A DEEP LEARNING METHOD FOR MELODY EXTRACTION FROM A POLYPHONIC SYMBOLIC MUSIC REPRESENTATION

Katerina Kosta **Wei Tsung Lu** **Gabriele Medeot** **Pierre Chanquion**
ByteDance
{name.surname}@bytedance.com

ABSTRACT

The task of identifying melodic lines in polyphonic music is a known active research topic in the symbolic and audio domain. Its importance has attracted the interest of researchers focusing on Music Information Retrieval and musicological applications and achieving high results is a common goal in industrial applications. Distinguishing the melody from the rest of the material in a written score can be a challenging task, however improvements have been reported in recent years using deep learning methods. In this paper, we present a lightweight deep bidirectional LSTM model for identifying the most salient melodic line of a music piece using handcrafted features without requiring the input score to be separated into multiple parts. We evaluate our model to measure the effectiveness of several data augmentation techniques and to compare performance to other state-of-the-art models. We also identify the features' importance and evaluate their incremental contribution on the model performance using evaluation metrics. Results on the POP909 dataset show that our model approximates or outperforms current state of the art models trained on the same dataset, based on different implemented metrics and observations.

1. INTRODUCTION

The concept of the note-to-note organization of music naturally evolves to perceiving larger structures such as phrases and melodic contours. Identifying what a listener perceives to be the melody in a piece of music and, more generally, examining the musicological concept of a melodic line has emerged in recent years as an important topic in the Music Information Retrieval community [1].

A melodic line incorporates musical properties which are rich in contextual information such as structure and rhythm and it embeds expressive characteristics from the perspective of the composer when constructing a piece as well as the perspective of the performer interpreting it. From a musicological point of view, being able to extract a melodic line with high accuracy can reveal new research directions, from analysing a composition style to classifying a performer's tendencies. In industrial or other research applications, high accuracy in melody extraction can improve the results of music search or recommendation algorithms as well as music generation systems.

In both audio and symbolic domains, the task of retrieving the notes of the melody from a polyphonic piece is not trivial. For example, during the act of listening, one may find it hard to distinguish note intervals from one another. It may happen that an interval judged as a major third when heard by itself, is judged as fourth in a separate music context instead [2] (p. 217). During the act of reading a music score or rendering it using a single type of timbre, the note intervals within separate voices might be perceived as interchangeable. The aforementioned challenges have inspired the research on voice extraction, i.e. partitioning the polyphonic piece into a set of monophonic melodies (more details in [3] and [4]).

In this paper, we are focusing on the task of identifying the main melodic line in a symbolic score. Past work on this specific task can be divided into two broad categories. The first includes models predicting a melody part from a multiple-part score input, meaning that a series of notes have been separated to individual parts and the task is to predict which part, either globally or in segments, contains the main melodic line. Most common features used for this purpose are related to notes pitch, intensity, duration, as well as the total number of notes among a part. This type of melody identification is out of scope for this paper, however we refer to [5–7] for details.

Our approach belongs to the second category that is to identify the notes that constitute the main melody in the score without any prior knowledge of separated score parts. Hence, we process an unsegregated set of notes, as we believe that this approach give more flexibility on how it can be used in various applications.

Related work. There is a limited number of previous work on this matter. One of the first attempts is the skyline algorithm [8] which keeps the highest note starting at any time, from all simultaneous note events. The other existing algorithms can be separated in the way the music is represented as input. We can pinpoint two main strands. One handles the music as a piano-roll visualisation, which is a semantic segmentation where musical scores are treated as two-dimensional images. Melody extraction systems presented in [9] and [10] use this type of representation, with the aim to classify the notes represented as pixels. The

piano-roll representation is mainly used in deep learning methods using convolutional neural networks. The other strand, which our approach belongs to, treats the music sequentially, representing a series of tokens or low and mid-level features that best describe the input characteristics. Recurrent neural network based models employ the note-sequence representation. In the case of the system in [11], the input representation constitutes note-to-note affinity values coming directly from their contextual notes, constructing a weighted undirected graph, having as edge weights the corresponding affinity values. In order to obtain a single melody outcome, they employ spectral clustering to obtain one cluster over the learned graph.

The use of features is presented in [12], where each note is characterised by a set of note properties such as note dissonance, how close a note is to a note ranked with a high probability of being a melody note, or properties from the score metadata such as the musical instrument that a particular note has been assigned to. In our approach, we adapt a set of features which contain note information such as pitch and duration, as well as properties of a note in its polyphonic context.

A recent approach MIDIBERT [13] adopts the mask language model training strategy which is largely used in the field of natural language processing. The model accepts an input representation of sequence of tokens where each token represents a musical event. By training the model to reconstruct the masked input sequence, the transformer model can learn a latent representation of symbolic music. The experiment results show that with such pretraining, the model can be easily fine-tuned for other down stream tasks such as melody extraction.

Previous work reports high results in melody extraction accuracy, especially on recent systems that use deep learning methods. However, the resources used are either not publicly available or their use is copyright protected. Therefore, it is hard to compare and assess results due to unpublished datasets, different metrics reported and different test sets used. Datasets that clearly identify the primary melodic part from accompaniment are rare. For this publication, we have used the open-source POP909 dataset [14], where the salient melody notes have been distinguished. Pop music has been also used in [6, 9]. Folk music has been used in [10, 11] and classical in [9, 10]. The system in [13] adapts POP909 for the task of melody extraction.

Proposed method. The melody extraction model that we introduce in this paper is a supervised bidirectional Long-Short Term-Memory (biLSTM) model called LStoM, standing for Large Score to Melody. It receives a set of computed features as input derived from a MIDI score and classifies which notes constitute the main melodic line. We adapted the note events representation as a set of features in the form inspired by [4].

For our model input, we compute the following features for each note in our dataset, as described in Table 1. The first two features, `pitch` and `dur`, contain the basic pitch and duration information of a note, respectively, where pitch is expressed in a MIDI note number and the duration in a crotchet level. For `pitch_dist_below` and `pitch_dist_above`, we compute the distance to the pitch of the next (neighbouring) higher or lower note sounded simultaneously with the note in use. `pos_in_bar` records the score beat where the note onset is located within the score bar. `pitch_in_scale` records whether the note pitch is in the diatonic scale of the key signature. The features `pitch_dist_below` and `pitch_dist_above` are inspired by the work in [4]. The feature `pitch_in_scale` is inspired by the work in [12].

Feature name	Description
`pitch`	note pitch (MIDI number)
`dur`	note duration (crotchets)
`pitch_dist_below`	absolute pitch distance (semitones)
`pitch_dist_above`	absolute pitch distance (semitones)
`pos_in_bar`	note onset position in bar (crotchets)
`pitch_in_scale`	note pitch in key scale (boolean)

Table 1. The features that have been selected and computed, along with their description.

More on the data that we have processed and the model we have built in Sections 2 and 3 respectively. Section 4 includes a set of experiments and observations to assess the model's capabilities. Finally, in Section 5 we summarise our findings and suggest potential future directions.

2. DATA DETAILS

For our experiments, we have used the MIDI files from POP909 dataset [14]. This dataset includes piano arrangements of Chinese pop songs created by musicians playing on a MIDI keyboard, and the melody part has been identified by manually transcribing the lead vocal melody. The scores include the remaining parts "bridge" and "accompaniment" which we merge and consider as a single accompaniment part throughout our experiments.

The scores are rich in expressive characteristics, such as the note timing and duration. In our processing, we have used the dataset metadata information regarding the beats and the key. For our purpose, we have pre-processed the MIDI files, to rectify the following dysfunctions. Many scores include a time signature of '1/4', so we have updated the time signature by considering the meter information from the audio beat metadata. Time signatures of "2/4" and "2/2" have been mapped to "4/4". Also, many scores include a misalignment in the downbeat level, so we have fixed the start time of the piece to reduce this issue.

As mentioned before, the scores are performative and this means that they accommodate fine details or imprecision in timing. In order to setup our model, we have created a time-grid of the notes' onset and duration that is flexible enough to keep performative characteristics and strict enough to not explode the dimensions of possible values which would be hard to be trainable later on. The time-grid is setup to align onset times in triplets resolution. The note durations are adjusted to the minimum between the distance of current note onset and next note onset, and the current note duration in semiquaver resolution.

The key information is extracted from the dataset metadata. When a song has more than one key signatures reported, we consider a single key signature extracted from the music21 key detection algorithm [15]. At the inference level of our model, we apply the latter method on the test score input. We do not apply any type of post-processing on the extracted melody. Also, we automatically merge all notes into a single MIDI part per piece, keeping the information of a note being a melody one as ground truth.

To facilitate a fair comparison of the models in our experiments, we use the same dataset subset and train-validation-test set split as in [13], which results to 865 songs. The steps above give us the amount of 1,408,056 notes, from which 19.8% are melody notes.

3. MODEL DETAILS

In this section, we present the details of our model architecture (3.1), the details about our training setup (3.2), as well as the metrics we have established (3.3). We have released the code of the model as open-source [1] .

3.1 Architecture

We implement a deep bidirectional LSTM architecture (biLSTM) [16] for our model, which is a type of Recurrent Neural Network (RNN). In our implementation, the biLSTM model has been setup after implementing a grid-search on the model hyper-parameters, using hyperopt [17]. After this process, the model results in having 6 layers, with hidden size of 140, followed by a forward layer. The Adam optimizer is used with a start learning rate of 0.001. We identified the set of hyper-parameters that produce the highest melody F-measure score (metric described in the following Section 3.3).

One characteristic of our dataset is the imbalance of the data labels (i.e. the two classification classes), meaning that non-melody notes outnumber the melody ones. To overcome this issue in our classification task, we adopt the focal loss [18] as the loss function for the model, which is defined as:

$$FL(p_t) = -a_t(1-p_t)^{\gamma} log(p_t), \tag{1}$$

where p_t denotes the model's estimated probability for an input to be classified to class t and $a_t \in [0, 1]$ is a weighting factor for the imbalanced classes which balances the importance of positive and negative examples. The term $(1-p_t)^{\gamma}$ acts as a modulating factor with γ controlling the rate at which over-weighted examples are down-weighted. We set $a_t = 0.25$ and $\gamma = 2$.

3.2 Training setup

The train-validation-test sets that we used from [13] contain a data percentage of 80-10-10%, respectively. The features have been scaled, given the data points in the train and the validation sets.

[1] https://github.com/bytedance/midi_melody_extraction

3.3 Metrics

The metrics that we have computed for this task are the following: `accuracy` indicating the overall accuracy of the predicted notes in percentage, as well as the `mel_P`, `mel_R` and `mel_F` for melody notes precision, recall and F measure values, respectively:

$$\texttt{mel\_P} = \frac{|\text{Correctly predicted melody notes}|}{|\text{Notes predicted as melody}|} \tag{2}$$

$$\texttt{mel\_R} = \frac{|\text{Correctly predicted melody notes}|}{|\text{Melody notes}|} \tag{3}$$

$$\texttt{mel\_F} = 2 * (\frac{\texttt{mel\_P} * \texttt{mel\_R}}{\texttt{mel\_P} + \texttt{mel\_R}}) \tag{4}$$

Also, we include the metric `Voice False Alarm` (VFA) from mir_eval [19], which is defined as the number of notes predicted as melody notes, although they are not, divided by the number of non-melody notes.

4. EXPERIMENTS

To evaluate the performance of our system, we have prepared a list of separate model setups, given the same train, validation and test sets. Also, we were interested in whether two types of data augmentation techniques would improve the accuracy of LStoM when applied in isolation as well as together. The first type is to shift the score key by altering the note pitches, where each piece has been transposed by n semitones, for n in $\{-6, -5, ..., 4, 5\}$ (the model is tagged as "LStoM PSaugm" standing for pitch shifting augmentation). The second type is to shift the melody notes one octave lower than the original one (the model is tagged as "LStoM MOaugm" standing for melody octave augmentation). Both techniques are common in the literature for the task of symbolic melody extraction [10].

4.1 Comparison with the state of the art

We investigate how other models that report state of the art results in various datasets perform in the case of the specific train, validation and test sets. To this end, we were able to train two models. One is following the default settings of [11] tailored to the task of the melody identification, using the augmentation techniques that are provided from this work and the pitch proximity method as a segment merging mode among the created note clusters (the model is tagged as "Hsiao-Su"). The other is following the default settings of [10], again using the augmentation techniques that are provided from this work (the model is tagged as "Lu-Su").

The skyline algorithm [8], which is essentially picking the highest pitch at a given onset time, has been considered as our baseline. Also, we consider a skyline variation where we keep the duration of the note with the highest pitch, ignoring the lower-pitch notes that start after its onset and before its offset. Both original and variation skyline output are illustrated in Figure 1, using the input example as reported in [8].

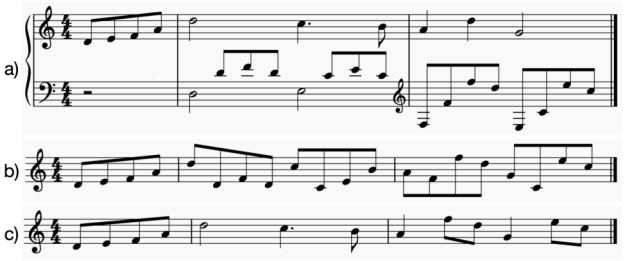

Figure 1. Illustration of our baselines. a) Input example, b) Output of skyline algorithm, c) Output of skyline variation algorithm.

Model	acc (%)	mel_F	mel_P	mel_R	VFA
LStoM*	92.3	0.816	0.778	0.872	0.065
LStoM* PSaugm	91.6	0.788	0.791	0.799	0.054
LStoM* MOaugm	92.3	0.800	0.815	0.811	0.066
LStoM* MO+PSaugm	91.9	0.798	0.797	0.811	0.054
MIDIBERT* [13]	**97.1**	**0.930**	**0.912**	**0.952**	**0.024**
LStoM	90.7	**0.774**	0.751	0.814	0.070
MIDIBERT [13]	**91.9**	0.772	**0.841**	0.727	**0.035**
skyline	63.1	0.499	0.353	**0.881**	0.435
skyline-variation	78.7	0.569	0.485	0.697	0.192
Lu-Su [10]	66.6	0.614	0.600	0.638	0.299
Hsiao-Su [11]	82.5	0.650	0.544	0.821	0.176

Table 2. Basic comparison results among testing models. An '*' is added to indicate that the MIDI files in the test set have been pre-processed.

Both LStoM and MIDIBERT [2] include pre-processing steps for note quantisation and alignment in downbeat level when preparing the MIDI files for training. At the testing stage, we considered as a fair comparison to first replicate the results of [13] where the MIDI files of the test set have been pre-processed, therefore our pre-processing steps were applied to the test set as well for LStoM and its augmentations (metrics reported at the top part of Table 2). The augmentations do not appear to improve the overall performance of LStoM. In the bottom part of Table 2, we report the metrics of the remaining comparison, where the notes of the test set have not been quantised nor aligned to the downbeat. Interestingly, MIDIBERT outperforms in the first scenario, while the results are mixed in the second one.

Lastly, LStoM is significantly lighter than MIDIBERT; the number of parameters are 875,462 compared to 111,298,052, respectively.

In the next two subsections, we are exploring LStoM in terms of how important the selected features for the training process are (Section 4.2) and we are discussing some observations from the models' outcome, respectively (Section 4.3).

[2] Function `align_midi_beats` in `https://github.com/wazenmai/MIDI-BERT/blob/CP/data_creation/preprocess_pop909/preprocess.py`, accessed 31 August 2022

4.2 Importance of features

We were interested to know how relevant the selected features are for our task and whether any of them are more important in a sense that they contain an amount of information that is crucial for such systems. Algorithmically it is feasible to examine and improve the interpretability of a predictive model to a degree, and to identify a type of ranking of importance for the input features. However, most recent feature ranking algorithms assume feature independence (for more examples and details we refer to [20]), an assumption which is not safe in our case.

Ranking	Feature	Ranking	Feature
1	`pitch`	4	`dur`
2	`pitch_distance_above`	5	`pos_in_bar`
3	`pitch_distance_below`	6	`pitch_in_scale`

Table 3. The ranking of feature importance obtained by RELIEF algorithm.

One algorithm that is not based on this assumption is RELIEF [21] which elaborates on a simple comparison idea whereby feature value differences between nearest neighbour instance pairs are identified. If a feature value difference is observed in a neighbouring instance pair with the same class, the feature ranking score decreases. However, the score increases when a feature value difference is observed in a neighbouring instance pair with different class values. We applied RELIEF to our dataset and the resulting feature ranking is reported in Table 3.

Not by surprise, the pitch has been ranked as the most important feature, followed by the one that computes how much is the distance from the pitch above. The third feature in the ranking list is the similar one, computing the pitch distance from the pitch below. The note duration comes to the fourth place, followed by the metrical-related feature of computing the position of the note within the score bar, and finally whether the note pitch is in scale.

In order to evaluate the incremental contribution of each of the features studied, we have considered feature subsets by incrementally adding features to the training set one at a time, starting with the most important one, i.e. the note pitch, and continuing to add features according to their rank order. The results obtained are reported in Table 4.

Model	acc	mel_F	mel_P	mel_R	VFA
LStoM1	86.5%	0.672	0.652	0.708	0.094
LStoM1-2	88.4%	0.718	0.697	0.752	0.082
LStoM1-3	89.4%	0.738	0.723	0.768	0.073
LStoM1-4	90.3%	0.742	0.780	0.724	0.052
LStoM1-5	92.0%	0.797	0.803	0.806	0.051
LStoM1-6	92.3%	0.816	0.778	0.872	0.065

Table 4. Basic comparison results among the variations of LStoM model, where LStoM*x-x* indicates the range of ranked features used at the training process.

It is worth noticing that the highest ranked feature contains on its own substantial predictive power: the melody F-measure score for the model induced with only pitch information alone is 0.672, which is slighter higher than the

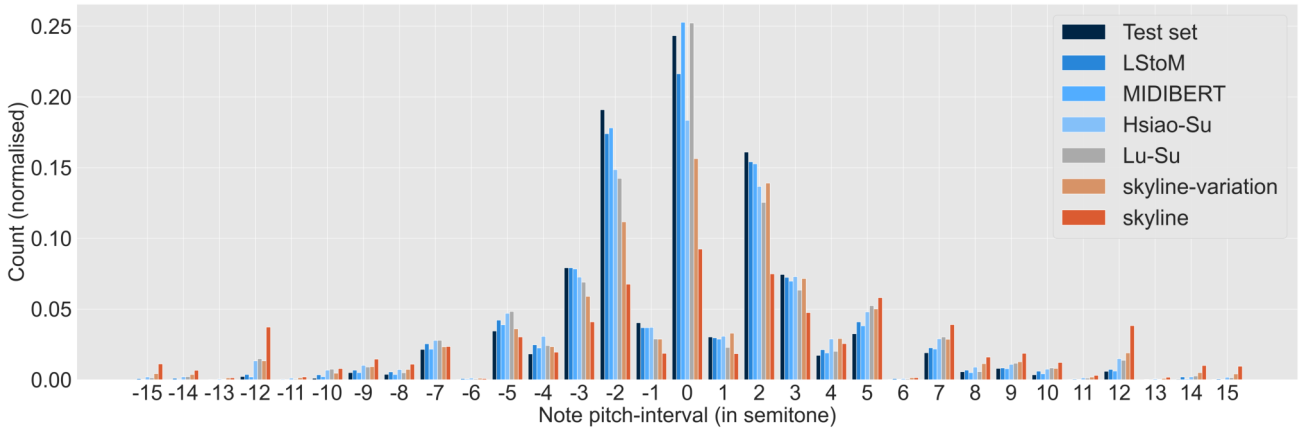

Figure 2. Pitch interval distribution among melodies predicted by LStoM, MIDIBERT, "Hsiao-Su", "Lu-Su", skyline and skyline-variation compared to the melodies from the test set.

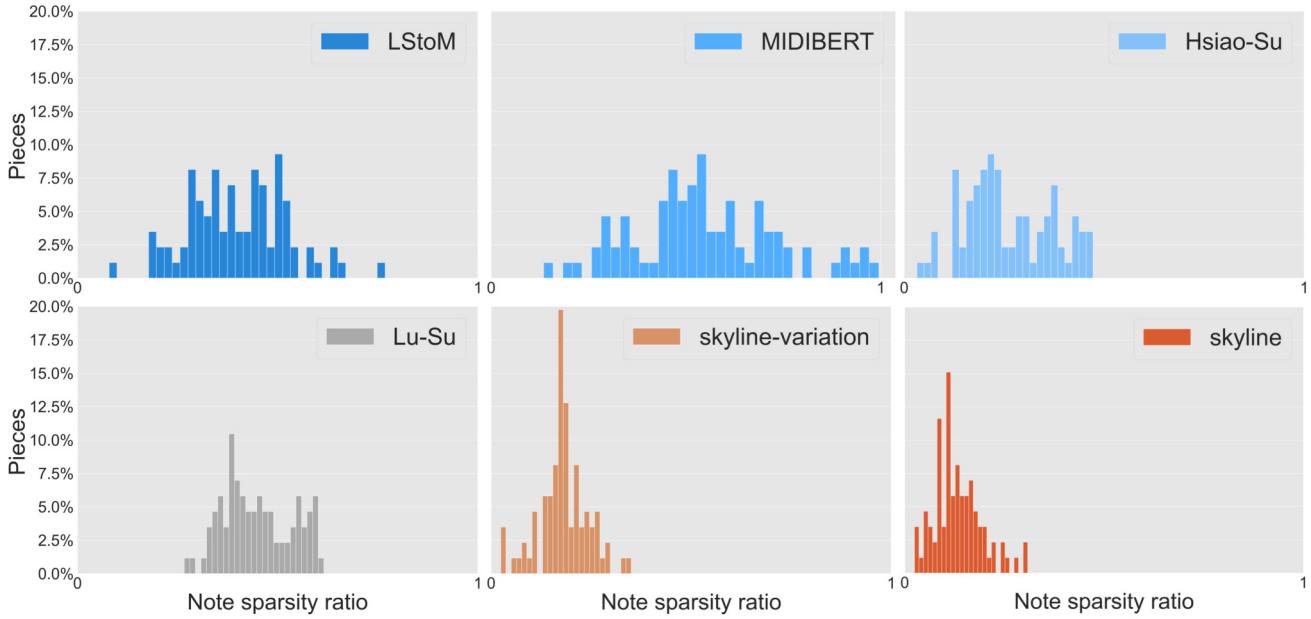

Figure 3a. Sparsity ratio among melodies predicted by LStoM, MIDIBERT, "Hsiao-Su", "Lu-Su", skyline and skyline-variation.

one obtained by our trainings on "Hsiao-Su" and "Lu-Su" systems. Interestingly, the precision is reduced by a very small margin and the voice alarm error value is slightly increased, when adding the feature `pitch_in_scale`, however the accuracy is increased at the same time.

4.3 Observations

Having a closer look to the predicted melodies by LStoM, MIDIBERT, "Hsiao-Su", "Lu-Su" and the baselines, we can highlight two separate statistics. One is the distribution of the note intervals among the consecutive melody note pairs (Figure 2 – the x axis has been limited to the range of around two and a half octaves for paper fitting purposes), where overall the melodies predicted by the models tended to reflect characteristics from the original melody contour. Another statistic is the percentage of predicted melodies with various degrees of sparsity in them. By sparsity, we mean the ratio of the total duration of the silent parts over the total duration of the notes in a score. Fig-

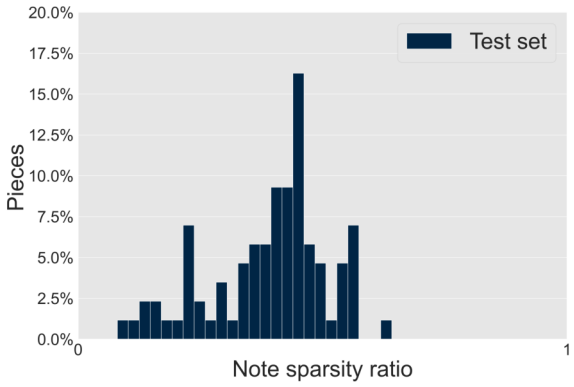

Figure 3b. Sparsity ratio among melodies in the test set.

ure 3a shows the sparsity distribution among the predicted melodies, while Figure 3b highlights the ground truth from the test set. A small amount of melodies from MIDIBERT tends to be more sparse than the scores from the test set.

The most challenging melody notes to classify are those that are not the highest note; i.e., those that the skyline method would certainly miss. We have used the same test set where around 28% of the melody notes are such challenging cases; of these, the LStoM system classifies correctly around 82%. What pattern in the feature data did the model use to achieve this high performance?

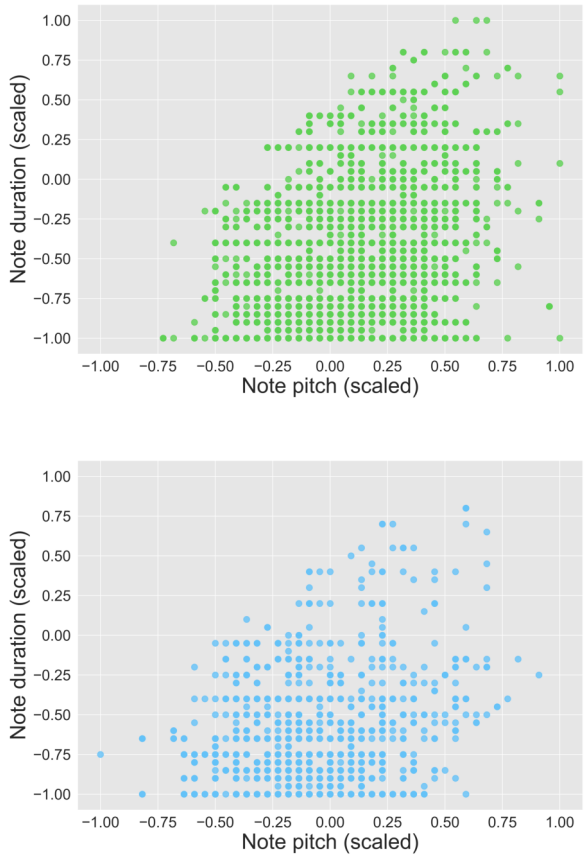

Figure 4. Scaled representation for the pitch-duration feature pair, for the melody notes that are not the highest note and have been either predicted correctly–top– or not (missed)–bottom.

It is not easy to examine the reason why sometimes such notes are not predicted as melody notes, but to investigate, we scaled the feature values of these true melody notes, and compared distributions of features between the correctly labelled (i.e., retrieved) and incorrectly labelled (i.e., missed) notes. One distribution that stood out is shown in Figure 4, a scatter plot of the pitch and duration of the melody notes. One may observe that although the distributions are similar, there are many more correctly labelled notes with high pitch and long duration. Such findings indicate that disentangling such points is not trivial, if at all feasible with our current feature set alone.

5. CONCLUSIONS AND PERSPECTIVES

In this paper we have presented a novel deep learning model of identifying the melody line from a polyphonic symbolic music. The key characteristics of the model is the input data representation as a set of features from the relevant task of multiple voice separation as well as the bidirectional nature of the architecture which provides information of future observations during the prediction process. The features have been examined regarding their ability to contain the information that the model needs to more accurately predict a melody note. Also, two data augmentation techniques are examined in isolation.

Results indicate that the proposed lightweight model which is trained and tested on a set of pop songs is capable of performing at the standard of existing benchmarks. Observations on the results reveal the ability of the model to reflect the concepts of score sparsity or the melodic pitch intervals from the test set. LStoM also performs well in situations where the melody note to be identified is not in the highest pitch of the score. One direction to explore is expanding the set of input features. Note velocity information or metrical representation extracted from the dataset could accommodate additional features for our model. Post-processing techniques have not been explored thoroughly, however they could help improving the results.

Setting research experiments for the task of melody extraction in the symbolic domain is challenging, in terms of data gathering and manipulating separate model architectures to fit the needs of a comparison task. Accumulating the available datasets and the existing systems to a common space for easy approach, such as in a dedicated MIREX page is a promising step forwards. To this end, examining how our model performs using different datasets in the training or inference process, is a step towards a more comprehensive review.

A direction that has been little explored is combining information from the symbolic domain to the prediction of melodies in the audio domain. Most of the existing literature for audio melody transcription relies only on information from the audio signal in isolation from other musical context explicitly. However, in [22], given a polyphonic music audio signal, the model is able to convert it to a piano arrangement; as part of this style transferring process, MIDI scores have been used as ground truth. Also, in [23], symbolic data representation has been used to pre-train a model which then operates to the audio domain. Tasks such as de-noising could be revisited following this perspective.

6. ACKNOWLEDGEMENTS

We thank Jordan B. L. Smith and Janne Spijkervet for their extensive paper review and support before submission.

7. REFERENCES

[1] J. Salamon, E. Gómez, D. P. Ellis, and G. Richard, "Melody extraction from polyphonic music signals: Approaches, applications, and challenges," *IEEE Signal Processing Magazine*, vol. 31, no. 2, pp. 118–134, 2014.

[2] N. Cook, *Music, Imagination, and Culture*, ser. ACLS Humanities E-Book. Clarendon Press, 1990.

[3] E. Cambouropoulos, "Voice and stream: Perceptual and computational modeling of voice separation," *Music Perception*, vol. 26, no. 1, pp. 75–94, 09 2008.

[4] R. de Valk and T. Weyde, "Deep neural networks with voice entry estimation heuristics for voice separation in symbolic music representations," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[5] R. Martín, R. A. Mollineda, and V. García, "Melodic track identification in midi files considering the imbalanced context," in *Pattern Recognition and Image Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 489–496.

[6] Z. Jiang and R. B. Dannenberg, "Melody identification in standard midi files," in *Proceedings of the 16th Sound and Music Computing Conference (SMC)*, 2019, pp. 65–71.

[7] S. Li, S. Jang, and Y. Sung, "Melody extraction and encoding method for generating healthcare music automatically," *Electronics*, vol. 8, no. 11, 2019.

[8] A. L. Uitdenbogerd and J. Zobel, "Melodic matching techniques for large music databases," in *Proceedings of the 7th ACM international conference on Multimedia (Part 1)*, 1999, pp. 57–66.

[9] F. Simonetta, C. E. Cancino-Chacón, S. Ntalampiras, and G. Widmer, "A convolutional approach to melody line identification in symbolic scores," *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[10] W.-T. Lu and L. Su, "Deep learning models for melody perception: An investigation on symbolic music data," in *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2018, pp. 1620–1625.

[11] Y.-W. Hsiao and L. Su, "Learning note-to-note affinity for voice segregation and melody line identification of symbolic music data," *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, pp. 285–292, 2021.

[12] H. Zhao and Z. Qin, "Tunerank model for main melody extraction from multi-part musical scores," in *Proceedings of the 6th International Conference on Intelligent Human-Machine Systems and Cybernetics - Volume 02*, ser. IHMSC '14. USA: IEEE Computer Society, 2014, p. 176–180.

[13] Y.-H. Chou, I.-C. Chen, C.-J. Chang, J. Ching, and Y.-H. Yang, "MidiBERT-Piano: Large-scale pre-training for symbolic music understanding," *arXiv preprint arXiv:2107.05223*, 2021.

[14] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, X. Gu, and G. Xia, "Pop909: A pop-song dataset for music arrangement generation," in *Proceeding of the 21st International Society for Music Information Retrieval (ISMIR)*, 2020.

[15] M. S. Cuthbert and C. Ariza, "Music21: A toolkit for computer-aided musicology and symbolic music data." in *International Society for Music Information Retrieval*, 2010, pp. 637–642.

[16] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005, iJCNN 2005.

[17] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *30th International Conference on Machine Learning, ICML 2013*, vol. 28, 2013, pp. 115–123.

[18] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[19] C. Raffel, B. Mcfee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. W. Ellis, C. C. Raffel, B. Mcfee, and E. J. Humphrey, "mir_eval: a transparent implementation of common mir metrics," in *In Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[20] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[21] R. J. Urbanowicz, R. S. Olson, P. Schmitt, M. Meeker, and J. H. Moore, "Benchmarking relief-based feature selection methods," *CoRR*, vol. abs/1711.08477, 2017. [Online]. Available: http://arxiv.org/abs/1711.08477

[22] Z. Wang, D. Xu, G. Xia, and Y. Shan, "Audio-to-symbolic arrangement via cross-modal music representation learning," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 181–185.

[23] W. T. Lu, L. Su *et al.*, "Vocal melody extraction with semantic segmentation and audio-symbolic domain transfer learning," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 521–528.