EURASIP Journal on Audio, Speech, and Music Processing

METHODOLOGY **Open Access**

Music time signature detection using ResNet18

Jeremiah Abimbola[1*], Daniel Kostrzewa[1] and Pawel Kasprowski[1]

Abstract
Time signature detection is a fundamental task in music information retrieval, aiding in music organization. In recent years, the demand for robust and efficient methods in music analysis has amplified, underscoring the significance of advancements in time signature detection. In this study, we explored the effectiveness of residual networks for time signature detection. Additionally, we compared the performance of the residual network (ResNet18) to already existing models such as audio similarity matrix (ASM) and beat similarity matrix (BSM). We also juxtaposed with traditional algorithms such as support vector machine (SVM), random forest, K-nearest neighbor (KNN), naive Bayes, and that of deep learning models, such as convolutional neural network (CNN) and convolutional recurrent neural network (CRNN). The evaluation is conducted using Mel-frequency cepstral coefficients (MFCCs) as feature representations on the Meter2800 dataset. Our results indicate that ResNet18 outperforms all other models thereby showing the potential of deep learning models for accurate time signature detection.

Keywords Music time signature, Machine learning, Residual networks, Deep learning

1 Introduction

Music information retrieval (MIR) is a multidisciplinary field that focuses on developing algorithms and tools to enhance users' browsing, searching, and organizing experiences within large music collections. It encompasses various applications such as music recommendation, music genre classification [1], singing voice detection [2, 3], music composition [4, 5], and music analysis [6], making it an area of continuous research and innovation. From personalized recommendations to seamless music streaming experiences, platforms like Spotify and Apple Music heavily rely on MIR techniques to cater to the diverse musical preferences of their users. Within this context, accurate time signature detection plays a crucial role in curating playlists that align with users' desired moods, genres, or activity levels. However, despite the wealth of research in MIR, the development of robust models specifically tailored for time signature detection remains a challenge, limiting the potential for innovative playlist curation. By accurately detecting time signatures, music streaming platforms can gain valuable insights into the rhythmic structure of songs, enabling the creation of tailored playlists that cater to users' specific preferences.

Time signatures, often referred to as meter, are fundamental to notated music as they provide a framework for measuring musical content, aiding in the segmentation of phrases and establishing rhythmic patterns [7]. Comprising a top number and a bottom number, time signatures convey vital information about the number of beats per measure and the corresponding note values. They can be categorized into simple and complex meters [8] For instance, a time signature of $\frac{2}{4}$ represents a simple duple meter with two beats per measure, where each beat divides into two equal parts. Conversely, $\frac{6}{8}$ signifies a complex (compound) meter with six beats per measure, where each beat divides into three equal parts.

Time signatures can also be said to be regular or irregular with the latter being the most challenging to detect

*Correspondence:
Jeremiah Abimbola
jeremiah.oluwagbemi.abimbola@polsl.pl
[1] Applied Informatics, Silesian University of Technology, Gliwice 44-100, Poland

[9]. Regular time signatures, such as $\frac{2}{4}$, $\frac{3}{4}$, $\frac{4}{4}$, and $\frac{6}{8}$, follow a predictable beat pattern and find wide use across music genres. In contrast, irregular time signatures like $\frac{5}{4}$, $\frac{7}{8}$, and $\frac{9}{8}$ introduce rhythmic patterns that differ from conventional meters, challenging human perception by presenting irregularities compared to what is typical or expected. For example, $\frac{5}{4}$ appears in progressive rock and jazz, $\frac{7}{8}$ in Balkan and Middle Eastern music, and $\frac{9}{8}$ in Irish, Balkan, and classical compositions.

In this study, our focus lies on the top number of the time signature as it directly relates to beat tracking, while the bottom number is deemed irrelevant for our purposes. For instance, in a simple duple meter like $\frac{2}{4}$, our attention is on the value 2, and similarly, in a compound meter such as $\frac{6}{8}$, we are interested in the value 6. It is worth noting that we considered all possible time signatures ranging from 2 to 12 as potential candidates for analysis. However, for simplicity, we have chosen to concentrate on meters 3, 4, 5, and 7. This is because compound time signatures can still be adequately represented by multiples of simple meters. For instance, $\frac{3}{4}$ and $\frac{6}{8}$ can be regarded as equivalent in some contexts, with the distinction lying in the emphasis on the notes. In $\frac{3}{4}$, emphasis typically falls on the first note of each measure, while in $\frac{6}{8}$, emphasis is often placed on both the first and fourth notes within the measure. Therefore, the meter 4 will represent meters of 2, 4, and 8, while 3 will represent 3, 6, 9, and 12. Meters of 5 and 7 are considered irregular and stand alone without representation by other numbers. These four main meters are treated as classes for the classification task.

Contrary to the assumption that humans effortlessly recognize time signatures and their rhythmic structures in music, accurately detecting and identifying them is often a challenging task. Reproducing this impressive human capability through algorithms based on raw signal processing techniques also presents significant challenges [10]. Researchers have made efforts in developing algorithms to analyze audio signals and extract the time signature value. These techniques employ sophisticated mathematical, signal processing [11–13], and machine learning strategies to decipher the underlying rhythmic patterns. While progress has been made, the results have varied in terms of accuracy and performance. Hence the need for more robust models.

Residual network (ResNet), introduced by He et al. [14] from Microsoft Research, is a deep learning architecture that has revolutionized image classification and was developed to overcome the problem of degradation in deep neural networks leveraging the ImageNet dataset [15]. It was observed that as networks grew deeper, accuracy would saturate and then rapidly decline [16]. This degradation was due to the difficulty of training deeper networks and the challenges in optimization. ResNet's breakthrough approach of utilizing residual connections, or skip connections, allows for the construction of very deep networks by bypassing layers, enabling more efficient training and improved accuracy. These skip connections effectively mitigate the vanishing gradient problem and enable the smooth flow of gradients throughout the network, allowing for the training of extremely deep models while preserving crucial information. Up till now, no study has used ResNet on MFCC data to detect music time signature.

Therefore, in this paper, we propose a novel approach that utilizes ResNet18, for time signature detection. By adapting ResNet18 to the domain of audio analysis, we aim to harness its remarkable feature extraction capabilities to accurately identify and classify time signatures. Our methodology incorporates leveraging this model to train large-scale audio datasets in order to enhance the model's generalization and performance. Through extensive experimentation and evaluation, we showcase the effectiveness of our approach across various musical genres.

The paper is structured as follows: a brief overview of related work is provided in Section 2. The dataset used for feature extraction and analysis is described in Section 3. Section 4 deals with automatic meter detection using the ResNet18 model, while Section 5 provides the results and discussion of the findings, and finally, Section 6 gives the conclusions and suggests potential directions for future improvements.

2 Related works

Most of the early work done in meter detection involved digital signal processing techniques. Fourteen years ago, Gainza, a pioneer in this domain, notably propelled the research, and the ongoing studies reflect the inherent complexity of this task [17]. He presented one that produced an audio similarity matrix (ASM) that shows how similar any two beats in a piece of music are. The ASM model, as described in Gainza et al. [18], employs a comparison strategy between longer audio segments, typically bars, and a mixture of shorter audio fragments, which could represent a fraction of a note. This approach is grounded on the assumption that different segments of an audio track contain repeating patterns or bars. By leveraging prior knowledge of the song's tempo, the model constructs a spectrogram with a frame length proportional to a certain percentage of the beat duration. This spectrogram aids in identifying the initial note of the song. Subsequently, a reference ASM as shown in Fig. 1 is generated by computing the Euclidean distance between two frames, denoted as $m = a$ and $m = b$, starting from the

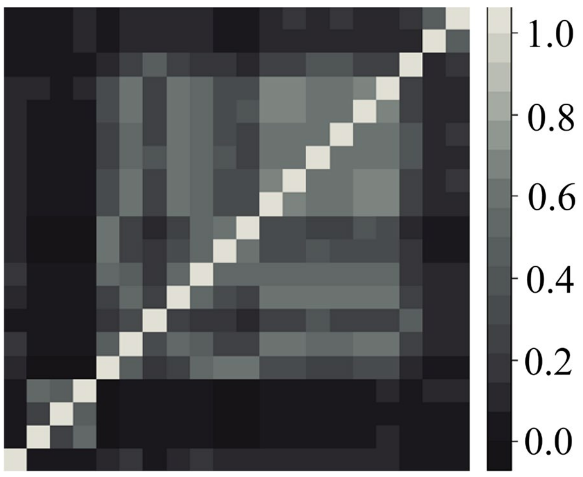

Fig. 1 The audio similarity matrix (ASM) visualizes the diagonal, where each box represents a spectrogram frame

first note obtained using Eq. (1). This method enables the model to capture subtle musical elements such as brief notes. Through a multi-resolution ASM approach, additional audio similarity matrices are produced, each representing different bar lengths. To summarize, the ASM model takes a spectrogram as input, constructs a similarity matrix, extracts the diagonal elements of the matrix, and utilizes these diagonals to infer the meter of a song.

$$ASM(a,b) = \sum_{k=1}^{N/2} [X(a,k) - X(b,k)]^2 \tag{1}$$

where a and b are spectrogram frames, N is the total number of frames, and X is a function of the frames and k is an index used in the summation.

Gainza also reported that the Euclidean distance could not be sufficient enough because of magnitude. The cosine distance measure is then taken to reduce the reliance on magnitude, which is given by (2).

$$ASM(a,b) = 1 - \frac{k = 1}{\sqrt{\sum_{k-1}^{s} X(a,k)^2} * \sqrt{\sum_{k-1}^{s} X(b,k)^2}} \tag{2}$$

The findings reported by Abimbola et al. in [13] indicated notably satisfactory results when utilizing MIDI files; however, the same level of performance cannot be guaranteed when dealing with audio samples. Gulati et al. [19] introduced a meter detection model tailored for Indian music, leveraging a two-stage comb filter-based approach initially designed for double/triple meter estimation. This model was further adapted to accommodate septuple meter signatures like $\frac{7}{8}$. Through evaluation on a comprehensive library of Indian music, the model demonstrated an average accuracy of 87%, highlighting its effectiveness across diverse musical contexts.

In a study similar to the original proposal by Gainza and extensively elaborated upon by Srinivasamurthy et al. in [20], the beat similarity matrix (BSM) model integrates beat tracking alongside the generation of a spectrogram (S) segmented into beat-synchronous frames (m), as depicted in Eq. 3. From these frames, a similarity matrix is derived, and the diagonal of this matrix is employed to ascertain the meter of the song.

$$Bi = Si(k, m) \tag{3}$$

Here, for the i-th beat Bi, ti represents the beat locations, with $ti \leq m < ti + 1$, and $t0 = 1$, and k indicates a particular frequency bin within a specific beat-synchronous frame of the spectrogram. This model was applied to estimate time signatures across various sets of Indian classical music tracks, achieving the highest accuracy of 68.8% for one of the datasets used in the study.

McLeod et al. [12] estimated the meter in symbolic music using a lexicalized probabilistic context-free grammar (PCFG) where each terminal is given a head that corresponds to the note that lasts the longest. To match a specific musical composition with understood metrical stress patterns, the grammar makes use of rhythmic signals. This makes the more complicated rhythmic dependencies found in musical compositions easier to model. Lexicalization is the process by which strong heads (those denoting longer notes) move up the metrical tree to the non-terminals. Instead of assuming independence as in a conventional PCFG, this enables the language to represent rhythmic interdependence, and the pattern of strong and weak beats and sub-beats is utilized to ascertain the underlying rhythmic stress pattern of a specific piece of music.

Bas de Haas et al. [21] proposed a model for meter detection using inner metric analysis (IMA), which determines a piece's meter and the initial downbeat location. The IMA model is used to understand a piece's metrical structure given the onset data. By doing a periodicity analysis, IMA determines the strong and weak metrical locations in a work, producing a weight profile for the entire piece. IMA is then transformed into a feature vector and probabilistically models the detection of the meter and initial downbeat position. They showed that PRIMA outperformed autocorrelation-based meter detection [22] as implemented in the MIDI toolbox [23] on the RAG dataset [24] and the FMpop collection (a proprietary dataset).

Eck et al. [25] showed that it is possible for music structures (one of which is meter) to be learned using long short-term memory (LSTM) recurrent neural network.

A search for correlation at significant gaps in a sequence can lead to an explosion of options regardless of design. Repetition is a particularly challenging problem for models like neural networks and graphical models (like Hidden Markov models) since it demands memory because music tends to repeat at intervals matching to the metrical hierarchy [26]. As a result, LSTM performed better in this sort of task. A notable success was achieved but this is attributed to the MIDI data used for training.

In another study, Varewyck et al. [27] approached the music meter estimation problem as a classification task. They used a support vector machine (SVM) classifier to predict the time signature of a piece of music based on a set of beat-level features. To obtain the beat-level features, they used an external beat tracker. They also conducted spectral envelope and pitch analysis. In addition to the beat-level features, Varewyck et al. incorporated the concept of inter-beat-interval (IBI) similarity, which was previously introduced by Gouyon et al. [28]. IBI similarity is a measure of the similarity between two successive IBIs as shown in Eq. 4. They found that IBI similarity is a useful feature for music meter estimation. By incorporating IBI similarity into their SVM classifier, they were able to improve the accuracy of their meter estimation system. By applying these methods and developing a hypothesis, they estimated the meter of the music.

$$CS(b) = \frac{\langle \mathbf{z}(b-1), \mathbf{z}(b) \rangle}{\|\mathbf{z}(b-1)\| \|\mathbf{z}(b)\|} \tag{4}$$

where b is the beat, and $\mathbf{z}(b-1)$ and $\mathbf{z}(b)$ are low-dimensional vectors grouped by related features. With a balanced collection of 30 song samples, they eventually developed an automated meter classification approach with the optimal feature combination that caused an error of around 10% in duple/triple meter classification and about 28% in meter 3, 4, and 6.

It is worthy to note that all the aforementioned studies use different dataset, often proprietary, thus making direct comparisons challenging.

3 Dataset

In this study, the Meter2800 dataset created by Abimbola et al. [29] was used. It was put together by merging three renowned datasets within the music information retrieval (MIR) domain: GTZAN [30], FMA [31], and MagnaTagATune [32]. Additional audio files were incorporated to supplement the deficiencies identified in these datasets. The collective compilation serves the purpose of providing an extensive and diverse resource for the investigation and exploration of meter detection methodologies.

It contains annotations of 2800 audio samples of 30 seconds grouped into 4 meter classes. The classes 3 and 4 have 1200 audio samples while 5 and 7 both used 200 audio samples respectively as summarized in Table 1. The annotated data is split into train, validation, and test in the ratio of 60:15:25 respectively and written into separate CSV files. As can be observed, the data is unbalanced, the obvious reason can be found if the GTZAN dataset is analyzed for example. It contains roughly 930 out of 1000 tracks having meter numerator values of 2, 3, and 4. This amounts to more than 90% of the dataset. The same can be said for other datasets. This is because hip-hop, rock, and pop are the three most popular musical genres worldwide [33], and their beat counts are simple to estimate. On the other hand, audio tracks with irregular meters are uncommon, which accounts for the dataset's imbalance.

Table 1 Summary of dataset annotated files

Class	Number of files annotated
3	1200
4	1200
5	200
7	200

3.1 Feature extraction

To utilize the ResNet model, we applied it to MFCC matrix values (where each row corresponds to a specific time frame and each column represents a coefficient value) [34]. MFCCs provide a compact representation of the spectral characteristics of an audio signal, capturing essential features for audio analysis and classification tasks [35]. While MFCCs are not interpretable beyond the first coefficient [36], they collectively provide valuable information for machine learning algorithms. The extraction of MFCCs from an audio file involves several steps. First, the audio signal is divided into short overlapping frames of 3 s from a 30-s audio file, in order to capture temporal information. Each frame is then multiplied by a windowing function $w(n)$ to minimize spectral leakage caused by abrupt changes at frame boundaries as shown in Eq. 5. Here, we chose the popular Hamming window as a time window for its computational simplicity. It is defined by the formula: $w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N}\right)$.

$$x[n] = w[n] \cdot \text{frame}[n] \tag{5}$$

Next, the fast Fourier transform (FFT) algorithm is applied to the windowed frames to convert them from the time domain to the frequency domain. This reveals the spectral content of each frame. Following the FFT, a Mel filterbank is employed to the power spectrum obtained. The Mel filterbank consists of a series of triangular filters evenly spaced on the Mel scale, which is a perceptual scale of pitches based on human hearing. After passing the power spectrum through the Mel filterbank, the logarithm of the filterbank energies is taken. This logarithmic compression helps mimic the human auditory system's sensitivity to loudness and reduces the dynamic range of the coefficients [37]. Finally, the discrete cosine transform (DCT) is applied to the logarithmically scaled filterbank energies to obtain the MFCCs as shown in Eq. (6). The DCT de-correlates the coefficients and represents them in a more compact form [36].

To generate the MFCC, we utilized the librosa [38] library in Python where various parameters are needed. These include the audio signal as the primary input, along with parameters such as the sampling rate, which in this case was set to 22050. Additionally, a hop length of 512 was specified, determining the number of samples between successive frames. The number of samples per frame for the discrete Fourier transform (DFT) was set to 2048. These parameters, along with the number of MFCC coefficients to compute, contributed to the extraction of MFCC features from the audio signal.

$$c_n = \sum_{m=0}^{M-1} \log_{10}(s(m)).\cos\left(\frac{\pi n(m-0.5)}{M}\right) \tag{6}$$

where c_n are the cepstral coefficients, $n = 0, 1, 2, ..., C - 1$, and M is the number of MFCCs set to 13 in this study. Consequently, the eventual dimensionality of the MFCC data is (130 × 13), that is, 130 audio frames of 13 coefficients each. The choice of 13 coefficients was determined through a Bayesian optimization process we carried out aimed at maximizing accuracy in the subsequent analysis. To do this, the objective function parameters are initialized with bounds for the search space. The parameters are as follows: the number of MFCCs denoted as mf where $10 \leq mf \leq 15, mf \in \mathbb{I}$; the hop length hp where $512 \leq hp \leq 2048, hp \in \mathbb{I}$; number of mels denoted as nm where $128 \leq nm \leq 1024, nm \in \mathbb{I}$; number of overlap which is a function of the beat duration given as

$$n = \left(\frac{60 \times t_p \times sr}{o}\right) \tag{7}$$

where n is the calculated number of overlaps, $n \in \mathbb{R}$, o is the overlap value $2 \leq o \leq 4, o \in \mathbb{I}$, sr is the sample rate of the signal, t_p is the tempo in bpm of the song, and 60 is the length of beats in a second. After optimization, the possible combination of these parameters that yielded the highest accuracy on the model was as follows: $mf = 13$, $nm = 1024$, $hp = 512$, and $o = 2$.

3.2 Data availability

The Meter2800 is accessible on the Harvard Dataverse via the link (https://bit.ly/meter2800). The dataset is licensed under CC0 1.0 Universal (CC0 1.0) Public Domain Dedication. This indicates that the creators of Meter2800 have waived all copyright and related rights to the extent allowed by law, placing the dataset in the public domain. Users are free to copy, modify, distribute, and perform the dataset, even for commercial purposes, without asking for permission.

4 Methodology

In this study, we opted to use ResNet18, a variant of the ResNet architecture known for its lightweight design compared to other variants. This choice was made due to ResNet18's ability to effectively handle smaller datasets while still capturing crucial features essential for the task at hand. Additionally, it was selected to mitigate the risk of over-fitting, which can be a concern when dealing with limited data. As the name suggests, it has 18 layers, including a fully connected layer, convolutional layers, batch normalization layers, and ReLU activation functions as shown in Fig. 2. The remaining connections are skip connections that allows the network learn implicit feature mappings rather than explicit feature mappings.

In order to establish a baseline for comparison with our proposed ResNet18 model, we conducted experiments with the ASM, BSM and several widely used machine learning algorithms on the same dataset (Meter2800). Specifically, we employed CNN, CRNN, SVM, KNN, naive Bayes, and random forest. The classification task was divided into two groups: one comprising four meter classes (unbalanced) and another consisting of two meter classes (balanced). This division allowed for the evaluation of the model's performance across these different class distributions.

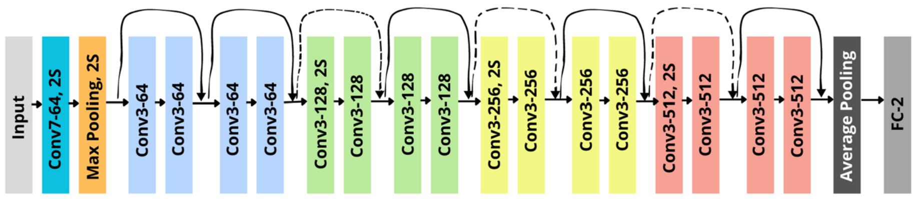

Fig. 2 The ResNet18 architecture developed by He et al. from Microsoft Research

The CNN architecture utilizes the convolutional operation, a fundamental component in extracting features from input data. In CNNs, convolutional layers consist of learnable filters or kernels that perform the convolution operation on the input data to produce feature maps. The convolution operation involves element-wise multiplication of the filter weights with localized patches of the input data, followed by a summation process to generate the output [39].

This convolution operation can be represented as follows:

$$\text{Output}(i,j,k) = \sum_{m=0}^{H-1}\sum_{n=0}^{W-1}\sum_{l=0}^{C_{\text{in}}-1}\left(\text{Input}(i+m,j+n,l)\times\text{Filter}(m,n,l,k)\right)+\text{Bias}(k) \tag{8}$$

Here, $\text{Output}(i,j,k)$ represents the activation at position (i, j) of the k-th feature map, H and W denote the height and width of the filter, C_{in} is the number of input channels, and $\text{Filter}(m,n,l,k)$ denotes the filter weights. The bias term $\text{Bias}(k)$ is added to each convolutional operation.

In the CNN architecture, the convolutional layers consist of three layers, each applying the convolution operation with different learned filters. These layers are followed by batch normalization, which normalizes the output of the convolutional layers to improve network stability and convergence. Subsequently, dropout layers are used to randomly deactivate a fraction of neurons during training, preventing overfitting by enhancing network generalization. The final layer in the CNN is a softmax layer, which computes the probabilities of each class prediction based on the extracted features from the convolutional layers. The CRNN extended the CNN by incorporating two additional layers of long short-term memory (LSTM) units.

The SVM utilized a kernel function (radial basis function——rbf) for non-linear classification, while KNN employed a k value of 3 for determining the nearest neighbors. These algorithms were applied to the MFCC feature data extracted from the audio signals, and their respective performance metrics were evaluated for comparison against our proposed ResNet18 model.

4.1 Detection process

The entire architecture consists of an initial Conv2d layer followed by the ResNet18 base model with modified fully connected layers. In our adaptation, the last layer of 512 × 1000, which was originally designed for ImageNet classification with 1000 classes, was removed and replaced with a new fully connected layers and a softmax layer tailored to our specific classification needs as seen in Fig. 3. The input data is initially passed through a convolutional layer with a 7 × 7 kernel and a stride of 2. This initial Conv2d layer performs a set of convolutions to extract low-level features from the audio data in a 2D format. Unlike images, which typically have multiple color channels, the MFCC input has only one channel being a representation of audio. However, the Conv2d layer outputs a tensor with three channels. It is important to note that although the models (CNN, CRNN, and RESNET) were initially designed for image processing, we adapted them to handle the MFCC input matrix, which contains values representing audio features. To better understand the whole architecture, the process is itemized as follows:

4.1.1 Data preparation

The dataset used is divided into training, validation, and test sets. The train, test, and validation datasets

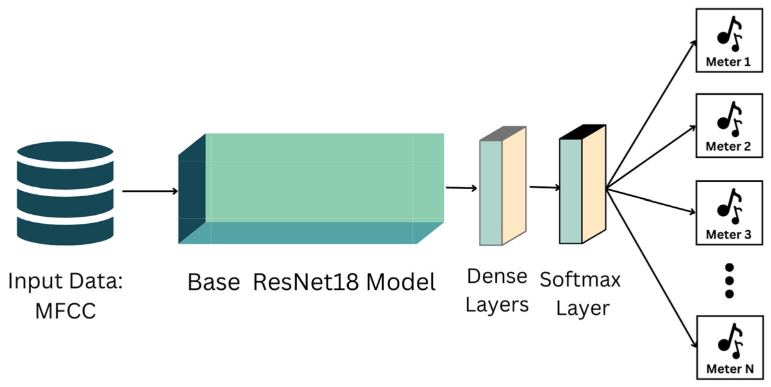

Fig. 3 The ResNet18 architecture adapted for music time signature detection using MFCC features

are organized based on classes, each with a batch size of 64. The training data is shuffled, while the evaluation data retains its original order for consistent evaluation.

4.1.2 Model initialization
While some prior approaches might have utilized pre-trained weights or architectures, this model takes a departure by initializing the ResNet18 model with initial random weights and its fully connected layers are adjusted to match the number of output classes. This strategy eliminates potential biases from pre-trained weights, ensuring a more specialized adaptation to the nuances of time signature detection tasks.

4.1.3 Hyper-parameters
The choice of hyper-parameters, including setting the epoch to 40 for the ResNet18 model and 200 epochs for both the CNN and CRNN, along with a learning rate of 0.001 for all deep learning models, has led to improved performance and convergence.

4.1.4 Optimizer and loss function
The Adam optimizer was used with the cross-entropy loss function, optimizing the model's parameters during training for multi-class classification. For binary classification (where the number of classes M equals 2 as in the case of $\frac{3}{4}$ and $\frac{4}{4}$ classes), cross-entropy is calculated as:

$$CE = -(y\log(p) + (1-y)\log(1-p)) \tag{9}$$

and in the case of $M > 2$ (i.e., multi-class classification), a separate loss is calculated for each class label per observation and sum the result:

$$Loss = -\sum_{c=1}^{M} y_{o,c}\log(p_{o,c}) \tag{10}$$

where y represents the target class (true label), p is the predicted probability for the positive class in binary classification, and $y_{o,c}$ and $p_{o,c}$ are the true and predicted probabilities, respectively, for each class c in multi-class classification for a specific observation o.

4.1.5 Validation
The model's performance was assessed on the validation dataset at the end of each epoch, comparing predictions to ground truth labels to determine generalization ability.

4.1.6 Evaluation
After training, the final model was evaluated on the test dataset, computing test loss and accuracy to gauge performance on unseen MFCC data.

5 Results and discussion
Following the comparison of the ASM, BSM, and various machine learning algorithms including SVM, random forest, KNN, naive Bayes, alongside CNN, CRNN, and ResNet18, the evaluation metrics such as *F*-score, recall, precision, and accuracy were employed to gauge the effectiveness of these models in time signature detection. *F*-score, recall, precision, and accuracy are fundamental evaluation metrics in classification tasks, each providing unique insights into the model's performance. *F*-score, which considers both precision and recall, is particularly valuable in scenarios where there is class imbalance, as it offers a balanced assessment by considering false positives and false negatives. In contrast, for balanced classification tasks where each class is represented fairly equally, accuracy can be a suitable metric as it measures the proportion of correctly classified instances among all instances. In such cases, precision and recall may still provide valuable insights into the model's performance, but accuracy remains a straightforward and intuitive measure of overall correctness.

For models such as ASM and BSM that are not machine learning-based, but rule-based systems or deterministic algorithms, the relevance of *F*-score and recall may be limited and their performance may be evaluated using another metric such as accuracy only as seen in Table 2.

5.1 Performance on 4 classes
When examining time signature detection among four classes (3, 4, 5, and 7), the deep learning models, particularly ResNet18, outperformed the classical models as well as the traditional machine learning algorithms significantly. ResNet18 achieved an *F*-score of 78%. This notably surpassed other deep learning models like CNN and CRNN and substantially outperformed SVM, random forest, KNN, and naive Bayes as shown in Table 2. The confusion matrix in Fig. 4 also provides a clear breakdown of how the model's predictions align with the true class labels, emphasizing areas where the model tends to struggle with specific class distinction too.

5.2 Performance on 2 classes
For the binary classification task involving classes 3 and 4, ResNet18 demonstrated good results achieving

Table 2 Classification report showing metrices for all models

Model	Accuracy	*F*-score	Recall	Precision
Classification report for 4 classes				
Classical models				
ASM	0.51	-	-	-
BSM	0.49	-	-	-
Supervised models				
SVM	0.74	0.71	0.74	0.75
KNN	0.65	0.66	0.65	0.67
Naive Bayes	0.63	0.66	0.63	0.69
Random forest	0.73	0.67	0.73	0.73
Deep learning models				
CNN	0.75	0.75	0.76	0.75
CRNN	0.76	0.75	0.75	0.75
RESNET18	**0.79**	**0.78**	**0.78**	**0.79**
Classification report for 2 classes				
Classical models				
ASM	0.53	-	-	-
BSM	0.50	-	-	-
Supervised models				
SVM	0.86	0.86	0.86	0.87
KNN	0.82	0.82	0.82	0.82
Naive Bayes	0.84	0.84	0.84	0.85
Random forest	0.86	0.86	0.86	0.86
Deep learning models				
CNN	0.87	0.87	0.87	0.87
CRNN	0.86	0.86	0.86	0.86
RESNET18	**0.88**	**0.88**	**0.88**	**0.88**

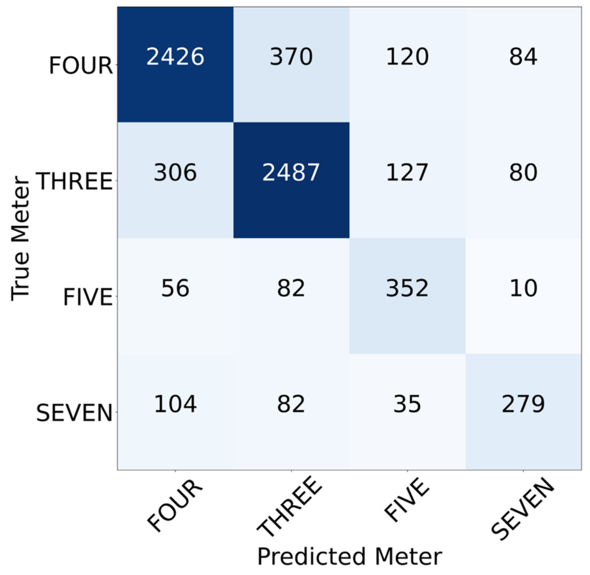

Fig. 4 Confusion matrix for ResNet18 showing 4 meter classes

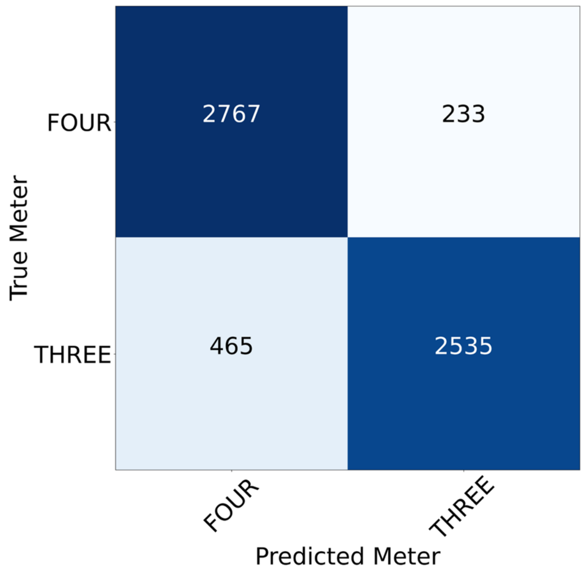

Fig. 5 Confusion matrix for ResNet18 showing 2 meter classes

Table 3 Trainable and non-trainable parameters for different deep learning models

Model	Trainable params	Non-trainable params
ResNet18	11,263,504	11,008
CNN	35,196	256
CRNN	76,612	96

an accuracy of 88%. Comparatively, this performance exceeded all other models, showcasing its effectiveness in distinguishing between these specific time signatures as seen in Table 2 and on the confusion matrix in Fig. 5

The investigation highlighted a distinct performance gap between classical machine learning algorithms (SVM, random forest, KNN, naive Bayes) and deep learning models. While the traditional methods demonstrated reasonable performance, they noticeably trailed behind deep learning architectures, indicating their limitations in capturing intricate patterns crucial for accurate time signature detection. Notably, ResNet18 showed consistent superiority over CNN and CRNN in this task, emphasizing the significance of its residual connections and skip connections.

The model's ability to effectively capture and retain essential features within the Meter2800 dataset underscores its robustness and adaptability in discerning nuanced patterns intrinsic to time signature classification tasks as well as it's potential for music signal processing applications. The obtained results also confirm that ResNet18 is excellent at handling the difficult problem of time signature detection especially because it has larger capacity for learning. Table 3 clearly shows this.

6 Conclusion

In conclusion, our study highlights the effectiveness of ResNet18, for time signature detection. This model significantly outperformed the ASM and BSM, while being slightly better than the traditional machine learning algorithms and other deep learning models used. These findings contribute to the field of music information retrieval and provide valuable insights for developing accurate and robust time signature detection systems. While the study demonstrates promising results, the absence of prior works using the Meter2800 dataset limits direct comparisons. Future research could focus on expanding the dataset and also exploring transfer learning techniques to leverage pre-trained models for time signature detection, further enhancing the model's generalization ability and performance across diverse musical compositions. Furthermore, it is worthwhile to explore the development of more advanced deep learning architectures to examine the potential impact of incorporating additional features for time signature detection which can potentially increase performance.

Abbreviations

SVM	Support vector machine
CNN	Convolutional neural network
CRNN	Convolutional recurrent neural network
KNN	K-nearest neighbors
ASM	Audio similarity matrix
BSM	Beat similarity matrix
RESNET	Residual network
MFCC	Mel-frequency cepstral coefficients
MIR	Music information retrieval
PCFG	Probabilistic context-free grammar
LSTM	Long short-term memory
IMA	Inner metric analysis

Acknowledgements
Not applicable.

Code availability
Not applicable.

Authors' contributions
JA contributed to the methodology and drafted the original manuscript. DK conceptualized the study and provided review and editing of the manuscript. PK contributed to the review and editing of the manuscript.

Funding
This work was partially supported by Statutory Research funds of Department of Applied Informatics, Silesian University of Technology, Gliwice, Poland.

Availability of data and materials
The dataset analyzed during the current study are available from the online url https://bit.ly/meter2800 [29].

Declarations

Consent for publication
All authors provided consent for publication.

Competing interests
The authors declare that they have no competing interests.

Received: 28 February 2024 Accepted: 16 April 2024
Published online: 13 June 2024

References

1. A.R. Rajanna, K. Aryafar, A. Shokoufandeh, R. Ptucha, in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. Deep neural networks: A case study for music genre classification (IEEE, 2015), pp. 655–660
2. S. Leglaive, R. Hennequin, R. Badeau, in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Singing voice detection with deep recurrent neural networks (IEEE, 2015), pp. 121–125
3. R. Monir, D. Kostrzewa, D. Mrozek, Singing voice detection: A survey. Entropy **24**(1), 114 (2022)
4. C. Hernandez-Olivan, J.R. Beltran, Music composition with deep learning: A review. Advances in speech and music technology: computational aspects and applications, pp. 25–50 (2022). arXiv preprint arXiv:2108.12290
5. N. Tokui, H. Iba, et al., in *Proceedings of the third international conference on generative art*. Music composition with interactive evolutionary computation, vol. 17 (2000), pp. 215–226
6. D. Meredith, *Computational music analysis*, vol. 62 (Springer, Aalborg, 2016), pp.57–80
7. C. Schmidt-Jones, *The basic elements of music* (Texas, Connexions, 2012), pp.10–57
8. A. Blatter, *Revisiting music theory: A guide to the practice* (Routledge, New York, 2012), pp.10–40
9. Z.J. Kan, A. Sourin, in *2020 International Conference on Cyberworlds (CW)*. Generation of irregular music patterns with deep learning (IEEE, 2020), pp. 188–195
10. J. Huang, D. Gamble, K. Sarnlertsophon, X. Wang, S. Hsiao, Feeling music: Integration of auditory and tactile inputs in musical meter perception. PLoS ONE **7**(10), e48496 (2012)
11. M. Gainza, E. Coyle, Time signature detection by using a multi resolution audio similarity matrix. Audio Eng. Soc. (2007), pp. 1-8
12. A. McLeod, M. Steedman, in *Proceedings of the 14th Sound and Music Computing Conference*. Meter detection in symbolic music using a lexicalized PCFG (2017), pp. 373–379
13. J. Abimbola, D. Kostrzewa, P. Kasprowski, Time signature detection: A survey. Sensors **21**(19), 6494 (2021)
14. K. He, X. Zhang, S. Ren, J. Sun, in *Proceedings of the IEEE conference on computer vision and pattern recognition*. Deep residual learning for image recognition (2016), pp. 770–778
15. J. Deng, W. Dong, R. Socher, L.J. Li, K. Li, L. Fei-Fei, in *2009 IEEE conference on computer vision and pattern recognition*. Imagenet: A large-scale hierarchical image database (IEEE, 2009), pp. 248–255
16. H.H. Tan, K.H. Lim, in *2019 7th international conference on smart computing & communications (ICSCC)*. Vanishing gradient mitigation with deep learning neural network optimization (IEEE, 2019), pp. 1–4
17. M. Gainza, in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. Automatic musical meter detection (IEEE, 2009), pp. 329–332
18. E. Coyle, M. Gainza, in *Audio Engineering Society Convention 122*. Time signature detection by using a multi-resolution audio similarity matrix (Audio Engineering Society, 2007)
19. S. Gulati, V. Rao, P. Rao, in *Speech, Sound and Music Processing: Embracing Research in India*. Meter detection from audio for Indian music (Springer, India, 2011), pp. 34–43
20. A. Srinivasamurthy, G. Tronel, S. Subramanian, P. Chordia, in *CompMusic Workshop*. A beat tracking approach to complete description of rhythm in Indian classical music (Georgia Tech Center for Music Technology, Atlanta, 2012), pp. 72–78
21. W.B. De Haas, A. Volk, in *International Society for Music Information Retrieval Conference*. Meter detection in symbolic music using inner metric analysis (2016), p. 441

22. D. Eck, in *10th Rhythm Perception and Production Workshop (RPPW'05)*. Meter and autocorrelation (Citeseer, 2005)
23. T. Eerola, P. Toiviainen, Midi toolbox: Matlab tools for music research, pp. 29–31 (2004)
24. A. Volk, W.B. de Haas, in *Proceedings of the International Society for Music Information Retrieval Conference*. A corpus-based study on ragtime syncopation (2013)
25. D. Eck, J. Lapalme, Learning musical structure directly from sequences of music. Univ. Montr. Dep. Comput. Sci. CP **6128**, 48 (2008)
26. N. Bouguila, W. Fan, M. Amayri, *Hidden Markov models and applications* (Springer, Canada, 2022), pp.103–55
27. M. Varewyck, J.P. Martens, M. Leman, Musical meter classification with beat synchronous acoustic features, DFT-based metrical features and support vector machines. J. New Music Res. **42**(3), 267–282 (2013)
28. F. Gouyon, P. Herrera, in *Audio Engineering Society Convention 114*. Determination of the meter of musical audio signals: Seeking recurrences in beat segment descriptors (Audio Engineering Society, 2003)
29. J. Abimbola, D. Kostrzewa, P. Kasprowski, Meter2800: A novel dataset for music time signature detection. Data Brief 109736 (2023), 51, pp. 1-6
30. B.L. Sturm, The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use. arXiv preprint arXiv:1306.1461, pp. 7–10 (2013)
31. M. Defferrard, K. Benzi, P. Vandergheynst, X. Bresson, FMA: A dataset for music analysis. (2016). arXiv preprint arXiv:1612.01840. https://arxiv.org/abs/1612.01840. Accessed March 2022
32. E. Law, K. West, M.I. Mandel, M. Bay, J.S. Downie, in *ISMIR*. Evaluation of algorithms using games: The case of music tagging. (2009), pp. 387–392. https://mirg.city.ac.uk/codeapps/the-magnatagatune-dataset. Accessed March 2022
33. S. Borthwick, R. Moy, *Popular music genres: An introduction* (Routledge, New York, 2020), pp.5–18
34. S. Gupta, J. Jaafar, W.W. Ahmad, A. Bansal, Feature extraction using MFCC. Signal Image Process. Int. J. **4**(4), 101–108 (2013)
35. A. Revathi, C. Ravichandran, P. Saisiddarth, G.R. Prasad, Isolated command recognition using MFCC and clustering algorithm. SN Comput. Sci. **1**, 1–7 (2020)
36. M.A. Hossan, S. Memon, M.A. Gregory, in *2010 4th International Conference on Signal Processing and Communication Systems*. A novel approach for mfcc feature extraction (IEEE, 2010), pp. 1–5
37. B. Kollmeier, T. Brand, B. Meyer, Perception of speech and sound. Springer Handb. Speech Process. **13**, 61–82 (2008)
38. B. McFee, C. Raffel, D. Liang, D.P. Ellis, M. McVicar, E. Battenberg, O. Nieto, in *Proceedings of the 14th python in science conference*. librosa: Audio and music signal analysis in python, vol. 8 (Citeseer, 2015), pp. 18–25
39. J. Wu, Introduction to convolutional neural networks. Natl Key Lab Nov. Softw. Technol. Nanjing Univ. China **5**(23), 495 (2017)