Journal of Physics A: Mathematical and Theoretical

PAPER

Classification of stochastic processes by convolutional neural networks

IOP Publishing Journal of Physics A: Mathematical and Theoretical

J. Phys. A: Math. Theor. **55** (2022) 274006 (33pp) https://doi.org/10.1088/1751-8121/ac73c5

Classification of stochastic processes by convolutional neural networks

Eman A AL-hada, Xiangong Tang and Weihua Deng*

School of Mathematics and Statistics, Gansu Key Laboratory of Applied Mathematics and Complex Systems, Lanzhou University, Lanzhou, 730000, People's Republic of China

E-mail: dengwh@lzu.edu.cn

Received 7 January 2022, revised 11 May 2022
Accepted for publication 26 May 2022
Published 24 June 2022

Abstract

Stochastic processes (SPs) appear in a wide field, such as ecology, biology, chemistry, and computer science. In transport dynamics, deviations from Brownian motion leading to anomalous diffusion (AnDi) are found, including transport mechanisms, cellular organization, signaling, and more. For various reasons, identifying AnDi is still challenging; for example, (i) a system can have different physical processes running simultaneously, (ii) the analysis of the mean-squared displacements (MSDs) of the diffusing particles is used to distinguish between normal diffusion and AnDi. However, MSD calculations are not very informative because different models can yield curves with the same scaling exponent. Recently, proposals have suggested several new approaches. The majority of these are based on the machine learning (ML) revolution. This paper is based on ML algorithms known as the convolutional neural network to classify SPs. To do this, we generated the dataset from published paper codes for 12 SPs. We use a pre-trained model, the ResNet-50, to automatically classify the dataset. Accuracy of 99% has been achieved by running the ResNet-50 model on the dataset. We also show the comparison of the Resnet18 and GoogleNet models with the ResNet-50 model. The ResNet-50 model outperforms these models in terms of classification accuracy.

Keywords: stochastic process, anomalous diffusion, classification, CNN

(Some figures may appear in colour only in the online journal)

1. Introduction

Stochastic processes (SPs) are mathematical objects usually defined as a set of random variables. SPs are commonly used as a mathematical model of systems and phenomena that vary

*Author to whom any correspondence should be addressed.

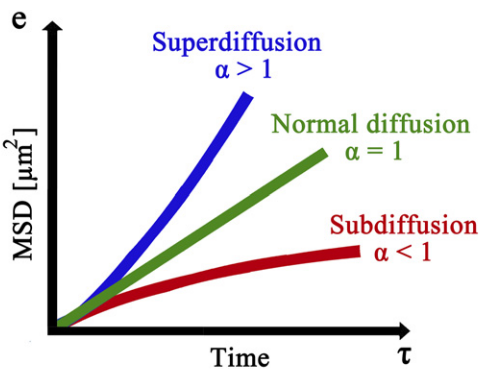

Figure 1. Displays examples of MSDs for normal diffusion (green line), super-diffusive (blue line) motion, and sub-diffusive (red line).

randomly. Examples include a gas molecule's movement, an electrical current fluctuating because of thermal noise, or a bacterial population's growth [23, 55]. SPs have applications in several disciplines, such as physics [56], biology [6], computer science [2], neuroscience [37], signal processing [21], and cryptography [31]. Furthermore, in finance, the apparent random change in financial business markets has motivated the use of SPs [65].

Recently, anomalous diffusion (AnDi) was discovered in numerous systems, for example, telomeres in a nucleus of cells [7], extreme-cold atoms [61], and colloidal particles in the cytoplasm [60]. Furthermore, in many biological systems were found AnDi, including DNA sequences and heartbeat intervals [8]. In addition, experiments achieved using one particle tracking [45] have revealed that chemically identical molecules can show different behaviors in biological media due to the complex environment in which diffusion occurs. For example, AnDi happens in the transport of molecules within the nucleus [19] and the cytosol [60] and the lipids diffusion and receptors in the cell membranes [36].

To distinguish between normal diffusion and AnDi is often through analysis of the mean-squared displacement (MSD) for diffusing particles, which rises linearly over time (MSD $\propto t$). The MSD is the ensemble average over a group of tracers (EA-MSD). When long tracks are available, the MSD can be obtained instead as a time average from the trajectory of a single tracer (TA-MSD) [48]. In normal diffusion $\alpha = 1$, while AnDi could be super-diffusive ($\alpha > 1$) or sub-diffusive ($\alpha < 1$) (figure 1) [9, 33].

Problematically, due to the stochastic nature of diffusional processes, unless a trajectory is long enough to display asymptotic behavior, a single trajectory cannot uniquely correspond with a type of diffusion. For example, continuous-time random walk (CTRW) and fractional Brownian motion (FBM) can both generate similar motion characteristics [48, 63]. Despite this, they come from two different fundamental biological mechanisms. Furthermore, in several instances, the trajectories of the actual are very short to extracting meaningful information from time-averaged MSDs. In addition, we see a lot of typical experiments also aimed to deduce the exponent of AnDi α to determine the model of underlying diffusion, which the MSD often estimates [33]. But, because the models with different physical characteristics can give the same exponent, it severely limits the unmistakable determination of the fundamental dynamics, based only on the MSD evaluation. Furthermore, using MSD to determine α can introduce significant errors and basis: (i) the estimation accuracy depends on fluctuations, which can just be decreased by increasing the tracers' number for EAMSD or trajectory length for TA-MSD, which is frequently not possible due to practical constraints; (ii) the α value is biased by noise, such as, the experimental trajectories localization precision [79], which must be estimated independently to introducing the proper correction [33, 46]; (iii) the MSD behavior from its asymptotic limit at short times or time lags may differ from its asymptotic limit; thus for the α correct estimation, required long trajectories [48].

Here, the difficulty arises in determining the underlying diffusion model, which is related to its physical driving mechanism. A lot of effort has been made to categorize experimental data that shows anomalous transport to resolve this ambiguity. For example, it has been suggested that alternative estimators can be used [41] to determine whether Golding and Cox's results are pioneering [24], arising from a CTRW [18, 34, 49] or FBM [44]. In addition, the moment scaling spectrum approach can be used to classify different modes of motion [22]. The fractionally integrated moving average framework [9], the mean maximum excursion method [67], and the distribution of directional changes [80] may efficiently substitute the estimator of MSD for the purposes of classification.

Recent advances in machine learning (ML) methods have increased the availability of new powerful data analysis tools and broadened the available method palette [5, 13, 25, 35]. Promisingly, many groups have applied deep-learning [20, 35] and ML [47] algorithms to classify the trajectories as normal, confined, and directed diffusion, showing more advantages over traditional methods. Many works have been focused on AnDi; e.g., Thapa *et al* [68] utilized the Bayesian analysis approach to discriminate between FBM, Brownian motion, and Brownian motion with diffusing diffusivity, further demonstrating its applicability on the mucus hydrogels biological data [12]. In another work, Munõz-Gil *et al* [52] used an algorithm of random forest to categorize a given trajectory as one of many the models of AnDi and to estimate the exponent of AnDi, with a 0.1 resolution, achieving an accuracy of 70%–90%, depending on noise and trajectory length. Monnier *et al* [50] used the Bayesian methods for MSD-based motion modes classification. Wagner and co-workers [70] created a random forest classifier for anomalous, normal, directed, and confined diffusion. Dosset *et al* [20] used a simple neural network (NN) of back-propagation to distinguish between different diffusion types. Qualitatively distinguish among directed, anomalous, normal, or confined motion [62]. Granik and *et al* [25] implemented a NN to categorize single-particle trajectories 'BM, RW, FBM and CTRW'. An open competition called the AnDi challenge had been conducted. It is divided into three different tasks: 'model classification, anomalous exponent inference, and trajectory segmentation' [51].

Here, we briefly introduce the diffusive models considered in this paper. For more details, we refer readers to appendix A; see simulation results: figures A1–A13, and the used algorithms: algorithms 1–11 given in appendix B. The first major class of diffusive models we consider is Lévy process. The Lévy process is a SP which has independent and stationary increments. The most representative Lévy processes are stable symmetric Lévy processes (including Brownian motion), non-decreasingly stable subordinators, and integer-valued Poisson processes. The second major class is CTRW, which has been widely used to model AnDi. In this paper, we focus on the four specific models, called CTRW1965, CTRW196502, CTRW196503, and CTRW196504. For CTRW1965, it has finite characteristic waiting time and jump length variance. Hence, its MSD is proportional to time. The second model, CTRW196502, with infinite characteristic waiting time (power-law) and finite jump length variance, is usually used to approximate the subdiffusion. And the superdiffusion can be modeled by CTRW196503, which also called the Lévy flight. At last, we also consider the case of infinite characteristic waiting time and jump length variance, CTRW196504. Besides, we also consider the FBM, alternating process, and multi-state processes here.

In this study, we classify the SPs using ML algorithms. We do our experiments on dataset generated from published research codes. The dataset is for 12 sub-SPs from five main SPs:

(i) alternating process 'LWandBM2019', (ii) Gaussian process 'fBm1968', (iii) Lévy process 'BM1905, beta-stable subordinator, homogeneous Poisson process, symmetric stable process', (iv) multi-state process 'CTRW2017, LW2018', (v) random walk 'CTRW1965, CTRW196502, CTRW196503, CTRW196504' (For more details, see appendix); as each process contains 4000 images, in total 48 000 images.

2. Convolutional neural network (CNN)

ML algorithms are used to learn the relationship of underlying data and make decisions without the need for explicit instructions [28]. ML methods may outperform human experts in many tasks without requiring human effort. These methods have gained popularity in recent years, and they have been applied to natural language processing [81] and speech recognition [17].

In 1989, a new NN type, called a convolutional neural network (CNN), was reported in [38]. CNN exhibits enormous potential in the tasks related to machine vision; it is a structure that is designed as a series of stages produced by layers. The first phases have two layers, namely, assembly and convolutional layers; the classification performance of the features that are extracted using layers is fully connected at the end of a network's structure [26]. In addition, the primary structure of CNN consists of five layers: (i) the input layer, (ii) the convolutional layer, (iii) the pooling layer, (iv) the fully connected layer and (v) the output layer (figure 2).

Different improvements in the CNN architecture have been reported from 1989 until the present. AlexNet is the first large-scale CNN architecture for obtaining good ImageNet classification results. The innovation of this network is its successful application of the rectified linear unit (ReLU) activation function, along with the use of the dropout mechanism and a data enhancement strategy to prevent overfitting. A local response normalization layer is used in the network to improve model generalization. Furthermore, the maximum pooling of the overlap is used to avoid the effect of blurring caused by average pooling.

The visual geometry group (VGG) of the University of Oxford proposed the VGG network. This network uses a deeper network structure with depths of 19, 16, 13 and 11 layers. The VGG network employs a small convolution kernel (33 pixels) instead of a large one, reducing parameters whilst increasing the expressive power of the network [62]. GoogLeNet is a deep NN model based on the inception module launched by Google. This network introduces an initial structure to increase the width and depth of a network whilst removing the fully connected layer and using average pooling instead of the fully connected layer to avoid the disappearance of the gradient. To carry the gradient forward, the network adds two more softmax [66]. A residual NN (ResNet) is a deep NN that uses a residual structure to solve the 'degradation' problem. ResNet uses multiple parameter layers to learn the representation of residuals between the input and the output instead of using parameter layers to try to learn mapping directly between the input and the output, such as in VGGs. ResNet is distinguished by its ease of optimization, and it can improve accuracy by adding a significant depth [27].

In addition, the random forest algorithm is an ensemble learning method for regression, classification, and other tasks; it operates by constructing a multitude of decision trees during training. For regression tasks, the average or mean prediction of individual trees is returned. For classification tasks, the output of the random forest is the class selected by most trees. Deep NN algorithms and the random forest algorithm are techniques that learn differently but can be used in similar domains. The random forest algorithm is used when data are structured and a dependent variable must be classified into a certain category. Meanwhile, deep NN algorithms are used when data are huge and mostly unstructured.

In the current study, the following three CNN methods are used:

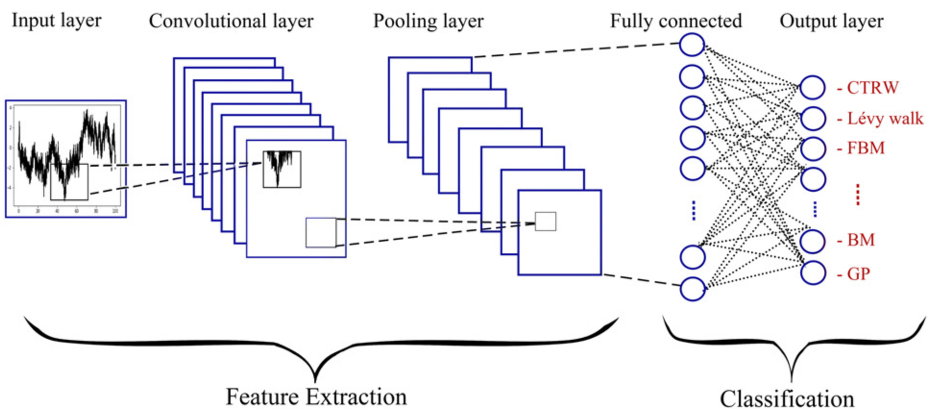

Figure 2. Design of CNN.

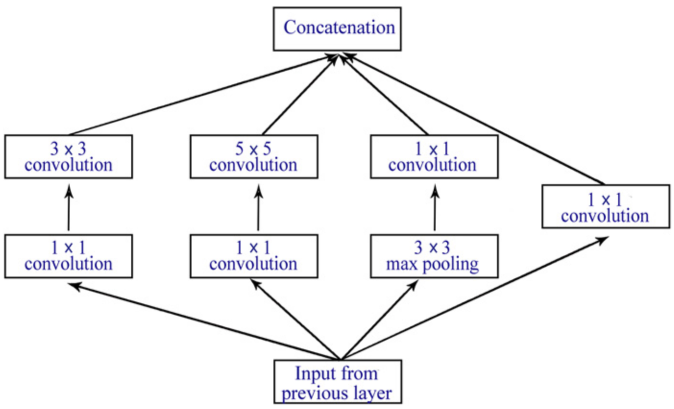

Figure 3. Basic architecture of the inception block showing the split, transform, and concept.

(a) GoogLeNet is a pretrained CNN with 22 layers [66]. It is trained on two datasets: Places365 and ImageNet [78]. The GoogLeNet version trained on ImageNet can classify images up to 1000 classes and that trained on Places365 can classify images up to 365 classes. The size of input images for the two networks is 224×224. The architecture of the inception block is shown in figure 3. In GoogLeNet, conventional convolutional layers are replaced with small blocks similar to the concept of substituting each layer with micro NN as proposed in network-in-network architecture. This block encapsulates filters of different sizes (1×1, 3×3 and 5×5) to capture spatial information at various scales. GoogLeNet regulates computations by adding a bottleneck layer of 1×1 convolutional filter before employing large kernels. Furthermore, it uses sparse connections to overcome the problem of redundant information and reduce cost by omitting irrelevant feature maps. In addition, the connection's density is reduced by using global average pooling at the last layer instead of a fully connected layer. These parameter tunings cause a significant decrease in the number of parameters from 138 million to 4 million parameters. However, the major drawback of the GoogLeNet is its heterogeneous topology that

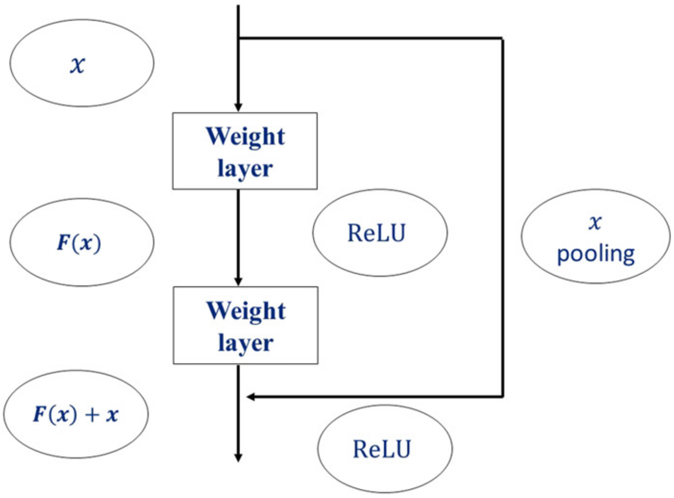

Figure 4. The ResNet building block.

requires to be customized from module to module. Another limitation of GoogLeNet is a representation bottleneck that drastically reduces feature space in the next layer and may occasionally lead to loss of useful information.

(b) ResNet is an artificial NN that uses a technique called skip connections. This technique skips training from a few layers and connects directly to the output. Typical ResNet models are implemented with double- or triple-layer skips that contain nonlinearities (ReLU) and batch normalization in between [27] (figure 4). The approach behind this network is as follows: instead of layers that learn underlying mapping, the network is allowed to fit into residual mapping. Skipping connections has two major reasons: to avoid the problem of vanishing gradients or to mitigate the degradation (accuracy saturation) problem wherein adding more layers to a suitably deep model leads to higher training error.

Figure 4 depicts a ResNet building block with an input parameter and target output $H(x)$. The block employs a shortcut connection, enabling it to learn the residual $F(x) = H(x) - x$ directly to generate the target output $[F(x) + x]$, avoiding the problems of performance degradation and accuracy reduction due to an excessive number of convolutional layers. In addition, the advantage of adding this type of skip connection is that if any layer negatively affects the performance of the architecture, then it will be skipped through regularization, resulting in training an extremely deep NN without the problems caused by a vanishing/exploding gradient. By using the ResNet building block shown in figure 4, ResNets with 18 and 50 layers, called ResNet-18 and ResNet-50, respectively, are proposed and evaluated. These networks can categorize images into 1000 classes.

We perform our experiments by using our data set, of which 80% of the data set is used for training, and the remaining 20% is used for test. We employ Adam's optimization algorithm approach to find the appropriate biases and weights for the NN to decrease the loss function. Adam's algorithm works by selecting a small number from training inputs during learning. The batch size is set at 10. In addition, the learning rate is set at 0.000 01. A small learning rate yields precise results; however, it necessitates a longer training time. These parameters are obtained empirically based on a series of trials on the dataset that produced the best classification results.

When evaluating the classification performance of the methods, we consider the most commonly used metrics such as train and test accuracies, precision, recall, and F1-score. Accuracy means how well the model predicts given input data. While the train accuracy metric is the accuracy of the CNN model over the training set, the test accuracy metric is calculated on the test set. Train loss and test loss values are of loss function outputs over the training and test sets. Precision is the proportion of correct results in all the returned results. Recall, also called sensitivity, is the proportion of the correct predictions to the total number of correct results that could have been returned. F1-score is the harmonic mean of precision and recall. All formulas of these metrics are given in the following equations, respectively.

$$\text{Accuracy (Acc)} = \frac{\text{True positives} + \text{True negatives}}{\text{Total data}}, \tag{1}$$

$$\text{Precision (P)} = \frac{\text{True positive}}{\text{True positive} + \text{False positive}}, \tag{2}$$

$$\text{Recall (R)} = \frac{\text{True positive}}{\text{True positive} + \text{False negative}}, \tag{3}$$

$$F1 - \text{score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{recall}}, \tag{4}$$

where true positive refers to the case a positive sample being predicted positive; true negative denotes the case of a negative sample being predicted negative; false positive denotes the case that a negative sample being predicted positive; false negative refers to the case that a positive sample being predicted negative.

3. Experimental setup

Experiments are conducted on a Lenovo Air 13IWL PC with 8 GB of RAM, CPU: Intel (R) Core (TM) i5-8265U, clocked at 1.60 GHz–1.80 GHz. It runs Microsoft Windows 10 64-bit operating system. We use the development language Python and ML library PyTorch to implement CNN.

4. Results and discussions

This paper introduces a multi-label classification for 12 SPs. All of the datasets are trained by three pre-trained CNNs with hyper-parameters. All the models show good results.

Figures 5 and 6 show the training and test accuracies and loss values obtained at the end of each epoch for the models. We can see that the training and test accuracies for the GoogLeNet model increase with increasing epoch until they remain almost constant. The test accuracy always shows better performance than the training accuracy (figure 5(c)). In addition, the training and test losses decrease until the end (figure 6(c)). The training and test accuracy of the ResNet-18 model increase with small fluctuations in between them (figure 5(b)). Furthermore, the training and test losses continue to decrease with minor fluctuations (figure 6(b)). Also, we can see small fluctuations between the training and test accuracies for the ResNet-50 model until 51 epochs, then the test accuracy shows better performance than the training accuracy to the end (figure 5(a)). The training and test losses are also decreasing to the end. Compared to

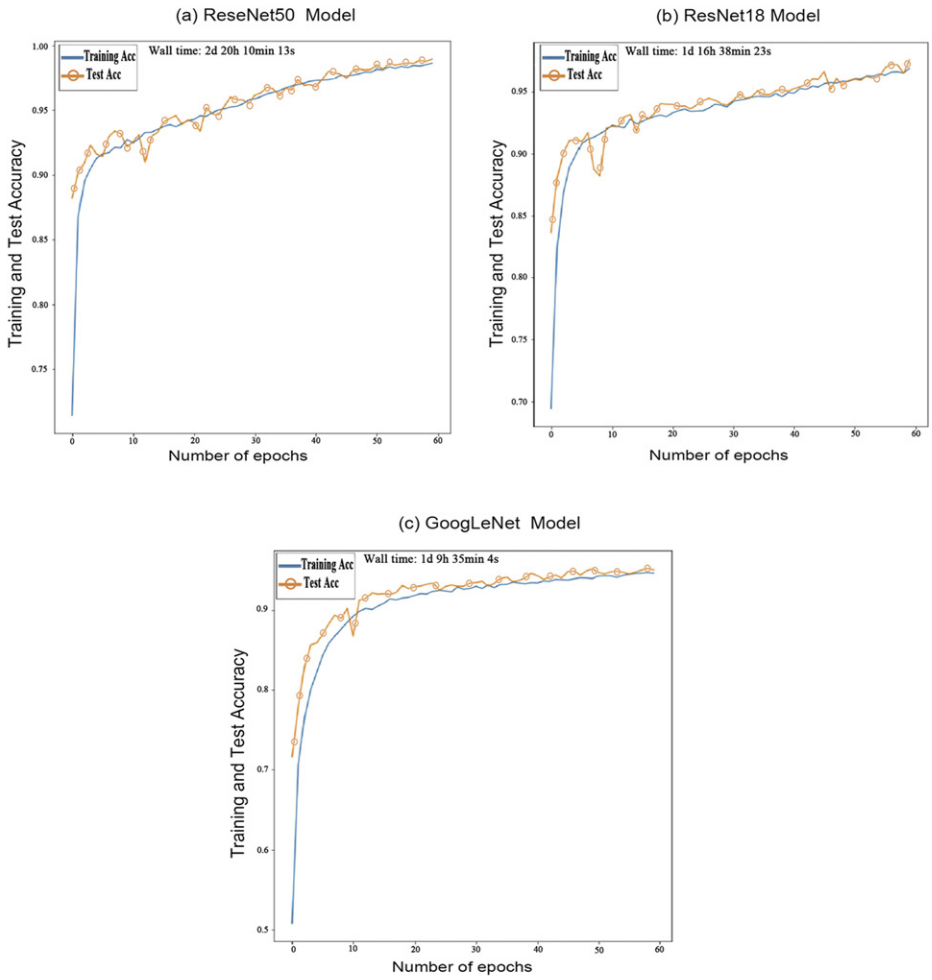

Figure 5. Graph of accuracy during training and test for all models.

the learning time between the GoogLeNet, ResNet-18, and ResNet-50 models, it can be seen that the training of the ResNet-50 model takes a bit more time.

Figure 7 displays the confusion matrix of all the models, which can be used to study individual misclassification rates. Columns represent the predicted classes, and rows represent the instances of known classes. The square matrix puts all the correct classifications along the upper-left to lower-right diagonal. It is observed from the confusion matrix that the models generally have no difficulty in identifying most classes. In comparison, some classes perform slightly worse, for example, random walk 'CTRW1965, CTRW196502, CTRW196503, CTRW196504'. However, in ResNet-50, these class-wise confusions are significantly reduced (figure 7(c)).

Table 1 presents the performance per class, which compares the models' performance results for every class. The models are able to provide good results for recall (R) and precision (P) rates. However, as can be noticed, the random walk-CTRW196504 is often mistaken for the random walk 'CTRW196502, and CTRW196503'. In addition, the random walk-CTRW196503 for

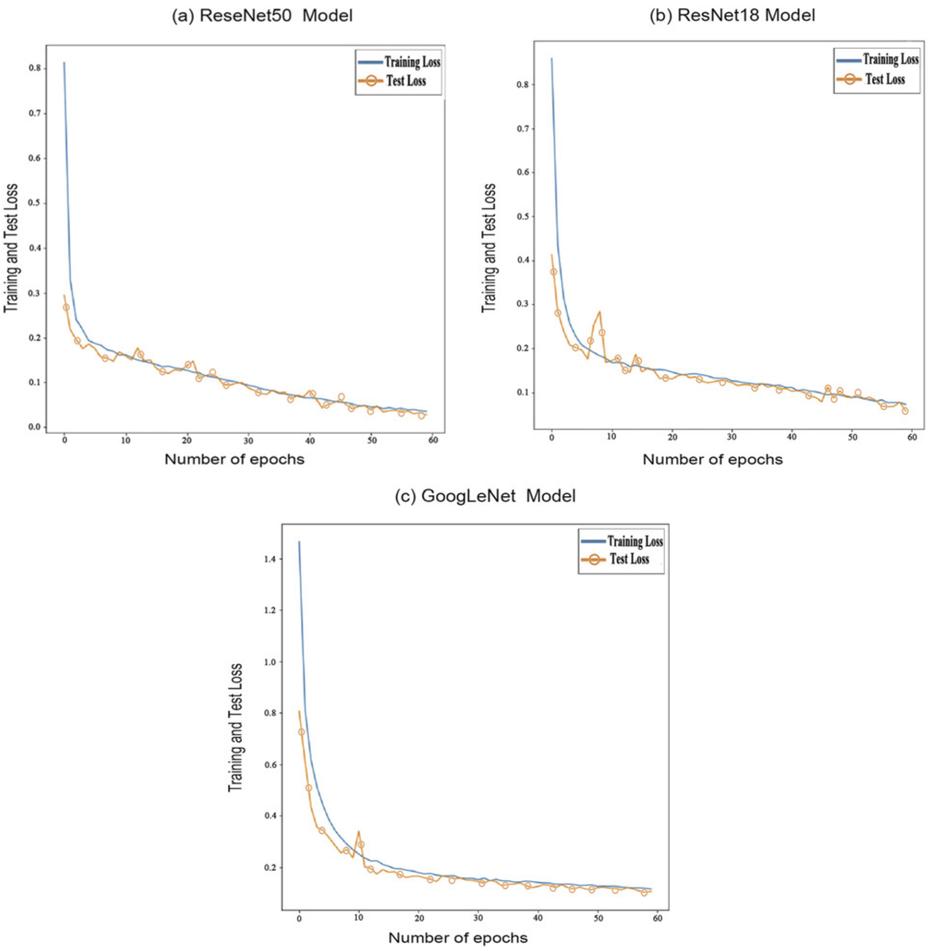

Figure 6. Graph of loss during training and test for all models.

'CTRW1965, CTRW196504', which is reasonable due to their similarity. Significantly, the image misclassification error rates in the ResNet-50 model drop more than 24% and 16% for the random walk-CTRW196504 compared with the GoogLeNet model and ResNet-18 model, respectively. In addition, it drops more than 10% and 7% for the random walk-CTRW196503 compared with the GoogLeNet model and ResNet-18 model, respectively. In general, the ResNet-50 model generally has no difficulty in identifying all classes.

Following completion of the classification process, the results recorded from the classification process for all algorithms have been summarized in table 2. The GoogLeNet and ResNet-18 models achieved 95% and 97% training accuracies, respectively. In addition, the training losses are 12% and 7%, respectively. While, for the test stage, the overall accuracies achieved by GoogLeNet and ResNet-18 is 95% and 98%, with a loss of 11% and 6%, respectively. But the ResNet-50 model surpasses all of them and gives the highest accuracy, achieving 99% for training and test accuracies. In addition, it gives less loss, achieving 4% and 3% for training and test losses, respectively.

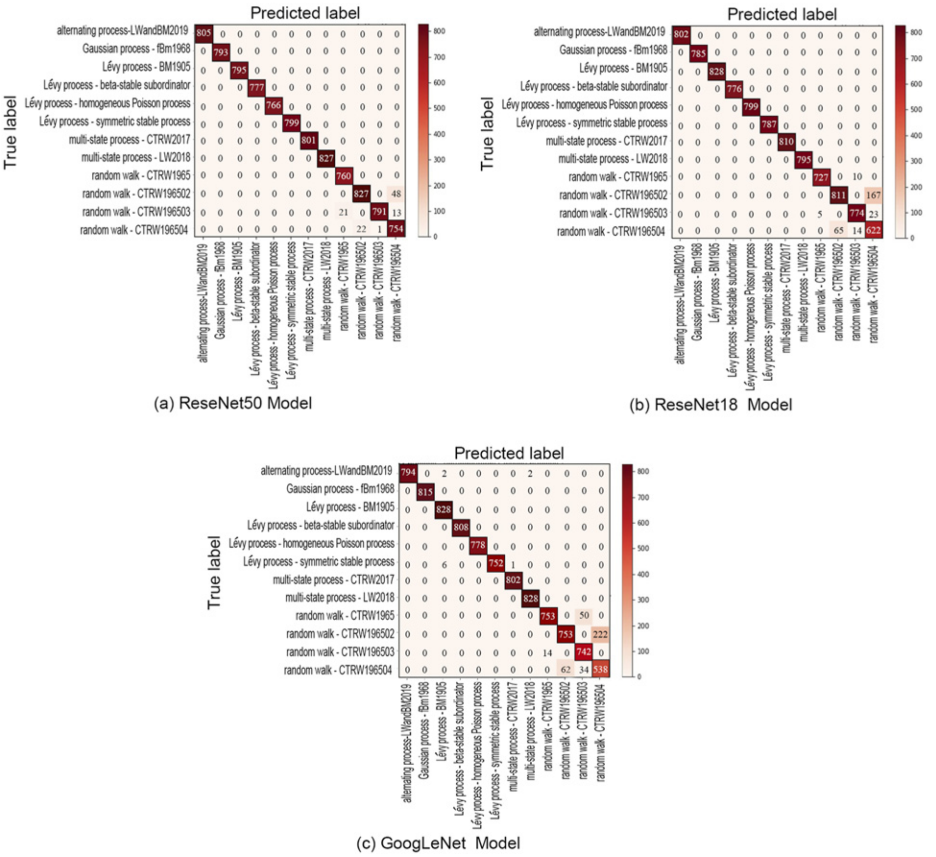

Figure 7. Confusion matrices for for all models.

Table 1. A brief summary of precision, recall and F1-score values obtained from the models, for 12 process classification. All results are rounded to two decimal digits.

Classes	ResNet-50			ResNet-18			GoogLeNet		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
0	100%	100%	100%	100%	100%	100%	100%	99%	100%
1	100%	100%	100%	100%	100%	100%	100%	100%	100%
2	100%	100%	100%	100%	100%	100%	99%	100%	100%
3	100%	100%	100%	100%	100%	100%	100%	100%	100%
4	100%	100%	100%	100%	100%	100%	100%	100%	100%
5	100%	100%	100%	100%	100%	100%	100%	100%	100%
6	100%	100%	100%	100%	100%	100%	100%	100%	100%
7	100%	100%	100%	100%	100%	100%	100%	100%	100%
8	97%	100%	99%	99%	99%	99%	98%	94%	96%
9	97%	95%	96%	93%	83%	87%	92%	77%	84%
10	100%	96%	98%	97%	97%	97%	90%	96%	93%
11	93%	97%	95%	77%	89%	82%	69%	85%	76%

Table 2. Summary of accuracy and loss in the three models, for 12 classes classification. All results are rounded to two decimal digits.

Model name	Train accuracy	Train loss	Test accuracy	Train loss
ResNet-50	99%	4%	99%	3%
ResNet-18	93%	7%	98%	6%
GoogLeNet	95%	12%	96%	11%

5. Conclusion

SPs are often observed in the natural world. This paper focuses on SPs classification using ML algorithms known as CNNs. We create the dataset for 12 sub-SPs from five major processes (alternating process, Gaussian process, Lévy process, multi-state process, and random walk). For training the CNNs, pre-trained models (GoogLNet, ResNet-18, and ResNet-50) are used, and the last convolutional layers are fine-tuned corresponding to the number of outputs. Despite all models performing excellently on the datasets, ResNet-50 performs best compared with other models. However, this comes at the expense of much longer training hours.

Acknowledgments

This work was supported by National Natural Science Foundation of China under Grant No. 12071195, AI and Big Data Funds under Grant No. 2019620005000775, NSF of Gansu under Grant No. 21JR7RA537, and Supercomputing Center of Lanzhou University.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/tangxiangong/ClassTop.

Appendix A. Models

In this appendix, we present a brief introduction to the concepts of SPs involved in this paper. We provide theoretical insights about the normal and AnDi models considered in this classification, as well as the description of the pseudocode used for simulations (see appendix B).

A.1. Stable distribution and power-law distribution

Before introducing the diffusion models, let us take a look at the stable distribution [1, 32, 43, 53, 54] and the power-law distribution [53, 54]. Let X_1 and X_2 be independent copies of a random variable X. Then X is said to be stable if for any positive constants a and b the random variable $aX_1 + bX_2$ has the same distribution as $cX + d$ for some constants $c > 0$ and d. Furthermore, a random variable X is stable if and only if its characteristic function $\mathbb{E}\left[\mathrm{e}^{\mathrm{i}kX}\right]$ can be written as [1, 32, 34]

$$\varphi(k;\alpha,\beta,\mu,\sigma) = \exp\left(-\sigma^{\alpha}|k|^{\alpha}\left(1 - \mathrm{i}\beta\omega(k;\alpha,\sigma)\,\mathrm{sgn}(k)\right) + \mathrm{i}\mu k\right) \tag{A.1}$$

with

$$\omega(k;\alpha,\sigma) = \begin{cases} \left(|\sigma k|^{1-\alpha} - 1\right)\tan\left(\dfrac{\pi\alpha}{2}\right), & \alpha \neq 1, \\ -\dfrac{2}{\pi}\ \ln|\sigma k|, & \alpha = 1, \end{cases} \tag{A.2}$$

where $\alpha \in (0, 2]$, is the index of stability, $\beta \in [-1, 1]$, called the skewness parameter, is a measure of asymmetry, $\mu \in \mathbb{R}$ is a shifted parameter, and $\sigma > 0$ is called the scaling parameter (without loss of generality, we will take $\sigma = 1$ in the following text).

When $\mu = \beta = 0$, it is called the symmetric α-stable distribution, and when $\alpha < 1$ and $\beta = 1$, it is called one-sided (or totally skewed) α-stable distribution. In this paper, we take $\mu = 0$ for totally skewed stable distribution, whose distribution is supported by $[0, \infty)$.

When $\alpha = 2$, the stable distribution is the normal distribution. If $\alpha < 2$, the stable distribution is power-law, that is, the probability density function (PDF) of symmetric α-stable distribution $L_\alpha(x)$, and the PDF of totally skewed α-stable distribution $S_\alpha(t)$ have the following asymptotic formulae [16, 53, 54]

$$L_\alpha(x) \propto \frac{1}{|x|^{1+\alpha}}, \quad |x| \to \infty, \tag{A.3}$$

$$S_\alpha(t) \propto \frac{1}{t^{1+\alpha}}, \quad t \to \infty. \tag{A.4}$$

To generate a symmetric α-stable random variable X, we can first generate a random variable V uniformly distributed on $(-\frac{\pi}{2}, \frac{\pi}{2})$ and an exponential random variable W with mean 1. Then compute X as following [30, 73]

$$X = \frac{\sin(\alpha V)}{(\cos\ V)^{\frac{1}{\alpha}}} \cdot \left(\frac{\cos(V - \alpha V)}{W}\right)^{\frac{1-\alpha}{\alpha}}. \tag{A.5}$$

Similarly, we can get a totally skewed α-stable S as below [30, 73],

$$S = c_1 \frac{\sin(\alpha(V + c_2))}{(\cos\ V)^{\frac{1}{\alpha}}} \cdot \left(\frac{\cos(V - \alpha(V + c_2))}{W}\right)^{\frac{1-\alpha}{\alpha}}, \tag{A.6}$$

where $c_1 = \left(\cos\frac{\pi\alpha}{2}\right)^{\frac{-1}{\alpha}}$ and $c_2 = \frac{\pi}{2}$.

The simulation algorithms for symmetric and totally skewed Stable Distribution are algorithms 2 and 3 respectively. In addition, one can also use the stable distribution in the MATLAB Statistic and Machine Learning Toolbox or levy_stable in the Python package SciPy to generate random numbers of stable distribution.

A.2. Continuous-time random walk

The CTRW model [14, 18, 34, 49] is based on the idea that the length of a given jump, as well as the waiting time elapsing between two successive jumps of a particle are drawn from a PDF $\psi(x, t)$. From $\phi(x, t)$, the jump length PDF $\lambda(x)$ and the waiting time PDF $\psi(t)$ can be deduced by marginal PDF. Here, we consider decoupled CTRW, that is, the waiting time and jump length are independent, $\phi(x, t) = \lambda(x)\psi(t)$. If $x(t)$ denotes the position of particle with initial position $x(0) = 0$, then

$$x(t) = \sum_{k=1}^{N(t)} \xi_k \tag{A.7}$$

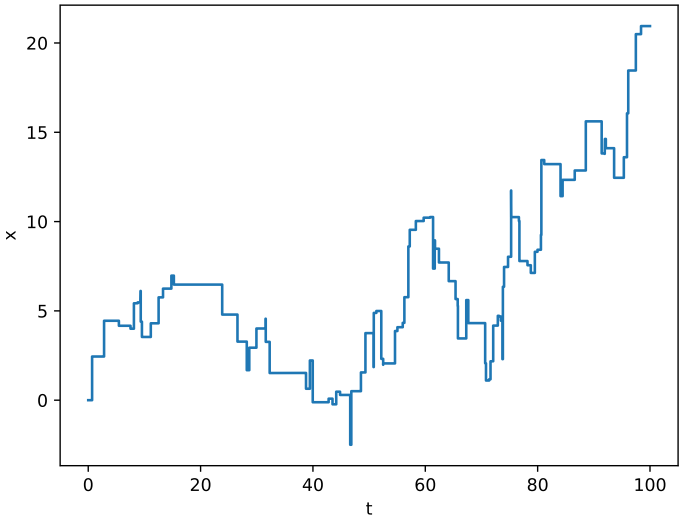

Figure A1. CTRW with finite characteristic waiting time and jump length variance.

with

$$N(t) = \max\left\{n \in \mathbb{N} : \sum_{k=1}^{n} \tau_k \leqslant t\right\}, \tag{A.8}$$

where waiting times $\{\tau_n\} \overset{\text{i.i.d.}}{\sim} \psi(t)$ and jump lengths $\{\xi_n\} \overset{\text{i.i.d.}}{\sim} \lambda(x)$.

Different types of CTRW processes can be categorised by the characteristic waiting time (mean-value of waiting time)

$$\int_0^{\infty} t\psi(t)\mathrm{d}t \tag{A.9}$$

and the jump length variance

$$\int_{\mathbb{R}} x^2\lambda(x)\mathrm{d}x \tag{A.10}$$

being finite or diverging [49], respectively. Therefore, we consider four cases of CTRW model. For finite and diverging characteristic waiting time, we choose exponential distribution and totally skewed β-stable distribution (power-law), respectively. For finite and diverging jump length variance, we choose symmetric α-stable distribution with $\alpha < 2$ and normal distribution ($\alpha = 2$), respectively.

For the finite characteristic waiting time and jump length variance CTRW model (see figure A1), it can model the normal diffusion in the sense of scaling limit with MSD $\langle x^2(t)\rangle \propto t$, where $\langle\cdot\rangle$ denotes the ensemble average. For the finite characteristic waiting time and diverging jump length variance CTRW model (see figure A3), also called the Lévy flight, its MSD is diverging. But we can regard $\langle|x(t)|^{\delta}\rangle \propto t^{\frac{\delta}{\alpha}}$ as MSD for $0 < \delta < \alpha \leqslant 2$ [49]. For the diverging characteristic waiting and finite jump length variance CTRW model (see figure A2), its MSD $\langle x^2(t)\rangle \propto t^{\beta}$ [49]. However, the equations governing PDFs for these four situation have

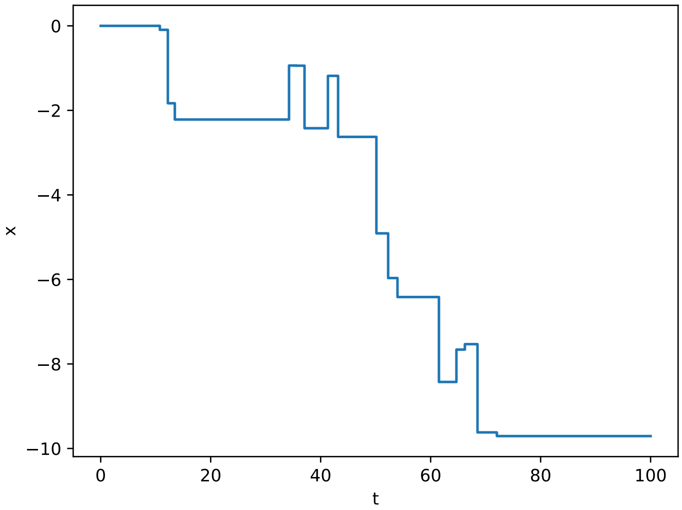

Figure A2. CTRW with diverging characteristic waiting time and finite jump length variance.

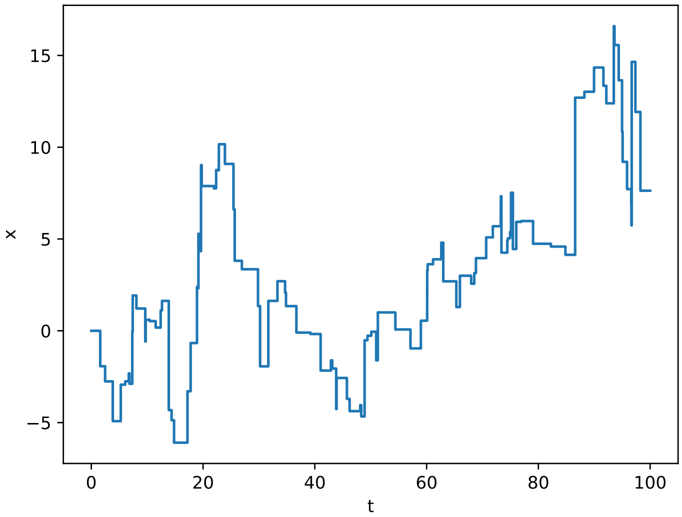

Figure A3. CTRW with finite characteristic waiting time and diverging jump length variance.

a unified form [11, 18, 49, 69], as below,

$$\frac{\partial}{\partial t}P(x,t) = KD_t^{1-\beta}\Delta^{\frac{\alpha}{2}}P(x,t), \tag{A.11}$$

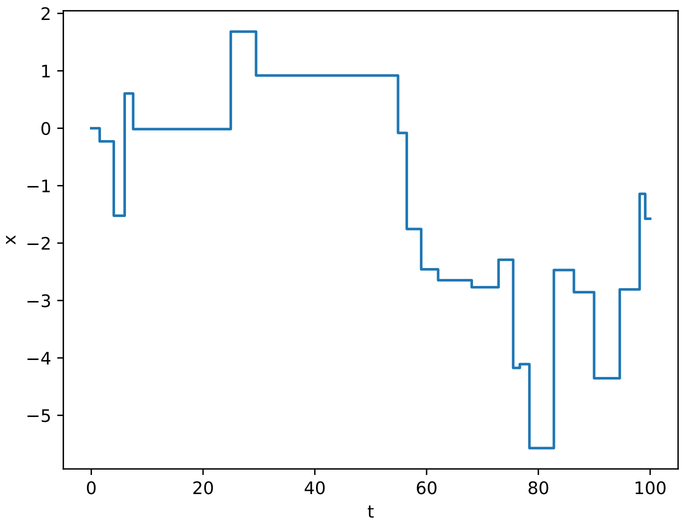

Figure A4. CTRW with diverging characteristic waiting time and jump length variance.

where K is the generalized diffusion coefficient, $D_t^{1-\beta}$ is the Riemann–Liouville fractional derivative operator [4, 57, 83] of oder $\beta \leqslant 1$,

$$D_t^{1-\beta} f(t) = \frac{\mathrm{d}}{\mathrm{d}t} \int_0^t \frac{(t-s)^{\beta-1}}{\Gamma(\beta)} f(s)\mathrm{d}s, \tag{A.12}$$

and $\Delta^{\frac{\alpha}{2}}$ is the fractional Laplace operator [39, 58]. Therein, $\beta = 1$ corresponds to the exponential waiting time, and $\alpha = 2$ corresponds to normal jump length. The forward Feynman–Kac equation [10, 11, 18, 49, 69, 77] is

$$\frac{\partial}{\partial t}\widehat{G}(x,\rho,t) = K\Delta^{\frac{\alpha}{2}}\mathcal{D}_t^{1-\beta,\kappa(x),\rho}\widehat{G}(x,\rho,t) + \mathrm{i}\rho\kappa(x)\widehat{G}(x,\rho,t), \tag{A.13}$$

where $G(x,A,t)$ is the joint PDF of $x(t)$ and its integral functional $A(t) = \int_0^t \kappa(x(s))\mathrm{d}s$ with prescribed function $\kappa(x)$, $\widehat{G}$ is the Fourier transform of G,

$$\widehat{G}(x,\rho,t) = \int_{\mathbb{R}} \mathrm{e}^{\mathrm{i}\rho A} G(x,A,t)\mathrm{d}A, \tag{A.14}$$

and $\mathcal{D}_t^{1-\beta,\kappa(x),\rho}$ is the fractional substantial derivative operator [18, 84, 85],

$$\mathcal{D}_t^{1-\beta,\kappa(x),\rho} u(x,t) = \left(\frac{\partial}{\partial t} - \mathrm{i}\rho\kappa(x)\right) \int_0^t \frac{\mathrm{e}^{\mathrm{i}\rho\kappa(x)(t-s)}}{\Gamma(\beta)(t-s)^{1-\beta}} u(x,s)\mathrm{d}s. \tag{A.15}$$

The simulation algorithm for these four CTRW models is given by algorithm 5.

A.3. Lévy process

Let $\boldsymbol{X} = \boldsymbol{X}(t,\omega)$ be a d-dimensional SP defined on a probability space $(\Omega, \mathscr{F}, \mathbb{P})$. We say that it has independent increments if for each $n \in \mathbb{N}$ and each $0 \leqslant t_1 < t_2 < \ldots < t_{n+1} < \infty$ the random variables $\{\boldsymbol{X}(t_{j+1}) - \boldsymbol{X}(t_j) : 1 \leqslant j \leqslant n\}$ are independent and that it has stationary increments if each $\boldsymbol{X}(t_{j+1}) - \boldsymbol{X}(t_j) \stackrel{d}{=} \boldsymbol{X}(t_{j+1} - t_j) - \boldsymbol{X}(0)$, where the symbol $\stackrel{d}{=}$ denotes identical distribution. Then $\boldsymbol{X}$ is a Lévy process [1, 32] if

(a) $\boldsymbol{X}(0) = 0$, a.s.;
(b) $\boldsymbol{X}$ has independent and stationary increments;
(c) $\boldsymbol{X}$ is stochastically continuous, i.e., for all $a > 0$ and for all $t \geqslant 0$

$$\lim_{s\to t} \mathbb{P}\left(|\boldsymbol{X}(t) - \boldsymbol{X}(s)| > a\right) = 0.$$

Because of its property of independent increments, Lévy process is Markovian.

According to Lévy–Khintchine theorem [1, 32, 42], we know that the characteristic function of Lévy process $\boldsymbol{X}$ has the following formula

$$\mathbb{E}\left[\mathrm{e}^{\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{X}(t)}\right] = \mathrm{e}^{t\phi(\boldsymbol{k})}, \tag{A.16}$$

$$\phi(\boldsymbol{k}) = \mathrm{i}\boldsymbol{b}\cdot\boldsymbol{k} - \frac{1}{2}\boldsymbol{k}\cdot\boldsymbol{A}\boldsymbol{k} + \int_{\mathbb{R}^d\setminus\{0\}} \left(\mathrm{e}^{\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{y}} - 1 - \mathrm{i}\boldsymbol{k}\cdot\boldsymbol{y}\mathbb{1}_{\{|\boldsymbol{x}|<1\}}(\boldsymbol{y})\right)\nu(\mathrm{d}\boldsymbol{y}), \tag{A.17}$$

where $\boldsymbol{b}$ is a vector in $\mathbb{R}^d$, $\boldsymbol{A}$ is a $d \times d$ real symmetric semi-positive definite matrix, $\mathbb{1}$ is the indicator function of set, and ν is a Lévy measure, satisfying

$$\int_{\mathbb{R}^d\setminus\{0\}} \min\left\{1, |\boldsymbol{y}|^2\right\}\nu(\mathrm{d}\boldsymbol{y}) < \infty. \tag{A.18}$$

A.3.1. Isotropic α-stable Lévy process. We say Lévy process $\boldsymbol{L}_\alpha(t)$ is isotropic α-stable if for any fixed $t \geqslant 0$, $\boldsymbol{L}_\alpha(t)$ is an isotropic stable random vector with index of stability $0 < \alpha \leqslant 2$. And its characteristic function has the following specific formula

$$\mathbb{E}\left[\mathrm{e}^{\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{L}_\alpha(t)}\right] = \mathrm{e}^{-\sigma^\alpha|\boldsymbol{k}|^\alpha t}, \tag{A.19}$$

where $\sigma > 0$ is the scaling parameter.

According to characteristic function (A.19), we can obtain that the PDF $P(\boldsymbol{x}, t)$ of $\boldsymbol{L}_\alpha(t)$ satisfies following equation

$$\frac{\partial}{\partial t}P(\boldsymbol{x}, t) = \Delta^{\frac{\alpha}{2}}P(\boldsymbol{x}, t). \tag{A.20}$$

Here, we focus on the one-dimensional isotropic (or symmetric) α-stable Lévy process.

When $\alpha = 2$, the one-dimensional stable Lévy process $L_\alpha(t)$ is the Brownian motion $B(t)$. Brownian motion $B(t)$ is the only continuous Lévy process, and is also a Gaussian process with mean $\mathbb{E}[B(t)] = 0$ and covariance $\mathbb{E}[B(s)B(t)] = 2\,\min\{t, s\}$ for $t, s \geqslant 0$. And the PDF of $B(t)$ is

$$P(x, t) = \frac{1}{\sqrt{4\pi t}}\exp\left\{-\frac{x^2}{4t}\right\}. \tag{A.21}$$

The algorithm used to simulate one-dimensional α-stable Lévy process trajectories is described in algorithm 6. Here, we use the equation

$$L_\alpha(t_{n+1}) = L_\alpha(t_n) + (t_{n+1} - t_n)^{\frac{1}{\alpha}}\xi_n \tag{A.22}$$

to simulate trajectories, where $\{\xi_n\}$ are independent random variables of symmetric α-stable distribution.

A.3.2. β-stable subordinator. A subordinator [1, 32] is a one-dimensional Lévy process that is non-decreasing (a.s.). And subordinator $T(t)$ is a β-stable subordinator if for any fixed $t \geqslant 0$, $T(t)$ is a totally skewed β-stable random variable with index of stability $0 < \beta < 1$. The Laplace transform of $T(t)$ is given by

$$\mathbb{E}\left[\mathrm{e}^{-\lambda T(t)}\right] = \mathrm{e}^{-\lambda^{\beta} t}. \tag{A.23}$$

According to above Laplace transform, one can get the PDF $P(T, t)$ of $T(t)$ satisfies

$$\partial_t P(T,t) = -\partial_T^{\beta} P(T,t) - \frac{T^{-\beta}}{\Gamma(1-\beta)}, \tag{A.24}$$

where ∂_t^{β} is the Caputo fractional derivative operator [4, 57] of order $\beta < 1$,

$$\partial_t^{\beta} u(x,t) = \frac{1}{\Gamma(1-\beta)} \int_0^t (t-s)^{-\beta} \partial_s u(x,s) \mathrm{d}s. \tag{A.25}$$

Similar to (A.22), we use equation

$$T(t_{n+1}) = T(t_n) + (t_{n+1} - t_n)^{\frac{1}{\beta}} \zeta_n \tag{A.26}$$

to simulate the trajectories of β-stable subordinator, where $\{\zeta_n\}$ are independent random variables of totally skewed β-stable distribution. The simulation algorithm is algorithm 7.

A.3.3. Poisson process. The (time homogeneous) Poisson process of intensity $\lambda > 0$ is a Lévy process $N(t)$ taking values in $\mathbb{N}$, and

$$\mathbb{P}\left[N(t) = n\right] = \frac{(\lambda t)^n}{n!} \mathrm{e}^{-\lambda t} \tag{A.27}$$

for each $n = 0, 1, 2, \ldots$. The probability distribution $P(n,t) = \mathbb{P}\left[N(t) = n\right]$ satisfies following equation

$$\frac{\partial}{\partial t} P(n,t) = -\lambda \left(P(n,t) - P(n-1,t)\right). \tag{A.28}$$

According to the properties of independent and stationary increments, the Poisson process can also be defined by stating that the time differences between events of the counting process are exponential variables with mean $\frac{1}{\lambda}$. So we can use independent random numbers of mean-$\frac{1}{\lambda}$ exponential distribution with $\{\pi_n\}$ to simulate Poisson process's trajectories, that is,

$$N(t_n) = n, \ t_n = t_{n-1} + \pi_n, \ t_0 = 0. \tag{A.29}$$

The simulation algorithm is algorithm 8.

See figures A5–A8.

A.4. Fractional Brownian motion

FBM [3, 15, 44, 59] $B_{\mathrm{H}}(t)$ with Hurst index $0 < H < 1$ is a Gaussian process with stationary increments, and satisfies following properties:

$$B_{\mathrm{H}}(0) = 0, \ \text{a.s.}, \tag{A.30}$$

$$\mathbb{E}\left[B_{\mathrm{H}}(t)\right] = 0 \tag{A.31}$$

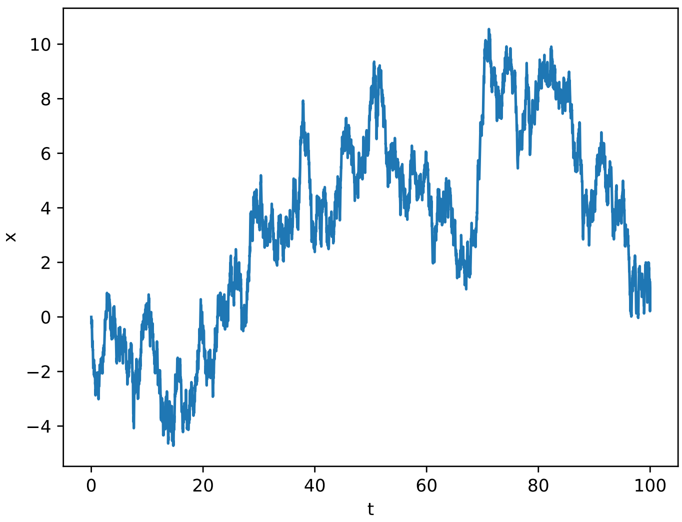

Figure A5. Brownian motion.

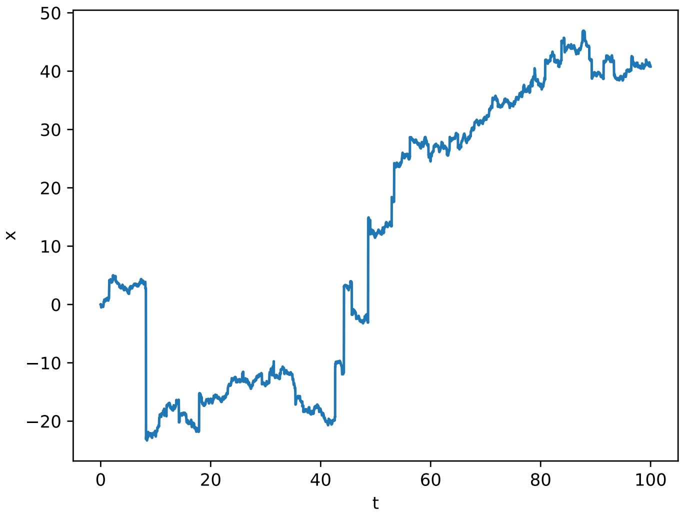

Figure A6. α-stable Lévy process with $\alpha < 2$.

for all $t \geqslant 0$, and

$$\mathbb{E}\left[B_{\mathrm{H}}(t)B_{\mathrm{H}}(s)\right] = \frac{1}{2}\left(t^{2H} + s^{2H} - |t-s|^{2H}\right) \tag{A.32}$$

for each t, $s \geqslant 0$. The mathematical definition of FBM is given by Mandelbrot and van Ness in [44], as following fractional stochastic integral of standard Brownian motion $B(t)$

$$B_{\mathrm{H}}(t) = \frac{1}{\Gamma\left(H+\frac{1}{2}\right)}\int_{\mathbb{R}}\left[(t-s)_{+}^{H-\frac{1}{2}} - (-s)_{+}^{H-\frac{1}{2}}\right]\mathrm{d}B(s), \tag{A.33}$$

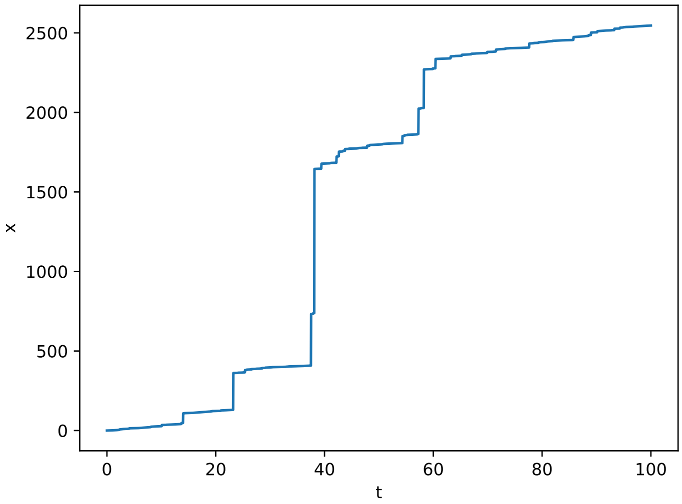

Figure A7. Stable subordinator.

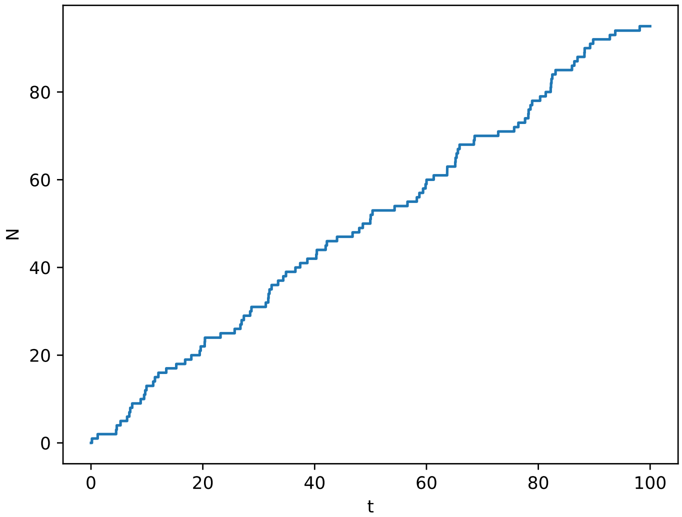

Figure A8. Poisson process.

where $x_+ = \max\{x, 0\}$. Specially, $B_{\frac{1}{2}}(t) = B(t)$.

The MSD of $B_{\rm H}(t)$ is $\mathbb{E}\left[(B_{\rm H}(t))^2\right] = t^{2H}$ for $t \geqslant 0$. Since $B_{\rm H}(t)$ is a Gaussian process, its characteristic function can be easily obtained by its mean and covariance,

$$\mathbb{E}\left[\mathrm{e}^{\mathrm{i}kB_{\rm H}(t)}\right] = \exp\left\{-\frac{1}{2}|k|^2 t^{2H}\right\}. \tag{A.34}$$

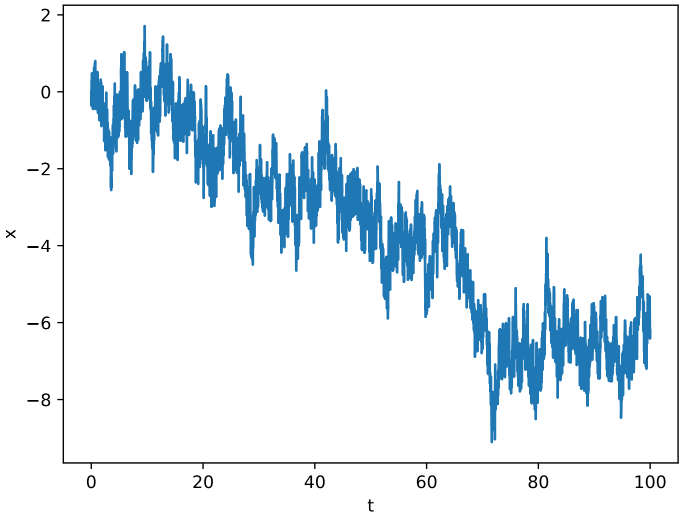

Figure A9. FBM with Hurst index $H < 1/2$.

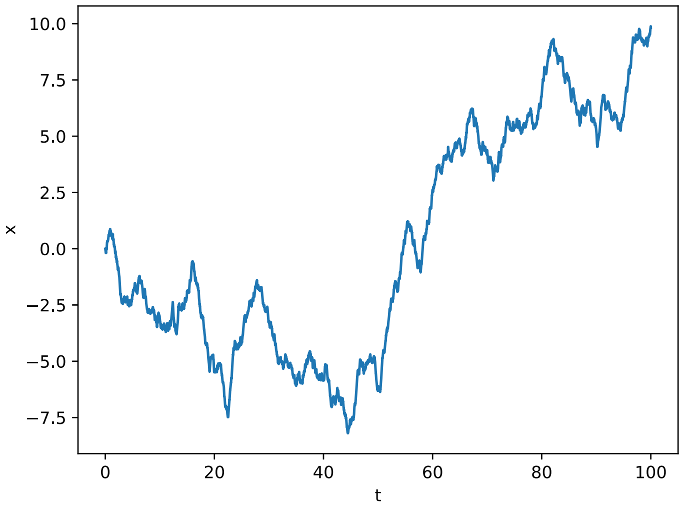

Figure A10. FBM with Hurst index $H > 1/2$.

Denote the PDF of $B_{\mathrm{H}}(t)$ by $P(x,t)$. Then we can get

$$P(x,t) = \frac{1}{\sqrt{2\pi t^{2H}}} \exp\left\{-\frac{x^2}{2t^{2H}}\right\}, \tag{A.35}$$

and

$$\frac{\partial}{\partial t}P(x,t) = Ht^{2H-1}\Delta P(x,t). \tag{A.36}$$

Various numerical approaches have been proposed to simulate FBM exactly. Here we use the Davies–Harte method [15] and the Hosking method [29] via the Python package stochastic. Details about the numerical implementations can be found in the associated references.

See figures A9–A10.

A.5. Alternating process

A two-state process [71] serves as an intermittent search process, which alternates between Lévy walk [40, 72, 76] and Brownian motion, i.e., Lévy walk $\rightarrow$ Brownian motion $\rightarrow$ Lévy walk $\rightarrow \ldots$. The searcher displays a slow active motion in the Brownian phase, during which the hidden target can be detected. While in Lévy walk phase, the searcher aims to relocate into some unvisited region to reduce oversampling. This kind of intermittent search process has wide applications in physical or biological problems [64].

The sojourn time distributions of the two-state process switching between Lévy walk and Brownian phase are $\psi^+(t)$ and $\psi^-(t)$, respectively. The subscripts '+' and '−' are introduced to represent the Lévy walk and Brownian phase, respectively. This process can be explicitly described by means of the velocity process $v(t)$ which also consists of two states: $v^+(t)$ for Lévy walk and $v^-(t)$ for Brownian motion. The PDF of $v^+(t)$ is $\delta(|v| - v_0)/2$ for some constant velocity v_0, whereas $v^-(t) = \sqrt{2}\xi(t)$ with $\xi(t)$ being Gaussian white noise.

Let the sojourn time distributions in the two states be a power-law form with exponents $\alpha_\pm$, that is,

$$\psi_\pm(t) \propto t^{-(1+\alpha_\pm)} \tag{A.37}$$

for large t. For $\alpha_\pm \in (0, 1)$, we know that one-sided $\alpha_\pm$-stable distribution has the above power-law form, so we can use (A.6) to get such a random variable. In addition, we can assume $\psi_\pm(t)$ has the following expression

$$\psi_\pm(t) = \begin{cases} C(t+1)^{-(1+\alpha_\pm)}, & t \geqslant 0, \\ 0, & t < 0. \end{cases} \tag{A.38}$$

By using the normalized property of the PDF, one can get the parameter $C = \alpha_\pm$. Hence, the probability distribution function

$$\Psi_\pm(t) = \begin{cases} 1 - (t+1)^{-\alpha_\pm}, & t \geqslant 0, \\ 0, & t < 0. \end{cases} \tag{A.39}$$

Therefore, if random variable $\xi \sim \Psi_\pm(t)$, then

$$\xi = (1 - X)^{\frac{-1}{\alpha_\pm}} - 1, \tag{A.40}$$

where X is a random variable uniformly distributed on (0, 1).

The MSD of this two-state process has the following expression [71]

$$\langle x^2(t) \rangle \sim K_1 t^{\nu_1} + K_2 t^{\nu_2}, \tag{A.41}$$

where K_1 and K_2 are constants, the values ν_1 and ν_2 are given in the following table.

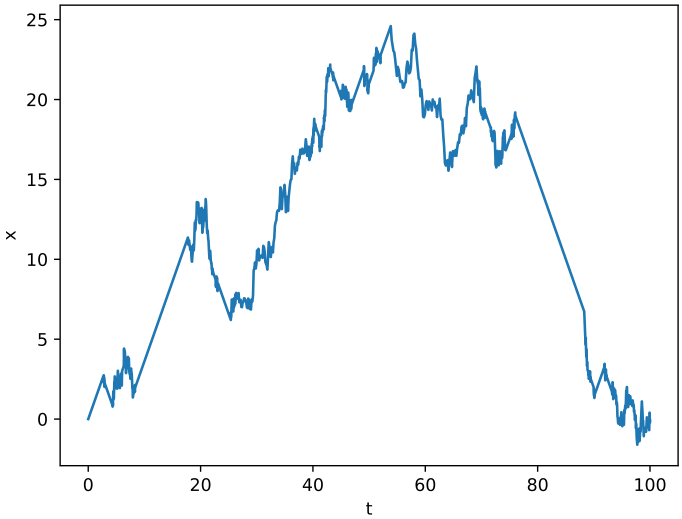

Figure A11. Alternating process with Lévy walk and Brownian motion.

Specific cases	ν_1	ν_2
$\alpha_+ = \alpha_- < 1$	2	1
$1 < \alpha_\pm < 2$	$3 - \alpha_+$	1
$\alpha_+ < \alpha_- < 1$	2	$\alpha_+ - \alpha_- + 1$
$\alpha_+ < 1 < \alpha_- < 2$	2	α_+
$\alpha_- < \alpha_+ < 1$	$\alpha_- - \alpha_+ + 2$	1
$\alpha_- < 1 < \alpha_+ < 2$	$\alpha_- - \alpha_+ + 2$	1

See figure A11.

A.6. Multiple internal states process

A.6.1. Fractional compound Poisson process with multiple internal states. Fractional Poisson process is a renewal process whose probability PDF of the holding/waiting times between two subsequent events has the asymptotic behavior $\phi(\tau) \sim \frac{1}{\tau^{1+\alpha}}$, $0 < \alpha < 1$ when time is long enough. Let $N(t)$ be the fractional Poisson process, i.e., there exist independent identically distributed random variables $\{\tau_n\} \sim \phi(\tau)$ for $0 < \alpha < 1$ and large τ, and $\{\xi_n\}$ a sequence of independent identically distributed variables (jump lengths). Then $X(t) = \sum_{n=1}^{N(t)} \xi_n$ is the fractional compound Poisson process.

Therefore, when the waiting time distribution is power-law with exponent $0 < \alpha < 1$, CTRW model (A.7) is a fractional compound Poisson process. Furthermore, we can generalize the renewal processes to have multiple internal states, where the waiting times for different internal states are drawn from different distribution. Here, we focus on fractional compound Poisson process $X(t)$ with multiple internal states [74], that is, $N(t)$ of $X(t)$ is a fractional Poisson process with multiple internal states. Each internal state has an own distribution of holding time, but the distribution of the jump lengths are all simply taken as normal distribution. In other words, $X(t)$ is a kind of CTRW process with multiple internal states whose characteristic waiting time is diverging and jump length variance is finite.

Suppose that $X(t)$ has N internal states. The transition of the internal states is described by a Markov chain with transition matrix $M \in \mathbb{R}^{N\times N}$. And the element M_{ij} of M represents the transition probability from state i to state j. Here, the bras $\langle\cdot|$ and kets $|\cdot\rangle$ denote the row and column vectors respectively. Let $\Phi(t) = \mathrm{diag}\left(\phi^{(1)}(t), \phi^{(2)}(t), \ldots, \phi^{(N)}(t)\right)$ be the waiting time distribution matrix and $\Lambda(x) = \mathrm{diag}\left(\lambda^{(1)}(x), \lambda^{(2)}(x), \ldots, \lambda^{(N)}(x)\right)$ be the jump length one. And we use the notation $|\mathrm{init}\rangle$ to represent the column vector of initial distribution of the internal states. In the Laplace space, $\widehat{\Phi}(s) \sim I - \Phi^*(s)$ where $\Phi^*(s) = \mathrm{diag}\left(B_{\alpha_1}s^{\alpha_1}, \ldots, B_{\alpha_N}s^{\alpha_N}\right)$, $0 < \alpha_1, \ldots, \alpha_N < 1$. Then the Fokker–Planck equation for $X(t)$ is given in [74].

As for renewal process $N(t)$ of fractional compound Poisson process $X(t)$, after each update, a Markov chain is used to determine which internal state it is in. Therefore, we should know how to generate a non-uniform discrete distributed random number.

Suppose that Y is a finite discrete random variable taking values in $\{1, 2, \ldots, N\}$, and $\mathbb{P}[Y = j] = p_j$, $j = 1, 2, \ldots, N$. Here we use the inverse transform method to sample Y. The probability distribution function of Y is

$$F_Y(x) = \begin{cases} 0, & x < 1 \\ \displaystyle\sum_{k=1}^{j} p_k, & j \leqslant x < j+1,\ j = 1, \ldots, N-1, \\ 1, & x \geqslant N. \end{cases} \tag{A.42}$$

The inverse of F_Y can be written as

$$F_Y^{-1}(u) = j, \quad \text{if} \sum_{k=1}^{j-1} p_k < u \leqslant \sum_{k=1}^{j} p_k, \quad j = 1, \ldots, N. \tag{A.43}$$

If U is a random variable uniformly distributed on (0, 1), then $F_Y^{-1}(U)$ is identically distributed with Y. Therefore, we can generate a sample ξ of U. Then $F_Y^{-1}(\xi)$ is a sample of Y. The pseudocode is given in algorithm 1.

After generating a sample of Y, we now can generate the trajectories of fractional compound Poisson process with multiple internal states $X(t)$. Firstly, we should determine the initial state by the initial distribution $|\mathrm{init}\rangle$. In fact, we can choose initial state by generating a sample I of probability distribution $\mathbb{P}[Y = j] = |\mathrm{init}\rangle_j$. Then the initial state is the Ith state. Next, after update in mth internal state, we can generate a sample n of probability distribution $\mathbb{P}[Y = j] = M_{mj}$ to determine that next update is occurred is in the nth internal state. The details is given by algorithm 10.

See figure A12.

A.6.2. Lévy walks with multiple internal states. As mentioned in appendix A.2, the Lévy flight has divergent MSD and the particle's jump does not cost any time [49]. In order to make the model fitter for the real natural phenomena, Lévy walk is used to describe superdiffusion properly and naturally [82]. The linearly-coupled Lévy walk with constant velocity v_0 is one of the most representative and important kinds of Lévy walks. Comparing to waiting time in CTRW, Lévy walk is given a random variable τ representing the duration of each step, and the total distance of each step of the particle's movement for linearly-coupled Lévy walk is then $v_0\tau$.

The Lévy walk with multiple internal states [75] is similar with fractional compound Poisson process with multiple internal states. The meanings of initial distribution vector $|\mathrm{init}\rangle$ and the transition matrix M are same to the ones in appendix A.6.1. However, the distributions of sojourn times $\{\phi_n(t)\}$ need not to be power-law, they can also be exponential. In addition,

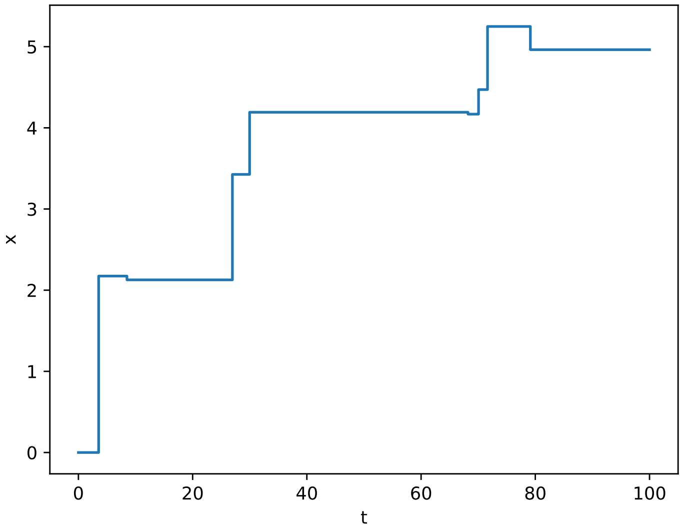

Figure A12. Fractional compound Poisson process with multiple internal states.

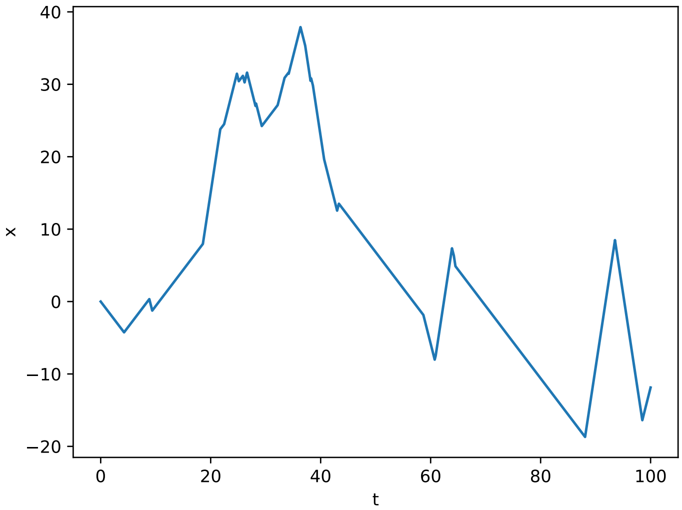

Figure A13. Lévy walk with multiple internal states.

compared to fractional compound Poisson process with multiple internal states, Lévy walk with multiple internal states removes the jump length distribution, but has an additional feature, distributions of speed $\{\lambda_n(v_0)\}$. In appendix B, we propose an algorithm (algorithm 11) to simulate the trajectories of Lévy walk with multiple internal states.

See figure A13.

Appendix B. Pseudocodes

B.1. Random number generator

See algorithms 1–4.

Algorithm 1. Generating finite discrete distribution random number.

Input: probability vector $\boldsymbol{p} = (p_1, \ldots, p_N)$ $\triangleright\ \mathbb{P}[Y = j] = p_j$
Output: a sample y of Y
compute the cumulative sum $\boldsymbol{q} = (q_1, \ldots, q_N)$ of $\boldsymbol{p}$
$q_0 \leftarrow 0$
generate a sample ξ uniformly distributed on $(0, 1)$
for $k \leftarrow 1$ to N **do**
    if $q_{k-1} < \xi \leqslant q_k$ **then**
        $y = k$
        break
    end if
end for
return y

Algorithm 2. Generating symmetric α-stable random number.

Input: index of stability α
Output: symmetric α-stable random number ξ
generate a random number V uniformly distributed on $(-\frac{\pi}{2}, \frac{\pi}{2})$ and an exponential random number W with mean 1
$\xi \leftarrow \frac{\sin(\alpha V)}{(\cos\ V)^{\frac{1}{\alpha}}} \cdot \left(\frac{\cos\,(V-\alpha V)}{W}\right)^{\frac{1-\alpha}{\alpha}}$ $\triangleright$ (A.5)
return ξ

Algorithm 3. Generating totally skewed α-stable random number.

Input: index of stability α
Output: totally skewed α-stable random number ξ
generate a random number V uniformly distributed on $(-\frac{\pi}{2}, \frac{\pi}{2})$ and an exponential random number W with mean 1
$c_1 \leftarrow (\cos(\pi\alpha/2))^{-\frac{1}{\alpha}}$ and $c_2 \leftarrow \frac{\pi}{2}$
$\xi \leftarrow c_1 \frac{\sin(\alpha(V+c_2))}{(\cos\ V)^{\frac{1}{\alpha}}} \cdot \left(\frac{\cos(V-\alpha(V+c_2))}{W}\right)^{\frac{1-\alpha}{\alpha}}$ $\triangleright$ (A.6)
return ξ

Algorithm 4. Generating power-law random number.

Input: the exponent of power-law distribution α
Output: power-law random number ξ
generate a random number X uniformly distributed on $(0, 1)$
$\xi \leftarrow (1 - X)^{\frac{-1}{\alpha}} - 1$ $\triangleright$ (A.40)
return ξ

B.2. CTRW

See algorithm 5.

Algorithm 5. Generating CTRW trajectory.

Input: length of trajectory T, index of waiting time β and index of jump length α, and initial position x_0 ▷ $\beta = 1$ means that the waiting time distribution is exponential
Output: time vector $\boldsymbol{t}$ and position vector $\boldsymbol{x}$

if $\beta = 1$ **then**
    set *random* as the random number generator of exponential distribution
else
    set *random* as the random number generator of totally skewed β-stable distribution ▷ Algorithm 3
end if
generate empty vectors $\boldsymbol{t}$ and $\boldsymbol{x}$ ▷ Variable-length vectors
$\boldsymbol{t}(1) \leftarrow 0$ and $\boldsymbol{x}(1) \leftarrow x_0$ ▷ Initial value
$t_{\mathrm{tot}} \leftarrow 0$ ▷ Total time
$x_{\mathrm{c}} \leftarrow x_0$ ▷ Current position
$n \leftarrow 1$ ▷ Counter
while true do
    generate a random number τ_n by generator *random*
    if $t_{\mathrm{tot}} + \tau_n > T$ **then**
        $\boldsymbol{t}(n+1) \leftarrow T$
        $\boldsymbol{x}(n+1) \leftarrow x_{\mathrm{c}}$
        break
    else
        $t_{\mathrm{tot}} \leftarrow t_{\mathrm{tot}} + \tau_n$
        $\boldsymbol{t}(n+1) \leftarrow t_{\mathrm{tot}}$
        generate a random number ξ_n of symmetric α-stable distribution
        $x_{\mathrm{c}} \leftarrow x_{\mathrm{c}} + \xi_n$
        $\boldsymbol{x}(n+1) \leftarrow x_{\mathrm{c}}$
        $n \leftarrow n + 1$
    end if
end while
return $\boldsymbol{t}$ and $\boldsymbol{x}$

B.3. Lévy process

See algorithms 6–8.

Algorithm 6. Generating α-stable Lévy process trajectory.

Input: length of the trajectory T, index of stability α and time-stepping size τ, and initial position x_0
Output: time vector $\boldsymbol{t}$ and position vector $\boldsymbol{x}$

$N \leftarrow \lceil \frac{T}{\tau} \rceil$
generate empty vectors $\boldsymbol{t}$ and $\boldsymbol{x}$ of length $N + 1$
$\boldsymbol{t}(n) \leftarrow (n-1)\tau$ for $n = 1, \ldots, N+1$
$\boldsymbol{x}(1) \leftarrow x_0$ ▷ Initial position
$x_{\mathrm{c}} \leftarrow x_0$ ▷ Current position
generate N random numbers of symmetric α-stable distribution $\{\xi_n\}_{n=1}^{N}$
for $n \leftarrow 1$ to N **do**
    $x_{\mathrm{c}} \leftarrow x_{\mathrm{c}} + \tau^{\frac{1}{\alpha}} \xi_n$
    $\boldsymbol{x}(n+1) \leftarrow x_{\mathrm{c}}$ ▷ (A.22)
end for
return $\boldsymbol{t}$ and $\boldsymbol{x}$

Algorithm 7. Generating β-stable subordinator trajectory.

Input: length of the trajectory T, index of stability β and time-stepping size τ
Output: time vector $\boldsymbol{t}$ and position vector $\boldsymbol{x}$

$N \leftarrow \lceil \frac{T}{\tau} \rceil$
generate empty vectors $\boldsymbol{t}$ and $\boldsymbol{x}$ of length $N+1$
$\boldsymbol{t}(n) \leftarrow (n-1)\tau$ for $n = 1, \ldots, N+1$
$\boldsymbol{x}(1) \leftarrow 0$ ▷ Initial value
$x_c \leftarrow 0$ ▷ Current position
generate N random numbers of totally skewed β-stable distribution $\{\zeta_n\}_{n=1}^{N}$
for $n \leftarrow 1$ to N **do**
  $x_c \leftarrow x_c + \tau^{\frac{1}{\alpha}} \zeta_n$
  $\boldsymbol{x}(n+1) \leftarrow x_c$ ▷ (A.26)
end for
return $\boldsymbol{t}$ and $\boldsymbol{x}$

Algorithm 8. Generating Poisson process trajectory.

Input: length of the trajectory T and intensity λ
Output: time vector $\boldsymbol{t}$ and position vector $\boldsymbol{x}$

generate empty vectors $\boldsymbol{t}$ and $\boldsymbol{x}$ ▷ Variable-length vectors
$\boldsymbol{t}(1) \leftarrow 0$ and $\boldsymbol{x}(1) \leftarrow 0$ ▷ Initial value
$t_{\mathrm{tot}} \leftarrow 0$ ▷ Total time
$x_c \leftarrow 0$ ▷ Current position
$n \leftarrow 1$ ▷ Counter
while true do
  generate a random number τ_n of exponential distribution with mean $\frac{1}{\lambda}$
  if $t_{\mathrm{tot}} + \tau_n > T$ **then**
    $\boldsymbol{t}(n+1) \leftarrow T$
    $\boldsymbol{x}(n+1) \leftarrow x_c$
    break
  else
    $t_{\mathrm{tot}} \leftarrow t_{\mathrm{tot}} + \tau_n$
    $x_c \leftarrow x_c + 1$
    $\boldsymbol{t}(n+1) \leftarrow t_{\mathrm{tot}}$
    $\boldsymbol{x}(n+1) \leftarrow x_c$
    $n \leftarrow n + 1$
  end if
end while
return $\boldsymbol{t}$ and $\boldsymbol{x}$

B.4. Alternating process

See algorithm 9.

Algorithm 9. Generating alternating process trajectory.

Input: length of trajectory T, sojourn time distributions' exponents α_+ and α_-, velocity of Lévy walk v_0 and initial position x_0
Output: time vector $\boldsymbol{t}$ and position vector $\boldsymbol{x}$

```
generate empty vectors t and x                                   ▷ Variable vectors
t(1) ← 0 and x(1) ← x_0                                          ▷ Initial position
t_tot ← 0                                                        ▷ Total time
x_c ← x_0                                                        ▷ Current position
n ← 1                                                            ▷ Counter
while true do
    generate a random number ξ_n uniformly distributed on (0, 1)
    if ξ_n < 0.5 then
        d ← −1                                                   ▷ Direction of Lévy walk
    else
        d ← 1
    end if
    generate a power-law random number τ_{n+} with exponent α_+  ▷ Algorithm 4
    if t_tot + τ_{n+} ⩾ T then
        t ← (t, T)
        x_c ← x_c + d v_0 (T − t_tot)
        x ← (x, x_c)
        break
    else
        t_tot ← t_tot + τ_{n+}
        t ← (t, t_tot)
        x_c ← x_c + d v_0 τ_{n+}
        x ← (x, x_c)
    end if
    generate a power-law random number τ_{n−} with exponent α_−  ▷ Algorithm 4
    if t_tot + τ_{n−} ⩾ T then
        generate a Brownian motion trajectory t̂_{n−} and x̂_{n−} with length T − t_tot and initial position x_c
        t_{n−} ← t_tot + t̂_{n−}(2 : end)
        t ← (t, t_{n−})
        x ← (x, x̂_{n−}(2 : end))
        break
    else
        generate a Brownian motion trajectory t̃_{n−} and x̃_{n−} with length τ_{n−} and initial position x_c
        t_{n−} ← t_tot + t̃_{n−}(2 : end)
        t ← (t, t_{n−})
        x ← (x, x̂_{n−}(2 : end))
        t_tot ← t_tot + τ_{n−}
        x_cur ← x(end)
    end if
    n ← n + 1
end while
return t and x
```

B.5. Multiple internal states process

See algorithms 10–11.

Algorithm 10. Generating fractional compound Poisson process with multiple internal states trajectory.

Input: length of trajectory T, index vector of waiting times $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)$, transition matrix M, initial state $\boldsymbol{I} = (I_1, \ldots, I_N)$ and initial position x_0

Output: time vector $\boldsymbol{t}$ and position vector $\boldsymbol{x}$

set *random* as the random number generator of totally skewed stable distribution ▷ Algorithm 3
generate empty vectors $\boldsymbol{t}$ and $\boldsymbol{x}$ ▷ Variable-length vectors
$\boldsymbol{t}(1) \leftarrow 0$ and $\boldsymbol{x}(1) \leftarrow x_0$ ▷ Initial value
$t_{\text{tot}} \leftarrow 0$ ▷ Total time
$x_{\text{c}} \leftarrow x_0$ ▷ Current position
$n \leftarrow 1$ ▷ Counter
generate a sample init of probability distribution $\mathbb{P}[Y = j] = I_j$ ▷ Algorithm 1
$\text{nums}_{\text{S}} \leftarrow \text{init}$
while true do
    generate a random number τ_n by *random* with parameter $\alpha_{\text{nums}_{\text{S}}}$
    if $t_{\text{tot}} + \tau_n > T$ **then**
        $\boldsymbol{t}(n+1) \leftarrow T$
        $\boldsymbol{x}(n+1) \leftarrow x_{\text{c}}$
        break
    else
        $t_{\text{tot}} \leftarrow t_{\text{tot}} + \tau_n$
        $\boldsymbol{t}(n+1) \leftarrow t_{\text{tot}}$
        generate a random number ξ_n of standard normal distribution
        $x_{\text{c}} \leftarrow x_{\text{c}} + \xi_n$
        $\boldsymbol{x}(n+1) \leftarrow x_{\text{c}}$
        generate a sample num_{next} of probability distribution $\mathbb{P}[Y = j] = M_{\text{nums}_{\text{S}}, j}$
        $\text{nums}_{\text{S}} \leftarrow \text{num}_{\text{next}}$
        $n \leftarrow n + 1$
    end if
end while
return $\boldsymbol{t}$ and $\boldsymbol{x}$

Algorithm 11. Generating Lévy walk with multiple internal states trajectory.

Input: length of trajectory T, vector of sojourn time distributions $\boldsymbol{w} = (w_1(t), \ldots, w_N(t))$, vector of velocity distributions $\boldsymbol{v}(x) = (v_1(x), \ldots, v_N(x))$, transition matrix M, initial state $\boldsymbol{I} = (I_1, \ldots, I_N)$ and initial position x_0

Output: time vector $\boldsymbol{t}$ and position vector $\boldsymbol{x}$ ▷ Variable-length vectors

Generate empty vectors $\boldsymbol{t}$ and $\boldsymbol{x}$
$\boldsymbol{t}(1) \leftarrow 0$ and $\boldsymbol{x}(1) \leftarrow x_0$ ▷ Initial value
$t_{\mathrm{tot}} \leftarrow 0$ ▷ Total time
$x_{\mathrm{c}} \leftarrow x_0$ ▷ Current position
$n \leftarrow 1$ ▷ Counter
Generate a sample init of probability distribution $\mathbb{P}[Y = j] = I_j$
$\mathrm{nums_S} \leftarrow \mathrm{init}$
while true do
  generate a random number τ_n of distribution $w_{\mathrm{nums_S}}(t)$
  generate a sample ζ_n uniformly distributed on $(0, 1)$
  if $\zeta_n < 0.5$**then**
    $d \leftarrow -1$
  else
    $d \leftarrow 1$
  end if
  generate a random number v_{0n} of distribution $v_{\mathrm{nums_S}}(x)$
  if $t_{\mathrm{tot}} + \tau_n \geqslant T$ **then**
    $\boldsymbol{t}(n+1) \leftarrow T$
    $x_{\mathrm{c}} \leftarrow x_{\mathrm{c}} + \mathrm{d}v_{0n}(T - t_{\mathrm{tot}})$
    $\boldsymbol{x}(n+1) \leftarrow x_{\mathrm{c}}$
    break
  else
    $t_{\mathrm{tot}} \leftarrow t_{\mathrm{tot}} + \tau_n$
    $x_{\mathrm{c}} \leftarrow x_{\mathrm{c}} + \mathrm{d}v_{0n}\tau_n$
    $\boldsymbol{t}(n+1) \leftarrow t_{\mathrm{tot}}$
    $\boldsymbol{x}(n+1) \leftarrow x_{\mathrm{c}}$
    generate a sample $\mathrm{num_{next}}$ of probability distribution $\mathbb{P}[Y = j] = M_{\mathrm{nums_S},j}$
    $\mathrm{nums_S} \leftarrow \mathrm{num_{next}}$
    $n \leftarrow n + 1$
  end if
end while
return $\boldsymbol{t}$ and $\boldsymbol{x}$

ORCID iDs

Weihua Deng https://orcid.org/0000-0002-8573-012X

References

[1] Applebaum D 2009 *Lévy Processes and Stochastic Calculus* (Cambridge: Cambridge University Press)
[2] Baron M 2019 *Probability and Statistics for Computer Scientists* (Boca Raton, FL: CRC Press)
[3] Biagini F, Hu Y Z, Ksendal B and Zhang T S 2008 *Stochastic Calculus for Fractional Brownian Motion and Applications* (Berlin: Springer)

[4] Bleanu D and Antnio M L 2019 *Handbook of Fractional Calculus with Applications* (Berlin: de Gruyter & Co)
[5] Bo S, Schmidt F, Eichhorn R and Volpe G 2019 Measurement of anomalous diffusion using recurrent neural networks *Phys. Rev.* E **100** 010102
[6] Bressloff P C 2014 *Stochastic Processes in Cell Biology* (Berlin: Springer)
[7] Bronstein I, Israel Y, Kepten E, Mai S, Shav-Tal Y, Barkai E and Garini Y 2009 Transient anomalous diffusion of telomeres in the nucleus of mammalian cells *Phys. Rev. Lett.* **103** 018102
[8] Buldyrev S V, Goldberger A L, Havlin S, Peng C-K and Stanley H E 1994 Fractals in biology and medicine: from DNA to the heartbeat *Fractals in Science* (Berlin: Springer) pp 49–88
[9] Burnecki K, Kepten E, Garini Y, Sikora G and Weron A 2015 Estimating the anomalous diffusion exponent for single particle tracking data with measurement errors—an alternative approach *Sci. Rep.* **5** 11306
[10] Carmi S and Barkai E 2011 Fractional Feynman–Kac equation for weak ergodicity breaking *Phys. Rev.* E **84** 061104
[11] Carmi S, Turgeman L and Barkai E 2010 On distributions of functionals of anomalous diffusion paths *J. Stat. Phys.* **141** 1071
[12] Cherstvy A G, Thapa S, Wagner C E and Metzler R 2019 Non-Gaussian, non-ergodic, and non-Fickian diffusion of tracers in mucin hydrogels *Soft Matter* **15** 2526
[13] Cichos F, Gustavsson K, Mehlig B and Volpe G 2020 Machine learning for active matter *Nat. Mach. Intell.* **2** 94
[14] Codling E A, Plank M J and Benhamou S 2008 Random walk models in biology *J. R. Soc. Interface* **5** 813
[15] Davies R B and Harte D S 1987 Tests for Hurst effect *Biometrika* **74** 95
[16] de Haan L and Resnick S I 1980 A simple asymptotic estimate for the index of a stable distribution *J. R. Stat. Soc.* B **42** 83
[17] Deng L *et al* 2013 Recent advances in deep learning for speech research at Microsoft *IEEE Int. Conf. Acoustics, Speech and Signal Processing* (IEEE) pp 8604–8
[18] Deng W H, Hou R, Wang W L and Xu P B 2020 *Modeling Anomalous Diffusion: From Statistics to Mathematics* (Singapore: World Scientific)
[19] Di Pierro M, Potoyan D A, Wolynes P G and Onuchic J N 2018 Anomalous diffusion, spatial coherence, and viscoelasticity from the energy landscape of human chromosomes *Proc. Natl Acad. Sci. USA* **115** 7753
[20] Dosset P, Rassam P, Fernandez L, Espenel C, Rubinstein E, Margeat E and Milhiet P E 2016 Automatic detection of diffusion modes within biological membranes using back-propagation neural network *BMC Bioinform.* **17** 1
[21] Dougherty E R 1999 *Random Processes for Image and Signal Processing* (Bellingham, WA: SPIE Optical Engineering Press)
[22] Ewers H, Smith A E, Sbalzarini I F, Lilie H, Koumoutsakos P and Helenius A 2005 Single-particle tracking of murine polyoma virus-like particles on live cells and artificial membranes *Proc. Natl Acad. Sci. USA* **102** 15110
[23] Gichman I I, Gikhman I I and Skorokhod A V 1969 *Introduction to the Theory of Random Processes* (Philadelphia, PA: Saunders)
[24] Golding I and Cox E C 2006 Physical nature of bacterial cytoplasm *Phys. Rev. Lett.* **96** 098102
[25] Granik N, Weiss L E, Nehme E, Levin M, Chein M, Perlson E, Roichman Y and Shechtman Y 2019 Single-particle diffusion characterization by deep learning *Biophys. J.* **117** 185
[26] Harangi B, Baran A and Hajdu A 2018 Classification of skin lesions using an ensemble of deep neural networks *40th Annual Int. Conf. IEEE Engineering in Medicine and Biology Society (EMBC)* (IEEE) pp 2575–8
[27] He K M, Zhang X Y, Ren S Q and Sun J 2016 Deep residual learning for image recognition *Proc. IEEE Conf. Computer Vision and Pattern Recognition* pp 770–8
[28] Heikkilä M, Pietikäinen M and Schmid C 2006 *Description of Interest Regions with Center-Symmetric Local Binary Patterns* (Berlin: Springer)
[29] Hosking J R M 1984 Modeling persistence in hydrological time series using fractional differencing *Water Resour. Res.* **20** 1898
[30] Janicki A and Weron A 1994 Can one see α-stable variables and processes? *Stat. Sci.* **9** 109
[31] Katz J and Lindell Y 2020 *Introduction to Modern Cryptography* (Boca Raton, FL: CRC Press)
[32] Ken-Iti S 1999 *Lévy Processes and Infinitely Divisible Distributions* (Cambridge: Cambridge University Press)

[33] Kepten E, Weron A, Sikora G, Burnecki K and Garini Y 2015 Guidelines for the fitting of anomalous diffusion mean square displacement graphs from single particle tracking experiments *PLoS One* **10** e0117722

[34] Klafter J and Sokolov I M 2011 *First Steps in Random Walks: From Tools to Applications* (Oxford: Oxford University Press)

[35] Kowalek P, Loch-Olszewska H and Szwabinski J 2019 Classification of diffusion modes in single-particle tracking data: feature-based versus deep-learning approach *Phys. Rev.* E **100** 032410

[36] Krapf D 2015 Mechanisms underlying anomalous diffusion in the plasma membrane *Curr. Top. Membr.* **75** 167

[37] Laing C and Lord G J 2010 *Stochastic Methods in Neuroscience* (Oxford: Oxford University Press)

[38] LeCun Y, Boser B, Denker J S, Henderson D, Howard R E, Hubbard W and Jackel L D 1989 Backpropagation applied to handwritten zip code recognition *Neural Comput.* **1** 541

[39] Lischke A *et al* 2020 What is the fractional Laplacian? A comparative review with new results *J. Comput. Phys.* **404** 109009

[40] Lomholt M A, Tal K, Metzler R and Joseph K 2008 Lévy strategies in intermittent search processes are advantageous *Proc. Natl Acad. Sci. USA* **105** 11055

[41] Magdziarz M and Weron A 2011 Anomalous diffusion: testing ergodicity breaking in experimental data *Phys. Rev.* E **84** 051138

[42] Mainardi F and Rogosin S 2008 The origin of infinitely divisible distributions: from de Finetti's problem to Lévy–Khintchine formula (arXiv:0801.1910)

[43] Mandelbrot B 1960 The Pareto–Lévy law and the distribution of income *Int. Econ. Rev.* **1** 79

[44] Mandelbrot B B and Van Ness J W 1968 Fractional Brownian motions, fractional noises and applications *SIAM Rev.* **10** 422

[45] Manzo C and Garcia-Parajo M F 2015 A review of progress in single particle tracking: from methods to biophysical insights *Rep. Prog. Phys.* **78** 124601

[46] Martin D S, Forstner M B and Käs J A 2002 Apparent subdiffusion inherent to single particle tracking *Biophys. J.* **83** 2109

[47] Matsuda Y, Hanasaki I, Iwao R, Yamaguchi H and Niimi T 2018 Estimation of diffusive states from single-particle trajectory in heterogeneous medium using machine-learning methods *Phys. Chem. Chem. Phys.* **20** 24099

[48] Metzler R, Jeon J-H, Cherstvy A G and Barkai E 2014 Anomalous diffusion models and their properties: non-stationarity, non-ergodicity, and ageing at the centenary of single particle tracking *Phys. Chem. Chem. Phys.* **16** 24128

[49] Metzler R and Klafter J 2000 The random walk's guide to anomalous diffusion: a fractional dynamics approach *Phys. Rep.* **339** 1

[50] Monnier N, Guo S-M, Mori M, He J, Lénárt P and Bathe M 2012 Bayesian approach to MSD-based analysis of particle motion in live cells *Biophys. J.* **103** 616

[51] Muñoz-Gil G *et al* 2021 Objective comparison of methods to decode anomalous diffusion *Nat. Commun.* **12** 6253

[52] Muñoz-Gil G, Garcia-March M A, Manzo C, Martín-Guerrero J D and Lewenstein M 2019 Machine learning method for single trajectory characterization (arXiv:1903.02850)

[53] Nolan J P 2003 *Handbook of Heavy Tailed Distributions in Finance* (Amsterdam: Elsevier)

[54] Nolan J P 2020 *Univariate Stable Distributions* (Berlin: Springer)

[55] Parzen E 2015 *Stochastic Processes* (New York: Dover)

[56] Paul W and Baschnagel J 2013 *Stochastic Processes* (Berlin: Springer)

[57] Podlubny I 1998 *Fractional Differential Equations: An Introduction to Fractional Derivatives, Fractional Differential Equations, to Methods of Their Solution and Some of Their Applications* (Amsterdam: Elsevier)

[58] Pozrikidis C 2018 *The Fractional Laplacian* (Boca Raton, FL: CRC Press)

[59] Rogers L C G 1997 Arbitrage with fractional Brownian motion *Math. Finance* **7** 95

[60] Sabri A, Xu X, Krapf D and Weiss M 2020 Elucidating the origin of heterogeneous anomalous diffusion in the cytoplasm of mammalian cells *Phys. Rev. Lett.* **125** 058101

[61] Sagi Y, Brook M, Almog I and Davidson N 2012 Observation of anomalous diffusion and fractional self-similarity in one dimension *Phys. Rev. Lett.* **108** 093002

[62] Simonyan K and Zisserman A 2014 Very deep convolutional networks for large-scale image recognition (arXiv:1409.1556)

[63] Sokolov I M 2012 Models of anomalous diffusion in crowded environments *Soft Matter* **8** 9043

[64] Song M S, Moon H C, Jeon J H and Park H Y 2018 Neuronal messenger ribonucleo protein transport follows an aging Lévy walk *Nat. Commun.* **9** 344
[65] Steele J M 2012 *Stochastic Calculus and Financial Applications* (Berlin: Springer)
[66] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V and Rabinovich A 2014 Going deeper with convolutions *Proc. IEEE Conf. Computer Vision and Pattern Recognition* pp 1–9
[67] Tejedor V, Bénichou O, Voituriez R, Jungmann R, Simmel F, Selhuber-Unkel C, Oddershede L B and Metzler R 2010 Quantitative analysis of single particle trajectories: mean maximal excursion method *Biophys. J.* **98** 1364
[68] Thapa S, Lomholt M A, Krog J, Cherstvy A G and Metzler R 2018 Bayesian analysis of single-particle tracking data using the nested-sampling algorithm: maximum-likelihood model selection applied to stochastic-diffusivity data *Phys. Chem. Chem. Phys.* **20** 29018
[69] Turgeman L, Carmi S and Barkai E 2009 Fractional Feynman–Kac equation for non-Brownian functionals *Phys. Rev. Lett.* **103** 190201
[70] Wagner T, Kroll A, Haramagatti C R, Lipinski H G and Wiemann M 2017 Classification and segmentation of nanoparticle diffusion trajectories in cellular micro environments *PLoS One* **12** e0170165
[71] Wang X D, Chen Y and Deng W H 2019 Aging two-state process with Lévy walk and Brownian motion *Phys. Rev.* E **100** 012136
[72] Wang X, Chen Y and Deng W 2019 Lévy-walk-like Langevin dynamics *New J. Phys.* **21** 013024
[73] Weron A and Weron R 1995 *Chaos—The Interplay between Stochastic and Deterministic Behaviour* (Berlin: Springer)
[74] Xu P and Deng W 2018 Fractional compound Poisson processes with multiple internal states *Math. Model. Nat. Phenom.* **13** 10
[75] Xu P and Deng W 2018 Lévy walk with multiple internal states *J. Stat. Phys.* **173** 1598
[76] Zaburdaev V, Denisov S and Klafter J 2015 Lévy walks *Rev. Mod. Phys.* **87** 483
[77] Zhang H, Li G-H and Luo M-K 2013 Fractional Feynman–Kac equation with space-dependent anomalous exponent *J. Stat. Phys.* **152** 1194
[78] Zhou B L, Khosla A, Lapedriza A, Torralba A and Oliva A 2016 Places: an image database for deep scene understanding (arXiv:1610.02055)
[79] Chenouard N *et al* 2014 Objective comparison of particle tracking methods *Nat. Methods* **11** 281
[80] Burov S, Tabei S M A, Huynh T, Murrell M P, Philipson L H, Rice S A, Gardel M L, Scherer N F and Dinner A R 2013 Distribution of directional change as a signature of complex dynamics *Proc. Natl Acad. Sci. USA* **110** 19689
[81] Moschitti A, Pang B and Daelemans W 2014 *Proc. 2014 Conf. Empirical Methods in Natural Language Processing (EMNLP)*
[82] Shlesinger M F and Klafter J 1986 Lévy walks versus Lévy flights *On Growth and Form* (Berlin: Springer) pp 279–83
[83] Sabatier J *et al* 2007 *Advances in Fractional Calculus* (Berlin: Springer)
[84] Cairoli A and Baule A 2015 Anomalous processes with general waiting times: functionals and multipoint structure *Phys. Rev. Lett.* **115** 110601
[85] Friedrich R, Jenko F, Baule A and Eule S 2006 Anomalous diffusion of inertial, weakly damped particles *Phys. Rev. Lett.* **96** 230601