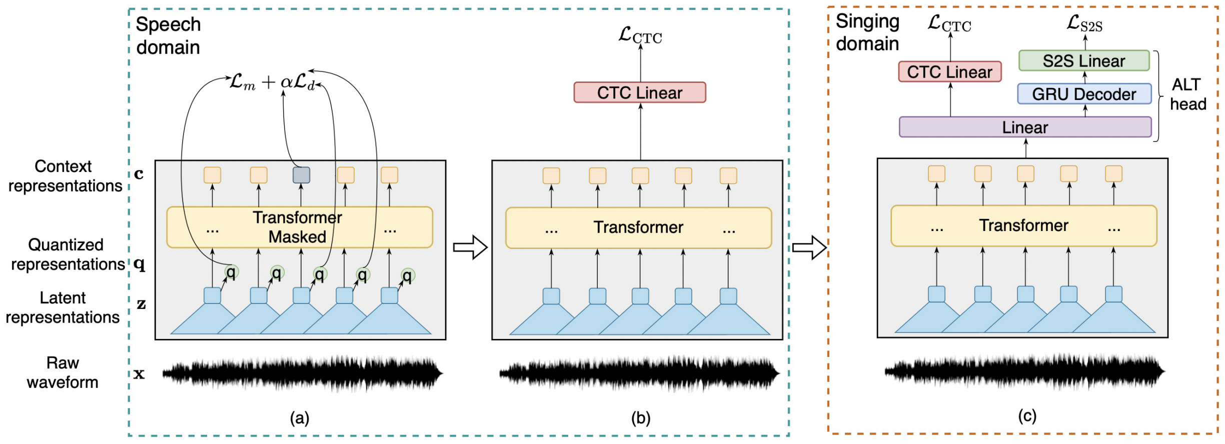

Figure 1. An overview of our training framework. (a) Stage I: Pretraining on speech data. (b) Stage II: Finetuning on speech data. (c) Stage III: Transferring on singing data.

3.3 Stage II: Finetuning on speech data

The process of finetuning requires labeled speech data. As shown in Fig. 1(b), the quantization module in wav2vec 2.0 is disabled since it is only used in Stage I, and a linear layer (CTC linear) is added on top of the Transformer. The whole model is trained by optimizing the connectionist temporal classification (CTC) loss $\mathcal{L}_{\text{CTC}}$ [40]. Suppose the ground-truth transcription is $\boldsymbol{w}^*$, which is a sequence of character tokens. The CTC loss is defined as:

$$\mathcal{L}_{\text{CTC}} = -\log \sum_{\boldsymbol{\pi} \in \mathcal{B}^{-1}(\boldsymbol{w}^*)} \prod_{t=1}^{T} p(\boldsymbol{\pi}_t|\boldsymbol{f}_t) \tag{2}$$

where $\boldsymbol{f}_t$ refers to $\boldsymbol{c}_t$ in this stage, T is the number of frames, $\mathcal{B}$ is a function to map an alignment sequence $\boldsymbol{\pi}_{1:T}$ to $\boldsymbol{w}^*_{1:N}$ (where N represents the number of character tokens) by removing duplicate characters and blanks while its inverse function $\mathcal{B}^{-1}(\boldsymbol{w}^*)$ refers to all the CTC paths mapped from $\boldsymbol{w}^*$. For speech data, there are 31 tokens for character targets, including 26 letters, the quotation mark, a word boundary token, $< bos >$, $< eos >$, and CTC blank token. The probability $p(\boldsymbol{\pi}_t|\boldsymbol{f}_t)$ is computed by the CTC linear layer followed by a softmax operation. Besides the supervised CTC loss, pseudo-labeling is also adopted during finetuning. Please refer to [16, 41] for more details.

3.4 Stage III: Transferring on singing data

3.4.1 From CTC to CTC/attention

To transfer the trained wav2vec 2.0 from the speech domain to the singing domain, we retain the weights of the feature encoder and the context network after Stages I and II. Furthermore, inspired by [42], we extend the original CTC system into a hybrid CTC/attention system through the addition of an ALT head on top of wav2vec 2.0 instead of a single CTC linear layer (Fig. 1(c)).

The context representations $\boldsymbol{c}$ are first fed into a linear layer followed by a leaky ReLU activation layer to obtain the features $\boldsymbol{f}$. Then $\boldsymbol{f}$ are sent to two network branches. One branch is a CTC linear layer, which aims to compute $p(\boldsymbol{\pi}_t|\boldsymbol{f}_t)$ as explained in sec. 3.2.2. Another branch is an attention-based GRU decoder [43] followed by a sequence-to-sequence (S2S) linear layer. The GRU decoder has a single layer with a hidden dimension of 1,024 and utilizes location-aware attention [43] with attention dimension 256. The decoder and S2S linear layer autoregressively compute the probability $p(\boldsymbol{w}_n|\boldsymbol{w}_{<n}, \boldsymbol{f}_{1:T})$, $n = 1, 2, ..., N$.

3.4.2 Training and Evaluation

During Stage III, wav2vec 2.0 and the ALT head are trained through the combination of CTC loss [40] and S2S loss [44]:

$$\mathcal{L}_w = \lambda_a \mathcal{L}_{\text{CTC}} + (1-\lambda_a)\mathcal{L}_{\text{S2S}} \tag{3}$$

$$\mathcal{L}_{\text{S2S}} = -\log \prod_{n=1}^{N} p(\boldsymbol{w}^*_n|\boldsymbol{w}^*_{<n}, \boldsymbol{f}_{1:T}) \tag{4}$$

where λ_a is a hyper-parameter to balance the CTC loss term and S2S loss term. To overcome catastrophic forgetting, we adopt a smaller learning rate for wav2vec 2.0 compared to the ALT head.

To evaluate the performance of the trained model, the most likely lyrics are predicted using beam search:

$$\begin{aligned}\boldsymbol{w}' = \arg\max_{\boldsymbol{w}} \lambda_b \log \sum_{\boldsymbol{\pi} \in \mathcal{B}^{-1}(\boldsymbol{w})} \prod_{t=1}^{T} p(\boldsymbol{\pi}_t|\boldsymbol{f}_t) \\ + (1-\lambda_b) \log \prod_{s=1}^{S} p(\boldsymbol{w}_s|\boldsymbol{w}_{<s}, \boldsymbol{f}_{1:T}) \\ + \lambda_c \log p_{LM}(\boldsymbol{w}) \end{aligned} \tag{5}$$

where λ_b and λ_c are two hyper-parameters in the decoding process. The language model is implemented by a 3-layer LSTM. The characters are firstly projected to embeddings and then fed into the LSTM with a hidden dimension of 2,048 to obtain RNN features. Finally, the RNN features are accepted by a 3-layer MLP with a hidden dimension of 1,024 to output the probability $p_{LM}(\boldsymbol{w})$.

Split	Dataset	# Utt.	Total Dur.
Train	DSing1	8,794	15.1 h
	DSing3	25,526	44.7 h
	DSing30	81,092	149.1 h
	DALI$^{\text{train}}$	268,392	183.8 h
Dev	DSing$^{\text{dev}}$	482	41 min
	DALI$^{\text{dev}}$	1,313	55 min
Test	DSing$^{\text{test}}$	480	48 min
	DALI$^{\text{test}}$	12,471	9 h
	Jamendo	921	49 min
	Hansen	634	34 min
	Mauch	878	54 min

Table 1. Statistics of segmented utterance-level datasets.

4. EXPERIMENTS

4.1 Datasets and preprocessing

We use various accessible mainstream lyric transcription datasets for our experiments, including DALI [19, 20], Hansen [22], Mauch [23], Jamendo [21], and a curated version of DAMP Sing! 300x30x2 [24] called DSing [5]. The train/development/test splits in our experiments are defined as follows. For the DSing dataset, we use the same split configuration as in [5]. Specifically, there are three different sizes of training sets (DSing1, DSing3, DSing30), as well as a development set DSing$^{\text{dev}}$ and a test set DSing$^{\text{test}}$. As for the DALI dataset, we divide all publicly available audio in DALI v2 [20] into training and development subsets (DALI$^{\text{train}}$ and DALI$^{\text{dev}}$ respectively). [1] We use DALI$^{\text{test}}$ [7], a subset of DALI v1 [19], as a test set for experiments involving DALI. The full Hansen, Mauch, and Jamendo datasets are used as additional out-of-domain test sets. In the training and development splits of the DALI dataset, songs that overlapped with any of our test sets are removed for more objective testing.

The speech data used to pretrain and finetune the wav2vec 2.0 is initially monophonic. Therefore, we expect wav2vec 2.0 to extract better representation features for singing data if monophonic audios are given as the inputs. Thus, we extract vocal parts from all of the polyphonic recordings in DALI, Mauch, and Jamendo datasets using Demucs v3 *mdx_extra* [45], which is the state-of-the-art source separation model that achieved the first rank at the 2021 Sony Music DemiXing Challenge (MDX). This ensures that we are consistent with the input requirements of wav2vec 2.0 and minimize the interference of musical accompaniment. We adopt utterance-level input for both training and testing. To facilitate the experiments, we perform utterance-level segmentation on all audios according to their annotations. Utterances with obvious faulty annotations are removed (e.g., utterances labeled as several words but have shorter than 0.1 s duration). The statistics of all datasets after utterance-level segmentation are listed in Table 1. The "total duration" column refers to the sum of durations of all utterances in the datasets, excluding the instrumental-only parts between utterances, hence resulting in shorter durations than [7]. We notice that the average utterance duration of DSing is longer than DALI (6.52 s vs. 2.47 s). Finally, lyric texts in training sets are normalized by converting all letters to upper case, converting digits to words, discarding out-of-vocabulary characters, discarding meaningless lines (e.g., "**guitar solo**"), and removing redundant space.

[1] At the time of this research, some audios are not retrievable through their YouTube links in the public-available metadata. Although audios containing the same titles and artist names can be found online, we cannot guarantee they perfectly match the annotations for the original audio versions in DALI. Discarding invalid audio is only performed for the training and the development sets.

4.2 Experiment setup

Our experiments are conducted through SpeechBrain toolkit [46] [2]. Before transferring on singing data, the wav2vec 2.0 has been pretrained on LibriVox (LV-60K) and finetuned on LibriSpeech (LS-960) [3] [16]. Then we randomly initialize the ALT head. During Stage III, we downsample all audios to 16 kHz and convert them to mono-channel by averaging the two channels of stereo audio signals. Then the singing data is augmented through SpecAugment [47]. wav2vec 2.0 and ALT head are trained using Adam optimizer [48]. The initial learning rates of ALT head and wav2vec 2.0 are 3×10^{-4} and 1×10^{-5} respectively. Learning rates are scheduled using the Newbob technique, with annealing factors of 0.8 and 0.9 respectively. The batch size is set as 4, and hyper-parameter λ_a is set as 0.2 during the experiments. We conduct our experiments on 4 RTX A5000 GPUs. Utterances whose duration is longer than 28 seconds are filtered out during training to prevent the out-of-memory issue. This filtering is not performed during the evaluation for a fair comparison with other existing methods.

We firstly utilize DSing30 to train the whole model for 10 epochs. We evaluate the performance of our model on DSing$^{\text{dev}}$ after each epoch. Finally, the best model is selected to be evaluated on the test split DSing$^{\text{test}}$. Word error rate (WER) is adopted as the evaluation metric. During the evaluation, WERs are averaged over all utterances in a test set. The RNNLM is trained for 20 epochs using an Adam optimizer [48] on texts of DSing30 split and validated on the DSing$^{\text{dev}}$ split after each epoch. The learning rate is 1×10^{-3} and the batch size is 20. During the decoding, the beam size is 512, and the hyper-parameters λ_b and λ_c are 0.4 and 0.5, respectively. The trained model is evaluated on the DSing$^{\text{test}}$ set.

For other test splits, we adopt both DSing$^{\text{train}}$ and DALI$^{\text{train}}$ to train the whole model. Since these two datasets are collected from different domains, we adopt a consecutive training strategy instead of training together in order to reduce the difficulty of training. Specifically, we continue training the whole model on DALI$^{\text{train}}$ split for 4 epochs. The weights of learnable parameters are initialized using the model trained on DSing30. After training, we evaluate the model on DALI$^{\text{test}}$ as well as the Hansen, Mauch, and Jamendo datasets. The configuration

[2] Our code is released at https://github.com/guxm2021/ALT_SpeechBrain
[3] https://huggingface.co/facebook/wav2vec2-large-960h-lv60-self

Method	DSing^{dev}	DSing^{test}	DALI^{test}	Jamendo	Hansen	Mauch
TDNN-F [5]	23.33	19.60	67.12	76.37	77.59	76.98
CTDNN-SA [6]	<u>17.70</u>	<u>14.96</u>	76.72	66.96	78.53	78.50
Genre-informed AM [3]	-	56.90	-	50.64	39.00	40.43
MSTRE-Net [7]	-	15.38	<u>42.11</u>	<u>34.94</u>	<u>36.78</u>	<u>37.33</u>
DE2 - segmented [4]	-	-	-	44.52	49.92	-
Ours	**12.34**	**12.99**	**30.85**	**33.13**	**18.71**	**28.48**

Table 2. WERs (%) of various ALT systems on different singing datasets. "-" refers to "non-applicable". We use **bold face** to highlight the best results, and <u>underline</u> to mark the second-best results. Note that the results of [5–7] on DALI^{test}, Jamendo, Hansen, and Mauch datasets are obtained without utterance segmentation.

of RNNLM is the same as above, except that we utilize texts of both DSing30 and DALI^{train} to train the model. To decode the lyrics, we set the beam size as 512 and the hyper-parameters λ_b and λ_c as 0.3 and 0.2, respectively.

5. RESULTS

In this section, we firstly compare our method with state-of-the-art ALT systems on multiple benchmark datasets. Then we conduct extensive ablation studies to show the benefits of our design choices on the DSing dataset, which has more accurate manual annotations on its development and test splits. Finally, we show that our method is still effective with a limited amount of singing data.

5.1 Comparison with the state-of-the-art

We compare the performance of the proposed method with previous approaches, as shown in Table 2. Our method outperforms all previously published results on all the evaluation datasets. Our method achieves 5.36%, 1.97%, 11.26%, 1.81%, 18.07%, 8.85% absolute WER reduction on the DSing^{dev}, DSing^{test}, DALI^{test}, Jamendo, Hansen, and Mauch datasets, compared with the best results among all the previous state-of-the-art approaches respectively. Especially, on DALI^{test}, Hansen, and Mauch datasets, our method significantly exceeds MSTRE-Net [7] by an average of 12.73% absolute WER.

5.2 Effects of pretraining & finetuning on speech data

As shown in Fig. 1, wav2vec 2.0 is pretrained and finetuned on speech data before transferring to singing data. The feature representation knowledge learned from speech data is the key to the success of our method. To validate this statement, we conduct ablation studies by comparing the proposed method to two alternative configurations. The first alternative configuration is that we randomly initialize the weights of wav2vec 2.0 and ALT head and then train the whole model on the DSing dataset as per the experiment setup in section 4.2. Note that the first alternative has no transfer learning from the speech domain (without both Stages I and II). The second alternative configuration is that we only perform pretraining on speech data [4] without

[4] https://huggingface.co/facebook/wav2vec2-large-lv60

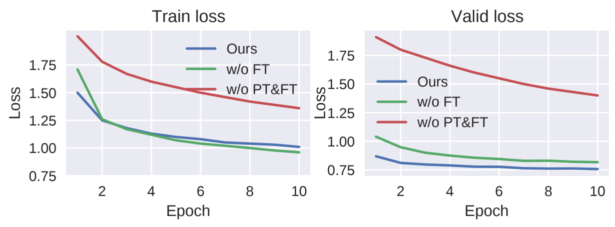

Figure 2. Comparison of different training configurations. "w/o PT & FT" refers to without Stages I and II. "w/o FT" refers to without Stage II. (Left) Training loss of all configurations for the first 10 epochs; (Right) validation loss of all configurations for the first 10 epochs.

Method	DSing^{dev}	DSing^{test}
Ours	**12.34**	**12.99**
- Finetuning	12.64 (+ 0.30)	14.58 (+ 2.59)
- Pretraining	35.61 (+24.27)	39.13 (+26.14)

Table 3. WERs (%) of different training configurations on DSing dataset.

Stage II before transferring the wav2vec 2.0 to the singing domain.

We evaluate the above training configurations on the DSing dataset. First, we show the curves of training loss and validation loss (loss of the development set) during the training for the first 10 epochs in Fig. 2. The losses are computed through Eq. 3. We observe that without Stages I and II, the training loss and validation loss are much higher than in the other two training configurations. In addition, its convergence is much slower. When we enable Stage I but disable Stage II, the behavior of the training loss is similar to the proposed configuration, except that the training loss is higher at the beginning. However, the validation loss in this setup is higher than that of the proposed configuration.

We continue training both alternatives until convergence and display the resultant performances in Table 3. We note that without Stage II, the performance drops by 0.30% higher WER on DSing^{dev} and 2.59% higher WER on DSing^{test}. Furthermore, without Stages I and II, the performance degrades severely as WERs on DSing^{dev} and

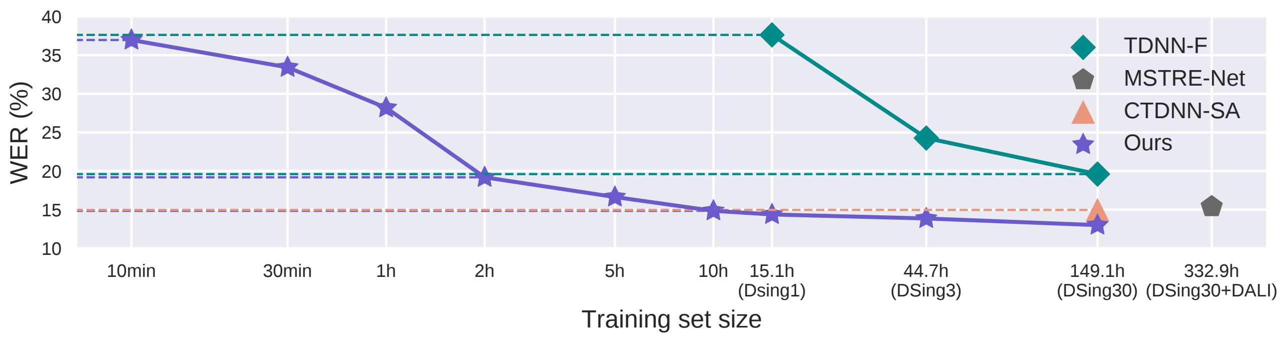

Figure 3. Transcription performance comparison when using different training set sizes, testing on the DSing^{test} dataset.

Method	DSing^{dev}	DSing^{test}
CTC	19.86	20.99
+ S2S	15.63 (-4.23)	16.95 (-4.04)
+ LM	**12.34** (-7.52)	**12.99** (-8.00)

Table 4. WERs (%) of CTC model and hybrid CTC/attention model on DSing dataset.

DSing^{test} increase by 24.27% and 26.14% respectively, compared to the proposed configuration. The results are consistent with our observations in Fig. 2. Therefore, we conclude that pretraining on speech data plays a significant role in transferring wav2vec 2.0 to the singing domain. Although finetuning on speech data is less crucial than pretraining, it also contributes to empirical performance gains.

5.3 Effects of extending CTC to CTC/attention model

To validate the effectiveness of changing from CTC to CTC/attention (as in Fig. 1), we compare the performance of our hybrid CTC/attention model with its CTC version, as shown in Table 4. We set $\lambda_b = 1$ in Eq. 5 to disable the branch of the GRU decoder and the S2S linear during the decoding. When $\lambda_c = 0$, the RNNLM is disabled.

We observe that the hybrid CTC/attention model achieves better performance by 4.23% and 4.04% WER on DSing^{dev} and DSing^{test} respectively than its CTC counterpart, which demonstrates the superiority of our ALT head design. Furthermore, we evaluate the benefits brought by the language model and find that the final model leads to 3.29% and 3.96% further absolute WER improvements compared to the hybrid CTC/attention model.

5.4 Effectiveness of transfer learning in low-resource scenarios

To explore the effectiveness of our transfer learning method in reducing the amount of required training data, we conduct an ablation study with different training set sizes. We first train our model on DSing30, DSing3, and DSing1, respectively, to observe the performance differences. Then, we further reduce the training set size to a minimum of 10 minutes to create more demanding low-resource setups. We report the WERs achieved on DSing^{test} as the performance measure. When training on 10-minute and 30-minute datasets, the GRU decoder converges too slowly; hence, WERs are computed according to the CTC outputs.

As shown in Fig. 3, our method achieves 14.84% WER with only 10 hours (about 6.7% size of DSing30) of labeled singing data, which surpasses the state-of-the-art results of 14.96% WER achieved by CTDNN-SA [6] trained on DSing30. It also has better performance with only 2 hours of training data (about 1.3% size of DSing30) than TDNN-F [5] trained on DSing30 (19.18% vs. 19.60% WER). Further, with only 10 minutes of data (1.1% size of DSing1), our method achieves better results than TDNN-F trained on DSing1 (36.97% vs. 37.63% WER). These results demonstrate the feasibility of achieving competitive results with much less training data by adopting the representation knowledge from the speech domain.

6. CONCLUSION

We have introduced a transfer learning approach for the automatic lyric transcription (ALT) task by utilizing the representation knowledge learned by self-supervised learning models on speech data. By performing parameter transfer on wav2vec 2.0 towards the singing domain and extending the original CTC model to a hybrid CTC/attention version, we achieved significant improvement compared to previous state-of-the-art methods on various singing datasets. We demonstrated that both pretraining and finetuning on speech data contribute to the final ALT performance and that pretraining brings more performance gains than finetuning on speech data. Additionally, our method still showed competitive performance using only a tiny proportion of training data, indicating its potential in low-resource scenarios.

7. ACKNOWLEDGEMENT

We would like to thank anonymous reviewers for their valuable suggestions. We also appreciate the help from Emir Demirel, Gabriel Meseguer-Brocal, Jens Kofod Hansen, Chitralekha Gupta for providing datasets. This project is funded in part by a grant (R-252-000-B78-114) from Singapore Ministry of Education.

8. REFERENCES

[1] T. Hosoya, M. Suzuki, A. Ito, S. Makino, L. A. Smith, D. Bainbridge, and I. H. Witten, "Lyrics recognition from a singing voice based on finite state automaton for music information retrieval." in *ISMIR*, 2005, pp. 532–535.

[2] H. Fujihara, M. Goto, and J. Ogata, "Hyperlinking lyrics: A method for creating hyperlinks between phrases in song lyrics." in *ISMIR*, 2008, pp. 281–286.

[3] C. Gupta, E. Yılmaz, and H. Li, "Automatic lyrics alignment and transcription in polyphonic music: Does background music help?" in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 496–500.

[4] E. Demirel, S. Ahlbäck, and S. Dixon, "Low resource audio-to-lyrics alignment from polyphonic music recordings," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 586–590.

[5] G. R. Dabike and J. Barker, "Automatic lyric transcription from karaoke vocal tracks: Resources and a baseline system." in *Interspeech*, 2019, pp. 579–583.

[6] E. Demirel, S. Ahlbäck, and S. Dixon, "Automatic lyrics transcription using dilated convolutional neural networks with self-attention," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[7] E. Demirel, S. Ahlbäck, and S. Dixon, "Mstre-net: Multistreaming acoustic modeling for automatic lyrics transcription," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021*, J. H. Lee, A. Lerch, Z. Duan, J. Nam, P. Rao, P. van Kranenburg, and A. Srinivasamurthy, Eds., 2021, pp. 151–158.

[8] X. Gao, C. Gupta, and H. Li, "Genre-conditioned acoustic models for automatic lyrics transcription of polyphonic music," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 791–795.

[9] B. Sharma and Y. Wang, "Automatic evaluation of song intelligibility using singing adapted stoi and vocal-specific features," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 319–331, 2019.

[10] L. Torrey and J. Shavlik, "Transfer learning in handbook of research on machine learning applications (eds. soria, e., martin, j., magdalena, r., martinez, m. & serrano, a.) 242–264," 2009.

[11] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[12] P. Sullivan, T. Shibano, and M. Abdul-Mageed, "Improving automatic speech recognition for non-native english with transfer learning and language model decoding," *arXiv preprint arXiv:2202.05209*, 2022.

[13] B. Zoph, D. Yuret, J. May, and K. Knight, "Transfer learning for low-resource neural machine translation," *arXiv preprint arXiv:1604.02201*, 2016.

[14] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021.

[15] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," *arXiv preprint arXiv:1904.05862*, 2019.

[16] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 449–12 460, 2020.

[17] M. Riviere, A. Joulin, P.-E. Mazaré, and E. Dupoux, "Unsupervised pretraining transfers well across languages," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7414–7418.

[18] S. Khurana, A. Laurent, and J. Glass, "Magic dust for cross-lingual adaptation of monolingual wav2vec-2.0," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 6647–6651.

[19] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, "Dali: A large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm," *arXiv preprint arXiv:1906.10606*, 2019.

[20] ——, "Creating dali, a large dataset of synchronized audio, lyrics, and notes," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, 2020.

[21] D. Stoller, S. Durand, and S. Ewert, "End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 181–185.

[22] J. K. Hansen and I. Fraunhofer, "Recognition of phonemes in a-cappella recordings using temporal patterns and mel frequency cepstral coefficients," in *9th Sound and Music Computing Conference (SMC)*, 2012, pp. 494–499.

[23] M. Mauch, H. Fujihara, and M. Goto, "Integrating additional chord information into hmm-based lyrics-to-audio alignment," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 200–210, 2011.

[24] Ccrma.Stanford.Edu, "Smule sing! 300x30x2 dataset," Sep. 2018. [Online]. Available: https://ccrma.stanford.edu/damp/

[25] S. Basak, S. Agarwal, S. Ganapathy, and N. Takahashi, "End-to-end lyrics recognition with voice to singing style transfer," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 266–270.

[26] C. Zhang, J. Yu, L. Chang, X. Tan, J. Chen, T. Qin, and K. Zhang, "Pdaugment: Data augmentation by pitch and duration adjustments for automatic lyrics transcription," *arXiv preprint arXiv:2109.07940*, 2021.

[27] X. Gu, L. Ou, D. Ong, and Y. Wang, "Mm-alt: A multimodal automatic lyric transcription system," *arXiv preprint arXiv:2207.06127*, 2022.

[28] A. M. Kruspe, "Training phoneme models for singing with "songified" speech data," in *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, M. Müller and F. Wiering, Eds., 2015, pp. 336–342.

[29] S. Pascual, M. Ravanelli, J. Serra, A. Bonafonte, and Y. Bengio, "Learning problem-agnostic speech representations from multiple self-supervised tasks," *arXiv preprint arXiv:1904.03416*, 2019.

[30] Y.-A. Chung, W.-N. Hsu, H. Tang, and J. Glass, "An unsupervised autoregressive model for speech representation learning," *arXiv preprint arXiv:1904.03240*, 2019.

[31] D. Jiang, X. Lei, W. Li, N. Luo, Y. Hu, W. Zou, and X. Li, "Improving transformer-based speech recognition using unsupervised pre-training," *arXiv preprint arXiv:1910.09932*, 2019.

[32] R. Zhang, H. Wu, W. Li, D. Jiang, W. Zou, and X. Li, "Transformer based unsupervised pre-training for acoustic representation learning," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6933–6937.

[33] L. Pepino, P. Riera, and L. Ferrer, "Emotion recognition from speech using wav2vec 2.0 embeddings," *arXiv preprint arXiv:2104.03502*, 2021.

[34] J. Zhao, R. Li, Q. Jin, X. Wang, and H. Li, "Memobert: Pre-training model with prompt-based learning for multimodal emotion recognition," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 4703–4707.

[35] D. Seo, H.-S. Oh, and Y. Jung, "Wav2kws: Transfer learning from speech representations for keyword spotting," *IEEE Access*, vol. 9, pp. 80 682–80 691, 2021.

[36] Z. Fan, M. Li, S. Zhou, and B. Xu, "Exploring wav2vec 2.0 on speaker verification and language identification," *arXiv preprint arXiv:2012.06185*, 2020.

[37] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[38] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[40] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

[41] Q. Xu, A. Baevski, T. Likhomanenko, P. Tomasello, A. Conneau, R. Collobert, G. Synnaeve, and M. Auli, "Self-training and pre-training are complementary for speech recognition," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3030–3034.

[42] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid ctc/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

[43] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," *Advances in neural information processing systems*, vol. 28, 2015.

[44] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2016, pp. 4945–4949.

[45] A. Défossez, "Hybrid spectrogram and waveform source separation," in *Proceedings of the ISMIR 2021 Workshop on Music Source Separation*, 2021.

[46] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong *et al.*, "Speechbrain: A general-purpose speech toolkit," *arXiv preprint arXiv:2106.04624*, 2021.

[47] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[48] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

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-

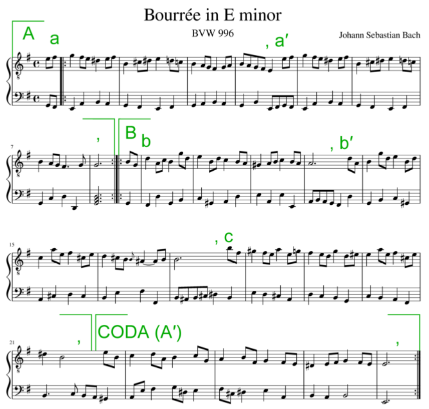

Figure 1. Example of phrase labeling from analysis of Bourrée from J.S. Bach's BWV 996 on a human-annotated score, where the relation between phrases and their respective part can be seen hierarchically. On an analyzed score, it is standard that only the first instance of a structure is labeled, although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