EMITTER International Journal of Engineering Technology
Vol. 11, No. 2, December 2023, pp. 246~264
DOI : 10.24003/emitter.v11i2.827

IRAWNET: A Method for Transcribing Indonesian Classical Music Notes Directly from Multichannel Raw Audio

Dewi Nurdiyah[1,2], Eko Mulyanto Yuniarno[1], Yoyon Kusnendar Suprapto[1], Mauridhi Hery Purnomo[1]

[1]Department of Electrical Engineering, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
[2]Faculty of Information Technology and Communication, Universitas Semarang, Semarang, Indonesia
Corresponding Author: ekomulyanto@ee.its.ac.id

Received September 7, 2023; Revised October 27, 2023; Accepted November 29, 2023

Abstract

A challenging task when developing real-time Automatic Music Transcription (AMT) methods is directly leveraging inputs from multichannel raw audio without any handcrafted signal transformation and feature extraction steps. The crucial problems are that raw audio only contains an amplitude in each timestamp, and the signals of the left and right channels have different amplitude intensities and onset times. Thus, this study addressed these issues by proposing the IRawNet method with fused feature layers to merge different amplitude from multichannel raw audio. IRawNet aims to transcribe Indonesian classical music notes. It was validated with the *Gamelan* music dataset. The Synthetic Minority Oversampling Technique (SMOTE) overcame the class imbalance of the *Gamelan* music dataset. Under various experimental scenarios, the performance effects of oversampled data, hyperparameters tuning, and fused feature layers are analyzed. Furthermore, the performance of the proposed method was compared with Temporal Convolutional Network (TCN), Deep WaveNet, and the monochannel IRawNet. The results proved that proposed method almost achieves superior results in entire metric performances with 0.871 of accuracy, 0.988 of AUC, 0.927 of precision, 0.896 of recall, and 0.896 of F1 score.

Keywords: Indonesian Classical Music, *Gamelan* Notes Transcription, Music Signal Processing, Multi-Channel Raw Audio, Deep Learning, IRawNet.

1. INTRODUCTION

AMT aims to automatically retrieve the musical source codes from a song. One such musical source code is the note composition of a song [1]. Machine learning has been utilized to recognize notes from isolated note recordings. Performing deep learning extends the capability of machine learning to transcribe sequences of notes from song recordings. Previously developed AMT methods involved a preprocessing step that transforms raw

audio to the time-frequency domain to produce more amplitude features in time frames. Transcription in the time-frequency domain works at the frame level or note level. The frame level only contains pitch information across the frame. Meanwhile, note level transcribes notes as a pitch that contains onset and offset information. Onset is assigned as the time of the beginning of a note. Meanwhile, offset is a time at the end of the note. Commonly, both are predicted as binary values along frames that contain pitch. The transcription result in the time-frequency domain requires post-processing to convert to the actual playing time in the time domain. The conversion probably induces significant timestamp shifts in the time domain if prediction errors occur at the frame level in the time-frequency domain because a frame is formed from a windowing process where each window is a combination of several timestamps.

Therefore, AMT requires an agile method to transcribe notes in the time domain to obtain pitch information that corresponds to the actual playing time. It is potentially applied for real-time music transcription because it does not involve the preprocessing of signal transformation and post-processing of transcription result conversion. It only relies on features of amplitude sequence.

Recently, an amplitude sequence of raw audio has been investigated as input representation for audio generation [2], [3]. TCN was examined for tracking the musical beat [4]. Furthermore, Deep WaveNet was introduced for transcribing piano notes directly from raw audio and classifying heart diseases [5], [6]. TasNet was presented for music source separation in time domain. Those methods simplified input by using monochannel raw audio.

Commonly, raw audio is composed of multichannel amplitudes. Developing a deep learning method using inputs derived from multichannel is challenging because the channels have Interaural Level Differences (ILD) and Interaural Time Differences (ITD). ITD and ILD induce time and sound level differences while signals arrive at every channel. An example of the ITD effect is the difference in the onset time between channels. Onset time means a time of musical note is starting. Whereas ILD produces the amplitude difference among the channels at each timestamp. Although ILD and ITD from the multichannel are useful for signal separation tasks [7], [8]. However, many AMT methods avoided them by simplifying the multichannel amplitude to the monochannel by averaging amplitude from the left and right channels. Exploiting amplitude features directly from multichannel raw audio has rarely been investigated in the AMT task. Stereo feature enhancement (SFE) was suggested for combining spectral features from the stereo channel to overcome ILD and ITD [9]. The main contributions of this study are described below:

1. Transcribing notes from song recording that contain a single instrument of an Indonesian classical music.

2. Proposing IRawNet as a deep learning method to transcribe notes directly from multichannel raw audio. IRawNet transcribes notes corresponding to the presence of pitches in the time domain.
3. Overcoming ILD and ITD effects from multichannel raw audio with fused features layers in the IRawNet architecture.
4. Presenting the Gamelan music dataset to evaluate the proposed method. Gamelan is the classical music of Indonesia, classified as polyphonic music [10], [11].
5. Balancing class distribution of the Gamelan music dataset using SMOTE.
6. Analyzing the best hyperparameters tuning of IRawNet and fused feature layers with statistical analysis.

2. RELATED WORKS

Short Time Fourier Transform (STFT), Constant Q Transform (CQT), Harmonic Constant Q-Transform (HCQT), Variable Q Transform (VQT), Log Magnitude Spectrogram, Differential Spectrogram, Scalogram, Chromagram, and Mel Spectrogram were widely used to transform raw audio signal [12]. Moreover, transforming the Mel spectrogram with Discrete Cosine Transformation (DCT) generates another feature, namely MFCC (Mel Frequency Cepstral Coefficients). MFCC is suitable for representing the timbral of note [13], [14]. Those signal transformations were utilized to transcribe notes in the time frequency domain. Another input representation that has been investigated is the latent feature of signal transformations obtained from the Autoencoder Decoder and Generative Adversarial Network (GAN) methods [15–17]. Moreover, binary vectors have been investigated as input representations [18]. A binary vector represents the active and inactive notes in each timestamp with values of 1 and 0, respectively.

Signal transformations, latent features, and binary vectors are sequence data. Therefore, a recurrent network is an appropriate deep-learning method for note transcription tasks. Deep Neural Network (DNN) has been used to transcribe classical Thai music [3]. A simple architecture of the Recurrent Neural Network (RNN) method has been used to learn note patterns from signal transformation sequences [9], [14]. However, the computational process of an RNN is too expensive because the transcription procedure in each frame depends on all previous frames. Long Short-Term Memory (LSTM) is an RNN refinement for transcribing notes in each frame based on relevant information from previous frames [7], [19]. Many studies developed hybrid methods to achieve enhanced transcription performance. Hybrid methods of Convolutional Neural Network (CNN) and RNN [6]; CNN, RNN, and LSTM [20]; CNN and Bidirectional LSTM [21]; Convolutional Recurrent Neural Network (CRNN) and Bidirectional Gate Recurrent Unit (BiGRU) have been proposed by combining various signal transformations [22], [23]. The results have proven that combining CNN in hybrid methods contributed to increasing the transcription performance. However, the hybrid methods relied on signal transformation as input representation. Hence, they are not

potentially applied for real-time music transcription. Thus, this study presents IRawNet to transcript music directly from raw audio without any handcraft signal transformation and feature extraction. Hence, it is potentially applied as a real-time music transcription method.

3. ORIGINALITY

This study proposed a new method IRawNet for transcribing musical note directly from multichannel raw audio without any handcraft feature extraction and signal transformation steps. IRawNet predicts musical notes based on features of amplitude sequence. Features of amplitude sequence contain pitches that have been annotated based on notes named by *a Gamelan* expert. IRawNet contains fused feature layer to tackle ILD and ITD effect that induce the difference of onset time and amplitude level from the left and right channels by using equations (2) and (3).

Moreover, we present our Gamelan music dataset [24] that has been published in the Zenodo repository [25].

4. SYSTEM DESIGN

4.1 Dataset Preparation

This study used a public *Gamelan* music dataset. We collected solo recordings of *Saron* instruments from *Gamelan* music dataset. Collected dataset, label, and source codes are available at the link https://github.com/dewinurdiyah05/*Gamelan*-Notes-Transcription.

Saron recordings have various durations between 76.814 to 204.823 seconds with a sampling rate of 44100 Hz. The sampling rate was downsampled to 11024 Hz to reduce computation. The transcription task classified a note in each timestamp of song recordings. Thus, a timestamp was considered as a sample. The number of datasets was the number of samples from entire song recordings. Therefore, the number of samples was (2 channels x 11024 of sampling rate x duration x 9 song recordings) 23,206,137 samples. Each sample contains an amplitude feature. The amplitude level of the *Gamelan* songs is not uniform. Therefore, these levels were normalized in the range between 1 to -1 according to the minimum and maximum limits of audio recordings.

Each sample of *Gamelan* songs was annotated based on the active note in each timestep. The notes of *Barang, Gulu, Dhada, Lima*, and *Nem* on the fifth octave were symbolized as 1, 2, 3, 5, and 6 respectively. *Barang* on the sixth octave was categorized as 11. Furthermore, the notes annotation was changed to a uniform scale of 1, 2, 3, 4, 5, 6, and 7, respectively. If there was no active note in a timestamp, it was categorized as 0.

4.2 Class Balancing

The note transcription task classifies notes in each sample. Class imbalance is a crucial issue that causes low classification performance by inducing misclassification of minority classes [26]. Each *Gamelan* song is

generated with different note compositions, thereby influencing the number of note classes. Therefore, analyzing class distribution is important before training the data to the deep learning method. SMOTE is an oversampling method that is suitable for class balancing because it does not change the original signal. It replicates samples from the minority classes. The main steps of SMOTE are :

- Choose the sample s from the minority class randomly
- Identify the neighborhood Ns that is closest to s.
- Pick the next neighborhood sample $\bar{s}: \ \bar{s} \in Ns$
- Generate a new sample sn with equation (1).

$$sn = s + \alpha(\bar{s} - s), \alpha \in [0,1] \quad (1)$$

- Repeat the first to fourth point until the synthetic sample is equal to the number of SMOTE expectations. Finally, the number of oversampled data is (41,745,408 samples).

4.3 Slicing Window

Previous studies used sliding windows with overlapping samples as input data for deep learning. Sliding windows improved deep learning performance because equal numbers of samples are repeatedly trained. In addition, deep learning is more reliable for learning short sequences in the slicing window than learning directly from song recordings that contain long sequences. Therefore, the oversampled data were sliced with a window size of 0.25 seconds, which was equal to 2756 samples. The results of the window-slicing process were raw audio chunks, denoted as X_1^{NxTx1} the left channel, and X_2^{NxTx1} the right channel. X_1 and X_2 represent multichannel inputs. N is the number of window slices, T is equal to the number of samples in the window size, and 1 is the number of amplitude features in each sample. Moreover, the sequence of categorical notes of each *Gamelan* song recording was sliced according to the window slicing size, which is denoted as Y^{NxTx8}. Y is the multiclass target. Eight is the number of categorical notes encoded in the binary values. A binary value of 1 represents an active note, whereas 0 indicates that no note is played in a sample.

4.4 IRawNet Architecture

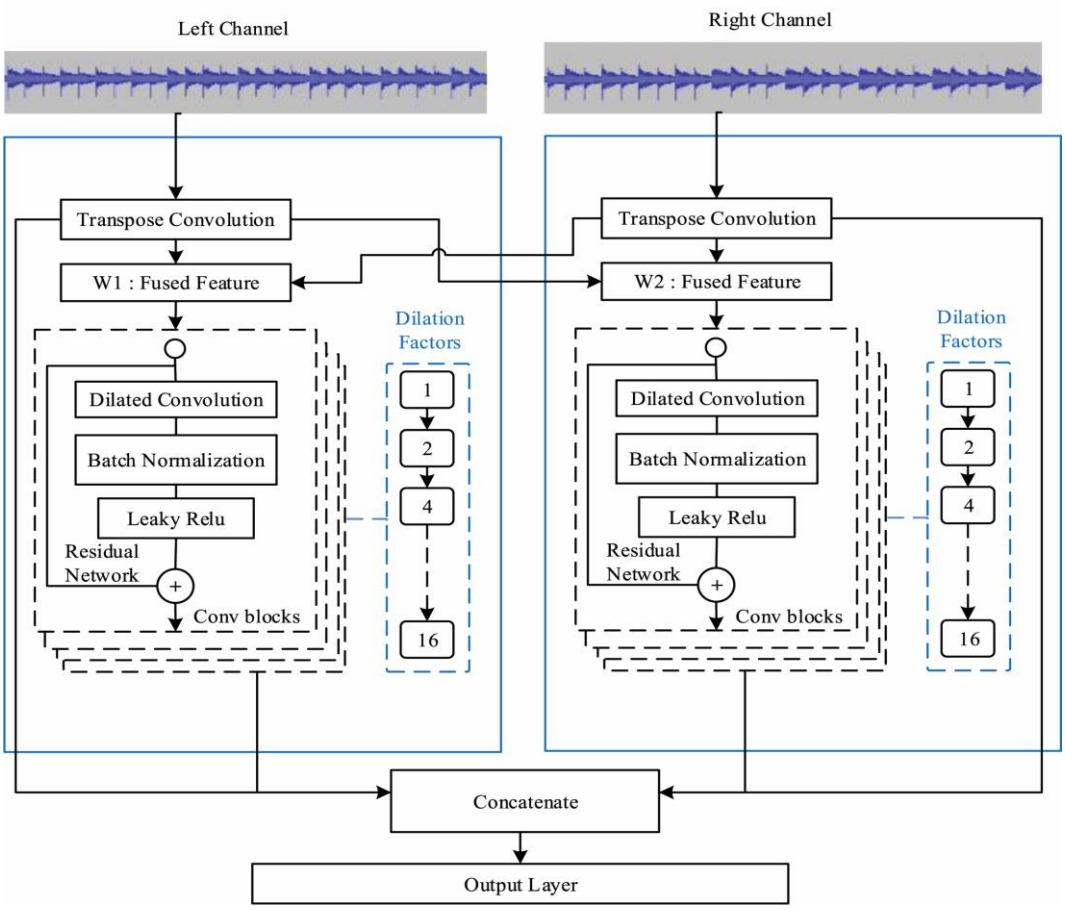


Figure 1. Proposed method: IRAWNET architecture

IRawNet is the deep learning method proposed in this study. Figure 1 visualizes IRawNet architecture. It consists of a multinetwork for extracting amplitude features from multichannel. Each sample of raw audio contains only one amplitude feature. Therefore, the transposed convolution layer enriches the amplitude features by increasing their dimensions [27]. The features from transposed convolution are denoted as W_H^1 and W_H^2, are associated with the fused feature layers to handle ITD and ILD effects in equations (2) and (3), respectively. The *tanh* activation function is applied to preserve the scale of amplitude features that do not change during feature integration. $\bar{R}$ in equation (4) is the average amplitude of the left and right channels. The amplitudes are averaged to prevent each of the channels from having a zero value. Moreover, Applying the direct sum among $\bar{R}$ and both W_1 W_2 avert too small features that are propagated to the convolutional block. Each convolutional block consists of a dilated causal convolution layer, a batch normalization, and a leaky ReLU activation function. The dilated causal convolutional layer classifies notes of the current sample by extracting previously selected samples based on the kernel size and dilation factor [28]. The batch normalization conserves weight value in the normal distribution

[29], and the leaky ReLU is a nonlinear activation function that is appropriate for the AMT task [17], [30].

Each convolutional block receives residual weights from previous convolutional blocks. The residual networks prevent small weights from being used in the subsequent convolutional blocks. Thus, it avoids overfitting in the deep network. The subsequent layer includes the concatenation feature of the last convolutional block and the transposed convolution layers from the multinetwork. It aims to combine the features derived from deep and shallow networks to achieve improved transcription performance. The output layer is a dense layer with eight neurons, which is equal to the number of categorical notes. The activation function of the last layer is the softmax function, as presented in equation (5). It is suitable for multiclass classification problems. Each multiclass vector Y_i has a score S_i for each element S_j. The combination of softmax and the categorical cross-entropy loss preserves only one element in $\bar{Y}_i$has a positive class prediction score S_p using equation (6). Table 1 shows initial hyperparameter of IRawNet. The input size of the left and right channels is 0.25 seconds or 2756 samples, the filter size is 32, the kernel size is 30, the alpha value of the leaky ReLU activation function is 0.03, and fused feature layers consist of W_1 and W_2. IRawNet was compiled with categorical cross-entropy loss, epochs of 100, and the RMSprop optimizer with a learning rate of 10-e5. The effective hyperparameter tuning is observed in the result section.

$$W_1 = \tanh\left(\bar{R} \oplus W_H^1\right) \tag{2}$$

$$W_2 = \tanh\left(\bar{R} \oplus W_H^2\right) \tag{3}$$

$$\bar{R} = \frac{1}{2}\left(W_H^1 \oplus W_H^2\right) \tag{4}$$

$$f(S_i) = \frac{e^{S_i}}{\sum_j^{Y_i} e^{S_j}} \tag{5}$$

$$Loss = -log\left(\frac{e^{S_p}}{\sum_j^{Y_i} e^{S_j}}\right) \tag{6}$$

Table 1. Initial hyperparameters tuning of the proposed method

Channel	*Layer Name*	*Output Shape*	*Kernel Size*	*Number of Filter*	*Stride*	*Dilation Factor*	*Alpha Value*
Left	Input 1	2756	-	-	-	-	-
Right	Input 2	2756	-	-	-	-	-
Left	1D Transpose Conv 1	2756	1	32	1	-	-
Right	1D Transpose Conv 2	2756	1	32	1	-	-
Left	W1	2756	-	-	-	-	-
Right	W2	2756	-	-	-	-	-
Left	Convolutional Block-1						
	1D Convolution 1	2756	30	32	1	1	-
	Batch Norm 1	2756	-	-	-	-	-
	Leaky Relu 1	2756	-	-	-	-	0.03
	Residual 1	2756	-	-	-	-	-
	Convolutional Block-2						
	1D Convolution 2	2756	30	32	1	2	-
	Batch Norm 2	2756	-	-	-	-	-
	Leaky Relu 2	2756	-	-	-	-	0.03
	Residual 2	2756	-	-	-	-	-
	…						
	Convolutional Block-5						
	1D Convolution 5	2756	30	32	1	16	-
	Batch Norm 5	2756	-	-	-	-	-
	Leaky Relu 5	2756	-	-	-	-	0.03
	Residual 5	2756	-	-	-	-	-
Right	Convolutional Block-1						
	1D Convolution 1	2756	30	32	1	1	-
	Batch Norm 1	2756	-	-	-	-	-
	Leaky Relu 1	2756	-	-	-	-	0.03
	Residual 1	2756	-	-	-	-	-
	Convolutional Block-2						
	1D Convolution 2	2756	30	32	1	2	-
	Batch Norm 2	2756	-	-	-	-	-
	Leaky Relu 2	2756	-	-	-	-	0.03
	Residual 2	2756	-	-	-	-	-
	…						
	Convolutional Block-5						
	1D Convolution 5	2756	30	32	1	16	-
	Batch Norm 5	2756	-	-	-	-	-
	Leaky Relu 5	2756	-	-	-	-	0.03
	Residual 5	2756	-	-	-	-	-
Combination of Left and Right	Concatenated Layer	2756	-	-	-	-	-
	Dense	2756	-	8	-	-	-
	Softmax	2756	-	-	-	-	-

4.5 Training, Validation, and Testing

Without overlapping raw audio chunks in training and testing, the raw audio chunks were split into 80% and 20% for training and testing randomly. The training stage used 5 K-fold, and 20% of training data was arbitrarily selected as validation.

4.6 Performances

4.6.1 Accuracy

As expressed in equation (7), accuracy is the correct prediction of positive and negative classes. The best note transcription performance has an accuracy score that is close to 1.

$$Accuracy = \frac{(TP+TN)}{(TP+FN+TN+FP)} \tag{7}$$

TP and FN are the numbers of true and false predictions of the positive class, respectively. FP is the false prediction of the negative class. Whereas TN is the true prediction of the negative class.

4.6.2 Precision

Precision is the ratio of the number of true predictions of the positive class to the total number of positive predictions, as expressed in equation (8). The positive class represents an active note with a value of 1 in Y.

$$Precision = \frac{TP}{(TP+FP)} \tag{8}$$

4.6.3 Recall

Recall is the ratio of the number of true predictions of the positive class to the total number of predicted samples that belong to the positive class, as expressed in equation (9).

$$Recall = \frac{TP}{(\mathrm{TP+FN})} \tag{9}$$

4.6.4 Area Under Curve (AUC)

AUC is a scalar value corresponding to the Receiving Operating Characteristic (ROC) performance. The ROC curve is a suitable curve for visualizing performance based on the true-positive and false-positive rates. A higher AUC implies higher transcription performance. The AUC value in equation (11) is obtained from the ROC area of Recall or True Positive Rate (TPR) and False Positive Rate (FPR) in equation (10).

$$FPR = \frac{FP}{(\mathrm{FP+TN})} \tag{10}$$

$$AUC = \int TPR\, d(FPR) \tag{11}$$

4.6.5 F1 Score

F1 score is the harmonic mean of precision and recall, as expressed in equation (12).

$$F1 = 2 \times \frac{(Precision \times Recall)}{(\mathrm{Precision+Recall})} \tag{12}$$

4.6.6 Statistical Analysis

In the experimental scenario, the most effective hyperparameter values were identified by tuning the filter size, kernel size, alpha value of the leaky ReLU activation function, optimizer type, and number of convolutional blocks. Increasing the filter size, kernel size, and number of convolutional

blocks increased the number of total parameters. Many parameters require long training times. In addition, Increasing total parameters is not guaranteed to produce the best overall performances. Hence, the statical analysis of ANOVA and t-test were performed to examine the significant performance differences in each hyperparameter scenario based on a confidence level of 95% with (α = 0.05). If (P value > α), H_0 is accepted. Under this hypothesis, there is no significant performance difference. Otherwise, if the (P value < α), then H_1 is accepted, indicating significant performance differences. The accepted hypothesis is described in each hyperparameter scenario.

5. EXPERIMENT AND ANALYSIS

In the experimental scenarios, the performance effects of oversampled data, hyperparameters tuning, and applying fused feature layers on the proposed method were analyzed.

5.1 Effect of oversampled data

The class distribution of the original Gamelan dataset is imbalanced in Figure 2(a) because each Gamelan song does not require playing whole notes, and the note composition of each Gamelan song is different from those of the other songs. Figure 2(b) shows the result of oversampled data with the SMOTE method. This scenario compared the performances achieved by utilizing inputs from the original data and oversampled data. In Figure 3 employing the oversampled data as input improved the accuracy by 16%, the AUC by 3.6%, the precision by 11%, the recall by 27%, and the F1 score by 19% over those performances obtained from the original data. Thus, in subsequent scenarios, the oversampled data were used as inputs.

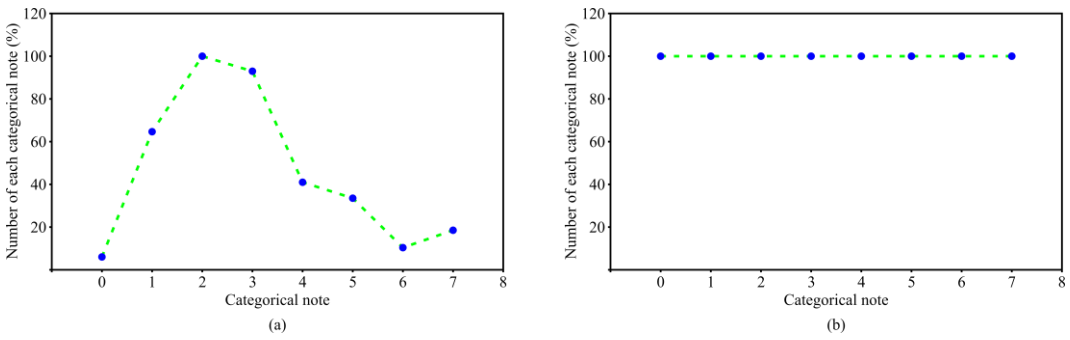
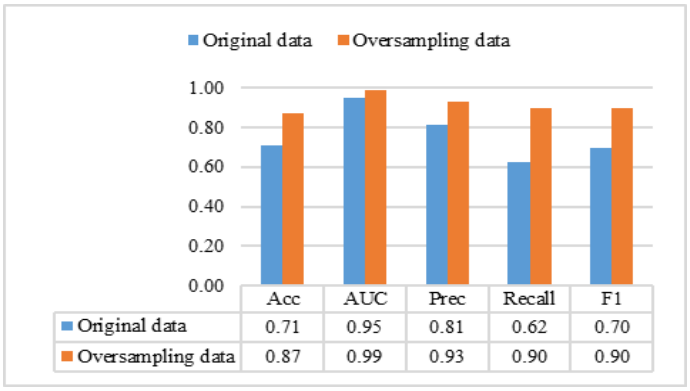


Figure 2. Class distribution of the original data (a) and oversampled data (b)



	Acc	AUC	Prec	Recall	F1
Original data	0.71	0.95	0.81	0.62	0.70
Oversampling data	0.87	0.99	0.93	0.90	0.90

Figure 3. Performances of utilizing input from the original and oversampled data

5.2 Hyperparameters Tuning

5.2.1 Effect of Filter Size

In this scenario, the performance was investigated using filter sizes 32, 64, and 128. A filter size of 32 generated a total of 310,216 parameters. A filter size of 64 corresponded to a total of 1,234,824 parameters, and a filter size of 128 raised the total number of parameters to 4,927,240. Figure 4 shows that increasing the filter size did not increase the overall performance. Moreover, based on the ANOVA test, H_0 was accepted because (P *value* = 0.278); hence, no significant performance differences were observed among filter sizes 32, 64, and 128. Thus, the most effective filter size is 32 because it yielded fewer parameters than the other options.

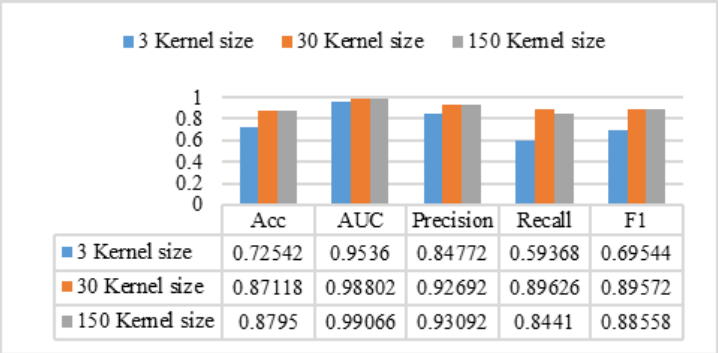
	Accuracy	AUC	Precision	Recall	F1
32 Filter size	0.87118	0.98802	0.92692	0.89626	0.89572
64 Filter size	0.88364	0.99146	0.9264	0.85474	0.88832
128 Filter size	0.88448	0.9915	0.92896	0.85444	0.89032

Figure 4. Performances of different filter sizes

5.2.2 Effect of kernel size

The kernel size determines the number of previous samples that are incorporated into the note transcription for the current sample. This scenario tested the performance achieved with kernel sizes of 3, 30, and 150. A kernel size of 3 yielded 33,736 total parameters, a kernel size of 30 generated 310,216 parameters, and a kernel size of 150 produced 1,539,016 parameters. The ANOVA test was rejected H_0 and accepted H_1 because of the (P value = 2.1e-21). Furthermore, a T-test was conducted to identify the most effective filter size by analyzing two kernel sizes.

- A paired t-test was used to examine significant differences between kernel sizes of 3 and 30. The result supported the acceptance of H_1 with (P value = 4.63513e-07). Figure 5 shows that the performances achieved with a kernel size of 30 were higher than kernel size of 3. Hence, the subsequence T-test considered kernel sizes of 30 and 150.



	Acc	AUC	Precision	Recall	F1
3 Kernel size	0.72542	0.9536	0.84772	0.59368	0.69544
30 Kernel size	0.87118	0.98802	0.92692	0.89626	0.89572
150 Kernel size	0.8795	0.99066	0.93092	0.8441	0.88558

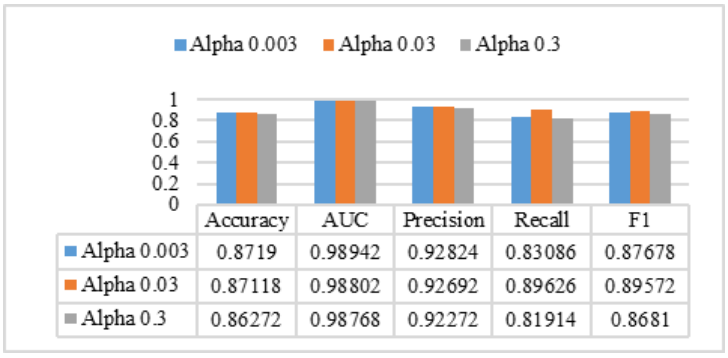
Figure 5. Performances of different tuning of kernel sizes

- The T-test with kernel sizes of 30 and 150 was approved H_0 with (P value = 0.189329433). No significant performance differences were observed between kernel sizes of 30 and 150. Moreover, the kernel size that corresponded to fewer parameters was identified as the more effective kernel size. Hence, the most effective kernel size is 30.

5.2.3 Effect of the alpha value of the leaky ReLU activation function

The leaky ReLU activation function aims to avoid zero weights via multiplication by alpha. Alpha values of 0.003, 0.03, and 0.3 were explored. The result of the ANOVA test acceptance H_1 with (P value = 0.0006). Hence, a T-test was required to examine the significant difference between two alpha values.

- A paired T-test of 0.3 and 0.03 justified H_1 with (P value = 0.025208516). It confirmed that there were significant performance differences between alpha values 0.3 and 0.03. Figure 6 shows that the alpha value of 0.03 reached higher performances than 0.3. Thus, the subsequent T-test examined alpha values of 0.03 and 0.003.
- According to the T-test with alpha values of 0.03 and 0.003, H_1 was accepted because (P value = 0.042834124). Hence, the more effective alpha value was determined based on the average performance. The alpha values of 0.03 and 0.003 achieved average performances of 0.91526 and 0.89944, respectively. Therefore, 0.03 is the recommended alpha value for the proposed method.



	Accuracy	AUC	Precision	Recall	F1
Alpha 0.003	0.8719	0.98942	0.92824	0.83086	0.87678
Alpha 0.03	0.87118	0.98802	0.92692	0.89626	0.89572
Alpha 0.3	0.86272	0.98768	0.92272	0.81914	0.8681

Figure 6. Performances achieved with different alpha values in the leaky ReLU activation function

5.2.4 Effect of the optimizer type

Exploring the optimizer type aimed to identify the best optimizer based on the convergence of the training and validation stage, metric performances, and statistical analysis results. This scenario examined optimizers of the Adaptive Moment Estimation (Adam), Root Mean Square Propagation (RMSprop), and Stochastic Gradient Descent (SGD) optimizers. Figure 7 shows that SGD achieved excellent convergence. However, SGD produced the lowest performance in accuracy. In contrast, Adam and RMSprop had high accuracy and achieved similar convergence in the training and validation stage. Figure 8 shows that Adam and RMSprop outperformed SGD in terms of all performances. Thus, a T-test was conducted to analyze the significant performance differences between Adam and RMSprop optimizers. The result approved H_1 with (P value = 0.04657), indicating significant performance differences between Adam and RMSprop optimizers. The total average performance of RMSprop and Adam were 0.91562 and 0.899768, respectively. Hence, the best optimizer is RMSprop.

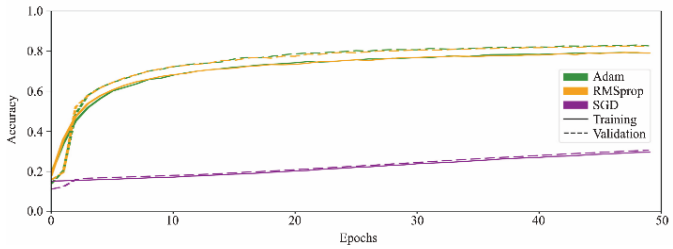
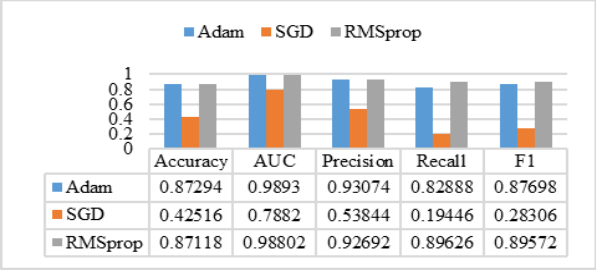


Figure 7. Accuracy convergence in the training and validation stage

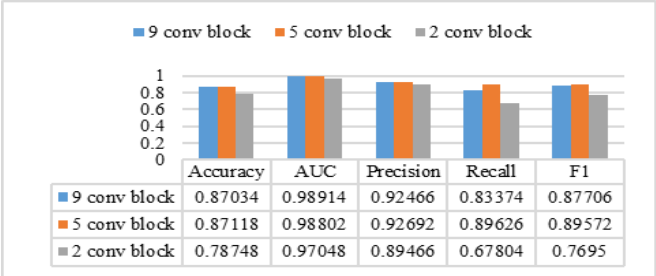


	Accuracy	AUC	Precision	Recall	F1
Adam	0.87294	0.9893	0.93074	0.82888	0.87698
SGD	0.42516	0.7882	0.53844	0.19446	0.28306
RMSprop	0.87118	0.98802	0.92692	0.89626	0.89572

Figure 8. Performance of the optimizer tuning

5.2.4 Effect of the number of convolutional blocks

The number of convolutional blocks influences the resulting feature extraction. IRawNet with deep network stacks uses more convolutional blocks to extract patterns from larger previous samples by increasing the dilation factor, hence, generating more parameters. In contrast, the shallow network only utilizes a few convolutional blocks and produces few parameters. This scenario investigated the performance of stacking two, five, and nine convolutional blocks. The result of the ANOVA confirmed H_1, there were significant performance differences among two, five, and nine convolutional blocks with (P value = 3.46963E-14). Figure 9 visualizes that the performance of five convolutional blocks is almost equivalent to nine convolutional blocks. Therefore, the T-test inspected the significant performance differences between these options. The result affirmed H_1 with (P value = 0.02854). It explains significant performance differences between five and nine convolutional blocks. The five and nine convolutional blocks attained an average performance of 0.91562 and 0.898988, respectively. Moreover, five and nine convolutional blocks generated 310,216 and 557,256 total parameters, respectively. Therefore, the most effective number of the convolutional blocks is five.

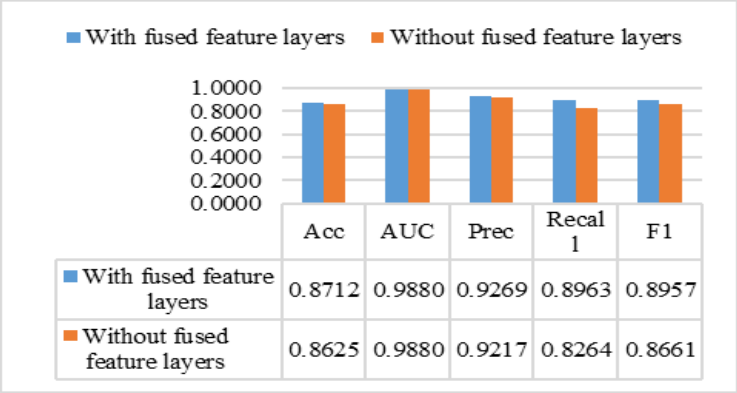


	Accuracy	AUC	Precision	Recall	F1
9 conv block	0.87034	0.98914	0.92466	0.83374	0.87706
5 conv block	0.87118	0.98802	0.92692	0.89626	0.89572
2 conv block	0.78748	0.97048	0.89466	0.67804	0.7695

Figure 9. Performances of different numbers of convolutional blocks

5.3 Effect of The Fused Feature Layers

The fused feature layers do not influence the number of total parameters. Fused feature layers contribute to overcoming ILD and ITD effects by combining amplitude features derived from multichannel. This scenario examined the versions of IRawNet with and without applying fused feature layers. Figure 10 shows that stacking the fused feature layer increased the performance by 0.8% in terms of accuracy, 0.5% in terms of precision, 6% in terms of recall, and 2% in terms of F1 score.



	Acc	AUC	Prec	Recal l	F1
With fused feature layers	0.8712	0.9880	0.9269	0.8963	0.8957
Without fused feature layers	0.8625	0.9880	0.9217	0.8264	0.8661

Figure 10. Comparison performance between versions of IRawNet using and not using fused feature layers

5.4 Comparison with the other methods

The proposed method was compared with Deep WaveNet, TCN, and the monochannel IRawNet. Table 2 shows that the proposed method achieves superior performance on all metrics. Mono channel IRawNet necessitates the shortest time to transcribe notes in the music song recording with a duration of 3.5 seconds because it has the fewest total parameters. Based on the number of total parameters, our proposed method desires 4 seconds more than mono channel IRawNet. Nonetheless, our proposed method can potentially be applied for real-time music transcription because the computation time is less than that of previous methods. Furthermore, Figure 11 visualizes the results of note transcription from a song recording. It shows that the proposed method produces lower error transcription than TCN, Deep WaveNet, and the mono channel IRawNet.

Table 2. Comparison performances with the other methods

Method	Accuracy	AUC	Precision	Recall	F1	Total Parameters	Time Computation (second)
TCN [3]	0.659	0.938	0.757	0.545	0.633	410,108	14.535
Deep WaveNet [5][31]	0.846	0.977	0.410	0.974	0.579	659,208	68.350
Mono channel IRawNet	0.8	0.976	0.870	0.733	0.795	155,112	7.006
Proposed method	**0.871**	**0.988**	**0.927**	**0.896**	**0.896**	310,216	11.397

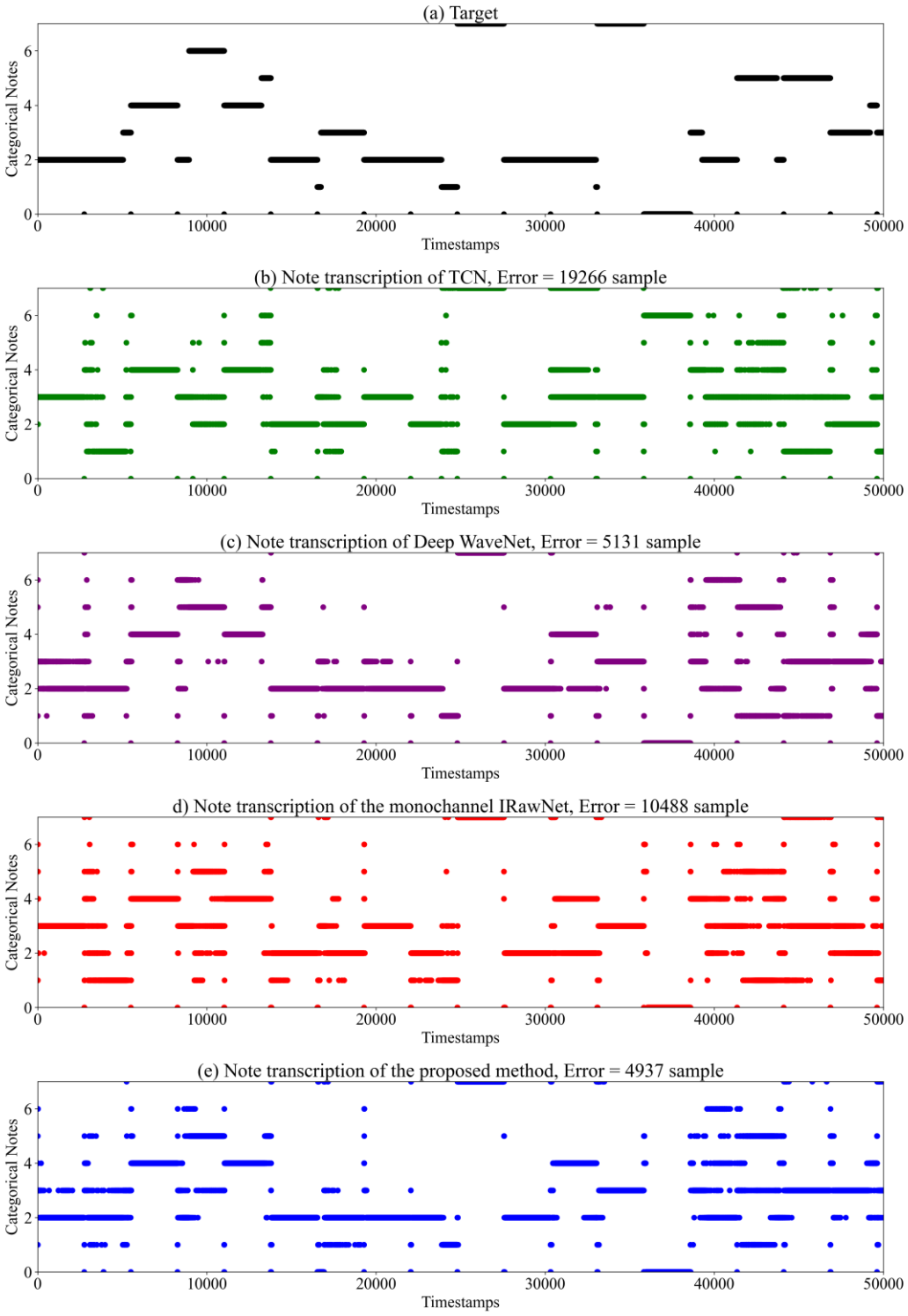


Figure 11. The comparison result of note transcription between the proposed method and the other methods. The error indicates false predictions from FP and FN.

6. CONCLUSION

This study presented IRawNet as a deep learning method for transcribing Indonesian classical music notes directly from multichannel raw audio as input representation. IRawNet extracts the sequence of amplitude features that represent note patterns in the music recording. It was validated on the *Gamelan* music dataset. The minority classes of the *Gamelan* music dataset were oversampled with SMOTE. The effect of oversampled data, hyperparameters, and fused feature layers was examined. The findings showed that adopting oversampled data boosted the accuracy by 16%, the AUC by 3.6%, the precision by 11%, the recall by 27%, and the F1 by 19% compared to the performance of using the original data. Based on the metric performance and the statistical analysis, The most effective hyperparameters are a filter size of 32, a kernel size of 30, an alpha value for the leaky ReLU activation function of 0.03, the RMSprop optimizer, and five convolutional blocks. Furthermore, The effect of the fused feature layer increased by 0.8%, the precision by 0.5%, the recall by 6%, and the F1 score by 2%. The performance of proposed method was compared with those of TCN [3], Deep WaveNet [31], and the monochannel IRawNet. The comparison results proved that employing the amplitude feature from multichannel in the proposed method improved the accuracy by 2% to 21%, the AUC by 1% to 5%, the precision by 5% to 51%, and the F1 score by 10% to 31%.

Acknowledgments

This study is supported and funded by Lembaga Pengelola Dana Pendidikan (LPDP) scholarship, Indonesia.

REFERENCES

[1] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, **Automatic music transcription: An overview,** *IEEE Signal Process. Mag.*, vol. 36, no. 1, pp. 20–30, 2018.

[2] A. van den Oord *et al.*, **{WaveNet}: A Generative Model for Raw Audio,** no. {arXiv}:1609.03499. 2016. Accessed: Jul. 15, 2022. [Online]. Available: http://arxiv.org/abs/1609.03499

[3] S. Bai, J. Z. Kolter, and V. Koltun, **An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling,** *ArXiv*, vol. abs/1803.0, 2018.

[4] E. P. MatthewDavies and S. Böck, **Temporal convolutional networks for musical audio beat tracking,** in *2019 27th European Signal Processing Conference (EUSIPCO)*, pp. 1–5, 2019.

[5] L. S. Martak, M. Sajgalik, and W. Benesova, **Polyphonic note transcription of time-domain audio signal with deep wavenet architecture,** in *2018 25th International Conference on Systems, Signals and Image Processing (IWSSIP)*, pp. 1–5, 2018.

[6] S. L. Oh *et al.*, **Classification of heart sound signals using a novel deep WaveNet model,** *Comput. Methods Programs Biomed.*, vol. 196,

pp. 105604, 2020.
[7] L. Chen, M. Yu, D. Su, and D. Yu, **Multi-band pit and model integration for improved multi-channel speech separation,** in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 705–709, 2019.
[8] S. Gul, M. S. Khan, and S. W. Shah, **Integration of deep learning with expectation maximization for spatial cue-based speech separation in reverberant conditions,** *Appl. Acoust.*, vol. 179, pp. 108048, 2021.
[9] W. Zhang, Y. Zhang, Y. She, and J. Shao, **Stereo feature enhancement and temporal information extraction network for automatic music transcription,** *IEEE Signal Process. Lett.*, vol. 28, pp. 1500–1504, 2021.
[10] S. Kristiawan, **THE GAMELAN AND ITS IMPACT ON DEBUSSY'S PAGODES,** *Mahasara-Journal Interdiscip. Music Stud.*, vol. 1, no. 1, pp. 24–32, 2021.
[11] J. Becker and A. H. Feinstein, ***Karawitan: Source Readings in Javanese Gamelan and Vocal Music**, Volume 1*. University of Michigan Press, 2020.
[12] K. Tanaka, T. Nakatsuka, R. Nishikimi, K. Yoshii, and S. Morishima, **Multi-Instrument Music Transcription Based on Deep Spherical Clustering of Spectrograms and Pitchgrams.,** in *ISMIR*, pp. 327–334, 2020.
[13] A. Huaysrijan and S. Pongpinigpinyo, **Deep Convolution Neural Network for Thai Classical Music Instruments Sound Recognition,** in *2021 25th International Computer Science and Engineering Conference (ICSEC)*, pp. 283–288, 2021.
[14] V. S. Pendyala, N. Yadav, C. Kulkarni, and L. Vadlamudi, **Towards building a Deep Learning based Automated Indian Classical Music Tutor for the Masses,** *Syst. Soft Comput.*, vol. 4, pp. 200042, 2022.
[15] C. Hawthorne, I. Simon, R. Swavely, E. Manilow, and J. Engel, **Sequence-to-sequence piano transcription with Transformers,** *arXiv Prepr. arXiv2107.09142*, 2021.
[16] B. Bahmei, E. Birmingham, and S. Arzanpour, **CNN-RNN and Data Augmentation Using Deep Convolutional Generative Adversarial Network for Environmental Sound Classification,** *IEEE Signal Process. Lett.*, vol. 29, pp. 682–686, 2022.
[17] H.-S. Choi, J. Lee, and K. Lee, **Spec2Spec: Towards the general framework of music processing using generative adversarial networks,** *Acoust. Sci. Technol.*, vol. 41, no. 1, pp. 160–165, 2020.
[18] A. Ycart and E. Benetos, **Learning and Evaluation Methodologies for Polyphonic Music Sequence Prediction With LSTMs,** *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 28, pp. 1328–1341, 2020.
[19] Q. Wang, R. Zhou, and Y. Yan, **Polyphonic piano transcription with a note-based music language model,** *Appl. Sci.*, vol. 8, no. 3, pp. 470, 2018.

[20] A. K. Sharma *et al.*, **Classification of Indian classical music with time-series matching deep learning approach,** *IEEE Access*, vol. 9, pp. 102041–102052, 2021.
[21] A. Sadekar and S. P. Mahajan, **Polyphonic Piano Music Transcription using Long Short-Term Memory,** in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–7, 2019.
[22] S. Shahriar and U. Tariq, **Classifying maqams of Qur'anic recitations using deep learning,** *IEEE Access*, vol. 9, pp. 117271–117281, 2021.
[23] M. A. Román, A. Pertusa, and J. Calvo-Zaragoza, **Data representations for audio-to-score monophonic music transcription,** *Expert Syst. Appl.*, vol. 162, pp. 113769, 2020.
[24] D. Nurdiyah, Y. K. Suprapto, and E. M. Yuniarno, **Gamelan Orchestra Transcription Using Neural Network,** in *2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*, pp. 371–376, Nov. 2020.
[25] D. Nurdiyah et al, September 12, 2023, **Gamelan Music Dataset**, Zenodo repository, doi: 10.5281/zenodo.8333916
[26] A. Arafa, N. El-Fishawy, M. Badawy, and M. Radad, **RN-SMOTE: Reduced noise smote based on DBSCAN for enhancing imbalanced data classification,** *J. King Saud Univ. Inf. Sci.*, vol. 34, no. 8, pp. 5059–5074, 2022.
[27] H. Gao, H. Yuan, Z. Wang, and S. Ji, **Pixel Transposed Convolutional Networks,** *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 5, pp. 1218–1227, 2020.
[28] L. Cheng, R. Khalitov, T. Yu, J. Zhang, and Z. Yang, **Classification of long sequential data using circular dilated convolutional neural networks,** *Neurocomputing*, vol. 518, pp. 50–59, 2023.
[29] M. Segu, A. Tonioni, and F. Tombari, **Batch normalization embeddings for deep domain generalization,** *Pattern Recognit.*, vol. 135, pp. 109115, 2023.
[30] Y.-N. Hung and A. Lerch, **Multitask learning for instrument activation aware music source separation,** *arXiv Prepr. arXiv2008.00616*, 2020.
[31] A. K. Sharma *et al.*, **Polyphonic note transcription of time-domain audio signal with deep wavenet architecture,** *IEEE Access*, vol. 28, no. 1, pp. 1–5, 2020.