*fractal and fractional*

MDPI

*Article*

# Optimizing Fractional-Order Convolutional Neural Networks for Groove Classification in Music Using Differential Evolution

**Jiangang Chen, Pei Su, Daxin Li, Junbo Han, Gaoquan Zhou and Donghui Tang ***

School of P.E. and Sports, Beijing Normal University, Beijing 100875, China
* Correspondence: 202131070010@mail.bnu.edu.cn

**Abstract:** This study presents a differential evolution (DE)-based optimization approach for fractional-order convolutional neural networks (FOCNNs) aimed at enhancing the accuracy of groove classification in music. Groove, an essential element in music perception, is typically influenced by rhythmic patterns and acoustic features. While FOCNNs offer a promising method for capturing these subtleties through fractional-order derivatives, they face challenges in efficiently converging to optimal parameters. To address this, DE is applied to optimize the initial weights and biases of FOCNNs, leveraging its robustness and ability to explore a broad solution space. The proposed DE-FOCNN was evaluated on the Janata dataset, which includes pre-rated music tracks. Comparative experiments across various fractional-order values demonstrated that DE-FOCNN achieved superior performance in terms of higher test accuracy and reduced overfitting compared to a standard FOCNN. Specifically, DE-FOCNN showed optimal performance at fractional-order values such as $v = 1.4$. Further experiments demonstrated that DE-FOCNN achieved higher accuracy and lower variance compared to other popular evolutionary algorithms. This research primarily contributes to the optimization of FOCNNs by introducing a novel DE-based approach for the automated analysis and classification of musical grooves. The DE-FOCNN framework holds promise for addressing other related engineering challenges.

**Keywords:** fractional-order convolutional neural networks; differential evolution; groove classification; music information retrieval; optimization algorithms



**Citation:** Chen, J.; Su, P.; Li, D.; Han, J.; Zhou, G.; Tang, D. Optimizing Fractional-Order Convolutional Neural Networks for Groove Classification in Music Using Differential Evolution. *Fractal Fract.* **2024**, *8*, 616. https://doi.org/10.3390/fractalfract8110616

Academic Editors: Zhouchao Wei and Lulu Lu

Received: 11 September 2024
Revised: 16 October 2024
Accepted: 21 October 2024
Published: 23 October 2024



## 1. Introduction

Groove is a complex and integrated aspect of music, noted for its ability to provoke both emotional and physical reactions in listeners [1,2]. It is generally described as the capacity to discern the beat, identify recurring rhythmic patterns [3,4]. According to psychological studies, groove is defined as a musical quality that makes us want to move with the rhythm or beat, characterized by induced body movement that is rhythm- and beat-dependent [3,5,6]. However, groove is not merely an emotional characteristic; its most distinctive feature is the synchronization it induces between music and movement [3,4]. Groove holds significant relevance in various fields, including sports rehabilitation, motor function promotion, and physical activity promotion [3,7,8]. Nonetheless, accurately measuring and predicting groove remains a challenging endeavor.

### 1.1. Literature Review

Groove is influenced by multiple dimensions, making it difficult to directly define using a single or a few musical characteristics. It is both a musical phenomenon and a musical feature, shaped by a range of factors [3]. These include rhythmic features such as syncopation, microtiming, tempo, beat salience, and event density, as well as acoustic features like harmonic complexity and bass sounds [3]. Moreover, the groove experience can vary significantly depending on the musical style, highlighting its multidimensional nature [3]. Due to this complexity, previous studies often measure groove by evaluating

the degree of sensorimotor coupling it induces, that is, the synchronization between music and movement. Janata et al., for example, employed both behavioral and computational techniques to establish groove ratings based on this coupling [9]. Their research included the creation of a curated playlist with detailed groove ratings, which has been validated and extensively utilized in practical applications [10–12]. However, these ratings are typically obtained through physical measurements, which can be cumbersome and resource-intensive. Therefore, there is a pressing need for new methodologies that simplify the evaluation process while providing an objective assessment of groove without relying solely on intricate behavioral approaches. Based on music information retrieval (MIR) techniques, analyzing audio signals to extract acoustic features—which reflect musical elements such as rhythm, beat, and rhythmic structure—has been shown to effectively classify music [13,14]. Therefore, investigating whether machine learning can accurately predict groove based on these acoustic features offers a promising approach to enhance our understanding and measurement of groove in music.

Convolutional neural networks (CNNs) have been proposed and successfully applied across various domains, demonstrating particular strength in processing images, audio, and video [15]. A notable example is OpenL3, a pre-trained CNN widely used for tasks such as music genre recognition, instrument identification, and emotion detection [16]. Fractional-order convolutional neural networks (FOCNNs) extend traditional CNNs by incorporating principles of fractional calculus into the network architecture. Unlike standard CNNs that utilize integer-order derivatives, FOCNNs employ fractional-order derivatives, such as the Caputo derivative, to update the weights and biases of fully connected and convolutional layers [17,18]. This approach has demonstrated higher accuracy in practical experiments [17,19,20]. In the context of groove classification, FOCNNs may offer a distinct advantage by enabling the model to capture subtle variations in rhythmic patterns and spectral features that are crucial to groove perception. This allows for a deeper understanding of the underlying acoustic features that define groove, making FOCNNs particularly well-suited for accurately classifying music into different groove levels. However, FOCNNs face challenges in practice, particularly in efficiently converging to an optimal solution when initial biases and weights are randomly selected [19]. This limitation necessitates the use of additional algorithms to optimize these parameters effectively.

Evolutionary algorithms have played a pivotal role in optimizing neural network structures and weights [21]. For instance, genetic algorithms (GAs) and grammatical evolution have been employed to automatically design CNN topologies, achieving promising results even without data augmentation or preprocessing [22]. Additionally, methods like population extremum optimization (PEO) have shown success in dynamically updating biases and weights during CNN training [17]. Differential evolution (DE) stands out as a robust optimization algorithm, beginning with a randomly initialized population of potential solutions [23]. At each iteration, DE generates new candidate solutions by applying differential mutation and crossover operations. These candidates are then evaluated using a fitness function, and the best solutions are retained for the next iteration. This process continues until convergence. In the context of optimizing FOCNNs for groove classification, DE is particularly advantageous due to its robustness and its ability to explore a broad solution space efficiently. Studies have shown that DE outperforms other algorithms, such as GA, PEO, and particle swarm optimization (PSO), by achieving lower variance in prediction accuracy on datasets like MNIST, which is critical for stable and accurate groove level classification [17]. Moreover, DE's independence from gradient information makes it especially suitable for optimizing FOCNNs, given the complex and non-convex nature of the objective function, particularly when fractional-order derivatives are involved [21,24].

### *1.2. Research Gap and Motivation*

Despite the success of previous approaches in predicting groove, current methods often rely on intricate physical measurements and cumbersome behavioral evaluations, which limit their practical applicability. There remains a need for a more streamlined approach that

can objectively predict groove without relying on resource-intensive methods. Additionally, existing CNN-based models have limitations in capturing the nuanced features necessary for accurate groove classification, which suggests the need for an enhanced model capable of better representing these complex rhythmic characteristics.

The novelty of this work lies in approaching groove classification as a novel engineering problem by applying fractional-order convolutional neural networks (FOCNNs), which provide an enhanced ability to capture the subtle, complex features of groove. Furthermore, we employ differential evolution (DE) to optimize the initial biases and weights of FOCNNs, thereby addressing the convergence issues faced by traditional FOCNN models. By doing so, we aim to provide a robust and efficient framework for groove classification that is both practical and highly accurate.

### *1.3. Contribution and Paper Organization*

This paper proposes the use of differential evolution (DE) to optimize FOCNNs utilizing Caputo fractional-order derivatives, with the aim of enhancing CNN performance for groove classification. Specifically, Caputo derivatives are employed to update weights and biases, while DE is applied to identify the optimal initial biases and weight settings. The key contributions of this work are as follows:

1. Development of a DE-based optimization approach for FOCNNs: We introduce a novel approach to identify the optimal selection from a broad range of initial biases and weights, effectively enhancing the model's convergence and accuracy.
2. Comparative experiments between DE-FOCNN and the original FOCNN: We perform extensive comparisons across various fractional-order values, highlighting the efficacy of DE in optimizing FOCNNs.
3. Implementation of experiments on the Janata dataset: We demonstrate that DE-FOCNN outperforms other popular optimization algorithms in terms of accuracy and robustness in groove classification

The remainder of this paper is organized as follows. Section 2 provides an overview of related work in groove classification and FOCNN optimization. Section 3 presents the proposed DE-based optimization approach for FOCNNs. Section 4 details the experimental setup, including data collection and preprocessing. Section 5 discusses the results, and Section 6 concludes the paper with potential future research directions.

This research contributes to the optimization of fractional-order convolutional neural networks by introducing a novel DE-based approach for the automated analysis and classification of musical groove. The proposed method offers potential applications in music recommendation, education, and therapeutic settings.

## 2. Related Works

### *2.1. Music Groove Classification*

The classification of emotional responses and perceived groove in music has been a focal point of research, with various methodologies explored to understand these complex phenomena. Hove et al. [25] identified key acoustic features that influence groove perception, revealing that enhancing low-frequency components significantly promotes groove. Madison et al. [5] utilized linear regression to classify groove by identifying beat salience as a significant predictor, while Stupacher et al. [1] found that variability in event density strongly correlates with groove perception. However, these studies faced limitations, particularly in accuracy, with the highest correlations barely exceeding 0.61 [5]. Additionally, the reliance on linear methods limited their ability to capture the nonlinear interactions inherent in groove classification, underscoring the need for more sophisticated approaches.

In the domain of emotion classification in music, researchers have employed a variety of machine learning techniques to classify emotional responses such as pleasure, sadness, and arousal. Mori et al. [13] utilized machine learning to classify emotional responses based on acoustic features, though they achieved limited accuracy with regression-based methods. Vempala et al. [26] compared linear regression, random forests, and neural

networks, finding that nonlinear approaches, particularly neural networks, were more effective in capturing the emotional nuances in music.

Recent advancements in deep learning have furthered the field, particularly in predicting valence and arousal. Keunwoo et al. [27] enhanced this approach with transfer learning using OpenL3, a pre-trained deep learning model, to extract high-dimensional features from Mel-spectrograms for emotion prediction. The Mel-spectrogram is particularly effective for mimicking human perception of different frequencies [28]. The use of CNNs to analyze Mel-spectrograms of audio signals, which capture both time and frequency patterns, is essential for classifying the rhythmic patterns and spectral features [28]. Despite the progress made, the complexity of the networks used in these studies was often insufficient to capture the intricate relationships between music and emotions, indicating potential benefits from more sophisticated CNN-based models for improved classification performance and efficiency.

### *2.2. Fractional-Order Neural Networks*

Fractional-order neural networks (FONNs) represent an extension of traditional neural networks, incorporating fractional calculus to enhance their modeling capabilities. This approach allows FONNs to handle complex, dynamic systems with greater precision, especially in scenarios where traditional integer-order networks may struggle [29]. FONNs have found applications in diverse fields, including system identification, control systems [30], and signal processing [31], demonstrating their utility in addressing a wide range of engineering challenges.

A notable application of fractional calculus in neural networks is the fractional-order backpropagation (BP) neural network. By leveraging fractional derivatives, such as the Caputo derivative, these networks optimize the learning process more effectively, leading to faster convergence and better overall performance compared to standard BP networks [32,33].

Recurrent neural networks (RNNs) have also seen significant improvements through the application of fractional calculus [34]. Researchers have introduced fractional stochastic gradient descent methods that utilize fractional derivatives like Caputo and Riemann–Liouville [34]. These methods make them more robust in managing complex, dynamic time-series data. Additionally, fractional-order Hopfield neural networks have been developed, leveraging the advantages of fractional calculus, such as long-term memory and nonlocality, to enhance their performance in capturing complex dynamics and improving stability [35]. By incorporating fractional-order activation functions, LSTMs have been optimized to improve predictive accuracy, even in challenging scenarios like wind power forecasting with missing data [36]. For instance, a hybrid LSTM-based FONN employing a fractional arctan activation function has demonstrated enhanced performance in handling complex, nonlinear time-series data [36]. These advancements underscore the potential of fractional-order techniques in boosting the effectiveness of neural networks, particularly for sequential and time-series applications.

Overall, the use of fractional derivatives in neural networks has significantly advanced their capability to model and optimize complex systems. These improvements are not limited to traditional networks but extend to CNNs as well. Fractional-order CNNs, by integrating fractional calculus, could offer enhanced feature extraction capabilities [19] and are likely to play a critical role in applications requiring detailed analysis of spatial and temporal patterns.

### *2.3. Differential Evolution in Neural Network Optimization*

DE has emerged as a powerful tool in the field of neural network optimization, applied to optimize both the topology and parameters of neural networks [21]. Since the late 1990s and early 2000s, DE-based optimizers for neural networks have been actively researched and developed, showing promising results in a variety of applications, including time-series forecasting and clinical dataset classification [37,38].

Evolutionary algorithms can optimize non-differentiable objective functions without relying on gradients. DE, in particular, has gained popularity due to its robustness, simplicity, and ability to efficiently search both global and local optima [17,23]. DE operates through simple operations—mutation, crossover, and selection—which enable it to explore the solution space effectively and converge rapidly towards optimal solutions. Recent developments in DE have focused on improving performance through adaptive control parameters and mutation strategies [21,39]. The introduction of individual dependent differential evolution, for example, enhances optimization by tailoring DE operations to the characteristics of neural network parameters, effectively reducing issues like premature convergence and stagnation [21].

Comparative studies have demonstrated the effectiveness of DE over other evolutionary algorithms like genetic algorithms (GAs) and estimation of distribution algorithms in training neural networks [37]. DE has been particularly successful in optimizing neural networks for time-series forecasting and other complex tasks [40], where it has outperformed other methods in terms of convergence speed and solution quality. Additionally, DE's flexibility in handling a variety of neural network architectures, from simple feedforward networks to more complex structures like CNNs [41] and RNNs [42], makes it a versatile tool in the arsenal of neural network optimization techniques.

In summary, the integration of DE into neural network optimization has significantly enhanced the ability to fine-tune network parameters and architectures, leading to improved performance in a wide range of applications.

## 3. Methodology

### *3.1. Fractional-Order Convolutional Neural Network Architecture*

#### 3.1.1. Caputo Fractional Derivatives

The Caputo fractional derivative, as defined by Equation (1), is applied to calculate the derivative of the input signal $f(x)$ with respect to the fractional order $v$, which can be non-integer. Unlike traditional integer-order derivatives, fractional derivatives accumulate information over the range of integration, providing greater sensitivity to prior inputs. In this context, the Gamma function $\Gamma(\cdot)$ normalizes the derivative, ensuring the proper scaling of fractional operators.

$$ {}^{c}D_{x}^{v} f(x) = \frac{1}{\Gamma(r-v)} \int_{\xi}^{x} (x-\tau)^{r-v-1} f^{(r)}(\tau) d\tau \tag{1} $$

where

- $v$ is the order of the derivative and can be a non-integer $(0 < v < r)$.
- $r$ is the smallest integer greater than $\alpha$, so $r-1 < v < r$.
- $\Gamma(\cdot)$ is the Gamma function, defined by $\Gamma(z) = \int_{0}^{+\infty} t^{z-1} e^{-t} dt$.
- $f^{(r)}(\tau)$ denotes the $r$-th derivative of $f(\tau)$.
- $\xi$ is the lower bound of the integration domain, which can be any value within the domain of $f(x)$.
- $\tau$ represents the integration variable, which accumulates the changes of the function $f(x)$ over the interval $[\xi, x]$.

We employ the Caputo fractional derivative because it incorporates historical information through fractional-order differentiation while maintaining the same initial conditions as integer-order derivatives [43]. This feature makes it particularly useful for updating gradients in neural networks, providing smoother and more adaptive parameter updates that improve convergence behavior.

#### 3.1.2. Convolutional Neural Network Architecture

The convolutional neural network (CNN) used in this study is designed to process Mel-spectrogram images of size $128 \times 199 \times 1$ and classify the groove level of music into three categories: low, moderate, and high. Therefore, the input to the model is image data,

and the output is the corresponding classification of the image into one of the three groove levels. The network architecture is composed of several sequential blocks, each performing a specific function to progressively extract and refine features from the input data.

The core of the network consists of multiple convolutional blocks. Each block is structured as follows: (1) Convolutional layer: Applies a series of $3 \times 3$ convolution filters to the input, producing feature maps that capture spatial hierarchies in the data. The number of filters increases progressively across blocks to allow for more complex feature extraction as the network deepens. (2) Pooling layers: These layers use $2 \times 2$ pooling windows to retain the most relevant information while reducing computational complexity.

The output of the last convolutional layer is flattened into a one-dimensional vector by the flatten layer. This vector is then passed to the fully connected layers, where the final classification into one of the three groove levels is made. The activation functions for the convolutional and fully connected layers are ReLU functions and the activation function for the output layer is the SoftMax function. See Figure 1.

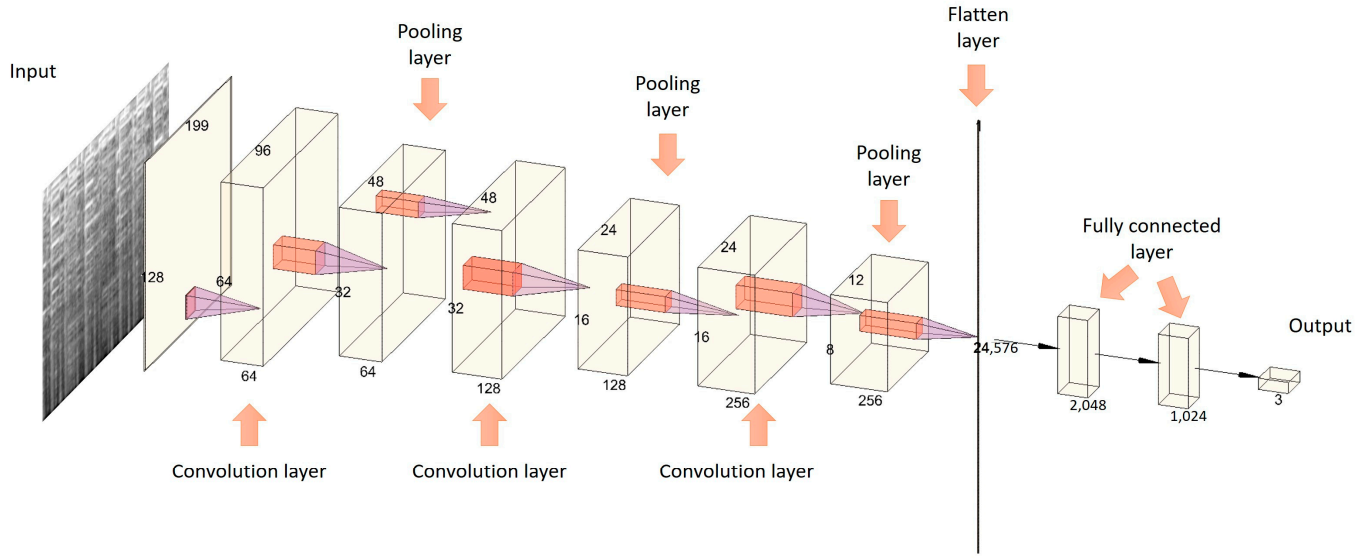



**Figure 1.** The structure of a convolutional neural network. The input to the model is image data, and the output is the corresponding classification of the image into one of the three groove levels.

### 3.1.3. Application of Caputo Derivative in CNN

In the convolutional neural network (CNN) architecture, the Caputo fractional-order gradient method (CFOGM) is applied to both convolutional and fully connected layers, as these are the layers where parameter updates are required during training. Pooling layers and the flattening layer do not involve learnable parameters, and thus, CFOGM is not applied to them.

1. Forward Propagation

Forward propagation is used to calculate the output of each layer based on the inputs and current weights and biases.

Fully Connected Layer:

In the fully connected layer, forward propagation is described by Equation (2), which computes the activation values for each neuron. The input from the previous layer $a^{l-1}$ is linearly combined with the weight vector $w_i^l$ and bias $b_i^l$ for the $i$-th neuron, producing the pre-activation output $z_i^l$. An activation function $f$ is then applied to yield the final output $a_i^l$. This process enables the network to extract progressively more abstract features across layers.

$$\begin{aligned} z_i^l &= w_i^l \cdot a^{l-1} + b_i^l \\ a_i^l &= f\left(z_i^l\right) \end{aligned} \tag{2}$$

where

- $z_i^l$ is the input to the $i$-th neuron in the $l$-th layer.
- $a_i^l$ is the output of the $i$-th neuron in the $l$-th layer after applying the activation function.
- $w_i^l$ is the weight vector connecting the $l-1$-th layer to the $i$-th neuron in the $l$-th layer.
- $b_i^l$ is the bias term for the $i$-th neuron in the $l$-th layer.
- $f$ is the activation function.

Convolutional Layer:

The forward propagation in the convolutional layer similarly computes the output feature maps via convolution operations. Equation (3) represents the application of the weight filter $w_m^l$ to the input features $a^{l-1}$, adding the bias $b_m^l$ to generate the pre-activation output $z_m^l$. The final result $a_m^l$ is obtained by passing $z_m^l$ through the activation function $f$. This architecture allows the convolutional layer to effectively capture local spatial features, and when combined with the updates from the fractional-order derivative method, significantly enhances the precision of the learning process [33].

$$\begin{aligned} z_m^l &= conv\left(w_m^l, a^{l-1}\right) + b_m^l \\ a_m^l &= f\left(z_m^l\right) \end{aligned} \tag{3}$$

where

- $\mathrm{conv}(\cdot)$ represents the convolution operation.
- $z_m^l$ is the output feature map before activation for the $m$-th filter in the $l$-th layer.
- $a_m^l$ is the activated output of the $m$-th filter in the $l$-th layer.
- $w_m^l$ is the weight matrix (filter) for the $m$-th filter in the $l$-th layer.
- $b_m^l$ is the bias term associated with the $m$-th filter.

2. Backward Propagation with Fractional-Order Gradients

Fractional-Order Gradient Calculation

To enable the network to more flexibly adapt to varying scales of feature changes, we employ the Caputo fractional derivative as a replacement for conventional gradient calculations. Using Equation (4), we can compute the fractional-order derivative $\mathcal{D}_v^w L$ of the loss function $L$ with respect to each weight and bias. This method allows for a more adaptable adjustment process based on the different states of the weights [44].

For both fully connected and convolutional layers, the gradients of the loss function $L$ with respect to weights and biases are approximated as

$$\begin{aligned} \mathcal{D}_v^w L &\approx \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial w} \cdot \frac{1}{\Gamma(2-v)} \cdot w^{1-v} \\ \mathcal{D}_v^b L &\approx \frac{\partial L}{\partial z} \cdot \frac{1}{\Gamma(2-v)} \cdot b^{1-v} \end{aligned} \tag{4}$$

where

- $\mathcal{D}_v$ denotes the Caputo fractional derivative operator.
- $v$ is the fractional order.
- $\frac{\partial L}{\partial z}$ is the gradient of the loss with respect to the neuron's pre-activation output $z$.
- $w$ and $b$ represent the current weights and biases.

Weight and Bias Update

Equation (5) illustrates the weight and bias update process using fractional-order gradients. Unlike traditional update methods that rely on integer-order gradients, the Caputo fractional derivative introduces memory effects by considering the cumulative history of parameter changes. This allows for smoother, non-local updates, improving convergence and reducing oscillations during training [33].

The weights and biases are updated using the fractional-order gradients as follows:

$$\begin{aligned} w^{t+1} &= w^t - \eta \cdot \mathcal{D}_v^w L \\ b^{t+1} &= b^t - \eta \cdot \mathcal{D}_v^b L \end{aligned} \tag{5}$$

where

- $w^{t+1}$ and $b^{t+1}$ are the updated weights and biases.
- $w^t$ and $b^t$ are the current weights and biases.
- $\eta$ is the learning rate.
- $\mathcal{D}_v^w L$ and $\mathcal{D}_v^b L$ are the fractional-order gradients with respect to weights and biases.

### *3.2. Differential Evolution for Optimization*

Differential Evolution Algorithm

Differential evolution (DE) is a population-based optimization algorithm, widely used for complex optimization tasks due to its simplicity and efficiency [45]. The DE algorithm was selected for this study because it is particularly well-suited to non-convex optimization problems such as neural network training, where traditional gradient-based methods often struggle with local optima [46]. DE's mutation, crossover, and selection mechanisms enable it to explore a wide solution space effectively, making it more likely to find global optima.

In our neural network architecture, DE is used to initialize the weights and biases before training begins. This initialization ensures that the network starts with parameters that are already close to an optimal configuration, thereby reducing the number of iterations required for convergence during backpropagation [41]. As a result, the combination of DE for initialization and fractional-order gradients for training leads to a more efficient and effective learning process; see Figure 2.

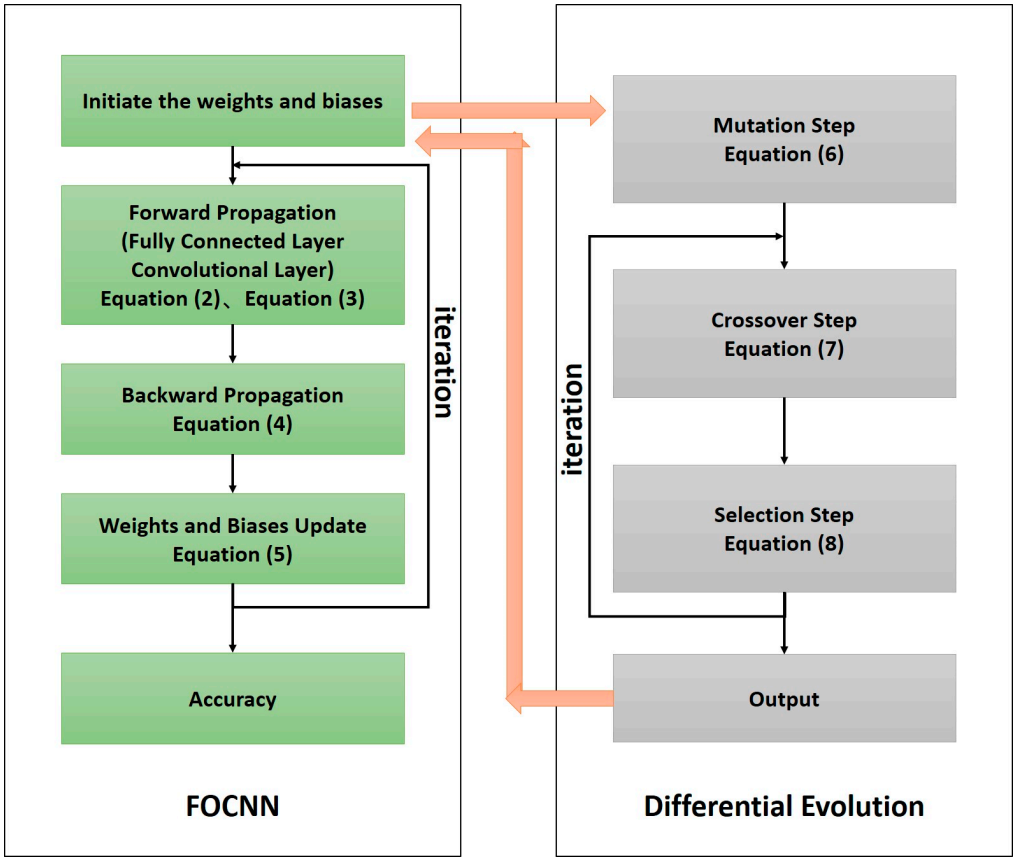



**Figure 2.** The flowchart of the proposed DE-FOCNN algorithm.

The optimization process in DE consists of three key steps: mutation, crossover, and selection, which iteratively improve the solutions within a population [24]. Each individual in the population represents a potential solution to the optimization problem, with the fitness function measuring the network's performance. Below, we explain the mathematical formulation behind each step.

1. Mutation Step

In the mutation step, new candidate solutions (mutant vectors) are generated by perturbing existing solutions. The basic formula for generating a mutant vector is given by

$$V_i^{(g+1)} = X_{r1}^{(g)} + F \cdot \left( X_{r2}^{(g)} - X_{r3}^{(g)} \right) \tag{6}$$

where

- $V_i^{(g+1)}$ is the mutant vector for the $i$-th individual at generation $g+1$.
- $X_{r1}^{(g)}$, $X_{r2}^{(g)}$, $X_{r3}^{(g)}$ are randomly selected individuals from the population.
- $F$ is a scaling factor controlling the amplification of the differential variation.

The purpose of this step is to explore the solution space more effectively by introducing diversity into the population. By using differences between randomly selected solutions, DE ensures that the exploration is adaptive and capable of escaping local optima.

2. Crossover Step

The crossover step combines the mutated vector with the current individual to generate a trial vector. This is governed by the following equation:

$$U_{i,j}^{(g+1)} = \begin{cases} V_{i,j}^{(g+1)} & \text{if } \text{rand}_j(0,1) \leq CR \text{ or } j = j_{\text{rand}} \\ X_{i,j}^{(g)} & \text{otherwise} \end{cases} \tag{7}$$

where

- $U_{i,j}^{(g+1)}$ is the trial vector, for the $i$-th individual at the $j$-th dimension.
- $CR$ is the crossover probability, determining how often components from the mutant vector are used.
- $j_{\text{rand}}$ is a randomly chosen index to ensure that at least one component from the mutant vector is inherited.

The crossover step balances exploration and exploitation by mixing information from both the mutant vector and the current individual.

3. Selection Step

In the selection step, the algorithm evaluates the fitness of the trial vector against the current individual. The individual with the better fitness value is retained for the next generation.

$$X_i^{(g+1)} = \begin{cases} U_i^{(g+1)} & \text{if } f\left(U_i^{(g+1)}\right) \leq f\left(X_i^{(g)}\right) \\ X_i^{(g)} & \text{otherwise} \end{cases} \tag{8}$$

where $f(\cdot)$ represents the fitness function, which measures the network's performance. This step ensures that only improved solutions are carried over to the next generation, gradually guiding the population toward an optimal or near-optimal solution.

## 4. Experiments

### *4.1. Settings*

#### 4.1.1. Datasets

The dataset used in this study consists of both a classic music dataset and additional real-world validation data. The classic dataset originates from the appendix of Janata's 2012 study [9], which includes 312 music clips spanning various genres. These tracks were

previously rated, with scores ranging from 40 to 107, and were subsequently normalized to a 0–100 scale. The tracks were then divided into three groove levels (low, moderate, and high) by splitting the scores into tertiles.

For real-world validation, 20 participants were recruited to perform a cycling task while listening to randomly selected music tracks from the dataset. Each participant was assigned 40 clips in total, presented in random order, during the cycling task. Their cadence (pedal stroke rate) was recorded and analyzed. Tracks were excluded from the dataset if the correlation between the participant's pedal stroke rate and the music's BPM was lower than 0.5; see Figure 3.

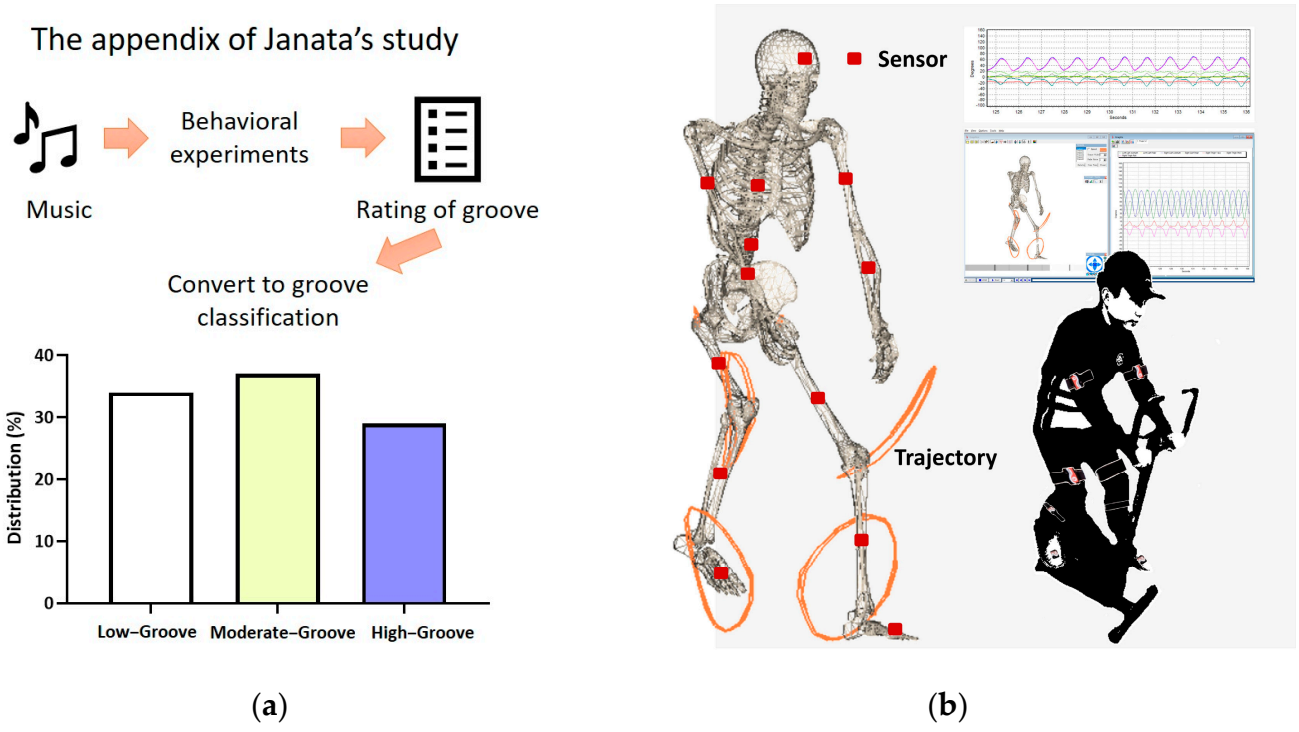



**Figure 3.** Dataset and experimental validation. (**a**) Janata and colleagues utilized behavioral and computational techniques to measure the sensorimotor coupling in response to grooves, thereby establishing a groove rating; (**b**) real-world validation using cycling task.

The use of pedal stroke rate to validate the groove levels in Janata's dataset was chosen because Janata also employed similar behavioral experiments to determine groove levels. Groove's most essential characteristic is its ability to induce synchronization between movement and music, rather than evoking emotional responses. Therefore, conducting such behavioral experiments provides the most direct way to assess groove.

Additionally, since Janata's playlist has already been validated [10–12], this study further refined the dataset by removing outdated tracks or those that failed to evoke a sense of movement synchronization in modern participants. No new dataset was created, as the purpose was to optimize the model training by filtering existing data effectively. Cycling was chosen as the experimental activity because groove has significant potential to enhance physical performance in cycling and improve motor rhythm in Parkinson's patients [47], where cycling has been shown to offer a safer and more effective intervention than other forms of movement.

Cycling Task Setup

The cycling task was conducted using a stationary bicycle. The seat height was adjusted to approximately 88% of each participant's inner leg length, ensuring a comfortable and natural pedaling posture. A standardized protocol was followed throughout each session. Before starting the formal task, participants completed a 3-minute warm-up phase at a torque of 2.5 N·m, maintaining a cadence of approximately 55–60 revolutions per

minute (rpm). Following the warm-up, the formal cycling task began, with the torque increased to 3 N·m. Participants began cycling at the start of each music stimulus, and after one minute of cycling (with the music clip looping), data from the last 20 seconds were recorded. The last 20 seconds were collected to ensure that participants had sufficient time to adapt and reach a stable rhythm before their cadence was measured.

Motion capture was performed using the FAB inertial measurement unit (IMU) system (BIOSYN, Surrey, BC, Canada) to continuously track joint trajectories and analyze participants' pedal cycles. Seven IMUs were placed on the pelvis, thighs, shanks, and feet, with a sampling frequency of 100 Hz, as shown in Figure 3b.

After the experiment, 54 music clips were excluded for not meeting the correlation threshold, leaving a final dataset consisting of 258 music clips with corresponding groove classification labels. These labels reflect the groove levels determined by participant ratings in Janata's study, where listeners evaluated how much each track encouraged bodily movement. The training dataset comprises 80% of the data and the testing dataset comprises 20%.

#### 4.1.2. Implementation Details

The model used in this study follows a basic CNN architecture, as outlined in Section 3.1.2 of the Methodology. The input to the model is image data, and the output is the corresponding classification of the image into one of the three groove levels. The loss function of the model is the cross-entropy loss function, as shown in Table 1.

**Table 1.** The parameters of FOCNN and DE-FOCNN.

| Parameter | |
|---|---|
| Loss function | $L = -\frac{1}{H}\sum_{S=1}^{H} y_S^{\mathrm{T}}\log(O_S)$ |
| Learning rate: | 0.001 |
| Batch size: | 8 |
| Initial weight: | $w \in [-0.5,\ 0.5]$ |
| Initial bias: | $b \in [-0.5,\ 0.5]$ |
| Number of iteration: | 100 |

The first convolutional layer of the model outputs 64 channels, with a $3 \times 3$ kernel size and a stride of 1, ensuring that the spatial dimensions of the input are preserved through 'same' padding. For the subsequent three convolutional layers, the kernel size and stride remain the same, but the number of channels doubles with each layer, progressively extracting more complex features from the input data. The model also includes two fully connected (dense) layers. The first fully connected layer consists of 2048 neurons, and the second fully connected layer consists of 1024 neurons, providing the capacity to learn intricate feature representations. For more details on the network architecture, please refer to Figure 1.

The model was implemented using the PyTorch framework (version 2.0.1) in Python (version 3.8.17), an open-source deep learning library widely used in research and industry. The training was conducted on a machine running Windows 10 with 32 GB of RAM. The model leveraged GPU acceleration for faster training, utilizing an NVIDIA RTX 4090 GPU with 24 GB of VRAM. All software frameworks and training hardware used in this study are open-source or commercially available and do not require any additional licenses for research or academic purposes.

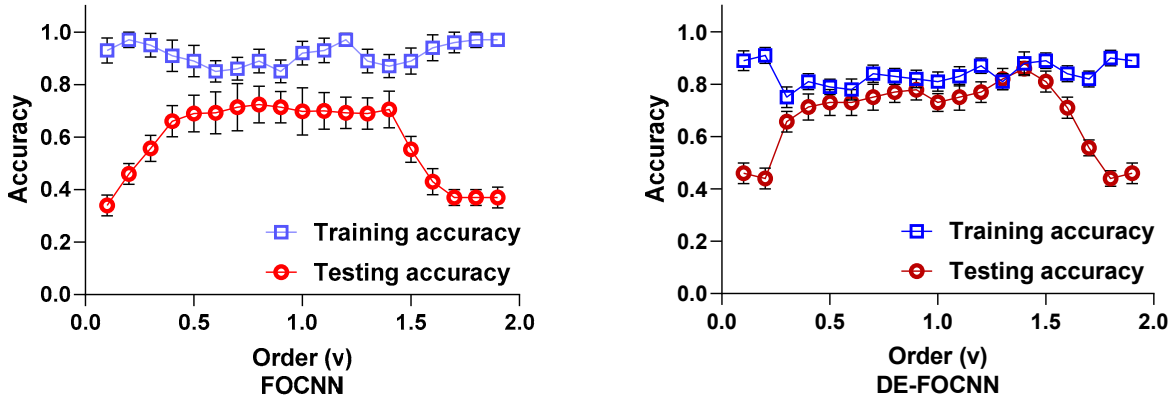
The specific parameters used during the iterative training process are detailed in Tables 1 and 2. Each experiment for the different algorithms was repeated 10 times to ensure consistency and reliability of the results. For each iteration, all parameters—including weights, biases, and the order of sample inputs—were randomly initialized to avoid any biases in the training process.

**Table 2.** The parameters of DE-FOCNN, GA-FOCNN, and PEO-FOCNN.

| Algorithm | Parameters |
|---|---|
| DE-FOCNN | $v = 1.4$, crossover probability $CR = 0.7$, scaling factor $F = 0.8$ |
| GA-FOCNN | $v = 1.4$, crossover probability $CP = 0.6$, mutation probability $m_p = 0.05$ |
| PEO-FOCNN | $v = 1.4$, mutation parameter $\lambda = 0.7$ |

### *4.2. Experimental Results*

We conducted experiments to compare the proposed DE-FOCNN with the standard FOCNN across a range of different fractional-order values. As shown in Figure 4, the DE-FOCNN consistently achieved higher average test accuracy than the standard FOCNN and exhibited less overfitting on the training set. This demonstrates the effectiveness of the DE algorithm in optimizing the performance of FOCNN. Additionally, for both DE-FOCNN and FOCNN, the Caputo fractional-order gradient method (CFOGM) outperformed traditional integer-order gradient methods at specific fractional orders, such as $v = 0.8$, $v = 0.9$, and $v = 1.3$. Notably, when the fractional order $v = 1.4$, DE-FOCNN achieved its best performance.




(**a**) (**b**)

**Figure 4.** The average training and testing accuracies of FOCNN and DE-FOCNN with different fractional orders. (**a**) Fractional-order convolutional neural network; (**b**) differential evolution—fractional-order convolutional neural network.

When $v = 1$, both FOCNN and DE-FOCNN function as integer-order neural networks, equivalent to traditional CNNs and DE-optimized CNNs, respectively. Our results indicate that the testing accuracy of FOCNN shows a slight downward trend compared to its performance at $v = 0.7$, $v = 0.9$, and $v = 0.9$, although the decline is not significant. In contrast, DE-FOCNN exhibits a more noticeable drop in performance at $v = 1$. Specifically, the testing accuracy at fractional orders 0.9 and 1.1 is significantly higher than at $v = 1$, further emphasizing the superiority of fractional-order models over integer-order ones.

A horizontal comparison between FOCNN and DE-FOCNN at $v = 1$ reveals a testing accuracy of 0.69 vs. 0.73, respectively, demonstrating that the DE-optimized CNN outperforms the traditional CNN.

To further validate the superiority of DE-FOCNN over other popular evolutionary algorithm-based FOCNN models (such as GA-FOCNN and PEO-FOCNN), multiple experiments were conducted on the Janata dataset. The upper and lower bounds of the initial biases and weights were set to 0.5. The parameters for DE-FOCNN, GA-FOCNN, and PEO-FOCNN are listed in Table 2.

The results for test accuracy across DE-FOCNN, GA-FOCNN, and PEO-FOCNN are presented in Table 3. It is evident from Table 3 that DE-FOCNN exhibited higher average

test accuracy compared to the other algorithms, and its variance was lower than that of GA-FOCNN and PEO-FOCNN, further demonstrating its stability. The confusion matrix of the 10 experiments for DE-FOCNN is shown in Figure 5.

**Table 3.** The statistical results of testing accuracy of DE-FOCNN, GA-FOCNN, and PEO-FOCNN.

| Algorithm | Average | Standard Deviation |
|---|---|---|
| DE-FOCNN | 0.859 | 0.027 |
| GA-FOCNN | 0.843 | 0.073 |
| PEO-FOCNN | 0.855 | 0.034 |

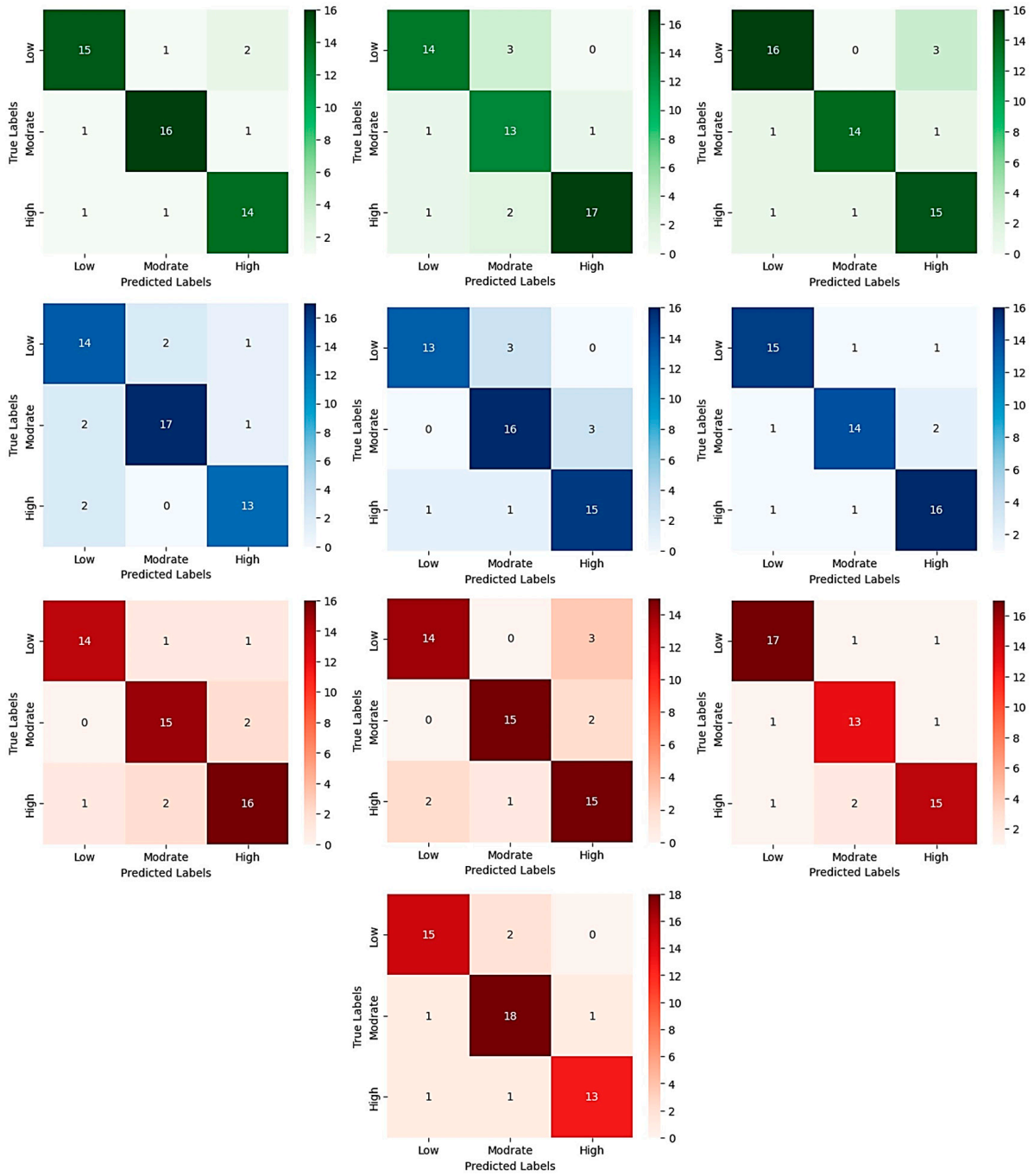
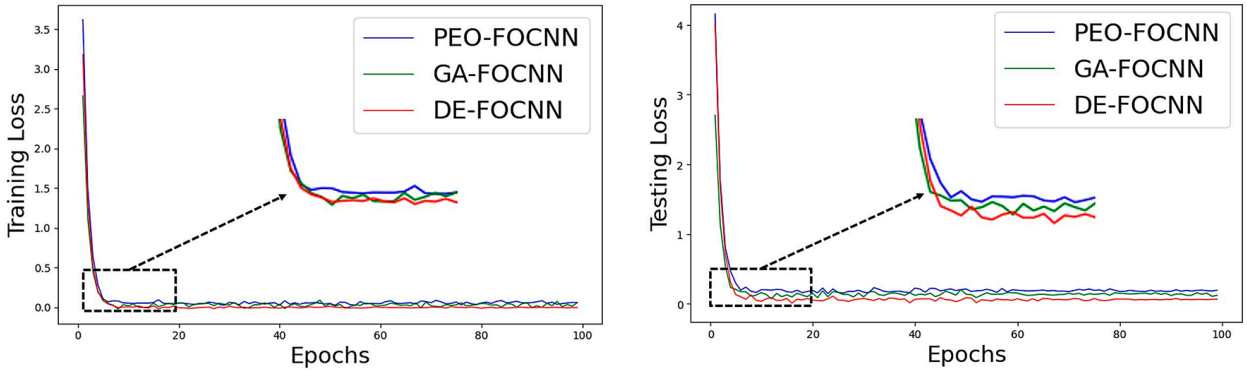



**Figure 5.** The confusion matrix of the DE-FOCNN model with a fractional order of 1.4 over 10 experimental trials.

Figure 6 illustrates the evolution of the loss functions for DE-FOCNN, GA-FOCNN, and PEO-FOCNN on both the training and test sets. It is clearly visible that DE-FOCNN achieves a lower loss value compared to the other three algorithms, indicating its superior optimization performance.




(**a**) (**b**)

**Figure 6.** The training and testing accuracies of different algorithms in evolutionary processes: (**a**) training loss; (**b**) testing loss.

Additionally, to verify the generalizability of DE-FOCNN and its effectiveness on a new dataset, we applied the model to classify the groove levels of music from the new dataset. We also conducted a cycling task to measure the synchronization between movement and music to validate the effectiveness of DE-FOCNN. The results indicated that DE-FOCNN achieved good classification performance on the new data, demonstrating strong generalizability. For detailed results, please refer to Supplementary Material, Figure S1.

## 5. Discussion

Groove plays a significant role in various fields, including motor rehabilitation, enhancement of motor function, and the promotion of physical activity. However, accurately measuring and predicting groove remains a challenging task. This study proposes a differential evolution (DE)-based optimization approach for fractional-order convolutional neural networks (FOCNNs) aimed at enhancing the accuracy of groove classification in music. Experimental results indicate that the Caputo fractional-order gradient method (CFOGM) outperforms traditional integer-order gradient methods. Furthermore, DE-FOCNN demonstrates significant performance improvement over standard FOCNN across various fractional-order values, achieving higher test accuracy and significantly reducing overfitting. Notably, DE-FOCNN achieved its best performance at the fractional-order value of $v = 1.4$. Additional experiments showed that DE-FOCNN not only had superior accuracy compared to other popular evolutionary algorithms (such as GA-FOCNN and PEO-FOCNN) but also exhibited lower variance.

The experimental results clearly indicate that, firstly, FOCNN performs better at specific fractional-order values compared to integer-order values ($v = 1$). This suggests that fractional-order convolutional neural networks have unique advantages in capturing the fine features present in Mel-spectrograms of music [28]. Secondly, DE-FOCNN demonstrates significant advantages in the optimization of fractional-order neural networks. Standard FOCNNs, due to limitations in the selection of initial parameters, tend to fall into local optima, resulting in slower convergence and suboptimal accuracy during training [19]. By contrast, DE, with its ability to effectively explore the global search space, significantly improves the selection of initial weights and biases in FOCNN [21,23,38], thereby greatly enhancing model training performance. This is particularly evident in reducing overfitting on the training set and improving accuracy on the test set, which is consistent with previous studies that utilized DE for optimizing neural networks [17,21,41].

When comparing DE-FOCNN with other optimization methods (GA and PEO), DE-FOCNN achieved slightly higher average test accuracy compared to PEO-FOCNN, with noticeably lower variance. This suggests that DE-FOCNN not only has an advantage in classification accuracy but also outperforms other optimization algorithms in terms of robustness. The evolution of the loss function shows that DE-FOCNN experiences less fluctuation on the training set and converges more quickly on the test set compared to GA-FOCNN and PEO-FOCNN, further demonstrating its effectiveness in the optimization process. Previous studies have also found that DE excels in terms of stability and robustness [17].

Despite the promising results, there are several limitations of this study that need to be acknowledged and addressed. This work is an exploratory study in a relatively new research area, specifically the application of fractional-order convolutional neural networks (FOCNNs) in music classification, which is still in its early stages. Our current optimization approach primarily focuses on the selection of initial parameters (weights and biases), while future work should explore further optimizations of the network structure itself. We also limited our experiments to relatively simple CNN architectures. Future research should investigate more complex CNN designs, such as multi-layer and multi-channel convolutional networks, incorporating fractional-order transformations and appropriate optimization techniques to achieve better performance. Additionally, DE-FOCNN faces challenges related to training time, particularly when dealing with large datasets and complex audio data. More efficient optimization algorithms could be explored in the future to reduce training time. Finally, to further enhance the credibility and generalizability of the results, future work should include a more diverse group of participants and larger datasets to ensure broad applicability of the findings.

## 6. Conclusions

In this study, we proposed a differential evolution (DE)-based optimization approach for fractional-order convolutional neural networks (FOCNNs) to enhance the performance of groove classification in music. This method leverages DE to identify the optimal selection of initial biases and weights from a wide range of values, improving the performance of the model.

Through extensive testing on the Janata dataset, our comparative experiments demonstrated that DE-FOCNN reduces overfitting and achieves higher test accuracy at specific fractional-order values compared to the original FOCNN. The experiments also showed that DE-FOCNN outperforms other popular optimization algorithms, such as GA-FOCNN and PEO-FOCNN, in terms of accuracy and variance reduction, confirming the effectiveness of DE as an optimization strategy for fractional-order models in groove classification tasks.

This research primarily contributes to the optimization of FOCNNs by introducing a novel DE-based approach for the automated analysis and classification of musical groove, offering potential applications in fields such as music recommendation, education, and therapy. The versatility of the DE-FOCNN framework holds promise for addressing other related engineering challenges. Future research could explore the application of DE-FOCNN to broader music-related tasks and larger datasets, with the aim of further optimizing its architecture and validating its generalizability.

**Supplementary Materials:** The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/fractalfract8110616/s1, Figure S1: Scatter plot of the correlation between tempo and cadence for music with different groove levels. (a) high-groove; (b) moderate-groove; (c) low-groove.

**Author Contributions:** Conceptualization, J.C.; methodology, J.C. and G.Z.; software, J.C. and P.S.; validation, J.C.; formal analysis, J.C. and G.Z.; investigation, J.H., D.L. and G.Z.; writing—original draft preparation, J.C.; writing—review and editing, J.C. and D.T.; supervision, D.T. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research is supported by the National Natural Science Foundation of China (No. 71874017).

**Data Availability Statement:** The code is available upon reasonable request to the corresponding author.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References


1. Stupacher, J.; Hove, M.J.; Janata, P. Audio Features Underlying Perceived Groove and Sensorimotor Synchronization in Music. *Music Percept.* **2016**, *33*, 571–589. [CrossRef]
2. Zatorre, R.J.; Chen, J.L.; Penhune, V.B. When the brain plays music: Auditory-motor interactions in music perception and production. *Nat. Rev. Neurosci.* **2007**, *8*, 547–558. [CrossRef] [PubMed]
3. Etani, T.; Miura, A.; Kawase, S.; Fujii, S.; Keller, P.E.; Vuust, P.; Kudo, K. A review of psychological and neuroscientific research on musical groove. *Neurosci. Biobehav. Rev.* **2024**, *158*, 105522. [CrossRef]
4. Witek, M.A.; Clarke, E.F.; Wallentin, M.; Kringelbach, M.L.; Vuust, P. Syncopation, body-movement and pleasure in groove music. *PLoS ONE* **2014**, *9*, e94446. [CrossRef] [PubMed]
5. Madison, G.; Gouyon, F.; Ullen, F.; Hornstrom, K. Modeling the tendency for music to induce movement in humans: First correlations with low-level audio descriptors across music genres. *J. Exp. Psychol. Hum. Percept. Perform.* **2011**, *37*, 1578–1594. [CrossRef]
6. Levitin, D.J.; Grahn, J.A.; London, J. The Psychology of Music: Rhythm and Movement. *Annu. Rev. Psychol.* **2018**, *69*, 51–75. [CrossRef]
7. Heng, J.G.; Zhang, J.; Bonetti, L.; Lim, W.; Vuust, P.; Agres, K.; Chen, S.A. Understanding music and aging through the lens of Bayesian inference. *Neurosci. Biobehav. Rev.* **2024**, *163*, 105768. [CrossRef]
8. Thorp, H.H. Music and the mind. *Science* **2024**, *383*, 1271. [CrossRef]
9. Janata, P.; Tomic, S.T.; Haberman, J.M. Sensorimotor coupling in music and the psychology of the groove. *J. Exp. Psychol. Gen.* **2012**, *141*, 54–75. [CrossRef]
10. Chang, A.; Kragness, H.E.; Tsou, W.; Bosnyak, D.J.; Thiede, A.; Trainor, L.J. Body sway predicts romantic interest in speed dating. *Soc. Cogn. Affect. Neurosci.* **2021**, *16*, 185–192. [CrossRef]
11. Stupacher, J.; Hove, M.J.; Novembre, G.; Schutz-Bosbach, S.; Keller, P.E. Musical groove modulates motor cortex excitability: A TMS investigation. *Brain. Cogn.* **2013**, *82*, 127–136. [CrossRef] [PubMed]
12. Wang, Y.; Guo, X.; Wang, H.; Chen, Y.; Xu, N.; Xie, M.; Wong, D.W.; Lam, W.K. Training and retention effects of paced and music-synchronised walking exercises on pre-older females: An interventional study. *BMC Geriatr.* **2022**, *22*, 895. [CrossRef] [PubMed]
13. Mori, K. Decoding peak emotional responses to music from computational acoustic and lyrical features. *Cognition* **2022**, *222*, 105010. [CrossRef]
14. Savage, P.E.; Brown, S.; Sakai, E.; Currie, T.E. Statistical universals reveal the structures and functions of human music. *Proc. Natl. Acad. Sci. USA* **2015**, *112*, 8987–8992. [CrossRef] [PubMed]
15. Schwendicke, F.; Golla, T.; Dreher, M.; Krois, J. Convolutional neural networks for dental image diagnostics: A scoping review. *J. Dent.* **2019**, *91*, 103226. [CrossRef]
16. Cramer, J.; Wu, H.; Salamon, J.; Bello, J.P. Look, Listen, and Learn More: Design Choices for Deep Audio Embeddings. In Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; pp. 3852–3856.
17. Chen, B.; Chen, Y.; Zeng, G.; She, Q. Fractional-order convolutional neural networks with population extremal optimization. *Neurocomputing* **2022**, *477*, 36–45. [CrossRef]
18. Fan, H.; Rao, Y.; Shi, K.; Wen, H. Time-Varying Function Matrix Projection Synchronization of Caputo Fractional-Order Uncertain Memristive Neural Networks with Multiple Delays via Mixed Open Loop Feedback Control and Impulsive Control. *Fractal Fract.* **2024**, *8*, 301. [CrossRef]
19. Sheng, D.; Wei, Y.; Chen, Y.; Wang, Y. Convolutional neural networks with fractional order gradient method. *Neurocomputing* **2020**, *408*, 42–50. [CrossRef]
20. Alsadi, W.; Wei, Z.; Moroz, I.; Alkhazzan, A. Existence and stability of solution in Banach space for an impulsive system involving Atangana-Baleanu and Caputo-Fabrizio derivatives. *Fractals* **2023**, *31*, 2340085. [CrossRef]
21. Ikushima, N.; Ono, K.; Maeda, Y.; Makihara, E.; Hanada, Y. Differential Evolution Neural Network Optimization with Individual Dependent Mechanism. In Proceedings of the 2021 IEEE Congress On Evolutionary Computation (CEC 2021), Kraków, Poland, 28 June–1 July 2021; pp. 2523–2530.
22. Baldominos, A.; Saez, Y.; Isasi, P. Evolutionary convolutional neural networks: An application to handwriting recognition. *Neurocomputing* **2018**, *283*, 38–52. [CrossRef]
23. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
24. Ahmad, M.F.; Isa, N.A.M.; Lim, W.H.; Ang, K.M. Differential evolution: A recent review based on state-of-the-art works. *Alex. Eng. J.* **2022**, *61*, 3831–3872. [CrossRef]

25. Hove, M.J.; Martinez, S.A.; Stupacher, J. Feel the bass: Music presented to tactile and auditory modalities increases aesthetic appreciation and body movement. *J. Exp. Psychol. Gen.* **2020**, *149*, 1137–1147. [CrossRef]
26. Vempala, N.N.; Russo, F.A. Modeling Music Emotion Judgments Using Machine Learning Methods. *Front. Psychol.* **2017**, *8*, 2239. [CrossRef]
27. Choi, K.; Fazekas, G.; Sandler, M.; Cho, K. Transfer learning for music classification and regression tasks. *arXiv* **2017**, arXiv:1703.09179.
28. Choi, S.; Park, J.I.; Hong, C.H.; Park, S.G.; Park, S.C. Accelerated construction of stress relief music datasets using CNN and the Mel-scaled spectrogram. *PLoS ONE* **2024**, *19*, e0300607. [CrossRef] [PubMed]
29. Aguilar, C.J.Z.; Gomez-Aguilar, J.F.; Alvarado-Martinez, V.M.; Romero-Ugalde, H.M. Fractional order neural networks for system identification. *Chaos Solitons Fractals* **2020**, *130*, 109444. [CrossRef]
30. Yin, C.; Chen, Y.; Zhong, S. Fractional-order sliding mode based extremum seeking control of a class of nonlinear systems. *Automatica* **2014**, *50*, 3173–3181. [CrossRef]
31. Aslam, M.S.; Raja, M.A.Z. A new adaptive strategy to improve online secondary path modeling in active noise control systems using fractional signal processing approach. *Signal Process.* **2015**, *107*, 433–443. [CrossRef]
32. Bao, C.; Pu, Y.; Zhang, Y. Fractional-Order Deep Backpropagation Neural Network. *Comput. Intell. Neurosci.* **2018**, *2018*, 7361628. [CrossRef]
33. Wang, J.; Wen, Y.; Gou, Y.; Ye, Z.; Chen, H. Fractional-order gradient descent learning of BP neural networks with Caputo derivative. *Neural Netw.* **2017**, *89*, 19–30. [CrossRef] [PubMed]
34. Yang, H.; Fan, R.; Chen, J.; Xu, M. Recurrent Neural Networks with Fractional Order Gradient Method. In Proceedings of the 2022 14th International Conference on Advanced Computational Intelligence (ICACI), Wuhan, China, 15–17 July 2022; pp. 49–55.
35. Pu, Y.; Yi, Z.; Zhou, J. Fractional Hopfield Neural Networks: Fractional Dynamic Associative Recurrent Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 2319–2333. [CrossRef] [PubMed]
36. Ramadevi, B.; Kasi, V.R.; Bingi, K. Hybrid LSTM-Based Fractional-Order Neural Network for Jeju Island's Wind Farm Power Forecasting. *Fractal Fract.* **2024**, *8*, 149. [CrossRef]
37. Donate, J.P.; Li, X.; Sanchez, G.G.; de Miguel, A.S. Time series forecasting by evolving artificial neural networks with genetic algorithms, differential evolution and estimation of distribution algorithm. *Neural Comput. Appl.* **2013**, *22*, 11–20. [CrossRef]
38. Leema, N.; Nehemiah, H.K.; Kannan, A. Neural network classifier optimization using Differential Evolution with Global Information and Back Propagation algorithm for clinical datasets. *Appl. Soft Comput.* **2016**, *49*, 834–844. [CrossRef]
39. Eltaeib, T.; Mahmood, A. Differential Evolution: A Survey and Analysis. *Appl. Sci.* **2018**, *8*, 1945. [CrossRef]
40. Wang, L.; Zeng, Y.; Chen, T. Back propagation neural network with adaptive differential evolution algorithm for time series forecasting. *Expert Syst. Appl.* **2015**, *42*, 855–863. [CrossRef]
41. Huang, J.; Zeng, G.; Geng, G.; Weng, J.; Lu, K.; Zhang, Y. Differential evolution-based convolutional neural networks: An automatic architecture design method for intrusion detection in industrial control systems. *Comput. Secur.* **2023**, *132*, 103310. [CrossRef]
42. Duchanoy, C.A.; Moreno-Armendariz, M.A.; Urbina, L.; Cruz-Villar, C.A.; Calvo, H.; Rubio, J.d.J. A novel recurrent neural network soft sensor via a differential evolution training algorithm for the tire contact patch. *Neurocomputing* **2017**, *235*, 71–82. [CrossRef]
43. Shin, Y.; Darbon, J.; Karniadakis, G.E. Accelerating gradient descent and Adam via fractional gradients. *Neural Netw.* **2023**, *161*, 185–201. [CrossRef] [PubMed]
44. Liu, Y.; Shao, Q.; Liu, Y.; Yang, D. An interval neural network-based Caputo fractional-order extreme learning machine applied to classification. *Appl. Soft Comput.* **2024**, *167*, 112310. [CrossRef]
45. Halder, U.; Das, S.; Maity, D. A cluster-based differential evolution algorithm with external archive for optimization in dynamic environments. *IEEE T. Cybern.* **2013**, *43*, 881–897. [CrossRef] [PubMed]
46. Liu, X.F.; Zhan, Z.H.; Zhang, J. Resource-Aware Distributed Differential Evolution for Training Expensive Neural-Network-Based Controller in Power Electronic Circuit. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 6286–6296. [CrossRef] [PubMed]
47. Nombela, C.; Hughes, L.E.; Owen, A.M.; Grahn, J.A. Into the groove: Can rhythm influence Parkinson's disease? *Neurosci. Biobehav. Rev.* **2013**, *37*, 2564–2570. [CrossRef] [PubMed]