A NOVEL DATASET AND DEEP LEARNING BENCHMARK FOR CLASSICAL MUSIC FORM RECOGNITION AND ANALYSIS

Daniel Szelogowski **Lopamudra Mukherjee** **Benjamin Whitcomb**
University of Wisconsin — Whitewater
{szelogowdj19,mukherjl,whitcomb}@uww.edu

ABSTRACT

Automated computational analysis schemes for Western classical music analysis based on form and hierarchical structure have not received much attention in the literature so far. One reason, of course, is the paucity of labeled datasets — which, if available, could be used to train machine learning approaches. Dataset curation cannot be crowdsourced; one needs trained musicians to devote sizable effort to carry out such annotations. Further, such an analysis is not simple for beginners; obtaining labeled data that can capture the nuances of a musician's reasoning acquired over years of practice is fraught with challenges. To this end, we provide a system for computational analysis of classical music, both for machine learning and music researchers. First, we introduce a labeled dataset containing 200 classical music pieces annotated by form and phrases. Then, by leveraging this dataset, we show that deep learning-based methods can be used to learn Form Classification as well as Phrase Analysis and Classification, for which few (if any) results have been reported yet. Taken together, we provide the community with a unique dataset as well as a toolkit needed to analyze classical music structure, which can be used or extended to drive applications in both commercial and educational settings.

1. INTRODUCTION

Musical form analysis is the process of analyzing classical music pieces based on structure, themes, harmonies, and the relationship between them. This includes the large form (i.e., the musical "template" defining the piece's overall structure) and the hierarchical breakdown of themes, phrases, and often other substructures, including cadences. It has applications in various areas of music, such as music education, music analysis, forensic musicology, and so on, which we will discuss shortly. Typically, this task is carried out by music theorists benefiting from formalizations that have evolved over the centuries. However, form analysis is considered challenging, both for human analysts and signal processing algorithms. For humans, it takes years of training to learn to *understand* (not just read) classical music well enough to classify its structure. On the other hand, developing an ML-based approach is hindered by the difficulty of formulating this in a way that can be successfully learned as a computational task. Indeed, the curation of labeled datasets, a central ingredient in the success of supervised machine learning models, is prohibitive in terms of both time and money. First, it would require highly skilled musicians as annotators. Second, the dataset size is open-ended. With little guidance from the literature, it is also not obvious what sample sizes may be meaningful to get a basic model operational. Thus, budgeting is risky, even for a feasibility study, with assured cost overruns.

Motivation: The above challenges notwithstanding, let us consider the potential value for some key applications, *if* such a resource were publicly available.

Audio Thumbnail and Fingerprint Generation: Such a system can enable audio thumbnail/fingerprint generation for classical music where the user can quickly grasp the key sections without listening to the entire piece. This can be beneficial in marketing a piece of music as a product in a streaming service or web store (iTunes, Amazon Music) to draw in revenue for content creators [1–3].

Copyright Detection and Forensic Musicology: Such a system can also facilitate Forensic Musicology, which compares numerous pieces for similar or exact replications of musical phrases, motives, or other structures [1, 4].

Data Mining for Anthologies: Such a system can drive the production of musical form/analysis-based anthologies, alongside other fields of musicology where significant computational research is still in its early stages [5–7].

Music Education and Pedagogy: So far, music education and evaluation are purely human-guided. The availability of such a tool can facilitate the development of music practice and analysis software, such as dividing a piece by themes for rehearsal, assignment generation, or a grading system for human-analyzed scores. During a transitional time where many educational formats are moving to a hybrid setup, these developments will be a net win [8, 9].

Other potential applications also include Audio classification and Generalized Audio Structure Analysis.

Limitations of existing methods: In spite of the clearly defined need, existing structural analysis methods have been investigated almost exclusively in the context of popular music and for such tasks as segmentation of a piece into intro, verse, chorus, bridge, and outro [10]. These ideas cannot be directly applied to classical music: the for-

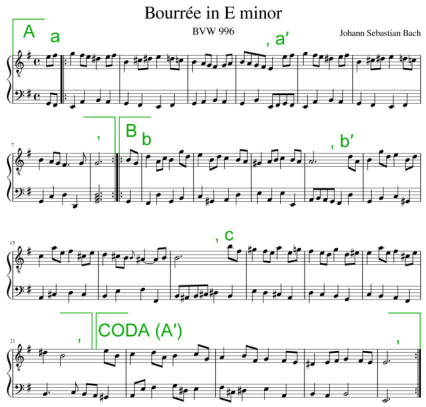

Figure 1. Example of phrase labeling from analysis of Bourrée from J.S. Bach's BWV 996 on a human-annotated score, where the relation between phrases and their respective part can be seen hierarchically. On an analyzed score, it is standard that only the first instance of a structure is labeled, although it may continue far beyond the initial instance.

mal structure in classical music is much more complicated and includes multiple forms which do not appear in popular music. Further, beyond form classification, one can also classify a musical piece into *phrases*, which typically correspond to smaller units within the piece. While classical music form analysis shares some similarities to poetic form at the *part/section* level(s) ($A\prime$ or stanza), which can be likened to popular music form (i.e., Verse-Chorus), the phrase-level analysis is drastically different due to the hierarchy of musical structures involved and a vast number of possible labels. In addition, classical music forms apply to the majority of pieces of classical music (given the large variety of large form structures). This means the form structure should be applicable to the audio (or the sheet music) by itself without the need for the segmentation of lyrical structure since a large majority of classical pieces are entirely instrumental or only partially lyrical.

Limitations of available datasets: The closest related dataset available, the Structural Analysis of Large Amounts of Music Information database, or SALAMI [11], lacks standardized conventions for the purpose of allowing for genre flexibility. It uses live recordings for time-relational analysis rather than basing timestamps on the sheet music or the score. This is not very useful for classical music since different conductors and performers often take drastically different tempi (which may be adjustable, but we seek the best overall system performance for this benchmark) and may omit or add repeats, therefore the audio cannot be compared to the sheet music directly. Hence, the use of this dataset for our goals is unfeasible.

Our Contributions: This paper provides a starting point for automated analysis of classical music forms and phrases. First, we introduce a new dataset, the **Standardized Musical Form and Structure Analysis (SMFSA) Database**, [1] consisting of 200 manually classified MIDIs, which use common analytical conventions and have categorical divisions by large musical forms. To demonstrate the use of the dataset, we also develop a deep learning-based framework to perform full form analysis, including form classification, segmentation, and part/phrase labeling. Together, this provides a comprehensive system for automated analysis of classical music, which is not otherwise available. This work provides the starting point for further development of computational techniques devoted to classical music as well as stimulating the development of numerous end-user applications.

[1] The database, research [12], and all code for this project can be found in [13]. The dataset was compiled from open sources, including [14–18].

2. RELATED WORK

While there are methods to perform genre classification, musical segmentation, and single-label segment classification in popular music, none have focused on the analytical process used by classical musicians specifically. These systems typically focus on popular music tasks including Verse-Chorus classification/segmentation [19] and genre classification [20], as well as segmenting a piece by phrases [21] — albeit without classifying. Next, we discuss a few related methods in chronological order.

Hörnel and Menzel [22] presented a melody and harmony generator using Feedforward Neural Networks to analyze musical and structural data in order to learn the characteristics of the writing style of a composer. However, their models were unable to learn higher-level musical structures occurring at multiple time scales simultaneously or recognize the melodic versus harmonic context of notes and intervals. Ponde de León and Iñesta [23] provided a framework for automatic musical style recognition of digital scores (MIDI) through the classification of rhythmic, harmonic, and melodic descriptors using k-Nearest Neighbors (k-NN), Bayesian classification, and Self-Organizing Maps (SOMs). Ullrich et. al. [24] discussed the importance of boundary recognition in structural music analysis using Convolutional Neural Networks (CNNs). Their model was trained on annotated Mel-scaled log-magnitude Spectrograms (MLS) [24] (from SALAMI) to peak-pick the onset boundaries of a given piece. Grill and Schlüter [25] further improved this model by assigning labels to digital audio using the MLS, Self-Similarity Lag Matrices (SSLMs), and human-annotated data (again from SALAMI). O'Brien [26] proposed an extension to the CNN architecture in [24] using the matrices derived from the Non-negative Matrix Factorization of a piece of music and identifying boundaries using segment association. O'Brien also noted that two major issues with their model were the lack of model memory in the CNN (i.e., the lack of LSTM or Recurrent cells) and the architecture had to be expanded to allow for a larger dataset [26]. De Berardinis et. al. [27] discussed the challenges of automatic musical structure detection and the issue of most current algorithms only being able to produce flat segmentations that cannot be applied to reveal the hierarchical structure of the piece. As such, they presented a new system for this task using multi-resolution community detection and graph theory to perform boundary detection and structural grouping, yielding a structural hierarchy. They noted that the method might also be used for structure visualization and finer-level musical structure analysis using tree representations to reflect additional structural relationships, and that CNNs will continue to lack improvement without recurrent layers or unsupervised methods.

3. METHODS

Next, we describe the key components of our model, starting with the dataset collection, the network architecture of the form analyzer, as well as a peak-picking scheme needed as input for the phrase analyzer.

3.1 Data Collection

We constructed a dataset — the **SMFSA Database** — consisting of 200 pieces of classical music in MIDI format [2] (for ease of score conversion, error correction, and signal processing). For each piece, we have an accompanying text document containing the form classification and part/phrase labels with their respective timestamps (obtained from the sheet music analysis performed by a human annotator), [3] written in a format similar to the SALAMI dataset. To represent the audio signals, the Mel Spectrogram was first generated by resampling the audio with a sample rate of 44.1 kHz, then computing the Short-Time Fourier Transform (STFT) over the entire signal with a hop length of 6144 (0.139 seconds per frame, see [28]) and 8192 samples per frame (a window size of 0.209 seconds per frame multiplied by the sampling rate). Further, to extract meaningful representations from the raw data, we obtain the 2D Self-Similarity Matrix (SSM) [25] of the Mel Spectrogram, constructed from various attributes and similarity metrics as well as the duration of the piece to create the set of features. The final tabulated dataset contains the mean and variance arrays for the following features: Mel Spectrogram SSM, Mel-Frequency Cepstral Coefficient (MFCC) Spectrogram SSLMs (Euclidean and Cosine distances), Chromagram SSLMs (Euclidean and Cosine distances) [24], as well as 15 other features. Since only 200 data samples may not be enough for many training tasks, we extend our dataset to 1200 by augmenting the dataset using speed shifting, pitch shifting, time shifting, and noise injection [12, pp. 41-46]. The features were extracted similarly for the augmented dataset as well.

3.2 Form Analyzer Architecture

Our goal is to identify a piece of classical music as one of the following 12 forms: Arch, Bar, Binary, Minuet & Trio, Ritornello, Rondo, Sonata (-allegro), Ternary, Theme & Variation, Through Composed, Unary/Strophic, and Unique. Note that not all of the forms are equally distributed in the dataset (reflecting real-world musical pieces) and this leads to a class-imbalanced multi-class classification problem (see Supplement Figure 3 for the relative proportion of each form in the dataset). To address this problem, we adopt a framework that implements an ensemble of decision trees using a neural network, which has been shown to work well for class-imbalanced multi-class datasets [29, 30]. We note that other alternatives such as focal loss [31] or imposing fairness in terms of model performance across each pair of classes can also be used [32].

In our implementation, we use the TreeGrad approach by [33], which implements Gradient Boosted Decision Trees as Neural Networks and has been shown to have a small computational footprint. TreeGrad is an extension of Deep Neural Decision Forests (DNDF), which treats the node split structure of a decision tree as a neural network architecture search problem. Similar to DNDF, they have three layers: a *decision node* layer, a *routing* layer, and a *prediction* layer [34]. The decision layer consists of decision stumps, each of which computes the probability of routing the data node x to (left/positive or right/negative) child nodes for a node n and is formulated as (shown here only for the positive route) $d_n^+(x;\theta_+) = \sigma(\theta_+^{\mathrm{T}} x + b)$, where θ_+ are the learnable parameters and σ is a temperature-controlled softmax function. These routing probabilities (both positive and negative) of all nodes are then concatenated to yield $D(x;\tilde{\theta})$, which is the output of the first layer. The parameter vector in $\tilde{\theta}$ forms the linear decision boundary that results in how each node is routed. This is followed by the routing layer, which computes the probability that each node uses to route a data point to a particular leaf. For this step, we use a binary preconstructed matrix Q of size $l \times 2n$, where l is the number of leaves, and n is the number of internal nodes. This allows for enumerating the relationship between nodes and leaves. This layer computes μ_l which indicates the probability of reaching a leaf l and can be written as

$$\begin{aligned}\mu_l(x|\theta) &= \textstyle\prod_{j=1}^{n}(D(x;\tilde{\theta}_j) \odot Q_l + (1 - Q_l)) \qquad (1)\\ &= \exp(Q_l^{\mathrm{T}} \log(D(x;\tilde{\theta}))\end{aligned}$$

where Q_l is the l-th row of Q and $\odot$ is the Hadamard Product [33]. We then pass it through a softmax activation function to produce the output of the second layer. The final output layer is the leaf layer which is a fully connected layer. The activation function used here is $\exp(x)$. See Figure 2 for an illustration of the network architecture.

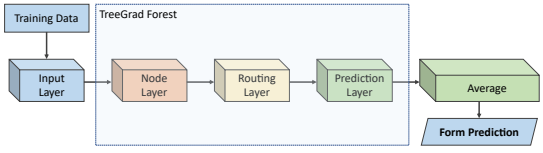

Figure 2. Architecture diagram of Form Analyzer

3.3 Phrase Analyzer Architecture

Recall that phrase analysis is a significantly more challenging task than form. To be able to learn phrase analysis similar to a musician, the model must be able to differentiate between musical events that can be labeled as having only the part label (*CODA*, *Episode 1*, *Middle Entry*, etc.), or only the phrase label (a, $a\prime$, b, *transition*, *melodic (variation)*, etc.) or both part and phrase labels. This presents several problems – the first deals with making the subtle distinction of when to assign both labels or just one, as well as choosing a suitable metric that captures the similarities correctly. Moreover, labels for varied phrases are also extremely difficult to grade, as a phrase (or part) may be repeated with a different harmonic goal (phrase a to a'

[2] While VSTs may differ across sequencing software, the Form Analyzer and Peak-Picking algorithm are intended to be instrument-neutral methods. As such, the timbral difference is negligible — including running the algorithm on different rendered sequences of the same pieces; we only seek to discover how the pitches are distributed structurally and temporally.

[3] We simply follow the human annotations from the original analyzed sheet music and copy these exactly at the time of their occurrence (in seconds, i.e., Part B with phrase d occurring at 15.27 seconds would be written as "15.270 B, d"), of course omitting the implicit labels as well.

or Parts A and A'). Variations of a can be either the same variation a' or a new variation a'' or a''' (these varied repetitions sometimes continue beyond 3, such as a^4 and so on). Since scoring such variations can be different based on the annotator, for experimentation on our dataset, we simply reduced the label set to remove prime marks and retain the phrase/part letter(s). Likewise, large and/or hybrid forms (such as Sonata-allegro form) in our dataset which have their own unique labels are simplified (during experimentation) to the letter label set with parts A, B, A'.

The task of Phrase Analysis requires labels to be provided with the form of the piece as well as the knowledge of where the musical phrase starts — a problem potentially solved using Onset or Peak Detection [25]. To do this, we implement a simple algorithm for peak-picking based on existing literature which is described next. The timestamps of the peaks, along with the output of the Form Analyzer (which is added as a feature), are provided as input to the Phrase Analyzer, where labels are classified sequentially (shown in Figure 4). We describe its components next.

3.3.1 *Onset Detection – Peak-Picking for Phrase Events*

To find the timestamps (in seconds) of the musical phrases, we use a reduced version of the peak-picking algorithm described in [35]. This algorithm computes the Mel Spectrogram and Self-Similarity Lag Matrix (SSLM) Chromagram (a group of pitch class distributions over time) to perform the onset detection. The Chromagram SSLM is computed using the Mel Spectrogram, which is clustered by pitch-class using k-Nearest Neighbors. The audio frames captured by the Short-Time Fourier Transform (STFT) are represented as a computed vector of peaks, which is returned as an array of timestamps. The choice of this scheme was based on preference for ideas that are common in the community, and we verified that it was comparable to other CNN architectures [36] and also greatly reduced system design time, given that training was not needed (since the approach was unsupervised).

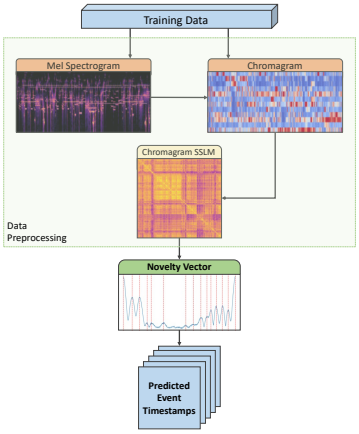

Figure 3. Schematic diagram of Peak-Picking algorithm

3.3.2 *Bidirectional LSTMs for Phrase Classification*

Using the onset detection method, the musical piece is essentially segmented into smaller phrases (i.e., the musical content between timestamps t and $t+1$ is denoted as feature vector x_t). Therefore, we think of a piece X as a collection of phrases $\{x_1, ..., x_T\}$, which occur in sequence. The task is to tag each phrase with a label for the phrase and/or the part. In all, we use 9 phrases, 9 parts, and 5 variations (phrase labels used for Theme & Variation pieces) for the labeling; specific names of these are provided in the Supplement. Clearly, the specific labeling associated with a phrase is dependent on what came before it, as well as what comes after. Therefore, we propose to use Bidirectional LSTMs to capture the dependence in such sequence data and replicate segment-segment similarity analysis.

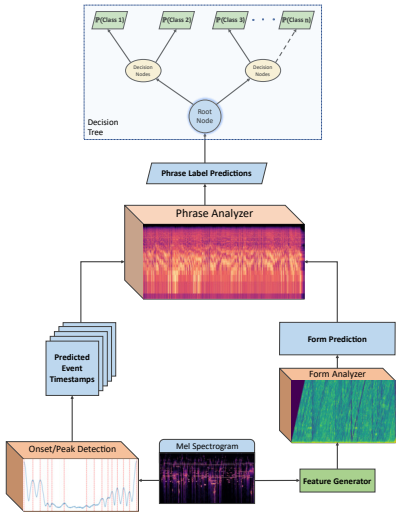

Figure 4. Architecture diagram of Phrase Analyzer

We now discuss the network architecture. X is passed as input to a Bidirectional Long Short-Term Memory (LSTM) [37] network. LSTM networks can capture long-term dependencies in temporal data and have been successfully used for a number of time series classification problems. LSTM (similar to Recurrent Neural Networks (RNNs) [38]) contains loops in its architecture that allow it to memorize previous states such that the network can effectively process temporal data. A typical LSTM layer consists of a set of recurrently connected blocks, known as memory blocks. Each block contains one or more recurrently connected memory cell (c^t) and three multiplicative units - the input (i^t), output (o^t), and forget gates (f^t) which regulate the extent to which data is propagated through the LSTM unit. The operations inside an LSTM block can be formulated using:

$$\begin{aligned} f_t &= \rho(W_f x_t + U_f h_{t-1} + b_f) \\ i_t &= \rho(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \rho(W_o x_t + U_o h_{t-1} + b_o) \\ \hat{c}_t &= \phi(W_c x_t + U_c h_{t-1} + b_f) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \hat{c}_t \\ h_t &= o_t \circ \phi(c_t) \end{aligned} \tag{2}$$

Here ρ and ϕ are activation functions, $\circ$ denotes the element-wise product operation, x_t is the input vector, W and U are weights, and h_t is the hidden state vector — also known as the output vector of the LSTM unit. Because of the dependence on both past and future phrases, we use a Bidirectional LSTM (Bi-LSTM), which includes both a forward and backward layer of LSTMs. Both the forward and backward layer outputs are calculated by using the standard LSTM update equations (2). Then, Bi-LSTM

connects the two hidden layers to the same output layer. More details about Bi-LSTMs can be found in [39, 40].

To perform the final classification, the Bi-LSTM is connected to a Decision Tree to provide the label(s) for each timestamp. To accomplish this, we used the feature vector output from the (Time Distributed) Dense layer as the training set for the tree, which is fit along with the original set of labels. Once the tree is combined with the Bi-LSTM, it can be used to provide the final prediction for new timestamps. Using this approach also helped greatly decrease the training time of the Phrase Analyzer model and reduced the overfitting that the LSTM would suffer from by itself.

4. EXPERIMENTS

We evaluate each component of our system individually as well as a whole for the Phrase Analyzer, which utilizes all the components. For all the experiments, we utilize the entire augmented dataset, where 85% of the data was used for training, and the rest is used for testing.

4.1 Form Analyzer Evaluation

Setup: The Form Analyzer model takes in the features as described in Section 3.1 (the duration and Mel Spectrogram SSM) and outputs the predicted classification, which is compared against the "ground truth" label in the dataset. To turn the Mel Spectrogram SSM into a usable feature vector, we calculate the mean of the SSM to obtain a 1D array, which the model receives with the duration appended. Note that we can use the Form Analyzer architecture as a standalone model to identify the form of the musical piece. This can be useful to search a musical database by forms or for a musicologist to understand when and why a composer may choose a particular form over another. On the other hand, it can also be used to provide input to the Phrase Analyzer. Therefore, we evaluate its efficacy independently to evaluate its performance compared to other methods. The augmented dataset was split into 85% training and 15% testing for cross-validation, [4] and the analysis model was evaluated using the classification accuracy in addition to other metrics such as Precision, Recall, and F1 scores.

Parameters: The TreeGrad model was tuned to 31 leaves per tree, an unbound max depth, 100 estimators (trees in the forest) with refit splits enabled, and the learning parameters include $\alpha = 0.1$ and a batch size of 32.

Results: The final Form Analyzer model achieved a classification accuracy of 83.9% — a surprisingly good performance given the subjective nature of the form classification, as many of the "inaccurate" classifications may be subjectively true based on personal bias. We compared our model with a number of other classification methods — 2D CNN [41], 2D CNN Ensemble [42], 1D CNN [43], Autokeras [44], Neural-SVM [45], XBNet [46], DJINN [47], and an implementation of Neural Decision Trees [48]. In the plot in Figure 5 (left), our model outperforms other methods significantly. The next closest approach is the decision tree method, illustrating that the decision tree-based approaches indeed works better for this data. Note that the model can be further fine-tuned by using a different boosting method. Furthermore, our method reported a Precision value of 85%, whereas both Recall and F-Score were 84%.

In addition to numerical accuracy, we wanted to study the mistakes in our method and how others performed in identifying the forms, and whether some specific forms were commonly misclassified as others. For this, we compared the confusion matrices from each approach. To compare the confusion matrices numerically, we compute the confusion entropy [49] for each — this is shown in Figure 5 (center and right). Confusion entropy is computed by exploiting the class distribution information of misclassifications of all classes as other classes (off-diagonal elements of the confusion matrices) and is an appropriate measure for this purpose. The results show overall, our method has the lowest entropy. The confusion entropy for each class (Figure 5 (right)) shows which classes were harder to classify. Here we found that our method has an entropy of $.24$ even for the worst-performing class. In addition, by quantitative analysis, a common theme we found is that most models tended to confuse forms that are typically similar in length (binary/unary, theme & variation/sonata, etc.) or hybrid/compound forms and their derived form (e.g., sonata/binary) even though for our model, such misclassifications are minimal.

4.2 Peak-Picking Algorithm Evaluation

Results: While the onset detection algorithm was not evaluated by a formal metric, it was tested against the training data, and the output timestamps were found to be nearly identical or had a low enough difference to be subjectively true (similar to the bias of a human analyst). The output of the algorithm was compared to numerous hand-labeled pieces from the dataset, and the difference was found to be negligible for most pieces (around $\approx \pm 5.269$ seconds on average), with the greatest absolute time difference being $\approx \pm 14.823$ seconds for pieces much greater than 6 minutes in duration. This metric was based on the approach found in [25]. The algorithm was also found to be comparable to other machine learning approaches such as SOMs [23] and CNNs [24]. Our result was based on a comparison between the "ground truth" and predicted timestamps in the validation dataset by index pairs rather than by the pair of closest timestamps since the algorithm may report more or fewer events than our own analysis.

4.3 Phrase Analyzer Evaluation

This model is much more difficult to evaluate using a classification evaluation metric due to numerous factors that vary from conventional machine learning classification models. First, the model gets the timestamps from the Peak-Picking algorithm, which is difficult to compare to human-annotated scores (our "ground truth") due to integrative disagreement (i.e., a group of analysts would need to collectively agree on a ground truth as their analyses will likely vary). The labels are also often highly subjective and vary by the judgment of the annotator. In addition, some of the labels are implicit (for example, Part A continues until timestamp n but is only labeled at the first occurrence).

[4] The dataset is sorted alphabetically by form per augmentation; other permutations typically performed the same or worse.

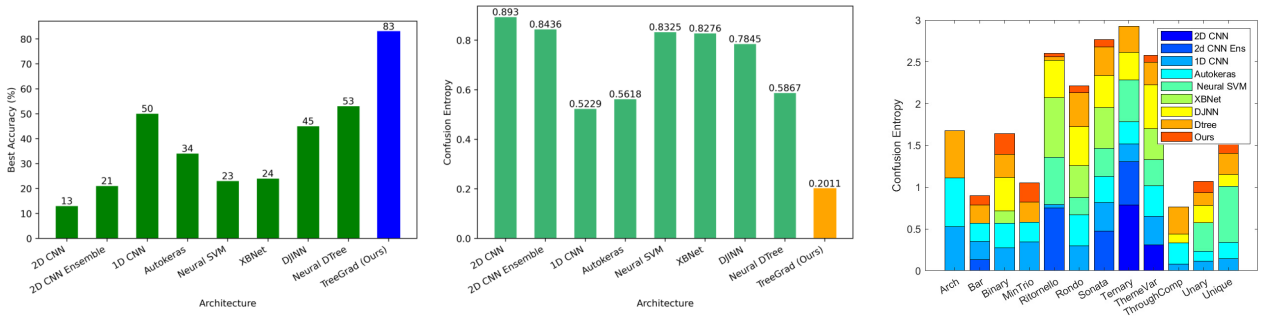

Figure 5. Form Analyzer Architecture Comparison with other methods (left), Confusion Entropy calculated for each method (center), and the class-wise Confusion Entropy for each method (right)

Therefore, we decided not to use a $0/1$ accuracy metric but a more realistic one based on how a music theory expert would grade the labeling of other musicians (such as in a music theory class) based on a rubric. Such rubric-based grading is commonly used in case of challenging problems such as automated essay and clinical note grading [50, 51].

Setup: We design a rubric as follows: we score each labeling in $[0, 1]$ giving a score of 1 if the part and/or phrase match perfectly, but also have a variety of other values in $[0, 1)$ for partially correct labeling. These partial score scenarios cover a range of possibilities, such as if either the part or phrase but not both has been guessed correctly, the degree of similarity of the predicted phrase to the actual phrase, and accounting for the fact that time stamping may be off by a few seconds. These partial scores are ordered from high to low depending on the extent and multiplicity of the issues mentioned above. The rubric is presented in the Supplement, and the code to do this grading on the output of the system will be provided with the dataset [13].

Parameters: The LSTM-Tree model has 4 LSTM units with a dropout rate of 0.2, a batch size of 10, sigmoid activation, and the learning parameters use the binary cross-entropy loss function, Adam optimizer, and 5 epochs.

Results: We evaluated the phrase analyzer using other state-of-the-art methods such as 1D CNN, RNN, and Feed-forward Neural Network and used the same rubric to come up with a weighted assessment score of the predicted phrases. We found that our LSTM-Tree architecture outperformed other NN architectures — our model reported a weighted score of 79.16%, whereas the best of the comparable methods reported a score of 77.75% (see Supplement Figure 1 for a plot related to this). A qualitative evaluation of the results reveals our LSTM-Decision Tree method produced labels that are more consistent with a human analyst (who is prone to some errors) even on one attempt, whereas for some of the other methods, the labelings are more random and less interpretable. These results show that the LSTM-Decision Trees not only substantially increased accuracy quantitatively for the Phrase Analyzer but also gave reasonably realistic phrase labels.

5. DISCUSSION

Here we briefly discuss some possible extensions and improvements to the dataset as well as methods presented in the paper. On the dataset front, we hope to expand the number of pieces as well as add additional annotators. The current dataset also features class imbalance; anthologies of classical music classified by form are lacking, [5] though our system could be employed to assist in the compilation of such a database. A more sophisticated system may also greatly benefit from restructuring the analysis labels in the database to the standards specified in [54]. On the method front, an obvious area of improvement is the Peak-Picking algorithm, which can be replaced by a deep learning approach such as CNN or LSTM so that the whole network can be considered as one large deep learning system and parameter weights learned jointly. The Phrase Analyzer model can also benefit from the inclusion of Curriculum Learning or a Human-in-the-Loop type of approach, much like that of a traditional Form and Analysis class — though a Sequence-to-Sequence or Autoencoder model may also be useful in developing a faster/more accurate system.

6. CONCLUSION AND FUTURE WORK

In this paper, we introduce a new dataset, the **SMFSA Database**, accompanied by deep learning benchmark methods to analyze classical music forms and phrases. To the best of our knowledge, this is the most comprehensive study of a computational approach used toward classical music structure analysis. While the current system is specific to classical music, it could be extended to classify numerous additional forms (e.g., through transfer learning or an extended architecture), such as those found in popular music, ethnomusicology, and more complex hybrid forms. Another difficult task lacking substantial research is Optical Music Recognition (OMR) — our methods could potentially be extended to perform both visual music analysis and the classification/segmentation on the score directly (i.e., in the style of a human analyst). The system may also be extendable for use in Forensic Musicology and Copyright Detection systems, using the output analysis from the system to compare multiple pieces of music for potentially similar or exact replications of musical phrases.

[5] Green's book on Form Analysis [52] was found to be the most useful and accurate resource resembling such an anthology, whereas other resources were presented as anthologies for analysis rather than classified works. As such, our dataset was built primarily on the pieces (and musical forms) selected in this book (including some from [53] as well) to serve as a common ground for analysis.

7. REFERENCES

[1] C. Burges, D. Plastina, J. Platt, E. Renshaw, and H. Malvar, "Using audio fingerprinting for duplicate detection and thumbnail generation," vol. 3, Apr 2005, pp. iii/9 – iii/12.

[2] J. Valinsky, "Facebook now lets users share music, listen to 30-second song snippets," *Digiday*, Nov 2015. [Online]. Available: https://digiday.com/?p=145138

[3] M. Müller, *Fundamentals of Music Processing: Using Python and Jupyter Notebooks*. Springer Nature, Apr 2021.

[4] R. Liao, "How china's acrcloud detects copyrighted music in short videos," *TechCrunch*, Aug 2020. [Online]. Available: https://techcrunch.com/2020/08/12/acrcloud-profile/

[5] T. Li, M. Ogihara, and G. Tzanetakis, *Music Data Mining*. CRC Press, Jul 2011.

[6] ——, "Special section on music data mining," *Multimedia, IEEE Transactions on*, vol. 16, pp. 1185–1187, Aug 2014.

[7] M. Gotham, "Moments musicaux," in *6th International Conference on Digital Libraries for Musicology*, ser. DLfM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 70–78. [Online]. Available: https://doi.org/10.1145/3358664.3358676

[8] R. Team, "Teaching the elements of music - form," Feb 2016. [Online]. Available: https://www.connollymusic.com/stringovation/teaching-the-elements-of-music-form

[9] M. L. Maher and D. Fisher, "Using ai to evaluate creative designs," in *Proceedings of the 2nd International Conference on Design Creativity (ICDC)*, vol. 1, Sep 2012, p. 45–54.

[10] H.-T. Cheng, Y.-H. Yang, Y.-C. Lin, and H. H. Chen, "Multimodal structure segmentation and analysis of music using audio and textual information," in *IEEE International Symposium on Circuits and Systems*, 2009, pp. 1677–1680.

[11] J. B. L. Smith, J. A. Burgoyne, I. Fujinaga, D. De Roure, and J. S. Downie, "Design and creation of a large-scale database of structural annotations." in *ISMIR*, vol. 11. Miami, FL, 2011, pp. 555–560.

[12] D. Szelogowski, "Deep learning for musical form: Recognition and analysis," Master's thesis, Apr 2022. [Online]. Available: https://doi.org/10.13140/RG.2.2.33554.12481

[13] ——, "Form-nn," Apr 2022. [Online]. Available: https://github.com/danielathome19/Form-NN

[14] C. M. Resource, "Welcome to the classical midi and mp3 resource." [Online]. Available: http://www.classicalmidiresource.com/

[15] B. Krueger, "Classical piano midi page - main page," 2018. [Online]. Available: http://www.piano-midi.de/midicoll.htm

[16] K. der Fuge Project, "kunstderfuge.com - the largest classical music midi collection." [Online]. Available: https://www.kunstderfuge.com/

[17] S. Ritchie, 2022. [Online]. Available: https://www.classicalmidi.co.uk/page7.htm

[18] T. C. A. Team, 2022. [Online]. Available: https://www.classicalarchives.com/midi.html

[19] E. Peiszer, T. Lidy, and A. Rauber, "Automatic audio segmentation: Segment boundary and structure detection in popular music," Tech. Rep., 2007.

[20] J. Yang, "Music genre classification with neural networks: An examination of several impactful variables," 2018.

[21] Y. Guan, J. Zhao, Y. Qiu, Z. Zhang, and G. Xia, "Melodic phrase segmentation by deep neural networks," 2018. [Online]. Available: https://arxiv.org/abs/1811.05688

[22] D. Hörnel and W. Menzel, "Learning musical structure and style with neural networks," *Computer Music Journal*, vol. 22, no. 4, pp. 44–62, 1998. [Online]. Available: http://www.jstor.org/stable/3680893

[23] P. J. Ponce de León and J. M. Iñesta, "Pattern recognition approach for music style identification using shallow statistical descriptors," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 2, pp. 248–257, 2007.

[24] K. Ullrich, J. Schlüter, and T. Grill, "Boundary detection in music structure analysis using convolutional neural networks," in *ISMIR*, 2014.

[25] T. Grill and J. Schlüter, "Structural segmentation with convolutional neural networks mirex submission," 2015.

[26] T. O'Brien, "Musical structure segmentation with convolutional neural networks," 2016.

[27] J. de Berardinis, M. Vamvakaris, A. Cangelosi, and E. Coutinho, "Unveiling the hierarchical structure of music by multi-resolution community detection," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, 2020.

[28] carlosholivan, "Selfsimilaritymatrix-serra.ipynb," Sep 2020. [Online]. Available: https://github.com/carlosholivan/SelfSimilarityMatrices/blob/master/SelfSimilarityMatrix_Serra.ipynb

[29] T. R. Hoens, Q. Qian, N. V. Chawla, and Z.-H. Zhou, "Building decision trees for the multi-class imbalance problem," in *Advances in Knowledge Discovery and Data Mining*, P.-N. Tan, S. Chawla, C. K. Ho, and J. Bailey, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 122–134.

[30] X. Yuan, L. Xie, and M. Abouelenien, "A regularized ensemble framework of deep learning for cancer detection from multi-class, imbalanced training data," *Pattern Recognition*, vol. 77, pp. 160–172, 2018.

[31] K. Pasupa, S. Vatathanavaro, and S. Tungjitnob, "Convolutional neural networks based focal loss for class imbalance problem: A case study of canine red blood cells morphology classification," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–17, 2020.

[32] V. S. Lokhande, A. K. Akash, S. N. Ravi, and V. Singh, "Fairalm: Augmented lagrangian method for training fair models with little regret," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 365–381.

[33] C. Siu, "Transferring tree ensembles to neural networks," in *Neural Information Processing*, T. Gedeon, K. W. Wong, and M. Lee, Eds. Cham: Springer International Publishing, 2019, pp. 471–480.

[34] P. Kontschieder, M. Fiterau, A. Criminisi, and S. Rota Bulò, "Deep neural decision forests," Dec 2015, pp. 1467–1475.

[35] J. Serrà, M. Müller, P. Grosche, and J. L. Arcos, "Unsupervised music structure annotation by time series structure features and segment similarity," *IEEE Transactions on Multimedia*, vol. 16, no. 5, pp. 1229–1240, 2014.

[36] C. Hernandez-Olivan, J. R. Beltran, and D. Diaz-Guerra, "Music boundary detection using convolutional neural networks: A comparative analysis of combined input features," Aug 2020. [Online]. Available: https://arxiv.org/abs/2008.07527

[37] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[38] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 2020.

[39] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks.*, vol. 4, 2005, pp. 2047–2052.

[40] Z. Cui, R. Ke, Z. Pu, and Y. Wang, "Stacked bidirectional and unidirectional lstm recurrent neural network for forecasting network-wide traffic state with missing values," *Transportation Research Part C: Emerging Technologies*, vol. 118, p. 102674, 2020.

[41] P. Nandi, "Cnns for audio classification," *Towards Data Science*, Dec 2021. [Online]. Available: https://towardsdatascience.com/cnns-for-audio-classification-6244954665ab

[42] L. Nanni, Y. M. G. Costa, R. L. Aguiar, R. B. Mangolin, S. Brahnam, and J. Silla, Carlos N., "Ensemble of convolutional neural networks to improve animal audio classification," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2020, no. 1, May 2020.

[43] S. Kiranyaz, T. Ince, O. Abdeljaber, O. Avci, and M. Gabbouj, "1-d convolutional neural networks for signal processing applications," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 8360–8364.

[44] J. Brownlee, "How to use autokeras for classification and regression," Sep 2020. [Online]. Available: https://machinelearningmastery.com/autokeras-for-classification-and-regression/

[45] M. A. Wiering, M. H. v. d. Ree, M. Embrechts, and L. Schomaker, "The neural support vector machine," Nov 2013. [Online]. Available: https://www.researchgate.net/publication/258435394_The_Neural_Support_Vector_Machine

[46] T. Sarkar, "Xbnet - an extremely boosted neural network," Jun 2021. [Online]. Available: https://arxiv.org/abs/2106.05239

[47] K. D. Humbird, L. Peterson, and R. McClarren, "Deep jointly-informed neural networks," Jul 2017. [Online]. Available: https://www.researchgate.net/publication/318205627_Deep_Jointly-Informed_Neural_Networks

[48] Y. Yang, I. G. Morillo, and T. M. Hospedales, "Deep neural decision trees," Jun 2018. [Online]. Available: https://arxiv.org/abs/1806.06988

[49] J.-M. Wei, X.-J. Yuan, Q.-H. Hu, and S.-Q. Wang, "A novel measure for evaluating classifiers," *Expert Systems with Applications*, vol. 37, no. 5, pp. 3799–3809, 2010.

[50] H. T. T. Nguyen, "Neural networks for automated essay grading," 2016.

[51] W.-w. Yim, A. Mills, H. Chun, T. Hashiguchi, J. Yew, and B. Lu, "Automatic rubric-based content grading for clinical notes," in *Proceedings of the Tenth International Workshop on Health Text Mining and Information Analysis (LOUHI 2019)*. Association for Computational Linguistics, 2019. [Online]. Available: http://dx.doi.org/10.18653/v1/d19-6216

[52] D. M. Green, *Form in Tonal Music: An Introduction to Analysis*. Cengage Learning, 1979.

[53] W. E. Caplin, *Classical Form: A Theory of Formal Functions for the Instrumental Music of Haydn, Mozart, and Beethoven*. Oxford University Press, Dec 2000.

[54] M. R. H. Gotham and M. Ireland, "Taking form: A representation standard, conversion code, and example corpora for recording, visualizing, and studying analyses of musical form," in *ISMIR*, 2019.