Alexandria Engineering Journal (2022) 61, 10183–10196

HOSTED BY

ELSEVIER

Alexandria University

Alexandria Engineering Journal

www.elsevier.com/locate/aej
www.sciencedirect.com

# A hybrid deep learning approach for musical difficulty estimation of piano symbolic music

Youssef Ghatas *, Magda Fayek, Mayada Hadhoud

*Computer Engineering Department, Cairo University, Egypt*

Received 16 October 2021; revised 4 March 2022; accepted 24 March 2022
Available online 1 April 2022

**KEYWORDS**

Machine learning;
Deep learning;
Classification;
Neural networks;
Computer music

**Abstract** Musical difficulty estimation is an essential part of musical learning. Without a precise estimate, a music learner cannot choose a piece to play according to their current level. This problem is highly complicated for its subjectivity and data scarcity. In this study, we investigate deep learning approaches to solve this problem. Our pipeline can be summarized as follows: firstly, we convert the symbolic music MIDI file of a piano performance to piano roll representation. Secondly, the piano roll is divided into smaller parts. Finally, a model is trained on parts accompanied by the corresponding difficulty labels. We test our models on both complete and partial track difficulty classification problems. Multiple deep convolutional neural networks are proposed and evaluated. Accompanied with handcrafted features, the proposed hybrid deep model yields a relative F1 score improvement of more than 10% compared to previous studies, achieving a state-of-the-art F1 score of 76.26%. Besides the direct application of the work to classify musical pieces, the promising results can be a starting point for more complicated applications, including automated difficulty-controlled music generation.

## 1. Introduction

Music learning is entertaining and beneficial. Researches showed that music education aids in the educational process [1] as it enhances linguistics [2], cognitive abilities [3,4], and memory [5,6]. Additionally, it has excellent life-long effects [7]. It also helps psychological development [8,9].

Besides theoretical aspects of music, practice is critical throughout the learning process. The learner should be trained on musical pieces increasing in difficulty. Choosing pieces more advanced than a student's current level can be disappointing. On the other hand, choosing less difficult pieces all the time can be non-beneficial. That is why it is critical to select a piece that fits each student.

Nowadays, e-learning is replacing traditional education. Music isn't an exception [10,11]. Even though the Internet contains many musical data, much fewer pieces have difficulty categorization, making them useless for learning. Assessing the difficulty of a musical piece is still a difficult task as it needs

* Corresponding author.
E-mail address: youssef.ghatas@eng.cu.edu.eg (Y. Ghatas).

Peer review under responsibility of Faculty of Engineering, Alexandria University.

https://doi.org/10.1016/j.aej.2022.03.060
1110-0168 © 2022 THE AUTHORS. Published by Elsevier BV on behalf of Faculty of Engineering, Alexandria University

to be done manually for each piece, which is time-consuming, error-prone, expensive, and subjective. According to the Associated Board of the Royal Schools of Music(ABRSM) report [12], the piano is the most commonly used instrument in the UK, and there is an increasing interest in learning piano and other keyboard instruments.

Musical Instrument Digital Interface (MIDI)[1] is the most commonly used format for symbolic music. It was initially designed to allow communication between a computer and electrical music devices. MIDI file mainly contains messages (instructions) required to produce music through electrical devices. Most musical notation systems include mapping to MIDI files, for example, Guido [13], LilyPond[14], and MusicXML[15].

Deep neural networks(DNNs) have been extensively used for computer vision applications, including image classification[16,17]. DNNs remove the need for handcrafted features. Audio deep learning has gained increasing attention during the last decade. DNNs achieved great results in various audio tasks, including genre classification [18,19], music style transfer [20,21], and music generation [22,23].

This study introduces the first complete deep learning pipeline to estimate piano musical pieces' difficulty. We examine multiple models to understand the effect of model size and complexity on our model given the small data available. Furthermore, the pipeline can be used for partial-piece classification, as the difficulty level might be different for each part of the music. Compared to previous studies, the approach achieves superior results.

The main contributions of this research can be summarized as follows:

- Introducing a novel deep learning approach to difficulty estimation of piano music pieces with better performance than handcrafted features.
- Combining the strengths points of both handcrafted features and deep learning, resulting in more than 10% relative improvement compared to single approaches.
- Evaluating the proposed pipeline on two different experiments: track parts classification and complete track classification.
- Thorough analysis and discussions for current challenges and future work.

This paper is structured as follows: Section 2 discusses the problem and essential musical concepts. Related work is illustrated in Section 3. The proposed pipeline is introduced in Section 4. Section 5 shows experiments and comparisons, considering different configurations and architectures. Section 6 provides a thorough analysis of results and discusses potentially future directions. Finally, Section 7 concludes the research.

## 2. Background

### 2.1. Musical concepts

A pitch is a musical term that corresponds to a specific frequency. Every 12 pitches are grouped in one octave. The piano can play each of the 12 pitches; seven are played using white keys, and the other five using the black keys, as shown in Fig. 1. A musical note is a symbol that indicates both the pitch and duration. Each musical piece is divided into notes to show how the musician should play it.

### 2.2. Data & representation

#### 2.2.1. MIDI files

MIDI protocol was invented to record and playback musical pieces. The main difference between MIDI format and audio formats is that the MIDI file does not contain the audio itself. However, it contains the instructions to re-play it through a music synthesizer. Music Synthesizer can be either software or physical instrument with MIDI interfacing. Messages are used to define how the playback should be performed. The most important messages are note-on and note-off, indicating when a note starts and finishes, respectively [24].

For any keyboard musical instrument, including piano, each note-on message contains a key number and velocity. The key number corresponds to pitch, indicating note frequency. Key numbers range from 0 to 127. Velocity indicates how hard the key was pressed. To stop the key press, a note-off message should be sent. MIDI files can contain other metadata that can be used to describe annotation and further modify the playback.

#### 2.2.2. Piano roll

Although the MIDI file format has become the standard instrumental format for musical playback, it is not directly applicable to matrix-based machine learning algorithms. As an alternative, piano roll representation is widely used in research [21,22,25,26]. It is a matrix-based representation for musical pieces. An example is shown in Fig. 2[2]. The x-axis represents time steps, and the y-axis represents pitches where c1 to c8 are octaves from 1 to 8, respectively. Piano roll representation includes onsets, offsets, and velocity for notes. There are multiple variations for piano roll [27], including Binary and onsets only representations.

### 2.3. Classification background

The classification problem aims to predict a label for a given input data. Classifiers can be mainly divided into two families:

#### 2.3.1. Feature-based classifiers

The feature-based classification mainly depends on handcrafted features selected by an expert in the field. After that, these features are fed into a classifier that learns to map the features to the corresponding labels. Feature-based classifiers mainly suffer from the needed expert knowledge to select the relevant features.

*Support Vector Machine (SVM).* The support vector machine [28] tries to find the separating hyper-plane that maximizes the margin between different classes in the feature space, where each feature is considered one dimension. A kernel trick is used to solve non-separable data by mapping the input features into higher dimensional space.

[1] https://www.midi.org/.

[2] Source: https://www.mutopiaproject.org.

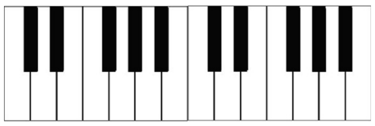

**Fig. 1** A Part of a piano keyboard - 2 octaves.

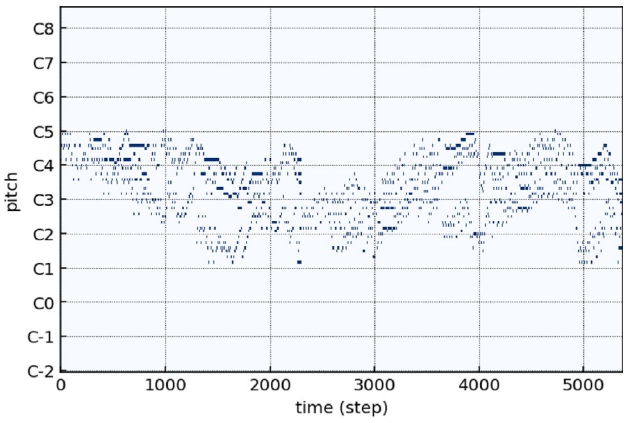

**Fig. 2** Piano roll example - French Suite No. 5 in G major.

*K-Nearset Neighbours (KNN)*. Unlike SVM, KNN doesn't need a training process. Instead, it uses distance in feature space. To assign a label for an input, KNN calculates the distance between the input and training samples. The algorithm finds the K nearest samples and determines the class label using the majority vote.

### 2.3.2. Deep neural networks

Unlike the feature-based approach, neural networks can work on the inputs directly. A neural network is simply a set of inputs passed through several layers to produce a set of outputs. For a classification problem, the output can be a class probability. The training objective is to minimize the loss. For a classification problem, the loss function mainly represents how far the network's outputs are from the correct labels. Loss is propagated to the layers using back-propagation, and so, as the training process progresses, the neural network should produce better outputs. A simple neural network is shown in Fig. 3.

*Fully Connected Layer*. The fully connected layer calculates each output using all values from the previous layer (or from the input if used as the first layer) using the following formula:

$$f(x) = \sum_{i=1}^{n}(W_i X_i + b_i) \tag{1}$$

Where $W$ and $b$ are the neuron weights and biases, respectively. Both $W$ and $b$ should be learned through the learning process.

*Activation Layers* Activation layers are used to add non-linearity to the model. Rectified Linear Units(ReLU) [29] tends to completely eliminate negative values. It uses the following formula:

$$f(x) = max(0, x) \tag{2}$$

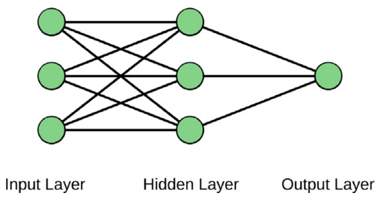

**Fig. 3** Basic neural network.

ReLU suffers from the dying problem, where gradients flowing through ReLU can become zero irreversibly. To tackle this problem [30], LeakyReLU uses the formula in Eq. 3.

$$f(x) = max(\alpha x, x) \tag{3}$$

Where $\alpha$ is a small positive value (commonly 0.01).

*Batch Normalization*. Batch normalization[31] was proposed to stabilize and accelerate the training process. It normalizes the input values to have a mean and variance close to 0 and 1, respectively, as illustrated in Eq. (4), which shows the calculation for one channel.

$$\mu_{\mathscr{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{4a}$$

$$\sigma^2_{\mathscr{B}} = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathscr{B}})^2 \tag{4b}$$

$$\hat{x}_i = \frac{x_i - \mu_{\mathscr{B}}}{\sqrt{\sigma^2_{\mathscr{B}} + \epsilon}} \tag{4c}$$

$$y_i = \gamma \hat{x}_i + \beta \tag{4d}$$

where $x_i, m$, and $B$ denote an input element to the layer, the mini-batch size, and the mini-batch data, respectively. $\beta$ and $\gamma$ parameters are learnable parameters.

*Pooling*. Pooling layers reduce the spatial size of the representation to reduce the number of parameters, reducing the size and computations of the neural network. Max pooling and average pooling are commonly used pooling layers to get the max and average value of the corresponding values accordingly.

*Regularization*. Regularization is used to reduce model overfitting by preventing the neural network from memorizing the exact input or part of it. $L1$ and $L2$ regularization penalize each network weight according to Eqs. 5 and 6, where $\lambda$ and $w$ denote the regularization parameter and the weight, respectively. Dropout [32] prevents overfitting by randomly dropping units from the neural network during training.

$$L1 = \lambda * |w| \tag{5}$$

$$L2 = \lambda * w^2 \tag{6}$$

## 3. Related work

Artificial intelligence has been used in various music tasks [33] ranging from analysis to modification and generation. This section will study common music representation, general approaches to music classification, and dive into symbolic music classification approaches in general. Finally, we will study previous piano difficulty problems in detail.

### 3.1. Music representation

Music representation is mainly categorized into audio-based and symbolic-based representations [27]. Audio representations are very general representations that capture audio itself to handle multiple types of audio recordings, including music, speech, and even animal sounds. Audio recordings are widely used in numerous applications, including speech recognition [34], speaker identification [35], music transcribing [36], speech emotion recognition [37], and animal sound classification [38].

Symbolic-based representations are more specific representations than audio representations. They can be considered as logical musical representations [33]. In the music domain, symbolic representations store higher-level musical data. For example, event-based musical representations, including MIDI[3], aim to provide ways to playback music. Inspired by musical sheets, musical notation systems are invented to store musical data with meta-data on how to playback to be both human-readable and machine-playable. Common notation systems include: MusicXML[15], Guido[13], and LilyPond[14].

### 3.2. Music general approaches

#### 3.2.1. Feature based approaches

In these approaches, a set of handcrafted features is defined and used to represent each track. Features defer depending on the input representation. For symbolic representation, high-level features are commonly used. Features can be divided into Pitch-related features, Rhythmic-related features, Harmonic-related features, Timbre-related features, and combined features [39]. Pitch-related features depend on the frequencies of the used musical notes. Rhythm-related features depend on musical patterns and repetitions. Harmonic-related features mainly depend on chords progression. Instruments define timbre-related features. Finally, combined features use multiple features concurrently.

After extracting multiple features, the features are fed into a classifier or simple neural networks. The main drawback of these approaches is the need to manually select and implement specific features based on the task and the input. That is why this approach is losing attention in the recent publication in favor of deep learning approaches.

#### 3.2.2. Deep learning approaches

Instead of extracting features, the input track is converted to an intermediate format and then fed directly into a deep neural network. Piano roll is a widely used musical representation [40]. The deep neural network handles the complete pipeline starting from the input representation until the final result, including the feature extracting step; however, the deep neural network features usually lack direct human explanation and are customized to the problem at hand. While deep learning dominates the visual domain for images and video classification, it is comparably less used in symbolic music classification problems, and traditional methods are still used in recent papers [41].

[3] MIDI can handle other metadata as shown in background section.

### 3.3. Symbolic music classification

In this subsection, we present several symbolic music classification studies, though different applications, they are relevant to our problem as they are classification problems and deal with symbolic music data. Şimşekli [42] used melodic interval histogram features of bass lines for genre classification. Melodic interval histograms are less sensitive to moving the entire music piece to higher or lower pitch values. The bass lines were manually extracted from the MIDI files. The author used KNN as a classifier and tested multiple distance metrics. The author used a 3-root 9-leaf dataset that consisted of 3 major genres Jazz, Rhythm & Blues, and Rock. Each of them is further divided into three subgenres. For comparison, multiple SVM classifiers with different kernel types were implemented. All models and distance metrics were tested on a dataset of 255 samples divided into 80% and 20% for training and testing. KNN has reached top accuracy with root and leaf accuracies of 100% and 86.67%, respectively.

Zhao et al. [43] used recurrent neural networks (RNN) for music emotion classification. They gathered a dataset of 196 samples. Each sample is given an emotional classification: aggressive, bittersweet, happy, humorous, or passionate. A melody trend feature matrix is extracted from each sample using rhythm and pitch. A forward neural network, a recurrent neural network (RNN), and an improved recurrent neural network with dropout layers were tested. The RNN model showed promising results with an accuracy of 64.1%. The improved RNN model with dropout layers further improved accuracy to reach 75.4%.

Chen et al. [41] implemented a two-step approach for music genre classification. Multiple features are extracted from each MIDI file: 80 pitch features, 12 musical interval features, seven chord features, three rhythm features, and two pitch entropy features. The authors use the intersection area under probability density function (PDF) curves to get each feature's significance. After that, the analytic hierarchy process (AHP) is used to rank the features. Finally, they use two-step classification models to classify each track into its main genre and subgenre. In total, they used 141 samples. They reached an accuracy of 87% for main genre classification. In their study, they show that features ranking improves accuracy. In addition, two-step classification showed much better results than one-step classification.

Kong et al. [44] targeted the problem of composer classification using a deep learning approach. Firstly, the input MIDI file is converted to three piano roll representations: the frame-roll, the onset roll, and the velocity roll. A convolutional recurrent neural network (CRNN) is introduced to use the benefits of CNN and capture long-time dependencies using a bidirectional gated recurrent unit(biGRU). For comparison, they implemented the same network for audio inputs after extracting log Mel-spectrograms features. The system is used with a different number of channels representing the different piano-roll representations. Multiple experiments were conducted to evaluate their work's performance on different numbers of classes (10 and 100 composers). The proposed models show different results across different experiments, yet, the MIDI approaches showed promising results.

Dervakos et al. [45] proposed two CNN models for MIDI symbolic music Genre recognition. The authors use piano rolls as input representations for the proposed models. The first model is Multiple Sequence Resolution Network (MuSeReNet) which uses multiple resolutions for the same input by converting the input to a tree of different resolutions. Another simpler model (the sequence model) is evaluated. The proposed models achieved great success for genre classification on two datasets. They evaluated their approaches against different sequence lengths. While both models produce comparable results for short lengths, MuSeReNet produces better results for longer lengths. They evaluated the effect of using dense and shallow blocks as the building blocks for both CNN models, and the shallow building block proved better performance.

### 3.4. Piano difficulty

Automatic difficulty assessment was previously studied in a few studies. In 2012, Sébastien et al. [46] studied piano difficulty using handcrafted features to evaluate difficulty. For input, they used MusicXML scores [15]. Seven features were introduced, including playing speed, displacement, and polyphony. A dataset of 50 elements was gathered from online music notation communities. Each element was assigned a difficulty level. Principal Component Analysis (PCA) and pianist opinions were used for objective and subjective evaluation, respectively. Their approach resulted in 66% accuracy on a dataset of 50 elements gathered from online music notation communities.

In 2012, Chiu and Chen [47] studied the difficulty assessment of piano music. They introduced multiple features. They formulated the problem as regression instead of classification. The authors filtered the old version of midi (format 0), as they needed separate tracks for left and right hand. For evaluation, they used two datasets contains 159 and 184 elements from 8notes[4] and Piano Street[5], respectively. They reached $R^2$ statistics of around 40%. The top five features were pitch entropy, pitch range, average pitch value, hand displacement rate, and playing speed. Table 1 summarizes the differences between Sébastien et al. [46] and Chiu and Chen [47] for automatic piano difficulty assessment.

In 2015, Holder et al. [48] computationally evaluated the difficulty of clarinet music pieces. The authors decomposed the input music piece into essential musical elements, including notes, intervals, and dynamics. Before evaluation, experts manually assigned a difficulty to each element (or group of elements) to construct a dictionary named "complexity parameters." For evaluation, they used MusicXML format as input. Each piece was assigned a specific difficulty based on the average evaluation of its parts. To evaluate their procedure, the authors used 32 manually graded pieces (between 1 −10). The results show a correlation with expected outputs.

Assessing the difficulty of a musical piece is subjective. While there are some common factors, including playing speed, frequency of clicks, range of played notes, hand stretching, and note reading complexity. There are many more factors to consider. Additionally, the effect of each factor in evaluating the difficulty is also subjective. That is why this problem is considered a high-level, complex problem. Moreover, that

[4] https://www.8notes.com/.
[5] https://www.pianostreet.com/.

**Table 1** Comparison between previous studies for piano difficulty estimation.

| | Sébastien et al. [46] | Chiu and Chen [47] |
|---|---|---|
| Input File | MusicXML | Midi file (Except format 0) |
| Dataset(s) | Online Music Notation Communities (50 files) | 8Notes (159 files) & Piano Street (184 files) |
| Problem category | Classification | Regression |
| Evaluation[a] | Accuracy (66%) | $R^2$ statistics (40%) |

[a] The evaluation values cannot be directly compared. As they are calculated using different metrics on different datasets. They are shown here for illustration only.

may be why it is less studied than other problems, including genre classification.

Previous studies suffer from the following:

1. They target each track as one unit. This limits each approach's application as it can not evaluate the difficulty on different parts of the track.
2. Each paper uses a different approach and evaluation metric to the problem, so no clear comparison is made between the two approaches.
3. They use handcrafted features, which limits the methodology to a specific instrument. In contrast, deep learning approaches can be applied to different instruments.
4. Small testing sets were used, which may limit the accuracy of the results.
5. Some features need particular data that does not exist in all musical files. For example, the fingering feature in Sébastien et al. [46] needs precise fingering annotation data that contains the best mapping between piano keys and the player's fingers. This is not always available and limits the usage of such a feature.
6. The features proposed in these studies cannot be used in end-to-end deep learning models such as generative models.

## 4. Proposed approach

The proposed deep learning pipeline is shown in Fig. 4. To the best of our knowledge, that is the first deep learning approach targeting piano difficulty assessment problem. Firstly, we convert the midi file to binary piano roll representation as explained in Section 2.2.2 using Pypianoroll [49]. Secondly, we take 72 pitches (6 octaves) starting from the 24 midi pitch. According to our preliminary experiments, taking 84 pitches (7 octaves) does not enhance classification performance. After that, the track is divided into 256-timesteps parts. Each part is labeled the same as its corresponding track.

The proposed model is described in Table 2 and Fig. 5. CNN architecture is chosen for its ability to tackle similar problems [44,45]. In addition, it can be a starting point for a generative model [22] as planned for future work (check Section 6.3). LeakyReLU [29] is used to add non-linearity to the model. We chose them instead of standard ReLU to prevent vanishing gradient problems. Batch normalization [31] is used to optimize the training process [50]. Multiple dropout [32] and average pooling [51] layers are used to prevent overfitting.

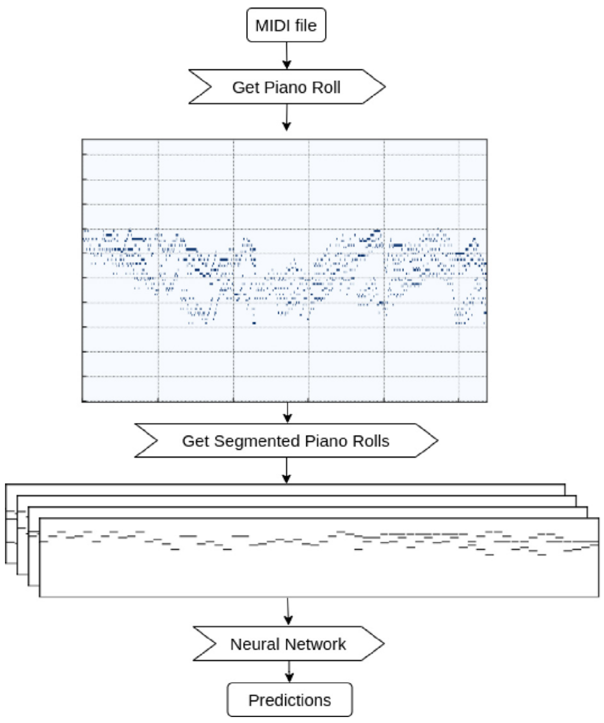

**Fig. 4** Deep learning pipeline.

For comparison, we have implemented two other models as summarized in Table 3. The small model (Model 2) is designed to reduce memory requirements (less than 2% of the proposed model parameters) and training time (reduced by around 70% of the proposed model training time). The first layer is supposed to capture the pitch-range-related features, and the higher layers are supposed to capture the time-related features. Each layer contains only ten filters. Average pooling layers are used to reduce the number of parameters further. For the full details of the small model, check Table 4 and Fig. 6.

Additionally, we have slightly modified the discriminator of MuseGAN[22] (we call it MuseCNN) to compare it with the proposed model. The original discriminator of MuseGAN was proposed to handle tracks of five musical instruments. We modified it to handle one track of 72 pitches and 256 time-steps and output three values to represent the three difficulty levels instead of the one value of the original discriminator. The modified architecture is described in Table 5. Furthermore, we have implemented the two architectures proposed in Dervakos et al. [45], namely the sequence model and the Multiple Sequence Resolution Network (MuSeReNet).

We have also implemented the top-5 features from Chiu and Chen [47]: pitch entropy, pitch range, average pitch value, hand displacement rate, and playing speed. All of them are implemented using music21 [52]. To visualize the data using the top-5 features from Chiu and Chen [47], we use t-SNE [53]; check Fig. 7 for 2D embedding visualization. The horizontal and vertical axes represent the first and the second embedding dimensions.

To compare the results with the state-of-the-art approaches, we have used two classification methods: (1) Support Vector Machine(SVM) [54] with the same parameters used in [47] except that we use the SVM for classification instead of regression and (2) K Nearest Neighbours (KNN) with k = 5. Check Fig. 8 for the pipeline.

Finally, we experiment with a fused approach that uses both extracted features and piano rolls. As shown in Fig. 9, the neural network takes the piano roll and the extracted features vector as an input and uses them to get the final output. We slightly modify the original architecture by concatenating the 5-vector handcrafted features and an output of an intermediate fully connected layer (20-length vector).

## 5. Experiments

Used software packages are summarized in Table 6.

### 5.1. Experimental setup

To the best of our knowledge, no public dataset is available for our problem. To gather data, we used 8notes.com. It is a website that contains musical pieces with subjective difficulty evaluation. Each piece is labeled as beginner, easy, intermediate, or advanced. For our experiments, beginner and easy tracks were considered the same class(easy). We have gathered around 1900 midi files divided as 30%, 60%, and 10% for easy, intermediate, and advanced difficulties, respectively.

For the training process, Adam optimizer [63] is used with the following parameters: $\beta_1 = 0.9, \beta_2 = 0.999$. We use a learning rate of 0.001 and a batch size of 128. To tackle overfitting problems in the deep learning approach, Both L1 [64] and L2 [65] regularization are used with 0.001 and 0.00001 penalties, respectively. For the fused approach, only L2 is used as the model showed more robustness against overfitting.

Dataset is split into training, validation, and test sets using stratified sampling [66] with a ratio of 8:1:1. To handle musical

**Table 2** Model 1 - Proposed model.

| Layer | Output Channels | Kernel Size | Stride | Activation | BN. | Pooling | Dropout | Output Size |
|---|---|---|---|---|---|---|---|---|
| Conv 1 | 32 | 32x16 | 1 | L. ReLU | - | - | ✓ | 225x57 |
| Conv 2 | 64 | 32x16 | 2 | L. ReLU | - | - | ✓ | 97x21 |
| Conv 3 | 64 | 32x16 | 1 | L. ReLU | ✓ | - | ✓ | 66x6 |
| Conv 4 | 32 | 3x3 | 1 | L. ReLU | ✓ | - | ✓ | 64x4 |
| Conv 5 | 16 | 3x3 | 1 | L. ReLU | ✓ | - | ✓ | 62x2 |
| Dense 1 | - | - | - | L. ReLU | - | - | ✓ | 20 |
| Dense 2 | - | - | - | - | - | - | - | 3 |
| Softmax | | | | 3 | | | | |

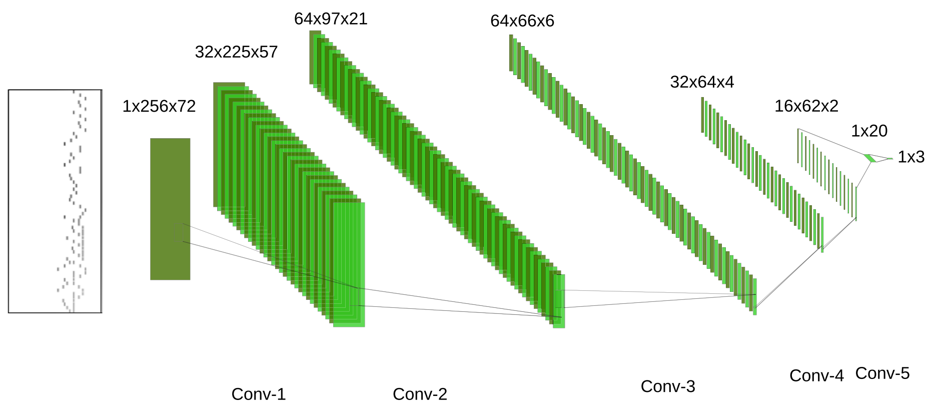

**Fig. 5** Proposed model architecture.

**Table 3** Models summary.

| Model Name | Description | # of parameters |
|---|---|---|
| Model 1 | Proposed Model - Large Model | 3.192 M |
| Model 2 | Small model - First layers are 1d Filters in pitch direction | 44 K |
| MuseCNN | Slightly modified version from MuseGAN Discriminator, as the original version was originally designed for five instrument tracks, but in our problem, we only deal with one instrument which is the piano | 155 K |
| Sequence Model | CNN Sequence Model[45] | 890 K |
| MuSeReNet Model | Multiple Sequence Resolution Network Model[45] | 1.1 M |

pieces' repetitive nature, we split songs into sets before extracting piano rolls. During the training phase, we extract overlapping 256-timesteps piano rolls with 128-timesteps increments from each track. We use categorical cross-entropy as our loss function.

For evaluation, we use accuracy, precision, recall, and F1 as given in Eqs. (7)–(10), respectively. Precision, recall, and F1 are calculated for each class and then averaged (macro averaging) to weigh every class equally despite the imbalanced dataset. For each classification class c, the prediction can be considered true positive (TP), true negative (TN), false positive (FP), or false negative (FN). True positive means a correct prediction. False positive means that an element that does not belong to c is falsely classified as c. False negative means that an element that belongs to c is falsely classified as not c. Finally, true negative means that an element that does not belong to c is correctly classified as not c.

$$Accuracy = \frac{Correct\ Predicions}{Total\ Number\ of\ Samples} \tag{7}$$

$$Precision = \frac{T_p}{T_p + F_p} \tag{8}$$

$$Recall = \frac{T_p}{T_p + F_n} \tag{9}$$

$$F1 = \frac{Precision * Recall}{Precision + Recall} \tag{10}$$

Small and large models are trained on batches of 128 elements for 30 and 65 epochs, respectively. At the end of each epoch, metrics are calculated for the validation set. The model with the best F1 is considered the best model. To get the prediction of the complete track, we calculate the average of parts' predictions.

**Table 4** Model 2 - Small model.

| Layer | Output Channels | Kernel Size | Stride | Activation | BN. | Pooling | Dropout | Output Size |
|---|---|---|---|---|---|---|---|---|
| Conv 1 | 10 | 1x10 | 1 | L. ReLU | ✓ | - | - | 256x63 |
| Conv 2 | 10 | 1x10 | 1 | L. ReLU | ✓ | - | - | 256x54 |
| Conv 3 | 10 | 10x10 | 1 | L. ReLU | ✓ | Avg. (2x2) | - | 123x22 |
| Conv 4 | 10 | 10x10 | 1 | L. ReLU | ✓ | Avg. (2x2) | - | 57x6 |
| Conv 5 | 10 | 3x3 | 1 | L. ReLU | ✓ | - | - | 55x4 |
| Conv 6 | 10 | 3x3 | 1 | L. ReLU | ✓ | - | - | 53x2 |
| Dense 1 | - | - | - | L. ReLU | - | - | ✓ | 20 |
| Dense 2 | - | - | - | - | - | - | - | 3 |
| Softmax | | | | 3 | | | | |

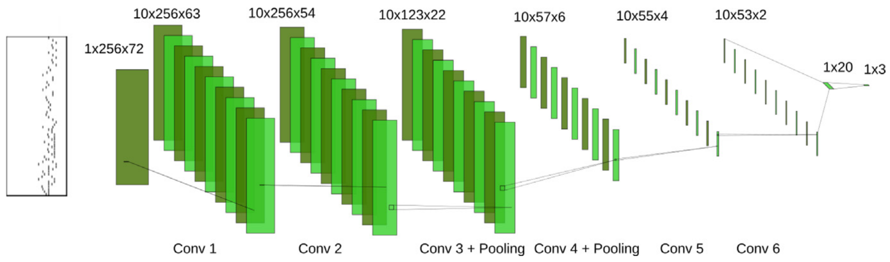

**Fig. 6** Small Model Architecture.

**Table 5** MuseCNN - Slightly modified from MuseGAN discriminator [22].

| Layer | Output Channels | Kernel Size | Stride | Activation | BN. | Pooling | Dropout | Output Size |
|---|---|---|---|---|---|---|---|---|
| Conv 1 | 16 | 1x1x12 | 1x1x12 | L. ReLU | ✓ | - | - | 4x64x6 |
| Conv 2 | 16 | 1x4x1 | 1x4x1 | L. ReLU | ✓ | - | - | 4x16x6 |
| Conv 3 | 64 | 1x1x3 | 1x1x1 | L. ReLU | ✓ | - | - | 4x16x4 |
| Conv 4 | 64 | 1x1x4 | 1x1x4 | L. ReLU | ✓ | - | - | 4x16x1 |
| Conv 5 | 128 | 1x4x1 | 1x4x1 | L. ReLU | ✓ | - | - | 4x4x1 |
| Conv 6 | 128 | 2x1x1 | 2x1x1 | L. ReLU | ✓ | - | - | 2x4x1 |
| Conv 7 | 256 | 2x1x1 | 2x1x1 | L. ReLU | ✓ | - | - | 1x4x1 |
| Dense | - | - | - | - | - | - | - | 3 |
| Softmax | | | | 3 | | | | |

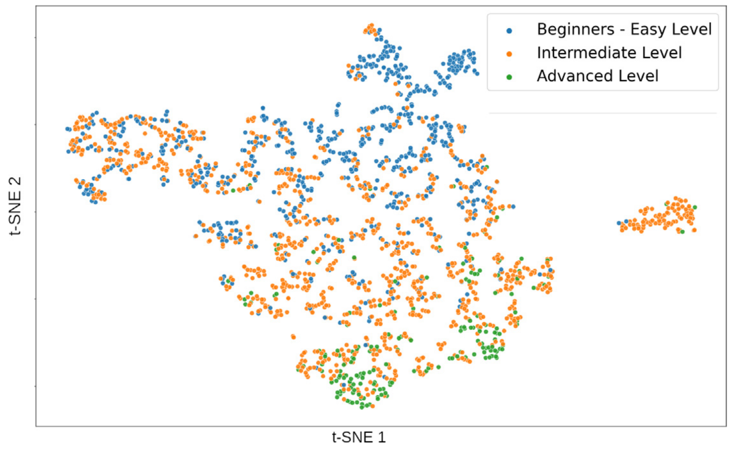

**Fig. 7** Data visualization using t-SNE.

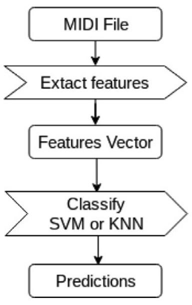

**Fig. 8** Handcrafted features pipeline.

## 6. Results & discussions

### *6.1. Results*

We compare the different approaches in two main experiments. In the first one, we evaluate the complete track classification. In the second experiment, we evaluate the classification for each part of the track. The second experiment does not apply to handcrafted global features. The results of both experiments are shown in Tables 7 and 8, respectively. These results can show the following:

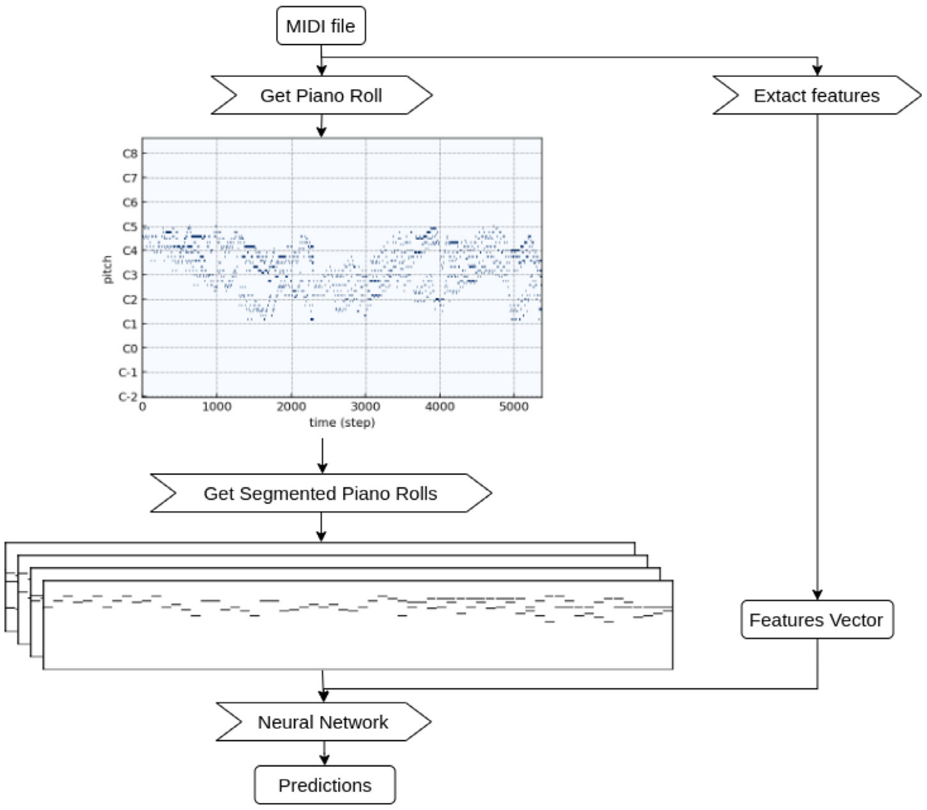

**Fig. 9** Fused approach.

**Table 6** Used software.

| Tool Name | Description |
|---|---|
| Numpy [55] | Matrices operations. |
| PyTorch [56] | GPU deep neural networks. |
| Jupyter notebook [57] | Visualization and implementation. |
| Python | Programming Language |
| Matplotlib [58] | Visualization and graphs |
| Seaborn [59] | Visualization and graphs |
| Pandas [60] | Data Analysis |
| Scikit Learn [61] | Machine Learning |
| tqdm [62] | Extensible progress meter |
| Music21 [52] | Computer Aided Musicology |
| Pypianoroll [49] | Working with pianorolls |

- End-to-end convolutional neural networks can achieve good results on the piano difficulty problem. Results show that deep neural networks are comparable to the handcrafted features method, and some networks are even superior. This encourages the usage of deep learning models as a part of a generative model to generate music in a given difficulty. Furthermore, the same methodology can theoretically work for other instruments as well.
- The small model 2 achieves acceptable results with F1 (68.08%) comparable to Handcrafted features F1(68.7%) with smaller memory requirements (44 K parameters) compared to the proposed model (3.192 M parameters) and MuseCNN (155 K parameters).
- Handcrafted features achieve good results which match the previously reported results in [46,47]. Check t-SNE visualization in Fig. 7 and results in Table 7. This encouraged fusion between the deep learning approach and handcrafted features.
- Combining both global features extracted manually and deep learning on piano roll parts achieves the best F1 score with more than 10% improvement over using either of them separately.
- There is a correlation between models' efficiency for parts and complete tracks; however, the relation is not completely proportional. For example, comparing the results of Models 2 and MuseCNN in Fig. 10 shows that while MuseCNN produces a better accuracy for complete track classification, it shows a slightly worse accuracy than Model 2 for parts classification. This can be the result of the naive average method used to get tracks classification from its parts. Another possible reason can be the different number of parts in each track. Advanced tracks will usually be longer than easy tracks, which makes them contain more parts per track. Consequently, each part's effect in beginner tracks is much more important for each part in advanced tracks. This problem may be solved using an equal number of samples from each track to calculate loss and validation performance.
- While MuseCNN is a modified version from a different application (Multi-track discriminator), it showed acceptable results compared to the state-of-the-art. This can encourage future work to use other neural networks

**Table 7** Complete tracks classification results.

| Model Name | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| Model 1 | 74.57 | 70.63 | 68.14 | 69.22 |
| Model 1 - Fused | **78.03** | 76.32 | **76.52** | **76.26** |
| Model 2 | 71.1 | 69.7 | 66.94 | 68.08 |
| Model 2 - Fused | 77.46 | **80.27** | 70.28 | 73.59 |
| MuseCNN[22] | 72.83 | 69.96 | 64.33 | 66.58 |
| Sequence Model[45] | 69.14 | 65.79 | 62.44 | 63.91 |
| MuSeReNet Model[45] | 69.75 | 64.36 | 70.06 | 66.28 |
| Features Set - SVM | 72.12 | 73.34 | 67.32 | 68.7 |
| Features Set - KNN | 67.27 | 57.7 | 54.9 | 55.49 |

**Table 8** Parts classification results.

| Model Name | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| Model 1 | 63.26 | 58.83 | 56.39 | 56.98 |
| Model 1 - Fused | **74.52** | **72.95** | **67.27** | **69.56** |
| Model 2 | 62.29 | 57.71 | 56.06 | 56.34 |
| Model 2 - Fused | 66.96 | 63.8 | 58.56 | 60.48 |
| MuseCNN[22] | 62.18 | 57.34 | 56.07 | 55.91 |
| Sequence Model[45] | 61.21 | 57.26 | 54.74 | 54.81 |
| MuSeReNet Model[45] | 59.34 | 54.83 | 58.15 | 55.97 |

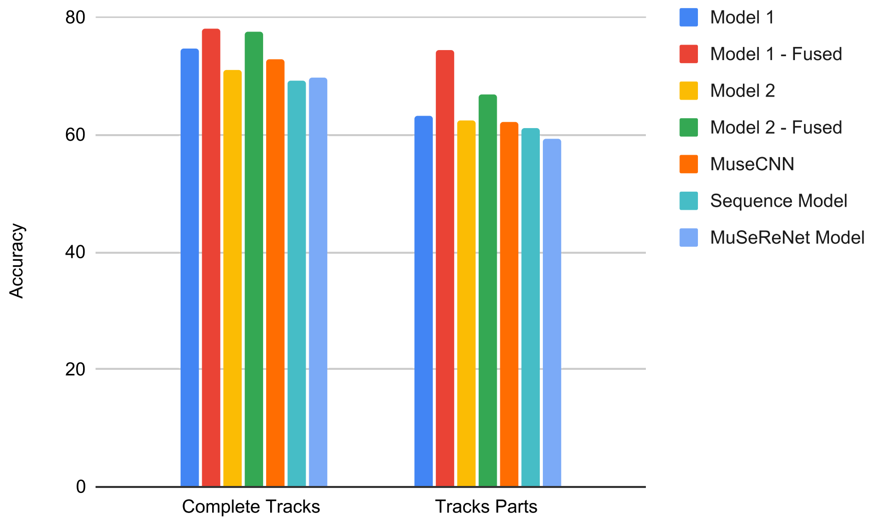

**Fig. 10** Accuracy comparison for complete tracks classification and parts classification.

designed for different symbolic music problems. On the other hand, customized approaches -like the one proposed in this research- can produce better results.

*6.2. Proposed model deep analysis*

In this section, we investigate the insights of the proposed model with fusion. Tables 9 and 10 illustrate confusion matrices for track parts and complete tracks, respectively. As expected, the proposed model has some difficulty separating intermediate from hard and easy classes; however, hard and easy separation is easier. This implies that the model could learn meaningful features for difficulty estimation.

Check Figs. 11 and 12 for correctly and incorrectly predicted examples, respectively. The confidence represents the probability produced by the model for the predicted class. The following can be concluded:

**Table 9** Parts confusion matrix.

| Class | Easy | Intermediate | Hard | Total | Recall |
|---|---|---|---|---|---|
| Easy | 279 | 229 | 1 | 509 | 54.81 |
| Intermediate | 108 | 1505 | 151 | 1764 | 85.32 |
| Hard | 17 | 252 | 433 | 702 | 61.68 |
| Total | 404 | 1986 | 585 | 2975 | 67.27 |
| Precision | 69.06 | 75.78 | 74.02 | 72.95 | |

**Table 10** Tracks confusion matrix.

| Class | Easy | Intermediate | Hard | Total | Recall |
|---|---|---|---|---|---|
| Easy | 38 | 19 | 0 | 57 | 66.67 |
| Intermediate | 12 | 86 | 4 | 102 | 84.31 |
| Hard | 0 | 3 | 11 | 14 | 78.57 |
| Total | 50 | 108 | 15 | 173 | 76.52 |
| Precision | 76.00 | 79.63 | 73.33 | 76.32 | |

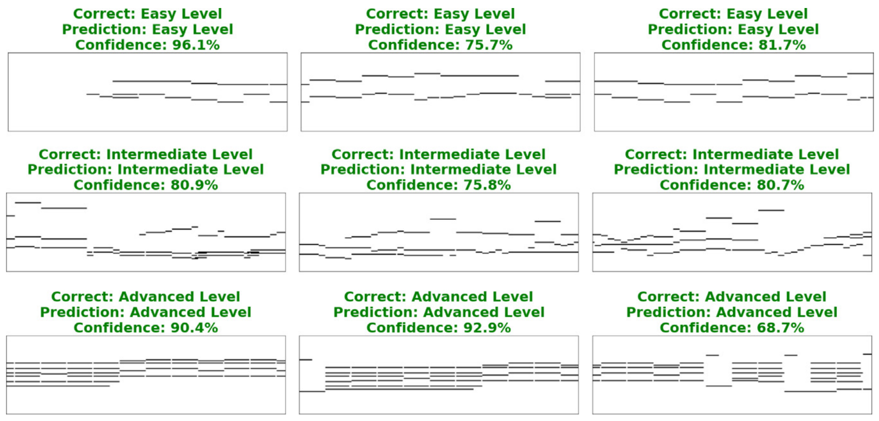

**Fig. 11** Correct Samples.

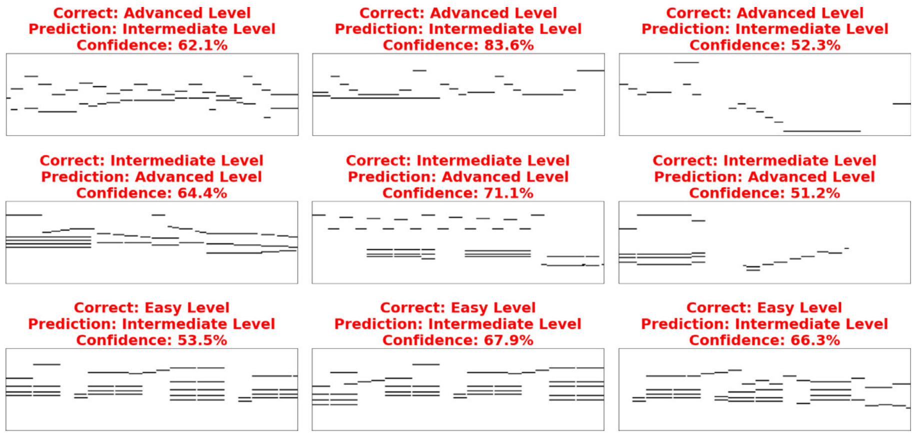

**Fig. 12** Incorrect Samples.

- In the correctly predicted samples, it can be seen that the prediction depends on the number of simultaneously pressed notes. In other words, as the number of concurrently played notes increases, the difficulty increases. However, the confidence is different based on their structure. This may be an indication that our model learns valuable information from both types of features. The second and third advanced samples are good examples for this observation.
- In the third advanced example, while the correct label is advanced, it seems easier than other advanced samples. It contains fewer concurrent notes than other advanced examples. However, the proposed model predicted it intermediate supposedly for its small duration pitches, which adds to its difficulty.
- The incorrectly labeled easy examples seem to trick the model because of the multiple concurrent notes.

### 6.3. Discussion

The main problem faced in our problem is related to data availability. Data suffers from the following:

1. Data is limited, compared to the available midi files available on the Internet. For a quick comparison, while we used less than 2000 tracks, The dataset introduced in Raffel [67] contains around 176 K tracks. In this research, we handled this problem using multiple tricks, including regularization penalty, dropouts, and batch normalization; however, the availability of more data can open doors for more advanced models.
2. Data is imbalanced; most of the gathered tracks are of intermediate difficulty.
3. Tracks are labeled as a unit, which seems a very optimistic assumption. While this might fit other problems, including style or genre classification where the style is consistent throughout the track, it does not seem to fit the problem. For example, advanced tracks might contain some intermediate parts or vice versa. This point is also noticed given the superiority of all models in the complete track classification compared to parts classification.

The problem is far from being solved. For future research, the authors suggest the following:

1. Gathering a new public dataset with as much finer granularity as possible. In other words, the dataset should contain labeling for each part, besides the global label. Additionally, if meaningful objective criteria can be used, it will reduce the subjectivity of the problem.
2. Defining augmentation techniques for this specific problem where the style and musicality are not the concern but the playing difficulty.
3. Testing other representations, including graph representation [68,69] and corresponding graph neural networks [70].
4. Implementing a model with larger time-steps to avoid the naive average combination for complete track classification. Hierarchical models might be a good alternative [40,71]. Also, the Music transformers concept [72] shows promising potential in similar problems.
5. Targeting related generation problems as the deep models proposed here can be a starting point for discriminators [73] or teacher models[74].

## 7. Conclusions

In this research, we investigated automatic piano difficulty estimation for symbolic music using deep neural networks. To the best of our knowledge, this is the first study regarding deep neural networks for piano difficulty estimation. Despite the complexities of the problem, especially data shortage and labeling subjectivity, the proposed model and its variation show promising results with a relative accuracy improvement of around 10% compared to traditional handcrafted features [46,47]. We have tested multiple models for both complete and partial track classification. The end-to-end deep learning approach produced an F1 score of 69.22% for the complete track classification task. Combining local features and handcrafted global features using a deep neural network, the proposed model reached an F1 score of 76.26% with around 10% relative F1 score improvement over single approaches for the complete track classification task.

In the future, we plan to improve the deep model architecture to handle longer musical pieces and combine track parts more efficiently. Additionally, we plan to target the small-size dataset by gathering a larger dataset with a more standard difficulty annotation technique. We consider this study an essential step towards automated instrumental learning.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

[1] M. Azizinezhad, M. Hashemi, S. Darvishi, Music as an education-related service to promote learning and skills acquisition, Procedia-Social Behav. Sci. 93 (2013) 142–145.
[2] T. Linnavalli, V. Putkinen, J. Lipsanen, M. Huotilainen, M. Tervaniemi, Music playschool enhances children's linguistic skills, Sci. Rep. 8 (1) (2018) 1–10.
[3] J.A. Bugos, D. DeMarie, The effects of a short-term music program on preschool children's executive functions, Psychol. Music 45 (6) (2017) 855–867.
[4] A.C. Jaschke, H. Honing, E.J. Scherder, Longitudinal analysis of music education on executive functions in primary school children, Front. Neurosci. 12 (2018) 103.
[5] F. Talamini, G. Altoè, B. Carretti, M. Grassi, Musicians have better memory than nonmusicians: A meta-analysis, PloS One 12 (10) (2017) e0186773.
[6] I. Roden, G. Kreutz, S. Bongard, Effects of a school-based instrumental music program on verbal and visual memory in primary school children: a longitudinal study, Front. Neurosci. 6 (2012) 572.
[7] J.D. Gómez-Zapata, L.C. Herrero-Prieto, B. Rodríguez-Prado, Does music soothe the soul? Evaluating the impact of a music education programme in Medellin, Colombia, J. Cult. Econ. (2020) 1–42.
[8] A.M. Croom, Music practice and participation for psychological well-being: A review of how music influences

positive emotion, engagement, relationships, meaning, and accomplishment, Musicae Scientiae 19 (1) (2015) 44–64.

[9] D. Shipman, A Prescription for Music Lessons, Federal Practitioner 33 (2) (2016) 9.

[10] J. Savage, Teaching music in England today, Int. J. Music Educ. (2020), 0255761420986213.

[11] K. Kelman, Current Approaches to Education-But What About the Music Industry?, in: Entrepreneurial Music Education, Springer, 2020, pp. 23–62.

[12] ABRSM: 4.2. Shifts in instrumental trends, https://gb.abrsm.org/en/making-music/4-the-statistics/42-shifts-in-instrumental-trends/, accessed: 2021-03-08, 2014.

[13] K. Renz, Algorithms and data structures for a music notation system based on guido music notation Ph.D. thesis, Technische Universität, 2002.

[14] H.-W. Nienhuys, J. Nieuwenhuizen, LilyPond, a system for automated music engraving, in: Proceedings of the XIV Colloquium on Musical Informatics (XIV CIM 2003), vol. 1, Citeseer, 2003, pp. 167–171.

[15] M. Good, MusicXML for notation and analysis, Virtual Score: Represent., Retrieval Restorat. 12 (113–124) (2001) 160.

[16] S.S. Yadav, S.M. Jadhav, Deep convolutional neural network based medical image classification for disease diagnosis, J. Big Data 6 (1) (2019) 1–18.

[17] N. Dematteis, D. Giordan, P. Allasia, Image Classification for Automated Image Cross-Correlation Applications in the Geosciences, Appl. Sci. 9 (11), ISSN 2076-3417, https://doi.org/10.3390/app9112357, https://www.mdpi.com/2076-3417/9/11/2357.

[18] S. Deepak, B. Prasad, Music Classification based on Genre using LSTM, in: 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), IEEE, 2020, pp. 985–991.

[19] H. Bahuleyan, Music Genre Classification using Machine Learning Techniques, arXiv e-prints, 2018, arXiv-1804.

[20] G. Brunner, A. Konrad, Y. Wang, R. Wattenhofer, MIDI-VAE: Modeling dynamics and instrumentation of music with applications to style transfer, arXiv preprint arXiv:1809.07600.

[21] G. Brunner, Y. Wang, R. Wattenhofer, S. Zhao, Symbolic music genre transfer with cyclegan, in: 2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, 2018b, pp. 786–793.

[22] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, Y.-H. Yang, Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, 2018a.

[23] L.-C. Yang, S.-Y. Chou, Y.-H. Yang, MidiNet: A convolutional generative adversarial network for symbolic-domain music generation, 2017, arXiv preprint arXiv:1703.10847.

[24] M.M. Association, et al., The complete MIDI 1.0 detailed specification, Los Angeles, CA, The MIDI Manufacturers Association, 1996.

[25] J. Wang, C. Jin, W. Zhao, S. Liu, X. Lv, An unsupervised methodology for musical style translation, in: 2019 15th International Conference on Computational Intelligence and Security (CIS), IEEE, 2019, pp. 216–220.

[26] H.H. Tan, ChordAL: A Chord-Based Approach for Music Generation using Bi-LSTMs., in: ICCC, 2019, pp. 364–365.

[27] S. Ji, J. Luo, X. Yang, A Comprehensive Survey on Deep Music Generation: Multi-level Representations, Algorithms, Evaluations, and Future Directions, 2020, arXiv preprint arXiv:2011.06801.

[28] C. Cortes, V. Vapnik, Support-vector networks, Machine Learn. 20 (3) (1995) 273–297.

[29] A.L. Maas, A.Y. Hannun, A.Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: Proc. icml, vol. 30, Citeseer, 3, 2013.

[30] B. Xu, N. Wang, T. Chen, M. Li, Empirical evaluation of rectified activations in convolutional network, 2015, arXiv preprint arXiv:1505.00853.

[31] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, PMLR, 2015, pp. 448–456.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Machine Learning Res. 15 (1) (2014) 1929–1958.

[33] E. Liebman, P. Stone, Artificial Musical Intelligence: A Survey, 2020, arXiv preprint arXiv:2006.10553.

[34] N. Moritz, T. Hori, J. Le, Streaming automatic speech recognition with the transformer model, in: ICASSP 2020–2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2020, pp. 6074–6078.

[35] S. Camacho, D. Renza, et al., A semi-supervised speaker identification method for audio forensics using cochleagrams, in: Workshop on Engineering Applications, Springer, 2017, pp. 55–64.

[36] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, D. Eck, Onsets and frames: Dual-objective piano transcription, 2017, arXiv preprint arXiv:1710.11153.

[37] A. Bhavan, P. Chauhan, R.R. Shah, et al, Bagged support vector machines for emotion recognition from speech, Knowl.-Based Syst. 184 (2019) 104886.

[38] K. Ko, S. Park, H. Ko, Convolutional feature vectors and support vector machine for animal sound classification, in: 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), IEEE, 2018, pp. 376–379.

[39] D.C. Corrêa, F.A. Rodrigues, A survey on symbolic data-based music genre classification, Expert Syst. Appl. 60 (2016) 190–210.

[40] J.-P. Briot, G. Hadjeres, F.-D. Pachet, Deep learning techniques for music generation–a survey, 2017, arXiv preprint arXiv:1709.01620.

[41] Y.-T. Chen, C.-H. Chen, S. Wu, C.-C. Lo, A two-step approach for classifying music genre on the strength of AHP weighted musical features, Mathematics 7 (1) (2019) 19.

[42] U. Şimşekli, Automatic music genre classification using bass lines, in: 2010 20th International Conference on Pattern Recognition, IEEE, 2010, pp. 4137–4140.

[43] W. Zhao, Y. Zhou, Y. Tie, Y. Zhao, Recurrent neural network for MIDI music emotion classification, in: 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), IEEE, 2018, pp. 2596–2600.

[44] Q. Kong, K. Choi, Y. Wang, Large-Scale MIDI-based Composer Classification, 2020, arXiv preprint arXiv:2010.14805.

[45] E. Dervakos, N. Kotsani, G. Stamou, Genre recognition from symbolic music with cnns, in: International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar), Springer, 2021, pp. 98–114.

[46] V. Sébastien, H. Ralambondrainy, O. Sébastien, N. Conruyt, Score analyzer: Automatically determining scores difficulty level for instrumental e-learning, in: 13th International Society for Music Information Retrieval Conference (ISMIR 2012), 2012, pp. 571–576.

[47] S.-C. Chiu, M.-S. Chen, A Study on Difficulty Level Recognition of Piano Sheet Music, 2012 IEEE International

Symposium on Multimedia, https://doi.org/10.1109/ism.2012.11, https://ieeexplore.ieee.org/document/6424624.

[48] E. Holder, E. Tilevich, A. Gillick, Musiplectics: computational assessment of the complexity of music scores, in: 2015 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward!), 2015, pp. 107–120.

[49] H.-W. Dong, W.-Y. Hsiao, Y.-H. Yang, Pypianoroll: Open source Python package for handling multitrack pianoroll, Proc. ISMIR. Late-breaking paper;[Online] https://github.com/salu133445/pypianoroll.

[50] S. Santurkar, D. Tsipras, A. Ilyas, A. Madry, How Does Batch Normalization Help Optimization?, Adv. Neural Informat. Process. Syst. (31) (2018).

[51] D. Scherer, A. Müller, S. Behnke, Evaluation of pooling operations in convolutional architectures for object recognition, in: International Conference on Artificial Neural Networks, Springer, 2010, pp. 92–101.

[52] M.S. Cuthbert, C. Ariza, music21: A toolkit for computer-aided musicology and symbolic music data, 2021.

[53] L. Van der Maaten, G. Hinton, Visualizing data using t-SNE, J. Machine Learn. Res. 9 (11) (2008).

[54] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, B. Scholkopf, Support vector machines, IEEE Intell. Syst. Their Appl. 13 (4) (1998) 18–28.

[55] S. Van Der Walt, S.C. Colbert, G. Varoquaux, The NumPy array: a structure for efficient numerical computation, Comput. Sci. Eng. 13 (2) (2011) 22–30.

[56] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, arXiv preprint arXiv:1912.01703.

[57] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, C. Willing, J. development team, Jupyter Notebooks - a publishing format for reproducible computational workflows, in: F. Loizides, B. Scmidt (Eds.), Positioning and Power in Academic Publishing: Players, Agents and Agendas, IOS Press, Netherlands, 2016, pp. 87–90.

[58] J.D. Hunter, Matplotlib: A 2D graphics environment, IEEE Ann. Hist. Comput. 9 (03) (2007) 90–95.

[59] M. Waskom, M. Gelbart, O. Botvinnik, J. Ostblom, P. Hobson, S. Lukauskas, et al., mwaskom/seaborn: v0. 11.1 (December 2020), Computer software]. Zenodo. doi 10.

[60] Wes McKinney, Data Structures for Statistical Computing in Python, in: Stéfan van der Walt, Jarrod Millman (Eds.), Proceedings of the 9th Python in Science Conference, 2010, pp. 56–61, https://doi.org/10.25080/Majora-92bf1922-00a.

[61] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine Learning in Python, J. Machine Learn. Res. 12 (2011) 2825–2830.

[62] C.O. da Costa-Luis, tqdm: A fast, extensible progress meter for python and cli, J. Open Source Softw. 4 (37) (2019) 1277.

[63] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[64] A.Y. Ng, Feature selection, L 1 vs. L 2 regularization, and rotational invariance, in: Proceedings of the Twenty-first International Conference on Machine Learning, vol. 78, 2004.

[65] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, 2017, arXiv preprint arXiv:1711.05101.

[66] R.J. May, H.R. Maier, G.C. Dandy, Data splitting for artificial neural networks using SOM-based stratified sampling, Neural Networks 23 (2) (2010) 283–294.

[67] C. Raffel, Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching Ph. D. thesis, Columbia University, 2016.

[68] B. Mokbel, A. Hasenfuss, B. Hammer, Graph-based representation of symbolic musical data, in: International Workshop on Graph-Based Representations in Pattern Recognition, Springer, 2009, pp. 42–51.

[69] D. Jeong, T. Kwon, Y. Kim, J. Nam, Graph neural network for music score data and modeling expressive piano performance, in: International Conference on Machine Learning, PMLR, 2019, pp. 3060–3070.

[70] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Networks 20 (1) (2008) 61–80.

[71] C.P. Tang, K.L. Chui, Y.K. Yu, Z. Zeng, K.H. Wong, Music genre classification using a hierarchical long short term memory (LSTM) model, in: Third International Workshop on Pattern Recognition, vol. 10828, International Society for Optics and Photonics, 108281B, 2018.

[72] C.-Z.A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A.M. Dai, M.D. Hoffman, M. Dinculescu, D. Eck, Music transformer, 2018, arXiv preprint arXiv:1809.04281.

[73] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, 2014, arXiv preprint arXiv:1406.2661.

[74] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, 2015, arXiv preprint arXiv:1503.02531.