulations, along the sequence (1) $\rightarrow$ (2) $\rightarrow$ (3) for Machine A and (1) $\rightarrow$ (2) $\rightarrow$ (3') $\rightarrow$ (4') for Machine B. After the machines are built, one can use the machines to predict the CR model parameters by inserting the pseudo-data following the sequence (i) $\rightarrow$ (ii) $\rightarrow$ (iii) $\rightarrow$ (iv) $\rightarrow$ (v).

We adjust our two machines by connecting the above training and prediction processes in a feedback loop. The first machine prediction of the CR model parameters in the feedback loop is via (i) $\rightarrow$ (ii) $\rightarrow$ (iii) $\rightarrow$ (iv) $\rightarrow$ (v), where the inputs are the pseudospectra. However, we can insert the first machine predicted model parameters back to the GALPROP simulation (1) to recompute the actual spectra. Then, these spectra with $<5\%$ random artificial fluctuations can be fed back to both machines to make the second prediction of CR parameters at the label (v). In this step, we can compare the first and second machine-predicted CR parameters to validate our training. If their averaged error does not satisfy $\frac{1}{10}\sum |\tau_i| < 0.1$, we have to update the mock data collection and modify our networks. The new data appended to the old mock data are the CR model parameters obtained from the first machine prediction and its GALPROP simulation. Therefore, our verification in the feedback loop is achieved by the sequence (v) $\rightarrow$ (vi) $\rightarrow$ (1) $\rightarrow$ (2) $\rightarrow$ (ii) $\rightarrow$ (iii) $\rightarrow$ (iv) $\rightarrow$ (v).

Finally, we emphasize that the experimental data can be updated. As long as the physical model of the propagation and source are not altered, our trained machines are reusable. Overall, our machines can quickly predict the allowed model parameters by plugging in new experimental data, even if the data have not been seen before or during the training.

5 Results

In this section, we show our results with the updated chi-square metric, $\chi^2_{\text{tot}}$ in Eq. (4.1). Note that our mock data are mainly prepared by performing the MCMC scan with $\chi^2_{\text{tot}}/10$, and hence the $1\sigma$ and $2\sigma$ regions based on the current data (determined by $\chi^2_{\text{tot}}$) could be less precise. Therefore, our only concern is the $3\sigma$, $4\sigma$, and $6\sigma$ regions. For the parameter constraints with $1\sigma$ and $2\sigma$ confidence levels, one can refer to Ref. [29].

The minimum $\chi^2_{\text{tot}}$ is 347.0 based on the total collected mock data. Considering the total 14 degrees of freedom as seen in Table 1, the confidence interval (CI) for $3\sigma$ (99.73%), $4\sigma$ (99.99%), and $6\sigma$ ($\approx$ 100%) are $\delta\chi^2_{\text{tot}}$ equal to 33.20, 43.82 and 69.91, respectively. Our presentation of parameter space for mock data is based on the Frequentist approach, i.e., the “profile likelihood” method (minimum chi-square), to eliminate uninteresting parameters. When plotting the contours in this work, we use the data generated from the MCMC scan and the feedback loop. However, the feedback loop data focus on the regions where the machines do not learn the inverse function well. Hence, utilizing the Bayesian marginal posterior approach is inappropriate because the prior distributions are unclear.
Figure 5: The systematic uncertainties introduced by the learning process. The left panel is based on Machine A, while the right panel is based on Machine B. The histograms are made by using 30,000 test samples.

5.1 The systematic uncertainties introduced by learning

We first discuss the systematic uncertainties introduced by Machines A and B. These results only depend on the mock data generated by CosRayMC with $\chi^2_{\text{scan}}$ as given in Eq. (4.1). Using this setup, we will know nothing about the current or future data, but the training is not affected. In Sec. 5.2, we will open the box so that the experimental chi-square value is updated to $\chi^2_{\text{tot}}$.

In Fig. 5, histograms of the $\Delta D_0/D_0$ systematic uncertainties introduced in the learning process of Machine A (left panel) and Machine B (right panel) are created from the test data. Both histograms are well-distributed Gaussian with central values close to zero, and the 1σ (68% CI) regions are given by $-0.25 \pm 0.75$ % for Machine A but $-2.33^{+0.700}_{-0.467}$ % for Machine B. Clearly, Machine B has more significant uncertainties due to the added artificial fluctuations. Therefore, adding 5% artificial fluctuations can contaminate the propagation inverter.

We summarize the systematic uncertainties of learning for the rest of the parameters in Table 3. Most of the parameters can be predicted very well, with the central values close to zero and small error bars. However, the networks do not predict the diffusion coefficient $D_0$ and the height of diffusion zone $z_h$ precisely, due to the weaker correlation between $D_0$ and $z_h$ at large values, as shown in Fig. 7. Typically the secondary-to-primary ratios of CR nuclei determine the ratio of $D_0/z_h$ [2]. Indeed, our machines learn this effect. This scaling may be broken when $z_h$ is so large that it is comparable to the boundary of the propagation halo. Therefore for $z_h > 10\, \text{kpc}$, the performance of our machines is not as good as that in other
| Source parameters | Machine A | Machine B |
|-------------------|-----------|-----------|
| $\nu_1$           | $0.00^{+1.75}_{-2.50}$ | $0.00 \pm 5.00$ |
| $\nu_2$           | $0.05^{+0.09}_{-0.14}$ | $-0.05 \pm 0.15$ |
| $\nu_3$           | $0.09^{+0.09}_{-0.09}$ | $-0.20 \pm 0.20$ |
| $\log R_{br,1}$   | $0.04 \pm 0.14$      | $0.00 \pm 0.30$ |
| $\log R_{br,2}$   | $-0.09 \pm 0.14$     | $0.15^{+0.30}_{-0.45}$ |
| Propagation parameters | Machine A | Machine B |
| $D_0$             | $-0.25 \pm 0.75$     | $-2.33^{+7.00}_{-4.67}$ |
| $\delta$          | $0.04^{+0.27}_{-0.18}$ | $-0.33 \pm 0.50$ |
| $z_h$             | $-0.50^{+1.25}_{-1.50}$ | $3.28 \pm 13.11$ |
| $v_A$             | $0.35 \pm 1.05$      | $0.00 \pm 3.33$ |
| $\eta$            | $0.00^{+0.70}_{-1.40}$ | $1.67^{+5.00}_{-1.67}$ |

Table 3: The systematic uncertainties of different parameters introduced by learning. The error bars are estimated by using test data.

### 5.2 Comparison between the network predictions and the reweighted MCMC method

In Fig. 6, we present the one-dimensional $\Delta \chi^2$ distribution for $D_0$ (upper left), $Z_h$ (upper right), $\nu_1$ (lower left), and $R_{br,1}$ (lower right). We use the profile likelihood method to remove other unwanted parameters. The solid red lines are based on the training samples, including the collected data from the initial MCMC and the feedback loops. The grey thick solid lines represent the $\Delta \chi^2$ distribution based on the initial MCMC scan. We collect the initial MCMC data using the likelihood function given in Eq. (4.1). For comparison, we randomly take $5 \times 10^4$ pseudospectra for the network predictions of **Machine A** (blue dashed lines) and **Machine B** (green dash-dotted lines). The orange horizontal dotted lines show the CI for 3$\sigma$, 4$\sigma$, and 5$\sigma$.

Generally, the network predictions perform well but **Machine A** perfectly captures the features of the $\Delta \chi^2$ distribution, even though the $\Delta \chi^2$ distributions are previously unseen information. Importantly, we can see that both **Machine A** and **Machine B** manage to find the trends from the training data, and they yield smoother curves than the MCMC method and our training samples. As expected, the feedback loop self-corrects the problems introduced by the poor coverage of the parameter space. In the initial MCMC scans, the networks cannot precisely predict the parameter space using extrapolation. Because of the condition $\sum_{i=1}^{10} |\tau_i| < 0.1$, we send extrapolating predictions back to the feedback loop even if they do not exist in the initial MCMC scans. Hence, the total training samples (red lines) are
Figure 6: The one-dimensional $\Delta \chi^2$ distribution for $D_0$ (upper left), $Z_h$ (upper right), $\nu_1$ (lower left), and $R_{br,1}$ (lower right). The $\Delta \chi^2$ distributions are projected to our CR parameter of interest, and the rest are profiled using the profile likelihood method. The red solid, grey thick solid, blue dashed, and green dash-dotted curves represent the $\Delta \chi^2$ distribution of the total training samples, the initial MCMC scan, the predictions of Machine A and the predictions of Machine B, respectively.

In Fig. 7, the propagation parameters $D_0$ and $z_h$ (left panel) and source parameters $R_{br,1}$ more smoothly distributed. Note that if our networks would not precisely predict the future data, they may extrapolate instead of interpolating the training samples and the feedback loop can be applied again for self-correction. For the sake of completeness, we also give the $\Delta \chi^2$ distributions of all other CR model parameters in Appendix A.2.
Figure 7: Two-dimensional allowed parameter space. The left panel is the allowed region projected on the \((Z_h, D_0)\) plane, while the right panel is for the source parameters \(R_{br,1}\) and \(\nu_1\). Based on the latest chi-square metric \(\chi^2_{\text{tot}}\), the green region is the 3σ confidence interval and the blue region is the 6σ confidence interval. The red circles and black crosses are 100 network predictions by Machine A and Machine B, respectively.

and \(\nu_1\) (right panel) are projected on the two-dimensional contour plots. We also illustrate the predicted scatter points by both machines with the pseudospectra within required a 4σ region (3σ mock data plus 1σ artificial fluctuations). The inner contour is the 3σ region \((\delta\chi^2_{\text{tot}} = 33.20)\) while the outer contour is the 6σ region \((\delta\chi^2_{\text{tot}} = 69.91)\). We take 100 representative samples of Machine A (red circles) and Machine B (black crosses). There is no dependence on \(D_0\) at large values of \(D_0\) and \(z_h\). Therefore, both Machine A and Machine B, but especially for Machine B, cannot perform well. However, because the correlation between \(R_{br,1}\) and \(\nu_1\) is clear, both machines can predict the contours quite well. For other combinations of CR model parameters, refer to Appendix A.2.

Finally, we compare the prediction efficiency for different methods if the chi-square is updated to the latest experimental value, then \(\chi^2_{\text{scan}} \rightarrow \chi^2_{\text{tot}}\). In Table 4 it is not surprising that only a minor group 6.5% of total MCMC samples can survive from 6σ cut of \(\chi^2_{\text{tot}}\), because the initial MCMC samples are created by using a loose condition \(\chi_{\text{scan}}^2 = \chi_{\text{tot}}^2/10\). Remarkably, Machine A achieves good performance in that approximately 70% of the total predictions fall into the 6σ region while this percentage is approximately 30% for Machine B. Even though we train both machines by using MCMC data, it is still much more efficient to generate samples in a new allowed region than simply reweighting the old MCMC data.

As an additional comparison, we build a typical classifier “Check Machine” to identify whether the model parameters fall into the 4σ region that is determined by using all the data samples collected in this study. The details of the “Check Machine” are given in Appendix B.
Table 4: The summary of prediction efficiency for different methods. The initial MCMC samples are collected based on $\chi^2_{\text{scan}}$ but re-weighted by using $\chi^2_{\text{tot}}$ where $\chi^2_{\text{scan}} = \chi^2_{\text{tot}}/10$. The network predictions of Machine A and Machine B are based on the pseudo-data created by using $\chi^2_{\text{tot}}$. For a comparison, we build a classifier “Check Machine” by combining total collected mock data.

We note that the Check Machine can identify the 4σ parameter space better than Machine B. Evidently, Machine A is still the most efficient network to predict the parameter space that falls into the required CI.

6 Summary and discussion

In this paper, we investigated the inverse function of the CR diffusion equation by using a CNN network. This function allowed us to efficiently and directly determine CR source functions and propagation mechanisms. Subsequently, we created two CNN machines to obtain the inverse function of the CR diffusion equation. In general, Machine A learned the mock data directly, while Machine B learned the mock data with added artificial fluctuations. The systematic uncertainties of the two machines introduced by the learning procedures were evaluated. We found that the procedure of identifying artificial fluctuations may somewhat disturb the inverse propagation in Machine B.

We prepared the learning data of the two CNN machines by using an MCMC global fitting based on the measured spectra from the secondary cosmic ray nuclei Li, Be, B, C, and O from AMS-02, ACE, and Voyager-1. However, we wanted to demonstrate that our networks can be used even for future updated data. Therefore, scans were performed with the total chi-square divided by 10, and the result was used as learning data. These two machines were adjusted as much as possible to obtain the best performance.

If the pseudospectra fed into the networks were created based on a new dataset with a different $\chi^2$, two machines could reasonably predict the CR model parameters at the required confidence interval. When data were updated, we only required a 4σ confidence interval as the best demonstration of our initial MCMC data. By giving a new experimental dataset with smaller error bars, both Machine A and Machine B performed much more efficiently in finding the required confidence interval in the multidimensional parameter space than the one obtained by reweighting old MCMC data. In addition, we also compared the two machines with a typical classifier that can identify the 4σ region in the multidimensional CR parameter
space by learning the reweighted old MCMC data. We found that our Machine A still performed better than this classifier, while Machine B performs slightly poorer.

In summary, the greatest advantage of our method is that when a new dataset is introduced or an existing dataset is updated, we do not need to retrain the networks nor repeat the whole MCMC analysis. The trained model can easily propose an approximate likelihood distribution of the CR model parameters. Furthermore, to demonstrate the usefulness of the inverse function of the propagation equation in this work, we have only fed pseudospectra into the network. However, if future detectors precisely measure the CR spectra for the whole energy range, the experimental spectra can be directly used for the network inputs. In addition, other spectra, such as anti-proton, Helium-3 and Helium-4 can be included in the training, but currently, these spectra suffer from large uncertainty [43].

Here, we comment on our choice of the initial mock data, where the chi-square metrics of the old data are designed to be ten times larger than the one being updated. This assumption is indeed conservative compared to the actual situation because future CR experiments might only improve the error bars by lesser factor. Under such an update, our pseudospectra shall be more reliable so that our networks can provide confidence interval with more controls.

Although we only focus on CR propagation in this paper, our approach can be easily adapted to solve the inverse problem of multidimensional parameter space in other fields, for example, searching for supersymmetry at the LHC [44]. Moreover, as soon as a standard CR propagation model is built, our machines can be simply adapted for searching for the signature of dark matter from precisely measured cosmic ray experimental data. Finally, we suggest three interesting applications of using our machines:

- Our machines can be generators of the model parameters with a proposed prior probability distribution. This usage is shown in this paper.

- To find the correlations between any physical spectra and theoretical model parameters, the architecture in Fig. 3 can be reused for new training. This can save much time in tuning the layer structure and hyperparameters.

- Based on the latest experimental measurements, our inverse function networks of CR propagation can suggest suitable CR model parameters for a nonstandard source search, such as DM or any new cosmic ray source. The machines allow us to put the latest experimental constraints on the DM parameter space by computing only the DM contribution.

Acknowledgments

We would like to thank Chi Chan for helpful discussions at the early stages of this work. We thank Roberto Ruiz de Austri Bazan, Sai-Ping Li and Wei-Chih Huang for useful comments and suggestions. Y.-L. S. Tsai was funded in part by the Taiwan Young Talent Programme of Chinese Academy of Sciences under Grant No. 2018TW2JA0005 and the Ministry of Science
and Technology, Taiwan under Grant No. 109-2112-M-007-022-MY3. Q.Y. is supported by the NSFC under Grants No. 11722328, No. 11851305, the 100 Talents program of Chinese Academy of Sciences, and the Program for Innovative Talents and Entrepreneur in Jiangsu. The work of K.C. was supported by the National Science Council of Taiwan under Grant No. MOST-107-2112-M-007-029-MY3.
A Supplemental figures

A.1 The kinetic energy distribution

Figure 8: The demodulated kinetic energy spectra of Li, Be, B, and O. The solar modulation parameter is fixed at $\phi = 696$ MV.

In Fig. 8 the demodulated fluxes when $\phi = 696$ MV for four different elements, Li (upper left), Be (upper right), B (lower left), and O (lower right), are shown. The green band indicates the 3σ uncertainty obtained by the global analysis of the latest experimental data from AMS02 (blue), Voyage (black), and ACE (purple). The 84 bins of pseudo-spectrum (red dots) are evenly located at the kinetic energy range $10^{-3}$ GeV < $\log E < 1.1 \times 10^5$ GeV.
A.2 One- and two-dimensional likelihood distribution

Figure 9: The one-dimensional $\Delta \chi^2$ distributions for CR model parameters, except for $D_0$, $Z_h$, $\nu_1$, and $R_{br,1}$, which are shown in Fig. 6. The grey thick solid, red solid, blue dashed, and green dash-dotted curves show the $\Delta \chi^2$ distribution of the initial MCMC scan, all samples, the predictions of Machine A and the predictions of Machine B, respectively. The CIs for $3\sigma$, $4\sigma$, and $5\sigma$ are shown by the orange horizontal dotted curves.

In Fig. 9 we present the one-dimensional $\Delta \chi^2$ distributions for the six remaining CR model parameters (Eq. (4.2)) in addition to Fig. 6. The colour coding is the same as before. The network predictions from both Machine A and Machine B mimic the $\Delta \chi^2$ distributions very well. Again, despite the wavy feature of MCMC caused by scan coverage, our networks already capture the CR model features and predict much smoother curves.

In Fig. 10 we show the distribution of all the propagation and source parameters. Similar to the colour scheme in Fig. 7, the green and blue regions represent the 3$\sigma$ and 6$\sigma$ CI of the latest cosmic ray data, respectively. As a comparison, we also show 100 predicted 4$\sigma$ points by Machine A (red circles) and Machine B (black crosses). Clearly, the majority of predicted 4$\sigma$ points (3$\sigma$ mock data plus 1$\sigma$ artificial fluctuations) agrees with the 3$\sigma$ contours. Although these predicted points are sampled on the experimental fluxes with respect to energies, our machines still know the correlation between parameters. Moreover, even the farthest points
are still included within the $6\sigma$ CI.

B Check Machine

This machine aims to determine whether the parameter set $\{\theta_i\}$ is within the $4\sigma$ region directly from the source and propagation parameter space. For the training data, we select those samples within the $4\sigma$ region and randomly select an equal number of samples outside the $4\sigma$ region from the total collected mock data. Unlike Machine A and Machine B, we

![Figure 10: Two-dimensional distributions of the propagation and source parameters. The colour scheme is the same as in Fig. 7.](image)

- $3\sigma$ (99.73%)
- $6\sigma$ ($\approx$ 100%)
- ○ Machine A
- × Machine B

28
use the source and propagation parameters as the input data of training. The data are divided into 90% and 10% for training and validation, respectively. The Check Machine contains six fully connected dense layers with the ReLU activation function, however, we use binary cross entropy as the loss function for binary class purposes. After adjusting the performance, we found that only 29% of the true data (inside the 4σ region) can be identified by the Check Machine, while 93% of the data outside the 4σ region can be identified as false. We show the confusion matrix in Fig. 11.
References

[1] E. C. Stone, A. C. Cummings, F. B. McDonald, B. C. Heikkila, N. Lal, W. R. Webber, Science 341, 150 (2013)

[2] D. Maurin, F. Donato, R. Taillet and P. Salati, Astrophys. J. 555, 585 (2001) [astro-ph/0101231].

[3] R. Trotta, G. Jóhannesson, I. V. Moskalenko, T. A. Porter, R. R. d. Austri and A. W. Strong, Astrophys. J. 729, 106 (2011) [arXiv:1011.0037 [astro-ph.HE]].

[4] Q. Yuan, S. J. Lin, K. Fang and X. J. Bi, Phys. Rev. D 95, no. 8, 083007 (2017) [arXiv:1701.06149 [astro-ph.HE]].

[5] A. W. Strong and I. V. Moskalenko, Astrophys. J. 509, 212 (1998) [astro-ph/9807150].

[6] C. Evoli, D. Gaggero, D. Grasso and L. Maccione, JCAP 0810, 018 (2008) Erratum: [JCAP 1604, E01 (2016)] [arXiv:0807.4730 [astro-ph]].

[7] M. Y. Cui, Q. Yuan, Y. L. S. Tsai and Y. Z. Fan, Phys. Rev. Lett. 118, no.19, 191101 (2017) [arXiv:1610.03840 [astro-ph.HE]].

[8] A. Cuoco, M. Krämer and M. Korsmeier, Phys. Rev. Lett. 118, no.19, 191102 (2017) [arXiv:1610.03071 [astro-ph.HE]].

[9] S. J. Lin, X. J. Bi and P. F. Yin, Phys. Rev. D 100, no. 10, 103014 (2019) [arXiv:1903.09545 [astro-ph.HE]].

[10] BOSER, B.E., GUYON, I.M. and VAPNIK, V.N. (1992). A Training Algorithm for Optimal Margin Classifiers. In Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory 144-152. ACM Press.

[11] CORTES, C. and VAPNIK, V. (1995). Support-Vector Networks. In Machine Learning 273-297.

[12] J. H. Friedman, Stochastic gradient boosting, Computational Statistics & Data Analysis 38 (2002), no. 4 367 - 378. Nonlinear Methods and Data Mining

[13] G. Ridgeway, Generalized boosted models: A guide to the gbm package, 2006.

[14] T. Chen and C. Guerstrin, Xgboost, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Aug, 2016)

[15] D. Guest, K. Cranmer and D. Whiteson, Ann. Rev. Nucl. Part. Sci. 68, 161-181 (2018) doi:10.1146/annurev-nucl-101917-021019 [arXiv:1806.11484 [hep-ex]].
[16] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, Maosong Sun, arXiv:1812.08434 [cs.LG].

[17] Alex Sherstinsky arXiv:1808.03314 [cs.LG].

[18] Polosukhin, Illia; Kaiser, Lukasz; Gomez, Aidan N.; Jones, Llion; Uszkoreit, Jakob; Parmar, Niki; Shazeer, Noam; Vaswani, Ashish, arXiv:1706.03762 [cs.CL].

[19] M. J. Fenton, A. Shmakov, T. W. Ho, S. C. Hsu, D. Whiteson and P. Baldi, arXiv:2010.09206 [hep-ex].

[20] J. Ren, L. Wu, J. M. Yang and J. Zhao, Nucl. Phys. B 943, 114613 (2019) arXiv:1708.06615 [hep-ph].

[21] M. Abdughani, J. Ren, L. Wu, J. M. Yang and J. Zhao, Commun. Theor. Phys. 71, no. 8, 955 (2019) arXiv:1905.06047 [hep-ph].

[22] J. Alsing, T. Charnock, S. Feeney and B. Wandelt, Mon. Not. Roy. Astron. Soc. 488, no. 3, 4440 (2019) arXiv:1903.00007 [astro-ph.CO].

[23] Y. K. Lei, C. Liu and Z. Chen, arXiv:2006.01495 [hep-ph].

[24] Q. Yuan, Sci. China Phys. Mech. Astron. 62, no.4, 49511 (2019) arXiv:1805.10649 [astro-ph.HE].

[25] A. W. Strong, I. V. Moskalenko and V. S. Ptuskin, Ann. Rev. Nucl. Part. Sci. 57, 285 (2007) astro-ph/0701517.

[26] D. Maurin, A. Putze and L. Derome, Astron. Astrophys. 516, A67 (2010) arXiv:1001.0553 [astro-ph.HE].

[27] G. Di Bernardo, C. Evoli, D. Gaggero, D. Grasso and L. Maccione, Astropart. Phys. 34, 274-283 (2010) arXiv:0909.4548 [astro-ph.HE].

[28] E. S. Seo, V. S. Ptuskin, Astrophys. J. 431, 705 (1994)

[29] Q. Yuan, C. R. Zhu, X. J. Bi and D. M. Wei, JCAP 11 (2020), 027 doi:10.1088/1475-7516/2020/11/027 arXiv:1810.03141 [astro-ph.HE].

[30] A. Putze, L. Derome and D. Maurin, Astron. Astrophys. 516, A66 (2010) arXiv:1001.0551 [astro-ph.HE].

[31] M. Aguilar et al. [AMS], Phys. Rev. Lett. 119, no.25, 251101 (2017)

[32] M. Aguilar et al. [AMS], Phys. Rev. Lett. 120, no.2, 021101 (2018)

[33] N. E. Yanasak et al., Astrophys. J. 563, 768 (2001)
[34] C. R. Zhu, Q. Yuan and D. M. Wei, Astrophys. J. 863, no.2, 119 (2018) [arXiv:1807.09470 [astro-ph.HE]].

[35] A. C. Cummings, E. C. Stone, B. C. Heikkila, N. Lal, W. R. Webber, G. Jóhannesson, I. V. Moskalenko, E. Orlando and T. A. Porter, Astrophys. J. 831, no.1, 18 (2016)

[36] Kouichi Yamaguchi, Kenji Sakamoto, Toshio Akabane, Yoshiji Fujimoto, (1990): A neural network for speaker-independent isolated word recognition, In ICSLP-1990, 1077-1080.

[37] J. Liu, Q. Yuan, X. J. Bi, H. Li and X. Zhang, Phys. Rev. D 85, 043507 (2012) [arXiv:1106.3882 [astro-ph.CO]].

[38] Daeho Kim, Jinah Kim, Jaeil Kim, Elastic exponential linear units for convolutional neural networks, Neurocomputing, Volume 406,2020

[39] Diederik P. Kingma and Jimmy Lei Ba, ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION, [arXiv:1412.6980v9 [cs.LG]].

[40] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[41] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, [arXiv:1406.2661 [stat.ML]].

[42] A. Butter, T. Plehn and R. Winterhalder, SciPost Phys. 7, no.6, 075 (2019) doi:10.21468/SciPostPhys.7.6.075 [arXiv:1907.03764 [hep-ph]].

[43] J. Wu and H. Chen, Phys. Lett. B 789, 292-299 (2019) doi:10.1016/j.physletb.2018.11.052 [arXiv:1809.04905 [astro-ph.HE]].

[44] N. Arkani-Hamed, G. L. Kane, J. Thaler and L. T. Wang, JHEP 08 (2006), 070 [arXiv:hep-ph/0512190 [hep-ph]].