LSTM and CNN-Based Detection of AI-Generated Classical Music from MIDI Features

Maecyntha Irelynn Tantra, Arya Wicaksana*
E-mail: maecyntha.irelynn@student.umn.ac.id, arya.wicaksana@umn.ac.id
*Corresponding author
Department of Informatics, Universitas Multimedia Nusantara
Tangerang 15810, Banten, Indonesia

Keywords: AI-generated music, Bach, classic music, CNN, detection, LSTM, MIDI

Received: [Enter date]

Detecting AI-generated classical music is a growing challenge as artificial intelligence continues to improve its ability to compose pieces that closely resemble human compositions. This study explores the use of deep learning methods like LSTM and CNN to classify whether a set of classical music is generated by AI or humans. The classification is based on sequential features extracted from MIDI files using beat-based segmentation, capturing statistical data of pitch, velocity, and duration over segments. The model was trained on a dataset comprising both AI-generated and human compositions, incorporating fine-tuning for optimal performance. Experimental results demonstrate that the proposed LSTM-based model achieves 99.00% accuracy on the primary test set, with an additional evaluation on an auxiliary dataset yielding 98.70% accuracy, confirming its reliability and strong generalization ability. Meanwhile, the CNN-based model attains accuracy scores of 97.00% and 97.10% on the primary and auxiliary datasets, respectively. Evaluation using confusion matrices and classification reports further validate both models' effectiveness, showing minimal misclassification rates. These findings suggest that while both LSTM and CNN achieve high classification performance in detecting AI-generated classical music, LSTM outperforms CNN in classification accuracy. Future research could explore integrating additional musical features or testing the model by expanding the dataset to cover a broader range of compositions, further improving model robustness and applicability.

Povzetek: "[Click here and Enter short Abstract in Slovene language]"

1 Introduction

In recent decades, as artificial intelligence (AI) has been one of the most significant innovations in the music industry, the growing interest from musicians and computer scientists in AI-based automatic music generation has led to rapid advancements in the field, with major companies actively contributing to its development [1]. This progress has enabled AI to generate musical compositions that mimic the styles of renowned composers while also producing original pieces, as demonstrated in studies related to the Flow Machines project [2]. While this innovation might have brought positive impacts [3], it has also raised concerns among composers and musicians regarding the originality and copyright of musical works, as the existing laws lack clarity on the boundaries of musical originality [4]. These concerns also include potential copyright infringement, its impact on royalties, and the ethical use of AI in music [3], similar to the problems surrounding AI applications in other fields [5], [6].

In addition to the previously mentioned concerns, another challenge is the increasing difficulty of distinguishing between content created by AI and by humans. While this study focuses on music, similar limitations in human judgment have been observed in other creative domains. For example, research on visual artwork found that people tend to evaluate art more positively when they believe it was created by a human, even though all the artworks were generated by AI [7]. This indicates not only that AI-generated content can be indistinguishable from human-made work, but also that perceptions of quality are often shaped by who is believed to be the creator. Similar limitations have been found in other fields, such as text analysis, AI tools like ChatGPT, Gemini, and Llama are becoming increasingly advanced at generating texts, making it more sophisticated to distinguish from human-written content [8]. While numerous studies explore AI-generated content detection in fields such as text analysis and image processing [9], [10], [11], research on detecting AI-generated music remains inadequate.

Since resources and prior research related to AI-generated music detection are still lacking, narrowing the detection scope to a specific genre is a practical starting point, as each genre has its own style. In this study, classical music was chosen as the primary subject as it is one of the fundamental genres, and the fact that most previous studies on AI-generated music have focused on this genre [12], [13]. As a result, more resources and references are available to support the better identification of AI-generated classical music in this study.

Additionally, building on previous findings that humans increasingly struggle to distinguish AI-generated content from human-made works [7], this study proposes an AI based detection approach to improve identification accuracy. Another research has also demonstrated the potential of using deep learning on symbolic music data for generative tasks. For example, a study explored chord progression generation using feature-based neural networks trained on this type of data, reinforcing the relevance of feature extraction and modeling in symbolic music tasks [14].

LSTM (Long Short-Term Memory) is one of the widely used architectures in AI music generation tools due to its ability to process sequential data. LSTM networks are a type of recurrent neural network (RNN) capable of learning and remembering over long sequences, making them particularly effective for tasks involving sequential data, such as rhythm learning and music composition [15]. Fudholi et al., in their research on enhancing classical music composition using LSTM algorithms, demonstrated the model's capability to learn complex musical structures, achieving an accuracy of 91.42% [16]. Although LSTM has primarily been used for music generation, its ability to capture long-term dependencies and recognize sequential patterns makes it a viable architecture for detecting AI-generated music.

CNN (Convolutional Neural Network) has also been studied in music classification tasks due to its ability to extract meaningful patterns from structured data. Unlike LSTM, which excels in modeling temporal relationships, CNN uses spatial hierarchies to identify distinguishing features across different segments of input data [17]. While CNN is often used in audio-based music classification, studies have demonstrated that it is also effective in symbolic music analysis [18], [19]. Given its ability to recognize structural patterns, this study also evaluates a CNN-based approach as an alternative to LSTM for AI-generated classical music detection. Both LSTM and CNN architectures have demonstrated strong performance in domains involving structured and sequential data, such as mental health prediction [20] and symbolic music classification [21] reinforcing their suitability for this detection task. Considering these findings, this study contributes an LSTM-based and CNN-based approaches in detecting AI-generated classical music.

The rest of this paper is structured as follows: Chapter 2 reviews related works; Chapter 3 and 4 introduce the theoretical background of the LSTM and CNN architecture used in this study; Chapter 5 details the proposed methodology; Chapter 6 presents and analyzes the results in the context of this study; Chapter 7 discusses the study results compared to previous related studies; Chapter 8 summarizes the paper with suggestions for future research.

2 Related works

Afchar et al. [22] introduced the first general-purpose AI-generated music detector using audio data, demonstrating its potential with an average detection accuracy of 97.4%. The result was achieved using a basic convolutional model, with the highest accuracy obtained when examining audio represented by amplitude-related features compared to other tested feature types. The study utilized a dataset containing around 25,000 music tracks across 16 genres.

Li et al. [23] compared the stacked LSTM and Bi-LSTM to distinguish AI-generated melodies from human-composed ones by analyzing the MIDI features. While pitch, position, duration, and velocity were initially considered the most critical features of notes, they eventually focused on the first three, as the dataset's velocity values were unfortunately unusable for effective analysis. They trained their models using a dataset sourced from Reddit and evaluated them using data provided by the competition committee. Both of their proposed algorithms secured the top two positions in the competition, with the LSTM model achieving a higher AUC (Area Under the Curve) score of 0.8812.

Deepak et al. [24] proposed an LSTM-based deep learning model to create a system for classifying different genres of music. To train and assess the model, the study used the GTZAN dataset, which comprises 1,000 audio tracks from ten distinct music genres. To efficiently handle the sequential nature of music data, the suggested model was composed of fully connected dense layers after recurrent LSTM layers. After 25 epochs of training, the system achieved an average accuracy of 96.17%.

Kong et al. [25] developed a large-scale MIDI-based composer classification system using Convolutional Recurrent Neural Networks (CRNN). They employed piano, onset, and velocity rolls as input representations to capture various musical features from the MIDI. Utilizing the GiantMIDI-Piano dataset, they evaluated the system's performance on both 10-composer and 100-composer classification tasks using a variety of input feature combinations. The system demonstrated higher accuracy in the 10-composer classification, achieving an average accuracy of 62.02% when evaluated on 30-second clips and an improved accuracy of 69.64% when assessed on entire music pieces.

Table 1: A summary of related works.

Literature	Year	Research Focus	Model	Dataset	Limitation	Average Result	
						Accuracy	AUC
Afchar et al. [22]	2025	AI-generated music detection using audio data	CNN	25,000 audio tracks	Limited interpretability due to lack of explicit musical features using audio	97.4%	-

Li et al. [23]	2020	Distinguishing AI-generated melodies from human-composed ones	LSTM & BiLSTM	10,000 MIDI files	Excluded velocity from final features due to dataset issues	**Accuracy**	**AUC**
						-	88.12% (LSTM) 80.32% (BiLSTM)
Deepak et al. [24]	2020	Music genre classification using audio data	LSTM	1,000 audio tracks	Used simple note-level inputs without modeling expressive or structural features	**Accuracy**	**AUC**
						96.17%	-
Kong et al. [25]	2020	MIDI-based music composer classification	CRNN	10,854 MIDI files	Relied on fixed-grid input representation, which may misalign with expressive timing	**Accuracy**	**AUC**
						62.02% (30s) 69.64% (full)	-

Compared to these studies, our work differs from previous research by highlighting several notable gaps. Afchar et al. [22] explicitly limited their focus to audio data, leaving symbolic or MIDI-based representations for future exploration. This presents a clear research opportunity in the symbolic domain, where event-level features such as pitch, duration, and velocity can be accessed directly, without being affected by performance or recording conditions. In contrast to Li et al. [23], who excluded velocity due to data limitations, our study retains velocity as a core expressive feature. We omit position, as our beat-based segmentation encodes rhythmic structure more effectively. Unlike Kong et al.'s study [25] which used fixed-grid representation that may misalign with expressive timing, our study adopts a beat-based segmentation approach, which allows features to align more naturally with musical timing and phrasing.

3 LSTM architecture

LSTM (Long Short-Term Memory) used in this study is a popular deep learning algorithm and a variant of Recurrent Neural Networks (RNNs). Unlike traditional regression methods, LSTM was designed to retain information over a lengthy period while discarding irrelevant data [26]. This algorithm is capable of handling sequential or time-related data by capturing both short-term and long-term dependency and modeling the complex, nonlinear relationship between variables. LSTM has also shown high effectiveness in various classification tasks involving time series data, further supporting its applicability in this context [27].Therefore, LSTM is well-suited for this study to learn musical patterns. As previously stated, musical features such as pitch, velocity, and duration can be extracted from MIDI files. By processing these features sequentially, LSTM can learn to identify underlying musical patterns, enabling it to differentiate between AI-generated and human-composed music. Moreover, LSTM is more resilient to missing data, noise, and irregularities without requiring assumption validation or initial hypothesis formulation like statistical methods [28].

Architecturally, LSTM comprises memory cells or cell states that store information over extended periods. Three different gates manage these memory cells, controlling the flow of information within the network, as outlined below [29].

- **Forget Gate** (f_t) controls part of the cell state ought to be forgotten.
- **Input Gate** (i_t) determines the new information to be kept later on.
- **Output Gate** (o_t) decides the output information produced by the cell state by combining the previous knowledge with the filtered new one.

Since cell states in LSTM function as the core component that allows information to flow unchanged, each cell state follows the steps illustrated in Figure 1.

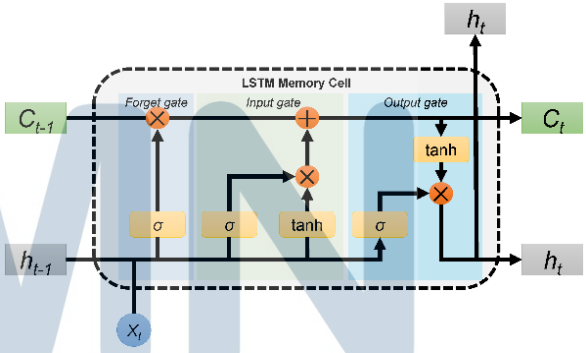

Figure 1: LSTM architecture. Source: [29], [30]

In the LSTM architecture, calculations occur within each gate, where the carried information consists of the previous hidden state (h_{t-1}) and the current input (x_t). First, part of the cell state (C_{t-1}) will go through the forget gate, where a sigmoid function will generate a value between 0 and 1. A value closer to 1 increases the likelihood of retaining the information, while a value closer to 0 increases the likelihood of discarding it. Next, the sigmoid layer of the input gate determines which information to update, followed by a tangent hyperbolic ($tanh$) layer that outputs a candidate vector for the cell state. Cell state (C_t) will then be updated by combining existing knowledge retained by the forget gate and new information selected by the input gate. Finally, the sigmoid layer of the output gate will decide which part of

the cell state will be passed as the new hidden state (h_t) for the next step in the network.

4 CNN architecture

CNN (Convolutional Neural Networks) offers an alternative approach to sequence modeling compared to LSTM. While LSTM excels at capturing long-term dependencies in sequential data, CNN focuses on learning local patterns through hierarchical feature extraction [31]. In music analysis, CNN can effectively identify structural patterns in symbolic music. By applying convolutional filters to MIDI features, CNN might be able to extract meaningful representations without relying on recurrent connections.

Basic CNN model structure consists of a convolution layer, activation layer, pooling layer, and fully connected layer as shown in Figure 2.

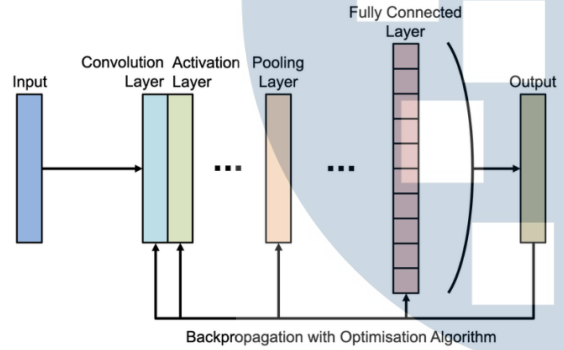

Figure 2: CNN architecture. Source: [32].

To extract feature maps, the first layer, known as the convolution layer, convolves the input pictures. There are some types of convolutional layers that are used commonly as listed below [33].

- **Conv 1D** is suitable for one-dimensional sequential data such as text and time-series data.
- **Conv 2D** can be used to process audio and image applications.
- **Conv 3D** is generally used for video and volumetric data.

After the convolution operation extracts spatial patterns, the activation layer then adds non-linearity to these feature maps, allowing the network to learn complex, non-linear classifiers for the input data. Some of the commonly used activation functions are ReLU (Rectified Linear Unit), sigmoid, and *tanh*. The feature maps are then abstracted by the pooling layer, which also modifies their dimensions as needed. Lastly, the fully connected layer performs classification using data gathered from the pooling and convolution layers.

5 Methodology

5.1 Data collection

The data collection involved gathering musical pieces from Hugging Face [34], JS Fake Chorales [35], and the Bach Doodle dataset from Magenta [36]. Only compositions by J.S. Bach were selected, as most publicly available AI-generated classical music datasets are based on his style. This ensures stylistic consistency, allowing the model to learn to detect music in the same style. The first two datasets were provided in MIDI format, while the Bach Doodle dataset is divided into 192 shards (000-191), each representing a separate shard of the overall dataset and can be downloaded via links. These shards enable researchers to download specific portions rather than the entire collection, improving data management efficiency. The Bach Doodle dataset was available in JSONL format, storing MIDI information as its values.

In total, 5,000 samples were collected, maintaining a balanced 50:50 ratio between AI-generated and human-composed pieces Specifically, two shards (e.g., 003 and 075) from the Bach Doodle dataset were randomly selected, and 2,000 samples were extracted from each shard. To complete the dataset, another 500 samples were collected from the Hugging Face MIDI dataset and 500 from the JS Fake dataset, resulting in a diverse and balanced collection.

Additionally, an auxiliary dataset was collected from two randomly selected shards of the Bach Doodle dataset that were not used in the primary dataset. This ensured that there was no overlap between the primary and auxiliary sets. Then, 500 samples were selected from each shard. In total, the auxiliary dataset consisted of 1,000 samples, with an equal proportion of AI-generated and human-composed music to maintain a balanced distribution.

5.2 MIDI feature extraction

Extracting the features of all the data required different approaches, as there are two types of file formats. Although some data was not in MIDI format, its values contained MIDI-related information, allowing for a similar and more efficient feature extraction process. To ensure that the music features could serve as meaningful input for the model, the data was divided into multiple segments based on beats. The feature extraction process involved the following steps.

- **Beat extraction**
  - For MIDI files, beats were extracted using a predefined library function.
  - For JSONL files, beats were computed based on each note's start and end.
  - The beat list was sorted and deduplicated to maintain temporal order.
- **Beat based segmentation**
  - The number of segments were determined by dividing the total beat count by a predefined 4 beats per segment, that aligns with the common practice in western music [37].
  - For example, a piece with 40 beats would be divided into 10 segments.
- **Feature extraction**
  - Three main features were extracted: pitch, velocity, and duration.
  - For MIDI files, these features were extracted using pretty_midi.

- For JSONL files, these features were either directly provided as values or calculated when necessary (e.g., duration calculation by subtracting a note's start time from its end time).

As we use pitch, velocity, and duration in this study, so to understand the nature of the dataset, here are the analytics of the global statistics from all data for each feature.

- **Pitch** ranged from 36 to 93, with the mean of 65.77 and standard deviation of 8.39.
- **Velocity** ranged from 24 to 127 with the mean of 95.79 and standard deviation of 10.74.
- **Duration** ranged from 0.0018s to 36.5s with the mean of 0.62s and standard deviation of 0.46s.

These statistical data confirm that all three features carry meaningful variance and are suitable for downstream learning tasks.

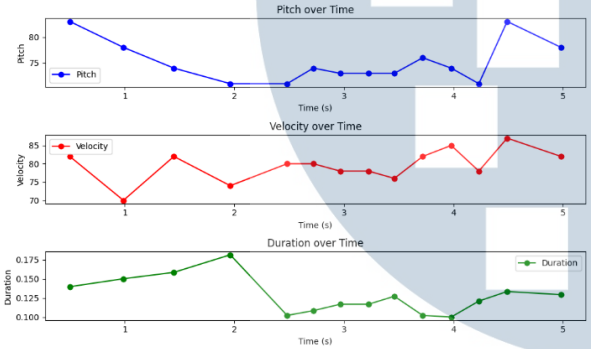

Figure 3: From top to bottom – Pitch, velocity, and duration trends in the first 5 seconds of a MIDI file.

Figure 3 visualizes the features' distribution example of one of the MIDI files over time in the first five seconds. Since these were sequential features, each was further processed to compute its mean, median, and standard deviation, resulting in a total of 9 features per segment. This statistical representation helps capture both the central tendencies and variations in musical patterns. Such statistical summarization methods have been shown to effectively represent musical data in symbolic music analysis tasks [38]. Mean values provide an overall summary of the feature distribution, the median helps mitigate the influence of extreme values, and the standard deviation quantifies the degree of variation within a segment.

5.3 Data pre-processing

The preprocessing process in this study was relatively brief. First, the extracted feature data was split into three sets with a 70:20:10 ratio for the training, validation, and test sets. This proportion was chosen to ensure the model had sufficient data to learn meaningful patterns (3,500 samples) while also allocating enough for validation (1,000 samples) to fine-tune hyperparameters and for testing (500 samples) to evaluate performance. Given the dataset size of 5,000 samples, this split was considered a practical balance between model training and reliable evaluation. After separating the dataset, normalization was applied to standardize the extracted features (pitch, velocity, and duration), as they had different value ranges. While note velocity can often be imbalanced or noisy, this study did not apply explicit outlier removal. Instead, standardization was considered sufficient to reduce the influence of extreme values while preserving expressive dynamics, as the earlier analysis showed a reasonable velocity distribution. Features with zero variance were identified within the training set and ignored from further processing since their values remained constant and did not require normalization. Using a standard scaler, each feature was normalized to have a mean value of 0 and a standard deviation of 1. Finally, the data was reshaped into a 3D format to match the input requirements of the LSTM and CNN architecture, which consists of these data:

- **Samples** – The number of samples.
- **Time steps** – The number of segments in the piece.
- **Features** – The number of features in each time step.

5.4 Model implementation

This study utilized two different models, one based on LSTM and the other on CNN.

5.4.1 LSTM base model

The LSTM-based model consists of four different layers, including the input, LSTM, dropout, and dense layers. Then to lessen overfitting, L2 regularization, often referred to as the Ridge penalty, was also applied to the LSTM and dense layers. L2 regularization encourages the model to control extreme weight values by penalizing the loss function when the weights get too large. By maintaining smaller and well-balanced weights, the model reduces dependence on specific features, enhancing its generalization ability and reducing overfitting.

The number of hidden units in the LSTM layer, dropout rate, and other architectural parameters were optimized through hyperparameter tuning. The search ranges and selected values are detailed in the hyperparameter tuning section.

5.4.2 CNN base model

While a basic CNN model typically consists of four main layers, the CNN-based model used in this study extends this structure to include six layers to improve performance and generalization. These six layers are input, Conv1D, MaxPooling1D, flatten, dropout, and dense.

The input layer serves as the entry point for the segmented sequential data. Then, by employing ReLU activation to extract local patterns, the Conv1D layer helps the model learn complicated features effectively while avoiding the vanishing gradient issue. MaxPooling1D then downsamples the feature maps, reducing computational complexity and preserving essential information. The Flatten layer then converts the pooled feature maps into a one-dimensional vector, preparing the data for the fully connected layers. L2 regularization is then applied in the convolutional and dense layers to

prevent overfitting. The dropout layer improves generalization by randomly deactivating neurons. Lastly, the dense layer with sigmoid activation classifies classical music as AI-generated or human-generated.

The specific architectural parameters such as the number of filters, kernel size, dropout rate, and L2 regularization strength were not fixed but optimized using a hyperparameter tuning process. The selected values and their respective search ranges are detailed in the hyperparameter tuning section.

5.5 Hyperparameter tuning

The training phase involved fine-tuning by utilizing different hyperparameter combinations. But first, to find the ideal hyperparameter configuration for the model, hyperparameter tuning was done using the training set. Hyperparameter tweaking by hand is a laborious, costly, and time-consuming process. Thus, this study used an automated approach offered by Keras Tuner, which is Bayesian Optimization. This hyperparameter tuning algorithm utilizes a probabilistic function to learn from past outcomes, allowing it to forecast and determine the next hyperparameter combination that is most likely to achieve optimal performance.

Here, a multi-objective hyperparameter tuning approach was employed to optimize two key metrics: minimizing validation loss to reduce overfitting and maximizing validation accuracy to enhance model accuracy. Although these goals can sometimes conflict, with improvements in one not always leading to improvements in the other, the tuning process aimed to find a balanced trade-off. Each trial was evaluated based on a combined consideration of both metrics, rather than optimizing one in isolation. The model that achieved the most favorable balance between validation loss and accuracy was selected as the optimal configuration for further training. Before the tuning process began, 25 initial trials were conducted randomly to establish a baseline for Bayesian Optimization. These initial trials provided the algorithm with preliminary insights, enabling it to make more informed predictions in subsequent iterations. The hyperparameter tuning process itself was limited to 25 trials, as Bayesian Optimization efficiently learns from past results, reducing the need for extensive experimentation and conserving computational resources.

During hyperparameter tuning, each trial was set to run for up to 80 epochs. To prevent unnecessary computations and overfitting, early stopping was implemented, allowing training to halt at the optimal epoch. This early stopping strategy is based on validation loss, which serves as the main indicator for deciding when to end training. Although both validation loss and validation accuracy are monitored throughout the process, only the validation loss is used to trigger early stopping. This ensures that the model continues to improve in generalization and avoids overfitting. The method follows these steps.

1. **Monitoring**
   Validation loss were tracked starting from at least three epochs.
2. **Best checkpoint**
   - Once the epoch with the lowest validation loss is identified, training continues for three additional epochs.
   - If no improvement occurs, the model reverts to the best checkpoint and stops.
   - If a better epoch is found, the process resets, extending evaluation by another three epochs.
3. **Repetition**
   The cycle repeats until no further improvements are observed, ensuring optimal performance.

The best-performing model and its corresponding hyperparameter configuration are stored. Tables 2 and 3 present the hyperparameter attributes, ranges, and optimal values identified through the tuning process for each model. The selected hyperparameter ranges, such as the number of units, dropout rates, and kernel sizes, were based on common practices reported in some prior studies across similar domains, ensuring a balanced search space without overly complicating the tuning process.

Table 2: LSTM hyperparameter tuning results.

Hyperparameter	Range	Best value
lstm_units	[16, 32, 64]	64
lstm_regularizer	0.005 - 0.05	0.015
dropout_rate	0.1 – 0.5	0.4
dense_regularizer	0.005 - 0.05	0.01
optimizer	[adam, rmsprop]	rmsprop
batch_size	[32, 64]	32

Table 3: CNN hyperparameter tuning results.

Hyperparameter	Range	Best value
filters	[16, 32, 64]	64
kernel_size	[3, 5]	3
cnn_regularizer	0.005 - 0.05	0.005
dropout_rate	0.1 – 0.5	0.2
dense_regularizer	0.005 - 0.05	0.005
optimizer	[adam, rmsprop]	adam
batch_size	[32, 64]	32

5.6 Fine-Tuning

Following the hyperparameter tuning process, the best-performing model from the optimal trial has been selected as the base model. Fine-tuning was then performed to further enhance the model's generalization performance on unseen data using the prepared validation set for monitoring. Unlike hyperparameter tuning, which explores multiple configurations, fine-tuning focuses on refining the training of the selected model while preserving its architecture. The number of epochs was reduced to 30 based on insights from the tuning stage, where top-performing models consistently achieved their best results by around the 30th epoch. This reduction helps prevent overfitting and streamlines training without

compromising performance. The batch size used matched the optimal value identified during tuning. Early stopping was also applied to monitor validation loss and halt training when no further improvement was observed. These focused adjustments allowed the model to converge more effectively using the best available settings, enhancing performance while minimizing overfitting risk.

5.7 Implementation and Replication Details

To support reproducibility and ensure that the experiments can be replicated by other researchers, this section outlines the software environment, core libraries, dataset sources, and code availability. All experiments were conducted on Google Colab's cloud computing platform, which provided a 2-core Intel(R) Xeon(R) CPU @ 2.20GHz and 13.61 GB of RAM. This configuration was sufficient for training and evaluating the deep learning models used in this study.

The models were implemented using the following Python libraries and tools.

- NumPy v2.0.2
- Pandas v2.2.2
- Matplotlib v3.10.0
- TensorFlow v2.18.0
- Keras v3.8.0
- Keras Tuner v1.4.7
- Scikit-learn v1.6.1
- pretty_midi v0.2.10

The dataset used in this study was compiled from the following publicly available sources.

- Hugging Face dataset [34]: https://huggingface.co/datasets/drengskapur/midi-classical-music
- JS Fake Chorales dataset [35]: https://github.com/omarperacha/js-fakes
- Bach Doodle dataset by Magenta [36]: https://magenta.tensorflow.org/datasets/bach-doodle

All preprocessing scripts, model training pipelines, and hyperparameter tuning configurations are available in the accompanying code repository https://github.com/maecyntha/ai-classical-music-detector.

5.8 Evaluation

In this study, the model was evaluated by using it to make predictions on the test set that was previously split during data preparation. A confusion matrix was then used to further analyze its performance. To validate the model reliability, it was also further tested on an auxiliary test set consisting of 1,000 samples from a different dataset. The confusion matrix for this auxiliary test set was also examined to assess the model's generalization ability.

6 Results and analysis

After undergoing multiple stages of processing, the following section presents a detailed analysis of the results—beginning with hyperparameter tuning, followed by fine-tuning, and concluding with the final evaluation. The performance results are organized into two subsections, each focusing separately on the LSTM and CNN models, followed by a final subsection that compares the performance of both models.

6.1 LSTM model performance

This subsection presents the performance analysis of the LSTM model, covering hyperparameter tuning, fine-tuning, and final evaluation.

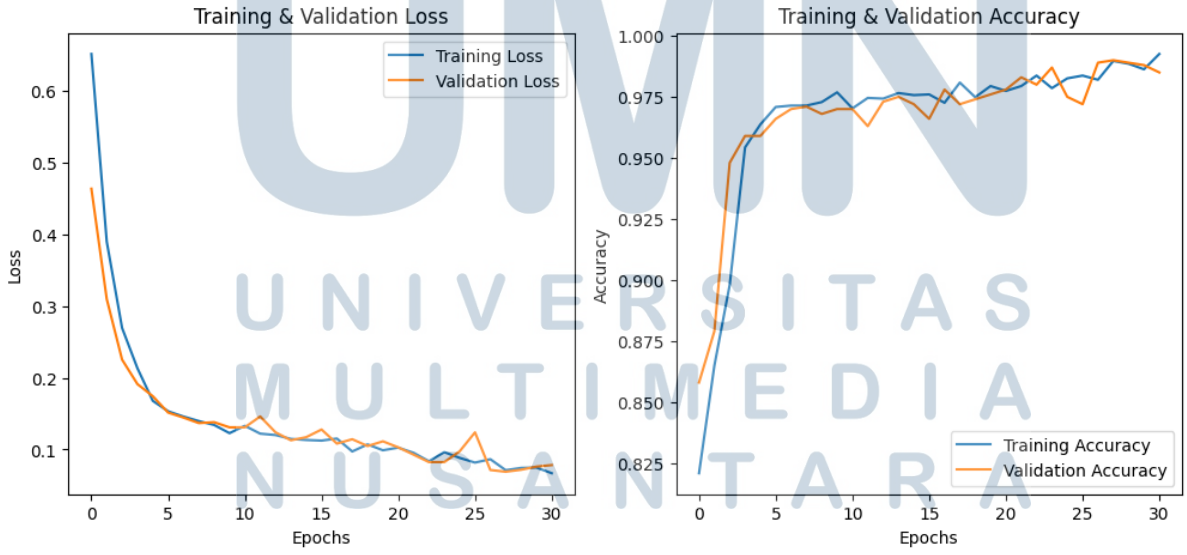

Figure 4: LSTM-based training & validation loss (left) and accuracy (right) over epochs.

Figure 4 illustrates the patterns in training and validation accuracy and loss throughout 31 epochs using LSTM-based model. As can be seen in the left plot, both curves exhibit a downward trend, indicating that the model progressively improves its predictions as training progresses, resulting in a lower loss. Despite minor

fluctuations in validation loss, its close alignment with training loss suggests minimal overfitting. On the other hand, the right plot, representing the accuracy trends, demonstrates a consistent increase in accuracy, with a sharp rise between epochs 5 and 10 before stabilizing around 97%–98%. This indicates that the model learns effectively in the early stages and refines its predictions as training progresses. The overall results confirm that the chosen hyperparameters facilitate efficient learning, as reflected in the significant early improvements, stable high accuracy, and consistent reduction in loss. The lack of a widening gap between training and validation metrics suggests the model generalizes well, maintaining robust performance on unseen validation data.

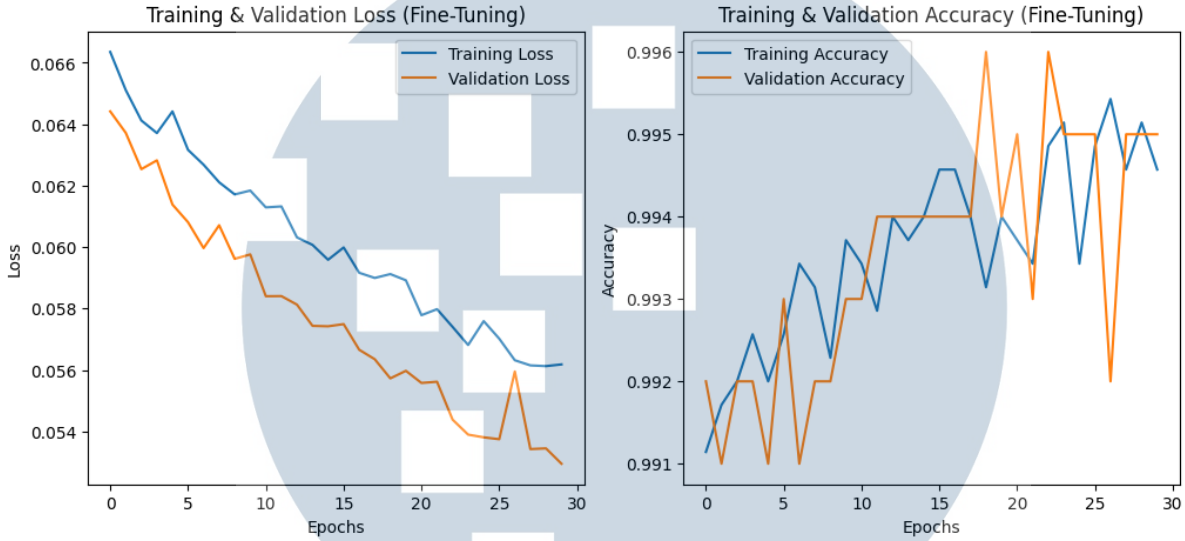

Figure 5: Fine-tuned LSTM-based training & validation loss (left) and accuracy (right) over epochs.

Figure 5 presents the performance of the fine-tuned model. In the left plot, which depicts training and validation loss, both losses continue to decrease steadily over 30 epochs. However, unlike the previous training phase during hyperparameter tuning, the two curves are not always closely aligned, suggesting slight discrepancies between the model's performance on training and validation data. Despite this, the overall downward trend indicates effective fine-tuning without severe overfitting. Meanwhile, the right plot, showing training and validation accuracy, exhibits noticeable fluctuations, particularly in validation accuracy, which sometimes spikes or drops significantly. This variability suggests sensitivity to different validation batches. However, the general trend remains consistent with the training accuracy, maintaining a high range between 99.1% and 99.6%. These results indicate that while fine-tuning enhances model performance, it introduces slight instability in validation accuracy. Nonetheless, the model maintains strong generalization, as reflected in the stable accuracy levels and continuous reduction in loss.

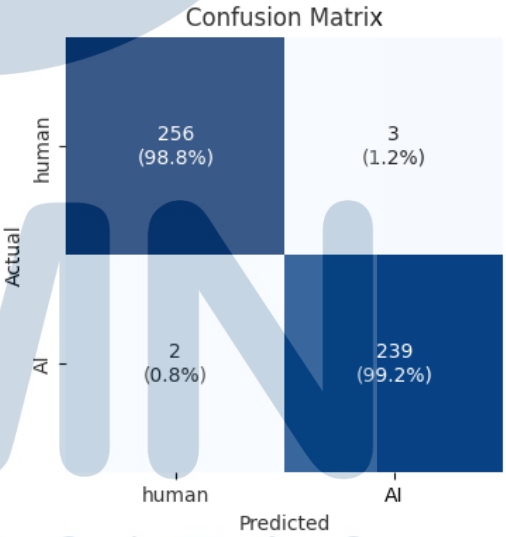

Figure 6: Confusion matrix of the LSTM-based model's predictions on the test set.

The model was then subsequently assessed using a specific test set prepared beforehand. The confusion matrix in Figure 6 illustrates that the model has a minimum number of false positives and false negatives, indicating strong predictive accuracy. This is further supported by the classification report, where the precision of 99.22% reflects the model's ability to make highly accurate predictions for both categories. A recall score of 99.01% confirms its effectiveness in minimizing false negatives. Furthermore, the F1 score and overall accuracy of 99.00% demonstrate a well-balanced performance

between precision and recall, ensuring reliable classification of both human and AI-generated compositions.

To statistically validate this performance, a one-sided proportion z-test was conducted to evaluate whether the model's accuracy of 99.00% was significantly better than random guessing (50%). The resulting z-score was approximately 69.296, with a p-value of less than 0.00001. Since the p-value is far below the commonly accepted significance level of 0.05, we can confidently reject the null hypothesis and conclude that the model's accuracy is statistically and significantly better than random classification. The entire training and validation process took approximately 21 minutes to complete.

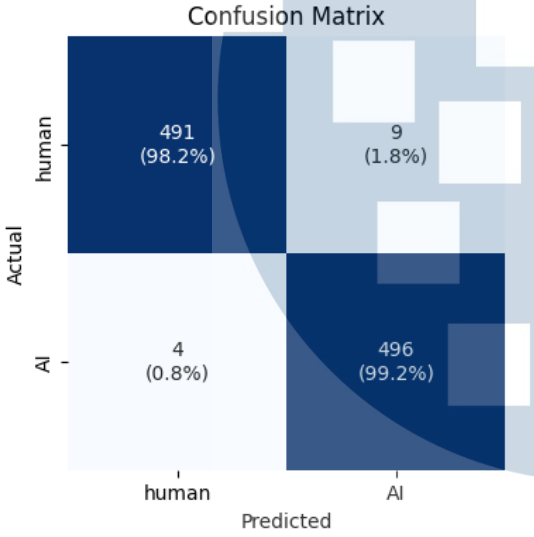

Figure 7: Confusion matrix of the LSTM-based model's predictions on the auxiliary test set.

The model was further validated using an auxiliary test set of 1,000 samples from a different dataset to assess its reliability. As shown in Figure 7, the confusion matrix reveals misclassification rates of 1.8% for human compositions and 0.8% for AI-generated compositions. With an overall accuracy of 98.70%, the model also achieves 98.70% in precision, recall, and F1 score, indicating a well-balanced performance. These results confirm that the model generalizes effectively to unseen data, maintaining high accuracy while minimizing errors, further reinforcing its reliability in distinguishing between human and AI-generated classical music.

6.2 CNN model performance

This subsection provides the performance analysis of the CNN model, following the same evaluation stages as the LSTM model, from hyperparameter tuning to final evaluation.

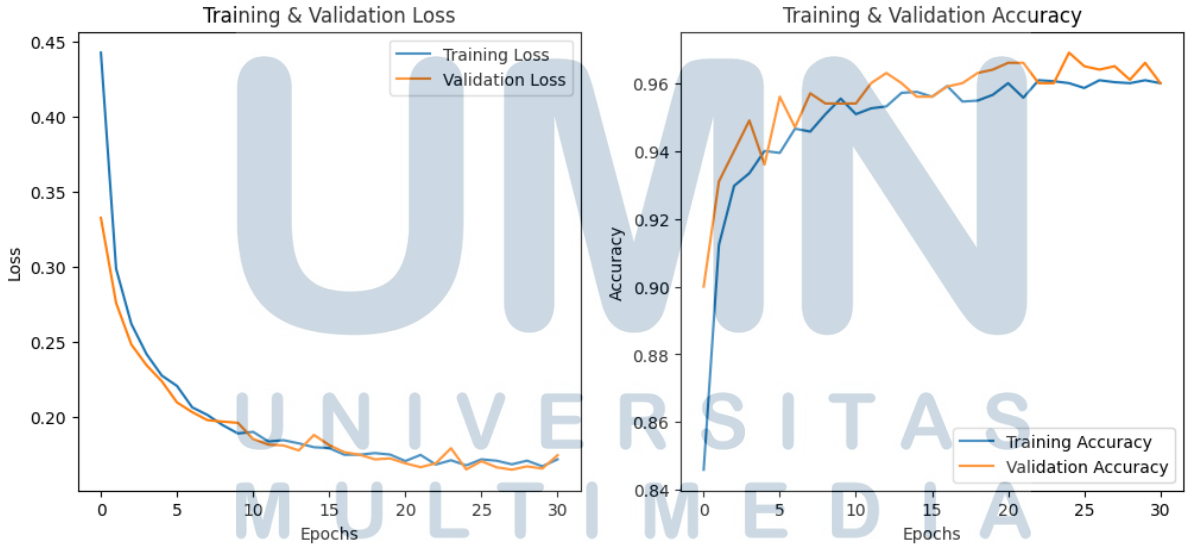

Figure 8: CNN-based training & validation loss (left) and accuracy (right) over epochs.

Figure 8 shows the CNN-based model's training and validation results across 31 epochs. The left plot shows a steady decline in training and validation loss, dropping from approximately 0.45 to below 0.20, indicating effective learning. Despite minor fluctuations, the validation loss remains closely aligned with the training loss, suggesting minimal overfitting. The plot on the right shows a steady rise in accuracy, improving quickly in the early epochs before leveling off at about 96% in the later phases. The consistent gap between training and validation accuracy further supports the model's strong generalization. These outcomes demonstrate how well the selected hyperparameters optimize the CNN model's learning process.

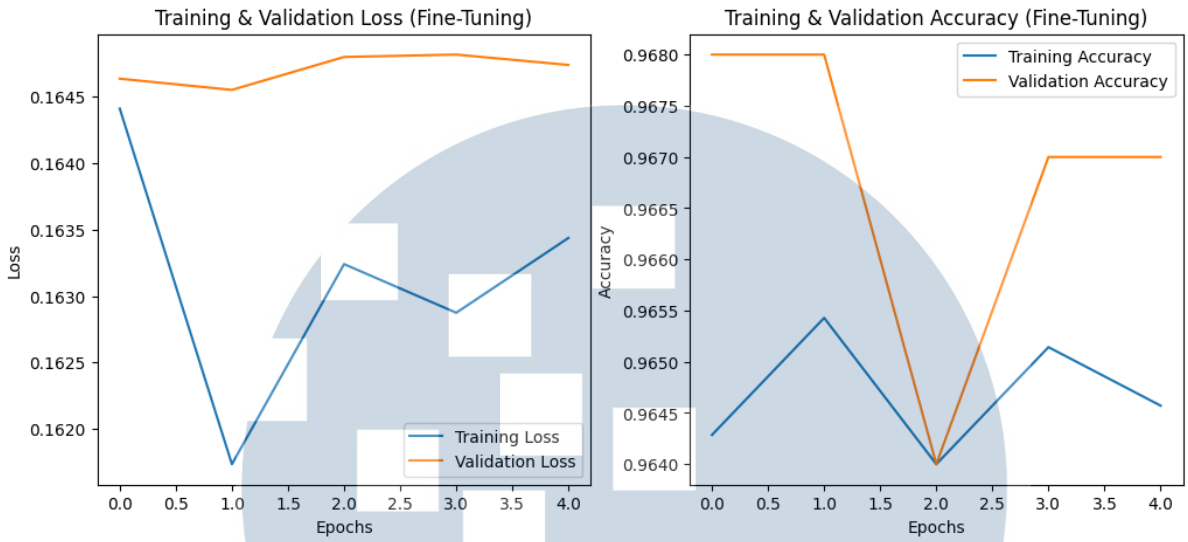

Figure 9: Fine-tuned CNN-based training & validation loss (left) and accuracy (right) over epochs.

Figure 9 illustrates the CNN model's performance during fine-tuning over five epochs. While validation loss stays comparatively constant at 0.1645, suggesting little improvement, training loss varies but tends to decline, as seen in the left plot. On the other hand, the right plot reveals significant accuracy variations, especially in validation accuracy, which momentarily declines at epoch 2 before leveling off at about 96.7%. These fluctuations imply that the model is sensitive to minor tweaks during fine-tuning.

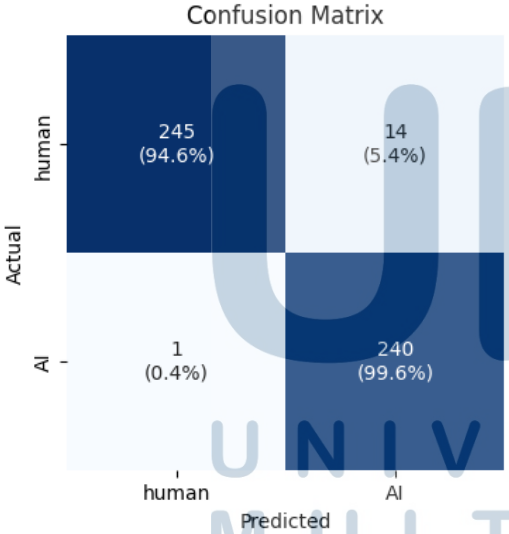

Figure 10: Confusion matrix of the CNN-based model's predictions on the test set.

Figure 10 displays the confusion matrix of the CNN model on the primary test set. With a high prediction accuracy, the model correctly identifies most examples; however, it misidentifies 14 human compositions as AI-generated and one AI-generated composition as human. With a precision of 97.04% and a recall of 97.09%, the model demonstrates a balanced ability to differentiate the two groups. The model's durability in maintaining good classification performance is demonstrated by its 97.00% overall accuracy and F1 score.

To confirm that the CNN model's 97.00% accuracy wasn't due to random chance, we performed a one-sided proportion z-test against a 50% baseline representing random guessing. The test returned a z-score of approximately 66.468 and a p-value far below 0.00001. Since this p-value is well below the standard threshold of 0.05, we rejected the null hypothesis, showing that the CNN model's performance is statistically significant. The complete model training and validation ran for approximately six minutes.

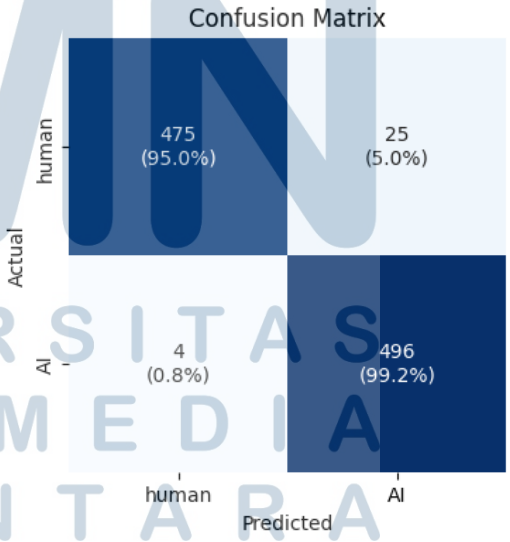

Figure 11: Confusion matrix of the CNN-based model's predictions on the auxiliary test set.

The confusion matrix for the CNN model that was evaluated using the extra dataset is shown in Figure 11. Despite misclassifying some pieces, the model continues

to exhibit good classification performance. The model achieves an overall accuracy of 97.10%, with a precision of 97.18%, a recall score of 97.10%, and an F1 score of 97.10%. These findings support the model's high reliability in distinguishing compositions created by humans and AI, as well as its capacity to generalize fresh data.

6.3 Comparative analysis

To provide a clearer comparison, Table 4 presents the classification performance of the LSTM and CNN models on the primary dataset. The LSTM model outperforms CNN in every metrics, with an accuracy of 99.00% as opposed to 97.00% for CNN. Additionally, LSTM retains a higher F1 score due to its superior precision and recall compared to CNN's. These results indicate that while both models perform well, the LSTM model demonstrates superior classification capability, likely due to its ability to capture long-term dependencies in sequential data, which is crucial for distinguishing between AI-generated and human-generated classical music.

Table 4: Comparison of classification performance metrics between LSTM and CNN.

Model	Accuracy	Precision	Recall	F1 Score
LSTM	99.00%	99.22%	99.01%	99.00%
CNN	97.00%	97.04%	97.09%	97.00%

7 Discussion

In this study, we achieve higher classification accuracy compared to previous works. Our first finding reinforces the effectiveness of LSTM-based models in capturing sequential dependencies in musical compositions. LSTM-based models are particularly well-suited for symbolic music analysis because they are designed to model temporal dependencies over long sequences. This aligns with the nature of symbolic music, where patterns unfold over time and require memory of prior context. In contrast, CNN primarily focuses on capturing local patterns through convolutional filters and may struggle to retain the broader sequential context necessary for interpreting compositional flow, especially in music where long-range structure and phrasing are key to stylistic identity.

Previously, Li et al. [23] have also proved that LSTMs effectively distinguish AI-generated melodies from human compositions, achieving high classification performance despite ignoring velocity due to dataset limitation. On the other hand, our approach includes velocity alongside pitch and duration, resulting in a more comprehensive representation of musical expression. This richer feature set appears to improve the model's ability to distinguish AI-generated classical music from that composed by humans.

Other than feature selection, our approach also differs in the representation of musical structures. Kong et al. [25] used piano rolls which rely on fixed time grids that divide time into uniform intervals regardless of the underlying musical context. This can lead to a misalignment between the grid and expressive timing in performances, especially when dealing with tempo changes, or uneven note spacing. On the other hand, our study uses beat-based segmentation which groups events based on musical beats rather than uniform time slices. This approach aligns more naturally with the timing and phrasing of music, allowing for a more accurate capture of rhythmic and expressive characteristics. As a result, it better preserves sequential dependencies that are important for classification, which may be blurred in fixed-grid representations.

Furthermore, while studies such as Deepak et al. [24] have explored deep learning models for symbolic music classification, our results indicate that temporal modeling remains crucial when the goal is to detect AI-generated classical music. The strong performance of our model emphasizes the significance of capturing long-term dependencies in musical sequences.

The CNN-based model utilized in this study aligns with the work of Afchar et al. [22], who used convolutional networks for AI-generated music recognition. However, whereas their study focused on amplitude-related audio features, our study, which focuses on symbolic music analysis using MIDI data, achieves higher accuracy. This distinction illustrates an intriguing difference between the two approaches: audio-based methods capture performance characteristics and timbral properties, whereas symbolic music analysis separates compositional structures, eliminating variations caused by recording conditions. Despite these differences, our CNN model produces competitive results, indicating that convolutional networks can effectively learn differentiating patterns from structured symbolic representations.

These findings from both LSTM and CNN models suggest that other than the choice of model architecture, the way musical data is structured and represented significantly impacts the effectiveness of AI-generated classical music detection.

8 Conclusion

In this study, a detection system made to distinguish AI-generated and human-generated classical music was successfully approached using LSTM and CNN algorithms. Through the experimental analysis, it was observed that both models demonstrated strong classification performance by achieving high accuracy, precision, recall, and F1 scores across multiple test sets. However, the results shows that LSTM performs better than CNN with minimal misclassification rates, indicating that the model is capable of effectively differentiating between the two composition types. This shows LSTM's superior ability to capture sequential dependencies, which is crucial in this detection task. Further validation on an auxiliary test set, which maintained a high level of accuracy, confirmed its robustness, suggesting strong generalization to unseen data.

Despite these promising results, some limitations remain. We acknowledge that training the model on a dataset representing only a single compositional style, specifically the works of J.S. Bach, may introduce

potential bias and limit the model's generalizability to broader musical genres. Additionally, the dataset used in this study may not fully capture the diversity of musical styles and compositions, potentially limiting generalizability to more complex or unconventional pieces. To address these limitations, future research could expand the dataset to include compositions from a wider range of composers or styles. For improved robustness, incorporating AI-generated music that emulates contemporary, jazz, or pop artists may help capture cross-genre characteristics. Furthermore, integrating more advanced musical attributes, including harmonic, timbral, and structural features, may also refine classification performance further. Eventually, these improvements would enhance the model's versatility and reliability, paving the way for more comprehensive AI-driven music classification. Finally, while ROC curves and AUC scores were not included in this study due to the absence of probability outputs, future models with probabilistic predictions could leverage these tools for more nuanced performance evaluation.

Acknowledgement

The authors acknowledge the support of Universitas Multimedia Nusantara for this study and appreciate the constructive feedback provided by the reviewers and editors, which significantly contributed to improving the quality of this paper.

References

[1] M. Civit, J. Civit-Masot, F. Cuadrado, and M. J. Escalona, "A systematic review of artificial intelligence-based music generation: Scope, applications, and future trends," *Expert Syst Appl*, vol. 209, p. 118190, Dec. 2022, doi: https://doi.org/10.1016/j.eswa.2022.118190.

[2] F. Pachet, P. Roy, and B. Carré, "Assisted Music Creation with Flow Machines: Towards New Categories of New," in *Handbook of Artificial Intelligence for Music*, Cham: Springer International Publishing, 2021, pp. 485–520. doi: 10.1007/978-3-030-72116-9_18.

[3] M. Fox, G. Vaidyanathan, and J. Breese, "THE IMPACT OF ARTIFICIAL INTELLIGENCE ON MUSICIANS," *Issues In Information Systems*, vol. 25, no. 3, pp. 267–276, 2024, doi: 10.48009/3_iis_2024_121.

[4] P. Jurcys and M. Fenwick, "Originality and the Future of Copyright in an Age of Generative AI," *Computer Law & Security Review*, Sep. 2023.

[5] N. Lucchi, "ChatGPT: A Case Study on Copyright Challenges for Generative Artificial Intelligence Systems," *European Journal of Risk Regulation*, vol. 15, no. 3, pp. 602–624, Sep. 2024, doi: 10.1017/err.2023.59.

[6] E. Bonadio and L. Mcdonagh, "Artificial intelligence as producer and consumer of copyright works: evaluating the consequences of algorithmic creativity," pp. 112–137, Jun. 2020, [Online]. Available: https://ssrn.com/abstract=3617197

[7] L. Bellaiche *et al.*, "Humans versus AI: whether and why we prefer human-created compared to AI-created artwork," *Cogn Res Princ Implic*, vol. 8, no. 1, p. 42, Jul. 2023, doi: 10.1186/s41235-023-00499-6.

[8] A. M. Elkhatat, K. Elsaid, and S. Almeer, "Evaluating the efficacy of AI content detection tools in differentiating between human and AI-generated text," *International Journal for Educational Integrity*, vol. 19, no. 1, p. 17, Sep. 2023, doi: 10.1007/s40979-023-00140-5.

[9] A. Akram, "An Empirical Study of AI Generated Text Detection Tools," *Advances in Machine Learning & Artificial Intelligence*, vol. 4, no. 2, pp. 44–55, Oct. 2023, doi: 10.33140/AMLAI.

[10] R. and S. R. S. and D. G. P. and B. N. and S. S. Tiwari Shreeji and Sharma, "Detecting AI Generated Content: A Study of Methods and Applications," in *Proceedings of International Conference on Communication and Computational Technologies*, S. and G. R. and P. S. D. Kumar Sandeep and Hiranwal, Ed., Singapore: Springer Nature Singapore, 2024, pp. 161–176. doi: 10.1007/978-981-97-7423-4_13.

[11] J. J. Bird and A. Lotfi, "CIFAKE: Image Classification and Explainable Identification of AI-Generated Synthetic Images," *IEEE Access*, vol. 12, pp. 15642–15650, 2024, doi: 10.1109/ACCESS.2024.3356122.

[12] P. Tiwari and S. Jha, "Music Generation with Long Short-Term Memory Networks from MIDI Data of Classical Music," in *2024 IEEE International Conference on Information Technology, Electronics and Intelligent Communication Systems (ICITEICS)*, 2024, pp. 1–4. doi: 10.1109/ICITEICS61368.2024.10625468.

[13] G. G. N. S and V. V. P. D, "Generating Creative Classical Music by Learning and Combining Existing Styles," in *2023 4th International Conference on Communication, Computing and Industry 6.0 (C216)*, 2023, pp. 1–7. doi: 10.1109/C2I659362.2023.10431294.

[14] M. Zhu, "Research on Chord Generation in Automated Music Composition Using Deep Learning Algorithms," *Informatica*, vol. 47, no. 8, Sep. 2023, doi: 10.31449/inf.v47i8.4885.

[15] F. Shah, T. Naik, and N. Vyas, "LSTM Based Music Generation," in *2019 International Conference on Machine Learning and Data Engineering (iCMLDE)*, 2019, pp. 48–53. doi: 10.1109/iCMLDE49015.2019.00020.

[16] D. R. Fudholi, D. N. A. Putri, R. B. M. A. A. Wijaya, J. E. Kusnadi, and J. C. Amarissa, "The Application of LSTM in the AI-Based Enhancement of Classical Compositions," *Journal of Informatics, Information System, Software Engineering and Applications (INISTA)*,

vol. 7, no. 1, pp. 107–117, Nov. 2024, doi: 10.20895/INISTA.V7I1.1628.

[17] M. Ahmad, M. Mazzara, and S. Distefano, "Regularized CNN Feature Hierarchy for Hyperspectral Image Classification," *Remote Sens (Basel)*, vol. 13, no. 12, 2021, doi: 10.3390/rs13122275.

[18] E. Dervakos, N. Kotsani, and G. Stamou, "Genre Recognition from Symbolic Music with CNNs: Performance and Explainability," *SN Comput Sci*, vol. 4, no. 2, p. 106, 2022, doi: 10.1007/s42979-022-01490-6.

[19] F. Simonetta, C. E. Cancino-Chacón, S. Ntalampiras, and G. Widmer, "A Convolutional Approach to Melody Line Identification in Symbolic Scores ," in *Proceedings of the 20th International Society for Music Information Retrieval Conference* , ISMIR, Nov. 2019, pp. 924–931. doi: 10.5281/zenodo.3527966.

[20] L. Pan and H. Ma, "A Computational CNN-LSTM-Based Mental Health Consultation System in a College Environment," *Informatica*, vol. 49, no. 10, Jan. 2025, doi: 10.31449/inf.v49i10.7136.

[21] I. A. Abdulmajeed and I. M. Husien, "MLIDS22-IDS Design by Applying Hybrid CNN-LSTM model on Mixed-Datasets," *Informatica*, vol. 46, no. 8, Nov. 2022, doi: 10.31449/inf.v46i8.4348.

[22] D. Afchar, G. Meseguer-Brocal, and R. Hennequin, "AI-Generated Music Detection and its Challenges," in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025, pp. 1–5. doi: 10.1109/ICASSP49660.2025.10890655.

[23] Y. Li and Z. Lin, "Melody Classifier with Stacked-LSTM," Oct. 2020.

[24] S. Deepak and B. G. Prasad, "Music Classification based on Genre using LSTM," in *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, 2020, pp. 985–991. doi: 10.1109/ICIRCA48905.2020.9182850.

[25] Q. Kong, K. Choi, and Y. Wang, "Large-Scale MIDI-based Composer Classification," Oct. 2020.

[26] N. Shenvi and H. Virani, "Forecasting of Ionospheric Total Electron Content Data Using Multivariate Deep LSTM Model for Different Latitudes and Solar Activity," *Journal of Electrical and Computer Engineering*, vol. 2023, pp. 1–13, May 2023, doi: 10.1155/2023/2855762.

[27] A. Liu, "Multi-genre Digital Music Based on Artificial Intelligence Automation Assisted Composition System," *Informatica*, vol. 48, no. 5, Feb. 2024, doi: 10.31449/inf.v48i5.5474.

[28] X. Jin *et al.*, "Time series forecasting of Valley fever infection in Maricopa County, AZ using LSTM," *The Lancet Regional Health - Americas*, vol. 43, p. 101010, Mar. 2025, doi: 10.1016/j.lana.2025.101010.

[29] A. Moghar and M. Hamiche, "Stock Market Prediction Using LSTM Recurrent Neural Network," *Procedia Comput Sci*, vol. 170, pp. 1168–1173, 2020, doi: 10.1016/j.procs.2020.03.049.

[30] H. Fan, M. Jiang, L. Xu, H. Zhu, J. Cheng, and J. Jiang, "Comparison of Long Short Term Memory Networks and the Hydrological Model in Runoff Simulation," *Water (Basel)*, vol. 12, no. 1, p. 175, Jan. 2020, doi: 10.3390/w12010175.

[31] Y. Liu, H. Pu, and D.-W. Sun, "Efficient extraction of deep image features using convolutional neural network (CNN) for applications in detecting and analysing complex food matrices," *Trends Food Sci Technol*, vol. 113, pp. 193–204, Jul. 2021, doi: 10.1016/j.tifs.2021.04.042.

[32] S. Lee, J. Kim, H. Kang, D.-Y. Kang, and J. Park, "Genetic Algorithm Based Deep Learning Neural Network Structure and Hyperparameter Optimization," *Applied Sciences*, vol. 11, no. 2, 2021, doi: 10.3390/app11020744.

[33] A. Stamoulakatos *et al.*, "A Comparison of the Performance of 2D and 3D Convolutional Neural Networks for Subsea Survey Video Classification," in *OCEANS 2021: San Diego – Porto*, 2021, pp. 1–10. doi: 10.23919/OCEANS44145.2021.9706125.

[34] Anonymous, "MIDI Classical Music." Accessed: Jan. 16, 2025. [Online]. Available: https://huggingface.co/datasets/drengskapur/midi-classical-music

[35] O. Peracha, "JS Fake Chorales: a Synthetic Dataset of Polyphonic Music with Human Annotation," in *Proceedings of the 2022 Sound and Music Computing Conference, SMC 2022*, 2022. doi: 10.48550/arXiv.2107.10388.

[36] C.-Z. A. Huang, C. Hawthorne, A. R. and M. Dinculescu, J. Wexler, L. Hong, and J. Howcroft, "The Bach Doodle: Approachable music composition with machine learning at scale," in *International Society for Music Information Retrieval (ISMIR)*, 2019. [Online]. Available: https://goo.gl/magenta/bach-doodle-paper

[37] A. Marmoret, J. E. Cohen, and F. Bimbot, "Barwise Music Structure Analysis with the Correlation Block-Matching Segmentation Algorithm," *Transactions of the International Society for Music Information Retrieval*, vol. 6, no. 1, pp. 167–185, Nov. 2023, doi: 10.5334/tismir.167.

[38] L. Jing, "Evolutionary Deep Learning for Sequential Data Processing in Music Education," *Informatica*, vol. 48, no. 8, May 2024, doi: 10.31449/inf.v48i8.5444.