# Exploring CNN-based Architectures for Vietnamese Traditional Music Genre Classification

Nguyen Do Nhat Huy*, Le Do Thanh Hung, Mai Anh Quan†,
Huynh Anh Dung, Tieu Nhat Thanh, Do Quang Huy, Bui Tung Hung
FPT University, Ho Chi Minh, Vietnam
*first author, †corresponding author
email: quanmase182417@fpt.edu.vn

***Abstract*—With the rapid advancement of artificial intelligence, the classification of music genres is also progressing at a similar rate. Numerous corporations, including Spotify and Apple, have garnered significant interest in this domain, leading them to recruit highly skilled architects for their respective undertakings. Nevertheless, it is worth noting that various manifestations of Asian traditional music, including Vietnamese traditional music, have not attained an equivalent standard of performance excellence when compared to other musical genres. The objective of this study was to assess the effectiveness of Convolutional Neural Network (CNN) models in categorizing Vietnamese traditional music genres. In this study, we assessed the efficacy of two architectural designs, namely Convolutional Recurrent Neural Networks (CRNNs) and Parallel Convolutional Recurrent Neural Networks (PCRNNs), by employing diverse spectrograms. The present study involved a comparative analysis conducted on a well-maintained dataset of Vietnamese traditional music. The results of our study revealed the architecture that is most suited for this task. In our experiment, it has been proven that Parallel Convolutional Recurrent Neural Networks are the most suitable alternative for this specific purpose. This study makes a valuable contribution to the field of music information retrieval (MIR) by investigating the effectiveness and precision of Convolutional Neural Networks (CNN)-based approaches in classifying Vietnamese traditional music genres.**

***Index Terms*—Music genre classification, Convolutional Neural Networks, Recurrent Neural Networks, Vietnamese traditional music, Music.**

## I. Introduction

Music genre classification serves as a fundamental task in music information retrieval (MIR) systems [1], enabling efficient browsing, searching, and recommendation of music. Convolutional Neural Networks (CNNs) have become invaluable in the automatic classification of music genres attributed to their proficiency in extracting highly discriminative features from audio spectrograms. [2]

However, most existing research has predominantly focused on Western music genres, overlooking the specific challenges and opportunities inherent in Asian traditional music in general [3] and Vietnamese traditional music in particular. Vietnamese music features a rich diversity of instruments, styles, and regional variations, often adding layers of complexity to genre classification tasks. This paper aims to bridge this gap by exploring the performance of different CNN architectures for Vietnamese traditional music genre classification. We compare two promising architectures:

- Convolutional Recurrent Neural Networks (CRNN) [4]: This architecture incorporates Gated Recurrent Unit (GRU) networks to capture temporal dependencies within spectrograms, potentially benefiting genres with distinct rhythmic patterns.
- Parallel Convolutional Recurrent Neural Networks (PCRNN) [5]: This architecture combines a classic CNN with a Bi-directional Recurrent Neural Network (Bi-RNN) integrated with GRU to address limitations of CNNs and improves model quality.

This study made a significant contribution to the field of MIR, particularly in the area of MGC, by conducting a comparative analysis of two architectures: PCRNN and CRNN. Moreover, this study suggests a more effective architectural design that can be implemented in future research in MGC.

## II. Related work

Vietnamese traditional music genre classification in particular and music genre classification in general are broadly researched fields in Music information retrieval [6] [7]. Numerous researches illustrate that CNNs, which is a class of deep learning models in various types of fields, including computer vision, and natural language processing, also have a superior role in Music genre classification [6] [7]. In addition, the CNNs model, known as a deep learning model works well with images, and can also process various types of musical transformations, such as short-term Fourier transform (STFT) [8] and log-mel spectrogram. This transformation, similar to images, served as inputs for the classification model. However, unlike normal images, these visual feature maps of musical patterns possess sequential features essential for the music classification task, which may be lost during CNN training. Thus, K. Choi designed an architecture that combines CNN with recurrent neural networks (RNNs). RNNs are typically used in the field of natural language processing, utilizing data such as word sequences for tasks like word prediction. Their model consists of a linear combination of 4 CNN blocks and a 2-layer RNN with gated recurrent units (GRU) [4]. The CRNN demonstrated higher performance in music genre categorization compared to other CNN-based models due to its ability to capture more sequential data. According to Rui Yang, the first author of PCRNN paper [5], they believed they could develop an architecture capable of preserving as much

as possible the spatial features and temporal frame orders in the visual transformation of musical patterns. Their model comprises two blocks: a Convolutional Neural Network (CNN) block and a Bi-RNN block. The CNN architecture comprises a total of ten layers, five convolution layers and five max-pooling layers. These layers are further followed by ReLU activation functions and batch normalization. Conversely, the Bi-RNN block consists of 3 layers: a max-pooling layer, an embedding layer, and a 1-layer bidirectional RNN with GRU activation units, transforming the feature map into a 256-dimensional vector. The output of both blocks is fused into a 512-dimensional vector and then passed to a fully connected layer with softmax activation for the classification task [5].

## III. Methodology

.

### A. Overview

In this study, two models, namely CRNN and PCRNN, were employed. CRNN is a sequential combination of CNNs and RNNs, whereas PCRNN combines the two architectures in parallel. In addition, we thoroughly examined the technique adopted by two models, with particular emphasis on the GRUs approach.

### B. Convolutional Recurrent Neural Networks

Initially, we investigated the architecture of CRNN. The CRNN model utilizes a series of two-dimensional CNNs with four layers, followed by a two-layer RNN with gated recurrent units (GRU) to compress temporal patterns. The CRNN algorithm operates by compressing the initial matrix into a feature map. After completing the task, the model then employs RNN to extract temporal information in order to make predictions.

$96 \times 1366 \times 1$ dimensional log-mel-spectrograms are passed through four sets of Convolutional blocks that consist of $3 \times 3$ Convolutional layers with ReLU activation functions, Same padding with stride $1 \times 1$, and max-pooling layers with sizes of $2 \times 2$ move with a stride of $2 \times 2$. The feature maps are subsequently fed into a dropout layer, resulting in an output of a feature map with dimensions of $1 \times 1 \times 128$. The output is then transformed into a $1 \times 128$ shape and subsequently fed into a two-layer GRU (Gated Recurrent Unit), where the final hidden state produces a $1 \times 32$ vector as output. This vector is connected to the network's output after the final drop-out layer, and the probabilities of the five classes are calculated using the $sigmoid$ function.

*1) GRU:* The GRU algorithm is widely recognized for its ability to address the issue of missing or exploding gradients by effectively capturing temporal correlations from musical samples.

The update gates of the GRU are derived from conventional input and forget gates, hence incorporating a "reset gate". [5, equation 3].

$$h_i^{(t)} = u_i^{(t)} \tilde{h}_i^{(t)} + (1 - u_i^{(t)}) h_i^{(t-1)} \tag{1}$$

In this equation 1 [5, equation 3], the activation $h_i^{(t)}$ of $i^{th}$ layer at time $t$ is calculated by $h_i^{(t-1)}$ and the current candidate activation $\tilde{h}(t)$. The value of the "update gate" $u$ determines the number of unit updates each activation, which is determined using a specific formula. [5, equation 4]:

$$u_i^{(t)} = \sigma(b_u + [W_u x^{(t)}]_i + [U_u h^{(t-1)}]_i) \tag{2}$$

In this context, the variables $b$, $W$, and $U$ represent the bias, weight, and recurrent weights of the $i_{th}$ GRU. An input vector at time t is denoted as $x(t)$. The candidate activation, denoted as $h_i^{(t)}$, follows the same formula format as the "update gate". [5, equation 5]:

$$\tilde{h}_i^{(t)} = \tanh\left(b + [W x^{(t)}]_i + [U(r^{(t)} \otimes h^{(t-1)})]_i\right) \tag{3}$$

In the context of the "reset gate," the variable $r$ represents an element-wise multiplication operation. When the value of $r(t)$ approaches 0, the previous information will be disregarded as the "reset gate" is deactivated. [5, equation 6]:

$$r_i^{(t)} = \sigma(b_r + [W_r x^{(t)}]_i + [U_r h^{(t-1)}]_i) \tag{4}$$

the "reset gates" decide which information should be reconsidered later while the "update gates" determine how impact past states can affect the current state.

### C. Parallel Convolutional Recurrent Neural Networks

In order to classify music genres, PCRNN (Parallel CNN-RNN) is a hybrid architecture that merges the strengths of recurrent and Convolutional neural networks. This combination enables the PCRNN to potentially outperform conventional CNNs by capturing both temporal relationships and spatial information inside music spectrograms [5].

Based on the original RNN architecture, Rui Yang and colleagues introduced a next-gen structure [5], a new way to approach this task: The PCRNN (Parallel Convolutional Recurrent Neural Networks-Based). The PCRNN In a single stage, this approach integrates feature extraction and time-series data classification. The parallel convolutional neural network (CNN) and bi-recurrent neural network (Bi-RNN) blocks are designed to extract spatial features and temporal frame orders, respectively. The outputs of both blocks are combined to provide a robust representation of the time-series data.

In the CNN block, alternating pairs of max-pooling and convolutional layers process $513 \times 128 \times 3$ STFT spectrograms, A max-pooling operation is performed after each convolutional layer to further process the output of the preceding convolutional layer. The utilization of rectified linear units (ReLUs) activation functions within each convolutional layer offers expedited convergence and demonstrate a noteworthy capacity to address the issue of vanishing gradients. Additionally, batch normalization is employed in this procedure to enhance efficiency.

Our CNN block produces a 320-dimensional vector, $X_{cnn}$, which is then flattened. [5, equation 8]:

$$X_{cnn} = (X_1, X_2, ..., X_{320})^T \tag{5}$$

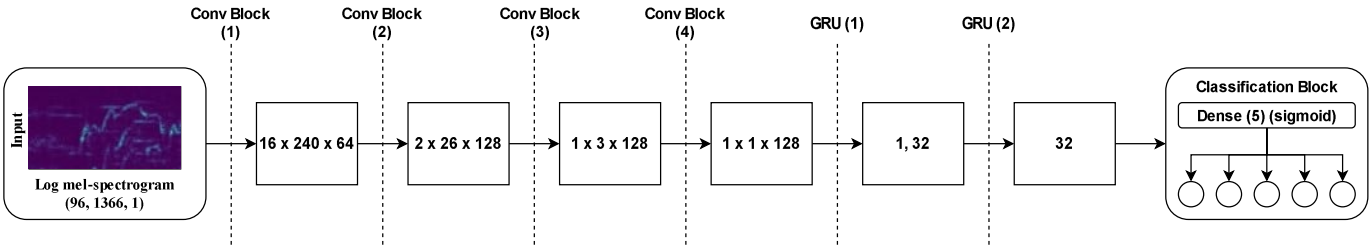


Fig. 1. Convolutional Recurrent Neural Networks

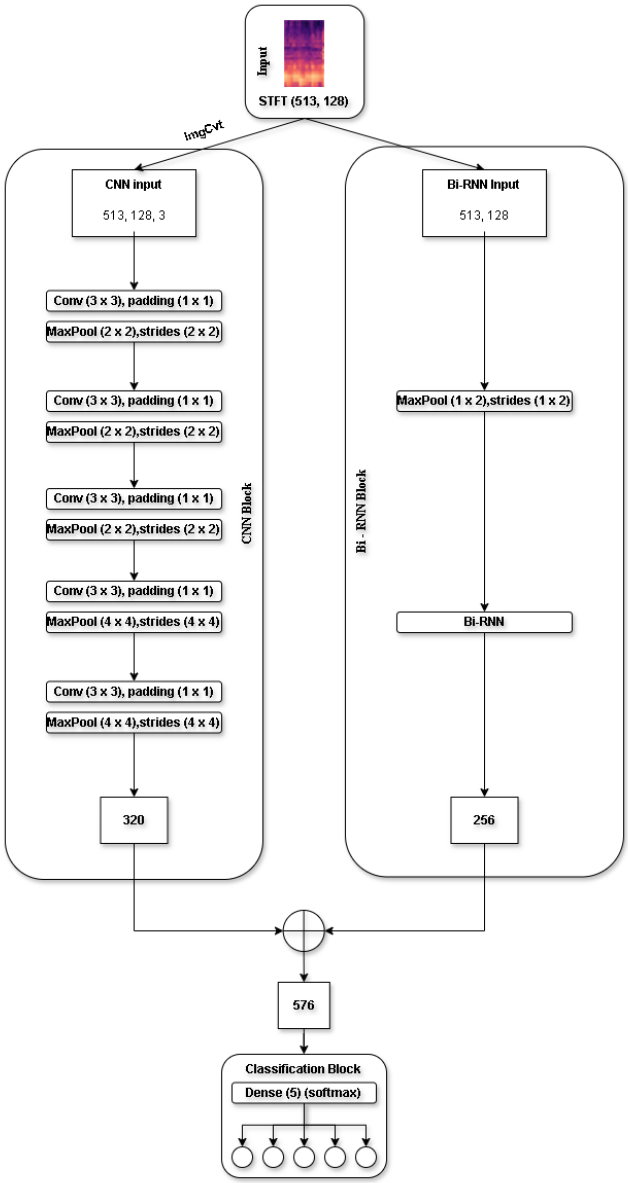


Fig. 2. Parallel Convolutional Recurrent Neural Networks

In this Bi-RNN block, the input spectrogram is subjected to dimension reduction using a max-pooling layer. By adhering to this methodology, the dimensions of the spectrogram are reduced from $513 \times 128 \times 3$ to $513 \times 128$. In the realm of feature extraction, conventional recurrent neural networks (RNNs) primarily rely on preceding contexts while neglecting the significance of reverse dependencies. However, a multitude of applications demonstrates that the prediction obtained from analyzing a sample is significantly impacted by the complete input sequence, which includes both historical and prospective data. Conventional RNNs suffer from the well-known problems of vanishing and ballooning gradients when dealing with long-term dependence. This issue was explicitly addressed with the development of the Bi-directional Recurrent Neural Network (Bi-RNN). The Bi-RNN block is designed based on two main factors: 1) the incorporation of an RNN with GRUs to capture additional temporal elements that may be disregarded by CNNs, and 2) the thorough utilization of past and future information within a complete sequence to extract more representative features.

In a bidirectional architecture, the determination of forward Gated Recurrent Units (GRUs) is based on the utilization of prior states in the positive time direction, whilst the computation of backward GRUs is based on the utilization of future values in the reverse time direction. [5, equation 7].

$$h_i^{(t)} = u_i^{(t)} \tilde{h}_i^{(t)} + (1 - u_i^{(t)}) h_i^{(t+1)} \tag{6}$$

Unlike the unidirectional architecture, the Bi-RNN with GRUs is capable of acquiring more robust representations by leveraging the entire sequence. In the end, this branch forms a 256-dimensional vector, ready for fusion.

In our paper, we want to preserve the integrity of the feature map from the previous layer for better results, so we decided to remove the Embedding layer, which compressed the feature map in the original study. We believe this action doesn't affect too much on the conclusion too much.

Having two outputs, we fused them into one 576-dimensional vector $\boldsymbol{F}$ [5, equation 8]:

$$\boldsymbol{F} = x_{cnn} \oplus x_{rnn} \tag{7}$$

follow by a softmax layer [5, equation 9]:

$$\hat{\mathbf{y}} = softmax(W^T F + b) \tag{8}$$

the affine transformation adding bias $(W^T F + b)$ reduces the vector from 576-dimensional into a 5-dimensional feature vector $\boldsymbol{v}$.

$$P(i) = \frac{exp(v^{(i)})}{\sum_{i=1}^{k} exp(v^{(i)})} \tag{9}$$

equation 9 [5, equation 10] calculates each value in vector $\boldsymbol{v}$ and represents a category probability $P(i)$ distribution over the various genres when $i$ run from 1 to $k$ (in this case, $k = 5$). When the backward process runs, the gradients of $X_{cnn}$ and $X_{rnn}$ are computed by the follow formula [5, equation 12]:

$$\begin{aligned}[\frac{\partial L}{\partial X_{cnn}}, \frac{\partial L}{\partial X_{rnn}}] &= W \frac{\partial L}{\partial F} \\ &= W \frac{\partial L}{\partial (X_{cnn} \oplus X_{rnn})} \\ &= W(\hat{y} - y)\end{aligned} \tag{10}$$

## IV. Experiment

### A. Overview

The hardware configuration employed in the investigation was as follows: The 12th Generation Intel(R) Core(TM) i5-12400F offers a clock speed of 2500MHz, 6 cores, 12 logical processes, 16GB of RAM DDR4, and 15.8GB of storage. Regarding the graphic card, the NVIDIA GTX GeForce 1650 is utilized. The software utilized in this work encompassed Librosa for data preprocessing, and spectrogram generation from audio files, as well as TensorFlow, Keras, and OS for model construction. The evaluations employed F1-Score and Accuracy metrics. We selected these indicators due to their efficacy in assessing models.

### B. Dataset

*1) Overview:* This dataset, which includes a variety of Vietnamese traditional music genres, was carefully selected and assembled for this research. The dataset contains 5 classes "Xam", "Cheo", "Ca Tru", "Chau Van", and "Cai Luong", thirty seconds per .wav file. Data has been cleaned and converted into the diversity of spectrograms for our models.

*2) Preprocessing:* Each audio file is loaded with a sampling rate of 12000 Hz. The duration of the audio is trimmed down to 29.12 seconds to ensure that the number of frames is 1366. The audio is then converted to a mel-spectrogram (with the number of FFT points set to 512, the number of Mel bins set to 96, and the hop length set to 256). The mel-spectrogram is then converted to a logarithmic scale. This conversion helps in visualizing the spectrogram. Often the model is also improved with log-mel-spectrogram as well*(cite Choi et al.).Next, a singleton dimension is added to make the data compatible with the expected input shape for the model. For each audio, both the aforementioned log-mel-spectrogram and the genre folder name are appended to a data list and a label list for training. stored as a *Numpy* array through *NumPy* library [9].

To implement the PCRNN Model, we segmented the original data into segments of 10 seconds each. This has been done by iterating through each audio file that ends with the .wav extension. This was accomplished by iterating through each audio file with a .wav extension. For each audio file, the contents were read using the Audio Segmentation method from the *PyDub* package [10]. Subsequently, based on the specified duration and the length of the audio file, the program determined the number of segments to be generated and stored them accordingly. Each segment was then extracted from the original audio file. After that, the algorithm proceeded to iterate over each sub-folder and repeat the aforementioned process. Next, the subsequent stage involves loading the data, using the same method as before. The PCRNN model will utilize data from the recently partitioned 10-second dataset. Next, we retrieved data from a 10-second audio file using the *librosa* library [11]. Then, we transformed the audio data from the time domain to the frequency domain using the discrete Fourier transform (STFT) [12] function. After that, we calculated the logarithm of the amplitude to obtain a spectral representation of the audio data. Finally, we initialized three lists using specific functions. The STFT image representations were stored as *NumPy* arrays, with each array having a size of (height, width, channels). The data was stored as a list of *NumPy* arrays, where each array represents the STFT representation of an audio file. The remaining part was a list of labels, which contained the labels for each audio file, including 5 different music types. Ultimately, a tuple was generated that consists of the list of STFT image representations, STFT representations, and their accompanying labels.

The data was subsequently divided into training, validation, and testing sets using the split arrays into random train and test subsets method from the *scikit-learn* library [13]. The data sets are separated randomly into three parts: a validation set consists of 10% of the data, a test set consists of another 10% of the data, and a training set consists of the remaining 80% of the data. Before being inputted into the training model, the image data is transformed into a *NumPy* array.

### C. CRNN

For each .wav file, we compute the log-mels-spectrogram and save the resulting data in a list. The extracted label name from the folder's name has been saved in a separate list. We then utilized these lists to generate training and testing data. We decided to employ the Adam optimizer with a learning rate of 0.001. Conducting further investigation on the Keras library [14] and its components (such as Maxpooling, Convolution block, and Batch Normalization), we understood that the model requires input in the form of an array and label encoding. We also modified the input data to match the shape required by the model and ensure that it passes through convolutional layers. After iterations of modifying the epoch from 10 to 100, the error range of accuracy and metrics is decreased. We also apply an early stopping technique with a patience of 5 to counter overfit. Our experiments show that the best size for each batch is 128. We adjusted the dropout rate during the propagation of data through the Convolutional Neural Network (CNN) layers. Increment the value from 0.1 to 0.2 and employ a checkpoint to save the weight in order to get the most optimal model.

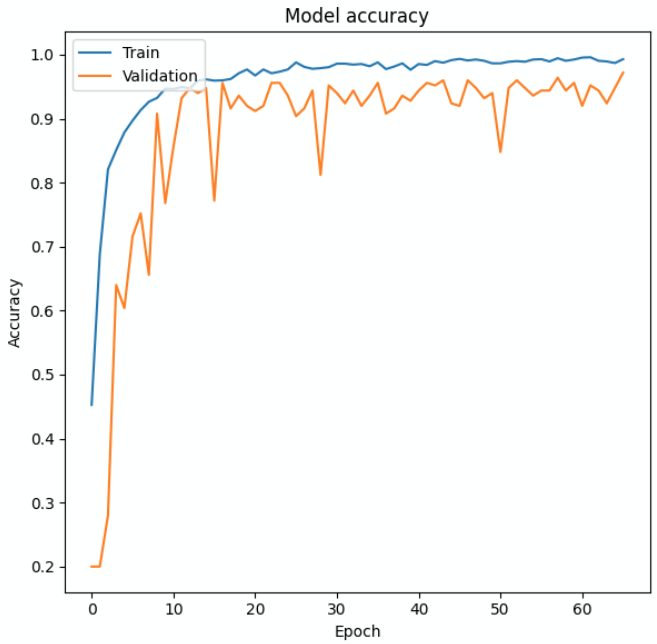


Fig. 3. Convolutional Recurrent Neural Networks

When using both the original dataset and an external dataset for prediction, the output of the model is a one-dimensional array consisting of five values that represent five labels. The label with the highest value is predicted as a result. So we tried to reverse the label transformation and successfully predict the categories. This process created a function that accurately forecasts data from an external dataset. To summarise, the boons of this model are minimal data loading and execution time, capable of executing on local systems with minimal processing resources and offering relatively good accuracy. Overall, this model has achieved an accuracy of $94.6\% \pm 1.2\%$ and F1-score of $95.1\% \pm 0.6\%$.

In Figure 3, we presented the training and validation accuracy for the CRNN model. The x-axis represents the number of epochs, and the y-axis denotes the accuracy. The blue curve illustrates the training accuracy, while the orange curve represents the validation accuracy.

### *D. PCRNN*

In the research report of the original PCRNN model, two types of *Numpy* arrays were inputted into the model, each representing 3 seconds of data, along with preprocessed data. Therefore, we first partitioned the original audio data into 3-second segments per file. During the architectural design of the model's feature map, it was seen that the removal of the embedding component resulted in the preservation of all the information contained inside the feature map. Hence, the decision was made to exclude the utilization of embedding in the RNN component of this model. Furthermore, dropout techniques were implemented after each layer in both branches to mitigate the issue of overfitting.

After the programming of the model architecture, an attempt was made to train it using the two input data streams. Nevertheless, we had constraints in terms of resources on our computing platform. Consequently, we opted to further divide the data, ensuring that each audio file had a duration of 10 seconds, employing the same preparation technique as before. By utilizing the 10-second data, we successfully addressed the computing resource constraints.

A trial training session was done, consisting of 20 epochs. Afterward, we conducted predictions on the test set using different label processing techniques, similar to those used for CRNN. As a result, the achieved accuracy was 95.6% and the F1 score was 96.02%. To enhance precision, we augmented the epoch count to 100 and used early stopping techniques, similar to Convolutional Recurrent Neural Networks (CRNN), to mitigate the risk of overfitting. The achieved outcomes encompassed an accuracy rate of $97.4\% \pm 1.3\%$ and an F1 score of $97.4\% \pm 1.1\%$.

Similar to Figure 3 above, Figure 4 demonstrated training and validation accuracy for the PCRNN model

### *E. Comparison*

After multiple iterations of training and testing, the efficacy of the PCRNN and RCNN models is demonstrated below table: Both CRNN and PCRNN attained remarkable indexes of $94.6\% \pm 1.2\%$ and $97.4\% \pm 1.1\%$ respectivelyI. RNN has effectively enhanced the algorithm's accuracy while employing parallel approaches has resulted in increased efficiency. The hypothesis posits that the simultaneous utilization of spatial and temporal information yields greater informational value compared to relying just on temporal information.

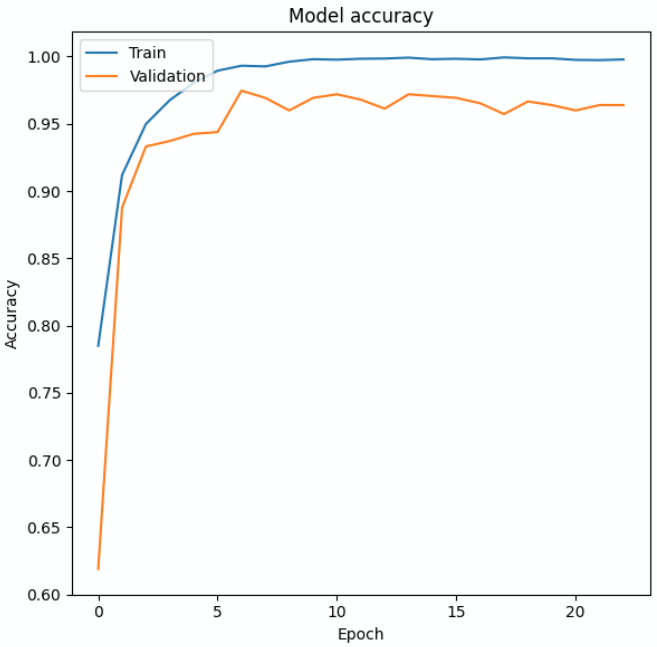


Fig. 4. Parallel Convolutional Recurrent Neural Networks

TABLE I
STATISTIC OF MODEL CRNN AND PCRNN

| | F1-score | Accuracy |
|---|---|---|
| **CRNN** | $95.1\% \pm 0.6\%$ | $94.6\% \pm 1.2\%$ |
| **PCRNN** | $97.4\% \pm 1.1\%$ | $97.4\% \pm 1.3\%$ |

## V. CONCLUSION

The field of music genre classification is receiving increasing attention. In response to this growth, architectural advancements have also emerged. We conducted a comparison of two CNN-based architectures using a dataset of Vietnamese traditional music, The result was shown by the experiment, opening a door for Asian traditional music in general and Vietnamese traditional music in particular in music genre classification.

## ACKNOWLEDGMENTS

We would like to express our sincere gratitude to Mr. Do Duc Hao from FPT University for his invaluable support and guidance throughout this project. His expertise and mentorship were instrumental in shaping our work.
We are also grateful to Mr. Nguyen Minh Dang for taking the time to review our paper and provide us with insightful feedback. His comments helped us to improve the clarity and quality of our work.

## REFERENCES

[1] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.
[2] A. Ghildiyal, K. Singh, and S. Sharma, "Music genre classification using machine learning," in *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 2020, pp. 1368–1372.
[3] K. Balachandra, K. Neha, S. Tushar, Swati, and S. Kumar, "Music genre classification for indian music genres," vol. 9, 08 2021.
[4] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," 2016.
[5] R. Yang, L. Feng, H. Wang, J. Yao, and S. Luo, "Parallel recurrent convolutional neural networks-based music genre classification method for mobile devices," *IEEE Access*, vol. 8, pp. 19 629–19 637, 2020.
[6] K. Palanisamy, D. Singhania, and A. Yao, "Rethinking CNN models for audio classification," *CoRR*, vol. abs/2007.11154, 2020. [Online]. Available: https://arxiv.org/abs/2007.11154
[7] Z. Nasrullah and Y. Zhao, "Music artist classification with convolutional recurrent neural networks," *CoRR*, vol. abs/1901.04555, 2019. [Online]. Available: http://arxiv.org/abs/1901.04555
[8] T. Toshniwal, P. Tandon, and N. P, "Music genre recognition using short time fourier tranform and cnn," in *2022 International Conference on Computer Communication and Informatics (ICCCI)*, 2022, pp. 1–4.
[9] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2
[10] J. Robert, M. Webbie *et al.*, "Pydub," 2018. [Online]. Available: http://pydub.com/
[11] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8, 2015.
[12] J. Allen, "Short term spectral analysis, synthesis, and modification by discrete fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 25, no. 3, pp. 235–238, 1977.
[13] 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, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[14] F. Chollet *et al.*, "Keras," https://keras.io, 2015.