3. BAF DATASET

This section describes the characteristics of BAF: Broadcast Audio Fingerprinting dataset. It is the only available dataset designed for broadcast monitoring. BAF contains TV recordings, reference tracks, and annotations done by 6 different annotators that cross-annotated matching queries and references. It is a self-contained dataset available upon user's access request. Open for non-commercial, research-only, with no adaptations or derivative works allowed and proper attribution. It must not be used for music generation or music synthesis research. Towards addressing ethical and sustainability concerns, we distribute a datasheet using the format proposed by Gebru et al. [39] with practical and detailed information about the dataset. Audio files, annotations, and the dataset datasheet are hosted in Zenodo [1] while the baseline code and evaluation scripts are in Github [2].

3.1 Methodology

The reference set contains 2,000 production music tracks (74 hours of audio) obtained directly from the Epidemic Sound [9] private catalog, described in Section 3.3.2. The queries are initially derived from TV stream monitoring on 478 TV channels from 43 countries, for a 2.5 months period at stereo maximum quality. We extract the audio with FFMPEG and we split the large audio files into 1-minute length queries.

As an automatic pre-annotation stage, we match queries with references relying on a proprietary stereo matching algorithm developed by BMAT. The algorithm is non-scalable and it relies on spectral peaks matching using stereo signals. It has been tailored to avoid false negatives, disregarding the presence of false positives and low computational efficiency. We then select all query segments that have at least one pre-annotation match. In addition, we discard queries matching more than 3 unique references since most of them contain false positives, which would be manually deleted during the later annotation process. Then, we shuffle all queries and select 3,425 of them, corresponding to 57 hours of TV broadcast audio from 203 TV channels across 23 countries. Finally, we convert all audios to publishing format: 8kHz mono, WAV `pcm_s16le`, a common specification in AFP [2, 14, 15, 34].

3.2 Annotation criteria

BAF has been annotated by six different annotators in a controlled environment. We built an in-house annotation web app in Django, secured by user credentials, in order to facilitate the annotation to remote users. The app displays all segments resulting from the automatic pre-annotation stage. Annotators were instructed to listen to the query and reference pairs, filter out false positives, and adjust the start and end times of the true positives with deciseconds precision. Queries with slight alterations with respect to the reference have also been annotated as true positives, such as versions with some missing stem (e.g. instrumental vs vocal), or 'edit' versions.

We ensured that each segment was annotated by three different annotators (sets of annotators created by random combination). We then created the cross-annotations with 3 different levels of agreement: *single, majority, unanimity*. These cross-annotations are the result of splitting annotations into segments and merging overlapping segments, assigning a tag depending on how many annotators marked a match in that time interval (1, 2, and all 3, respectively). Out of the 57 hours of queries, over 37 hours were marked as true positive by at least 1 annotator.

[1] `https://doi.org/10.5281/zenodo.6868083`
[2] `https://github.com/guillemcortes/baf-dataset`

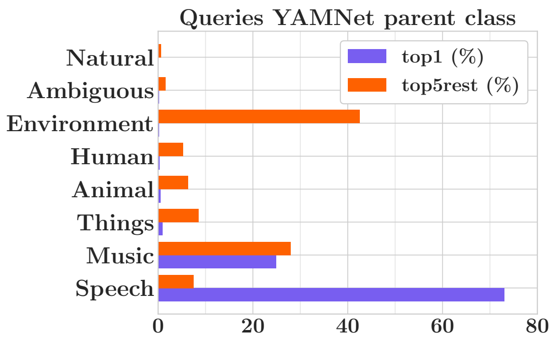

Figure 1. YAMNet classification of BAF queries. YAMNet uses a sliding window of length 0.96s and stride of 0.48s to generate one prediction for each step. Percentages are relative to each top classification.

3.3 Analysis

3.3.1 *Queries*

We have used the YAMNet sound event classifier [40] to study the most common sounds of BAF queries. YAMNet is a pretrained deep network that predicts 521 audio event classes based on the AudioSet-YouTube corpus [41]. Audioset ontology follows a tree structure so all classes are gathered under 7 different parent classes [42], from that, we extract *Speech* from *Human* as a separate class to better evaluate its presence. Figure 1 reflects that YAMNet's most predominant class is *Speech*, with 73% of the output predictions while only nearly 25% of them correspond to *Music*. Regarding the *top5rest* classes, *Environment* and *Music* are the most predominant, which means that noises and background music are common in the broadcast.

To obtain the YAMNet distributions we first run YAMNet for all queries and then select the outputs corresponding to the annotated segments. After that, we average the scores using a moving average window of length 2 to soften noisy scores. Lastly, we extract the top1 and top5 distributions and translate all labels to the corresponding parent class, following Audioset ontology. For each window, we remove from the top5 the parent class that matches that window's top1 parent class so with this, only the classes that are detected in the background remain. We name this distribution top5rest.

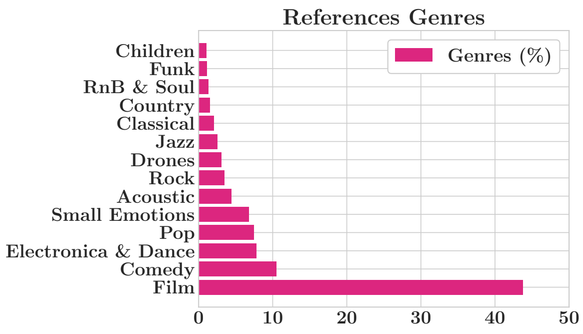

Figure 2. BAF references music genres. Only genres representing more than 1% are listed, the remainder represents 3% of the dataset.

In addition YAMNet predictions we used the MIREX 2019 Music Detection winner model [43, 44] to analyze BAF queries. The system is based on computing the Relative Music Loudness distribution [45] and classifies as *Foreground Music* only 18.07% of the total cross-annotated segments with the tag *unanimity* (segments all annotators agreed that there's music). Verifying the high presence of background music.

3.3.2 *References*

The BAF reference set is a selection of production music tracks from Epidemic Sound's private catalog of 35,000+ human-annotated tracks. In order to give valuable insights about instrumentalization, BPM, or genre, we have analyzed the tags and found that 7% of the selected tracks contain vocal elements like singing, while the remaining 93% are instrument-only versions. Figure 2 shows how the majority of the references are categorized as *Film music*, and cover styles/moods like Suspense, Drama, Build, Pulses, Small Emotions, or Solo Piano. Common keywords or tags found in this set of tracks include: comedic, piano, strings, driving, tension, corporate, guitar, and documentary. The references BPMs follow a Normal Distribution $\mathcal{N}(\mu, \sigma^2)$ with mean $\mu = 109$ and standard deviation $\sigma = 33$.

3.3.3 *Matches / Annotations*

Table 2 shows that 90.41% of the total annotated time (133,846 seconds) has been agreed by unanimity. Most of the differences between annotators come from divergences in start and end matching timestamps partially due to the difficulty to tell when a song starts or ends in a stream, especially with background music.

Additionally, to evaluate the reliability of annotators' agreement we compute the Fleiss' Kappa [46] indicator, a statistical measure of inter-rater reliability. It needs fixed-length elements to categorize them so we computed the Fleiss' Kappa indicator using 0.05 seconds length annotations. We obtained a factor of 0.9364 that can be interpreted as an almost perfect agreement [47].

3.4 Limitations

The size of the reference set is not large enough to mimic real production environments, where recordings are expected to be analyzed against tens of millions of tracks [25]. For future work, in order to study thoroughly the impact of False Positives (FP), a set of additional *noise* tracks should be added to the reference set. For this, public datasets mentioned in Section §2 could be used.

4. BENCHMARK

We benchmark the following available AFP algorithms: Audfprint [10], Panako [11, 12], Olaf [13], and NeuralFP [14] to study the performance of AFP in the broadcast monitoring use case. Additionally, we also propose a simple baseline that gives context to the results.

Audfprint is based on Shazam's algorithm [2]. It uses the locations of pairs of spectrogram peaks (local maxima points) as robust features for matching. **Panako** extends Shazam fingerprint by saving triplets of local maxima points in Constant-Q non-stationary Gabor transform. It uses time ratios to form a time-scale invariant fingerprint component. These components are hashed alongside a coarse value of the frequency position of the triplet, making it robust to time-scale and pitch modifications. **Olaf** is a lightweight AFP algorithm able to run in embedded systems. In the benchmarked version, it uses absolute exact frequencies and timestamps in the hash (like Shazam [2]) of triplets of peaks (like Panako [11]). It is not robust to pitch shifting or time distortions. **NeuralFP** is based on deep neural networks and created for high-specific audio retrieval using contrastive learning. It creates pairs of data applying distortions to short audio snippets so each batch of training data consists of randomly selected original samples and their augmented replicas. Then, it maximizes the inner product between pairs.

All algorithms except NeuralFP ran as they are published. NeuralFP, though, required changes in the indexing and matching modules to match the broadcast monitoring use case. The indexer now stores indexes on disk rather than on a memory map, and the matcher integrates the Maximum Inner Product Search used by the authors into PeakFP matcher pipeline, to be able to give start and end times of each match. The implementation [3] has been

[3] `https://github.com/guillemcortes/neural-audio-fp`

Class	%
single	3.69%
majority	5.90%
unanimity	90.41%

Table 2. Annotators agreement in percentage of annotation time length. *single* correspond to intervals where only 1 of the 3 annotators marked a match. In *majority* 2/3 annotators agreed while in *unanimity* there's full agreement.

validated by NeuralFP authors. Towards a fair comparison between systems, all algorithms return top1 matches. Audfprint and Olaf use the default configurations, Panako parameters are adjusted for 8kHz input signal and NeuralFP needs extra parameters for the custom matcher pipeline. Additionally, we study the impact of fingerprint density by increasing Audfprint (x2) and Panako (x1.5) peak density and also test two additional NeuralFP models *spcm1510* and *spc3000* that were trained with different levels of speech intensity [-15, 10] dB and [0, 10] dB, respectively. All configuration parameters used in this publication have been discussed with the authors of each respective algorithm and are available in the publication git repository.

Apart from the algorithms mentioned above, there are some others that we would have liked to benchmark, but no official public implementation of them was found. That is Fenet's et al. CQT approach [20], Google's Now Playing [26] lightweight, neural network-based, continuous monitoring system, and also Son's et al. FFMAP-based algorithm [27,48]. For QuadFP [23,49] and Waveprint [17] we tried to run third-party implementations but without success. We also plan to include Chromaprint [50], a public AFP implementation based on Ke et al. computer vision approach for music identification [51], and other algorithms to the benchmark.

4.1 Baseline: PeakFP

Available AFP systems aim at obtaining the best algorithm in terms of robustness, scalability, efficiency, etc. However, many existing systems are distributed as closed software packages or embedded into complicated frameworks which are difficult to adapt. Also, the literature lacks a simple and easy-to-use baseline that may be used as a starting point in AFP research. We address these issues by introducing PeakFP, a simple, open, non-data-driven, non-scalable, AFP algorithm based on spectral peak matching that it has been designed to be as simple as possible and not optimized for scalability, but at the same time, useful for detecting background music. While algorithms commonly use pairs or triplets of spectral peaks, PeakFP uses single-peak matching because pairs of spectral peaks are more prone to break when the SNR of the music signal is low due to music peaks being masked by other sounds. As a consequence, PeakFP is ineffective in front of pitch-shifting or time-scaling distortions.

PeakFP is divided into three modules: extractor, indexer, and matcher. They are designed to work independently following a simple pipeline we detail below. Note that the code of our implementation is available online (see Section 3). The extraction process involves finding peaks in a monaural audio magnitude spectrogram using a 2D max filter. Then, all the peaks (time-frequency tuples) are sorted by the time frame index and saved in a serialized binary file. The reference signature files are used to generate an inverted index on the peak frequency values. For a given frequency, the index contains the list of all occurrences of that frequency in every reference. The hash space comprises all the possible frequency values. This small hash space translates to a high quantity of hash matches that yield a high number of comparisons in the matching step. The matcher splits the peaks of the query recordings using a sliding temporal window defined by window length and hop size. Then, for all windows, it counts the common peaks for a specific query-reference time alignment similarly to Shazam [2]. After that, all matches go through a postprocessing stage in order to consolidate and resolve overlapping matches.

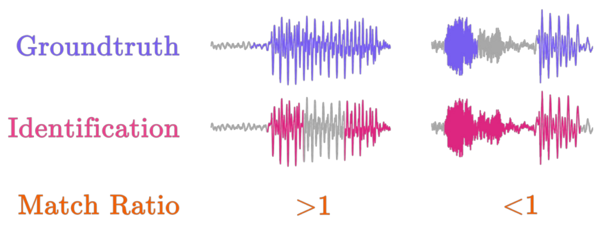

Figure 3. Proposed metric Match ratio. It defines the ratio between the number of identifications with respect to the number of annotations in the groundtruth.

4.2 Evaluation Metrics

A variety of metrics are used in the AFP literature for benchmarking and comparing algorithms, most of them based on classifying predictions into false positives/negatives or true labels [52], and using metrics derived from information retrieval such as true positive rate [11] precision, recall, and specificity [2,23]. Other papers use Top-1 Hit Rate [14] or TRECVID's proposed evaluation metric Normalized Detection Cost Rate (NDCR) [33–35].

In broadcast monitoring, it is typically required to provide exact start and end matching timestamps for each reference identification. For this reason, we propose to use the percentage of identified seconds alongside to match classification into Precision, Recall, and F1-score.

Some algorithms are prone to give short overly-split matches, while others tend to generate longer matches that include gaps without annotations. Towards quantifying this we introduce a new metric *Match Ratio*, defined in equation 1 and depicted in Figure 3, that represents the ratio between the total correct identified segments (TP matches), and the total unique annotations identified, groundtruth (GT) segments.

$$\text{Match Ratio} = \frac{\#\ \text{TP segments ID}}{\#\ \text{TP segments GT}} \tag{1}$$

A Match Ratio value bigger than 1 means that some of the identifications belong to the same annotation. Conversely, a value lower than 1 indicates that the algorithm generates a single identification for a segment in which there is more than one unique identification according to the groundtruth. The value for an algorithm that perfectly matches the annotations is 1.

Algorithm	Match Ratio	# matches			seconds identified		
		Precision	Recall	F1-score	Precision	Recall	F1-score
PeakFP	1.64	.96	**.70**	**.81**	.96	**.32**	**.47**
Panako2.0	1.85	**.98**	.33	.49	**.98**	.06	.12
Panako2.0 (x1.5)	2.12	.70	.60	.64	.69	.15	.25
Olaf	1.95	**.98**	.25	.39	**.98**	.06	.11
NeuralFP	1.39	.22	.30	.25	.37	.10	.15
NeuralFP-spcm1510	1.56	.23	.56	.33	.38	.22	.28
NeuralFP-spc3000	1.40	.69	.38	.49	.83	.13	.22
Audfprint	N/A*	.76	.05	.10	.86	.02	.04
Audfprint (x2)	N/A*	.71	.10	.17	.81	.04	.08

Table 3. Benchmark results on *unanimity* annotations. *Audfprint reports 1 match per query by default.

4.3 Results

Table 3 summarizes the performance of all benchmarked algorithms on *unanimity* annotations. All systems increase their Precision when considering the identified seconds because the False Positives identifications are shorter than the True Positives. At the same time, Recall decreases because the identifications are partial and do not cover the full annotation groundtruth. Hence the importance of studying also the performance in terms of identified seconds.

All algorithms except Audfprint obtain a Match Ratio > 1. This is caused because algorithms tend to detect small excerpts of the music (parts where the music SNR is higher) resulting in more than one identification per annotation. Audfprint only returns one identification per query by default, so its Match Ratio will always be 1 with this configuration. This also means that if a query has more than one annotation, Audfprint can't identify all of them.

The good Precision results (PeakFP, Panako, Olaf obtain over 0.96) should be analyzed taking into account that BAF is not challenging to False Positives since the reference set is limited to 2,000 references. The low Recall values show that there are a lot of identifications missed (False Negatives), manifesting that algorithms do not work well with background music or broadcast distortions. Increasing fingerprint density improves the F1-score a bit. It helps to boost the Recall in both Panako (x1.5) and Audfprint (x2) but at expense of Precision.

Additionally, to frame the computational cost of each algorithm, we have benchmarked their extraction, indexing, and matching times as well as index size. Table 4 shows that Olaf is the fastest benchmarked system. On the other hand, PeakFP is the slowest even though the extraction process is quick due to its simplicity, but the reduced hash space yields a high collision of hashes that slows down the matching. NeuralFP extraction process takes a lot of time compared to others, mainly due to the deep neural network model complexity. This process can be sped up by running it on a GPU, where deep learning models operate more efficiently. For both Panako and Audfprint, the matching step takes x2.5 times longer than the extraction.

Algorithm	Extraction & Indexing	Matching	Index size
Olaf	53m	3h 30m	349 MB
Panako 2.0	2h 17m	5h 24m	273 MB
NeuralFP	49h 34m	9h 30m	37 MB
Audfprint	9h 50m	23h 01m	19 MB
PeakFP	50m	98h 39m	160 MB

Table 4. Computational cost benchmark.

Regarding the index sizes, Audfprint and NeuralFP generate the smallest indexes (19MB and 37 MB) to represent a set of 2,000 references (74 hours of audio). Olaf generates the biggest index with almost 350 MB, 18 times the size of the Audfprint index. Olaf would take around 1.75 TB for an industrial database of 10 million references while Audfprint would take 95 GB.

Experiments have been run in a reproduceable, isolated environment. All audios have been loaded on the RAM disk and ran on a 98GB RAM server with two 16-cores CPUs at 2.60GHz. We have executed each algorithm with multiprocessing (when it was possible) and cleared the cache before each run. The results in Table 4 are normalized to one single thread.

5. CONCLUSIONS AND FUTURE WORK

We present a new dataset for broadcast monitoring with 57 hours of TV broadcast recordings, 74 hours of production music, and over 37 hours of human cross-annotations. All audios are monaural sampled at 8kHz and more than 80% of the annotated music is in the background. The Benchmark of state-of-the-art public algorithms shows that AFP for broadcast monitoring with a high presence of background music is yet to be solved. For this task, in addition to other metrics used in the literature, we propose using the Match Ratio and analyzing Precision, Recall, and F1-score not only for numbers of matches but also for seconds identified. We also provide a simple AFP baseline.

In future experiments, we plan to add noise tracks to the references set to study the evolution of False Positives and the scalability of each algorithm. We will get more insight into their behavior for different levels of background music, and we will benchmark more AFP algorithms.

6. ACKNOWLEDGEMENTS

The authors would like to thank Carl Thomé for making this collaboration possible. This research is part of *NextCore – New generation of music monitoring technology (RTC2019-007248-7)*, funded by the Spanish Ministerio de Ciencia e Innovación and the Agencia Estatal de Investigación. Also, has received support from Industrial Doctorates plan of the Secretaria d'universitats i Recerca, Departament d'Empresa i Coneixement de la Generalitat de Catalunya, grant agreement No. DI46-2020.

7. REFERENCES

[1] P. Cano, E. Batlle, E. Gómez, L. de C. T. Gomes, and M. Bonnet, "Audio fingerprinting: Concepts and applications," in *Computational Intelligence for Modelling and Prediction*, ser. Studies in Computational Intelligence, S. K. Halgamuge and L. Wang, Eds. Springer, 2005, vol. 2, pp. 233–245. [Online]. Available: https://doi.org/10.1007/10966518_17

[2] A. Wang, "An industrial strength audio search algorithm," in *ISMIR 2003, 4th International Conference on Music Information Retrieval, Baltimore, Maryland, USA, October 27-30, 2003, Proceedings*, 2003, pp. 7–13.

[3] J. R. Cerquides, "A real time audio fingerprinting system for advertisement tracking and reporting in fm radio," in *2007 17th International Conference Radioelektronika*. IEEE, 2007, pp. 1–4.

[4] E. Gomez, P. Cano, L. Gomes, E. Batlle, and M. Bonnet, "Mixed watermarking-fingerprinting approach for integrity verification of audio recordings," in *Proceedings of the International Telecommunications Symposium*, 2002.

[5] C. J. C. Burges, D. Plastina, J. C. Platt, E. Renshaw, and H. S. Malvar, "Using audio fingerprinting for duplicate detection and thumbnail generation," in *2005 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '05, Philadelphia, Pennsylvania, USA, March 18-23, 2005*. IEEE, 2005, pp. 9–12. [Online]. Available: https://doi.org/10.1109/ICASSP.2005.1415633

[6] IFPI, "Global music report 2022," https://www.ifpi.org/wp-content/uploads/2022/04/IFPI_Global_Music_Report_2022-State_of_the_Industry.pdf, 4 2022, [Accessed August 2022].

[7] B. Meléndez Catalán *et al.*, "Relative music loudness estimation in tv broadcast audio using deep learning: an industrial perspective," Ph.D. dissertation, Universitat Pompeu Fabra, 2021.

[8] A. G. Piotrowska, "Analyzing music in tv shows: some methodological considerations," *The science of television*, no. 14.3, pp. 10–27, 2018.

[9] "Epidemic sound," https://www.epidemicsound.com/, 2009, [Accessed August 2022].

[10] D. Ellis, "The 2014 labrosa audio fingerprint system," in *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, TW, October 27-31, 2014*, 2014.

[11] J. Six and M. Leman, "Panako - A scalable acoustic fingerprinting system handling time-scale and pitch modification," in *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014*, H. Wang, Y. Yang, and J. H. Lee, Eds., 2014, pp. 259–264. [Online]. Available: http://www.terasoft.com.tw/conf/ismir2014/proceedings/T048_122_Paper.pdf

[12] J. Six, "Panako 2.0-updates for an acoustic fingerprinting system," in *Demo / late-breaking abstracts of 22st International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 8-12, 2021*, 2021.

[13] ——, "Olaf: Overly lightweight acoustic fingerprinting," in *Demo / late-breaking abstracts of 21st International Society for Music Information Retrieval Conference, ISMIR 2020, Montréal, Canada, CA, October 11-16, 2020*, 2020.

[14] S. Chang, D. Lee, J. Park, H. Lim, K. Lee, K. Ko, and Y. Han, "Neural audio fingerprint for high-specific audio retrieval based on contrastive learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021*. IEEE, 2021, pp. 3025–3029. [Online]. Available: https://doi.org/10.1109/ICASSP39728.2021.9414337

[15] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system," in *ISMIR 2002, 3rd International Conference on Music Information Retrieval, Paris, France, October 13-17, 2002, Proceedings*, 2002, pp. 107–115. [Online]. Available: http://ismir2002.ismir.net/proceedings/02-FP04-2.pdf

[16] M. A. Bartsch and G. H. Wakefield, "Audio thumbnailing of popular music using chroma-based representations," *IEEE Trans. Multim.*, vol. 7, no. 1, pp. 96–104, 2005. [Online]. Available: https://doi.org/10.1109/TMM.2004.840597

[17] S. Baluja and M. Covell, "Waveprint: Efficient wavelet-based audio fingerprinting," *Pattern Recognit.*, vol. 41, no. 11, pp. 3467–3480, 2008. [Online]. Available: https://doi.org/10.1016/j.patcog.2008.05.006

[18] C. Bellettini and G. Mazzini, "Reliable automatic recognition for pitch-shifted audio," in *Proceedings of the 17th International Conference on Computer Communications and Networks, IEEE ICCCN 2008, St. Thomas, U.S. Virgin Islands, August 3-7, 2008.*

IEEE, 2008, pp. 838–843. [Online]. Available: https://doi.org/10.1109/ICCCN.2008.ECP.157

[19] ——, "A framework for robust audio fingerprinting," *J. Commun.*, vol. 5, no. 5, pp. 409–424, 2010. [Online]. Available: https://doi.org/10.4304/jcm.5.5.409-424

[20] S. Fenet, G. Richard, and Y. Grenier, "A scalable audio fingerprint method with robustness to pitch-shifting," in *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011*, A. Klapuri and C. Leider, Eds. University of Miami, 2011, pp. 121–126. [Online]. Available: http://ismir2011.ismir.net/papers/PS1-14.pdf

[21] M. Malekesmaeili and R. K. Ward, "A local fingerprinting approach for audio copy detection," *Signal Process.*, vol. 98, pp. 308–321, 2014. [Online]. Available: https://doi.org/10.1016/j.sigpro.2013.11.023

[22] X. Zhang, B. Zhu, L. Li, W. Li, X. Li, W. Wang, P. Lu, and W. Zhang, "Sift-based local spectrogram image descriptor: a novel feature for robust music identification," *EURASIP J. Audio Speech Music. Process.*, vol. 2015, p. 6, 2015. [Online]. Available: https://doi.org/10.1186/s13636-015-0050-0

[23] R. Sonnleitner and G. Widmer, "Robust quad-based audio fingerprinting," *IEEE ACM Trans. Audio Speech Lang. Process.*, vol. 24, no. 3, pp. 409–421, 2016. [Online]. Available: https://doi.org/10.1109/TASLP.2015.2509248

[24] R. Sonnleitner, A. Arzt, and G. Widmer, "Landmark-based audio fingerprinting for DJ mix monitoring," in *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016*, M. I. Mandel, J. Devaney, D. Turnbull, and G. Tzanetakis, Eds., 2016, pp. 185–191. [Online]. Available: https://wp.nyu.edu/ismir2016/wp-content/uploads/sites/2294/2016/07/187_Paper.pdf

[25] T. Walther and M. D. Gould, "Audio identification method," Worldwide Patent WO/2016/189 307, 2016.

[26] B. A. y Arcas, B. Gfeller, R. Guo, K. Kilgour, S. Kumar, J. Lyon, J. Odell, M. Ritter, D. Roblek, M. Sharifi, and M. Velimirovic, "Now playing: Continuous low-power music recognition," *CoRR*, vol. abs/1711.10958, 2017. [Online]. Available: http://arxiv.org/abs/1711.10958

[27] H. Son, S. Byun, and S. Lee, "A robust audio fingerprinting using a new hashing method," *IEEE Access*, vol. 8, pp. 172 343–172 351, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3024951

[28] Z. Yu, X. Du, B. Zhu, and Z. Ma, "Contrastive unsupervised learning for audio fingerprinting," *CoRR*, vol. abs/2010.13540, 2020. [Online]. Available: https://arxiv.org/abs/2010.13540

[29] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, "Evaluation of algorithms using games: The case of music tagging," in *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, Kobe International Conference Center, Kobe, Japan, October 26-30, 2009*, K. Hirata, G. Tzanetakis, and K. Yoshii, Eds. International Society for Music Information Retrieval, 2009, pp. 387–392. [Online]. Available: http://ismir2009.ismir.net/proceedings/OS5-5.pdf

[30] M. Ramona and G. Peeters, "Audio identification based on spectral modeling of bark-bands energy and synchronization through onset detection," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic*. IEEE, 2011, pp. 477–480. [Online]. Available: https://doi.org/10.1109/ICASSP.2011.5946444

[31] A. F. Smeaton, P. Over, and W. Kraaij, "Evaluation campaigns and trecvid," in *Proceedings of the 8th ACM SIGMM International Workshop on Multimedia Information Retrieval, MIR 2006, October 26-27, 2006, Santa Barbara, California, USA*, J. Z. Wang, N. Boujemaa, and Y. Chen, Eds. ACM, 2006, pp. 321–330. [Online]. Available: https://doi.org/10.1145/1178677.1178722

[32] G. Awad, P. Over, and W. Kraaij, "Content-based video copy detection benchmarking at TRECVID," *ACM Trans. Inf. Syst.*, vol. 32, no. 3, pp. 14:1–14:40, 2014. [Online]. Available: https://doi.org/10.1145/2629531

[33] X. Anguera, A. Garzon, and T. Adamek, "MASK: robust local features for audio fingerprinting," in *Proceedings of the 2012 IEEE International Conference on Multimedia and Expo, ICME 2012, Melbourne, Australia, July 9-13, 2012*. IEEE Computer Society, 2012, pp. 455–460. [Online]. Available: https://doi.org/10.1109/ICME.2012.137

[34] C. Ouali, P. Dumouchel, and V. Gupta, "A robust audio fingerprinting method for content-based copy detection," in *12th International Workshop on Content-Based Multimedia Indexing, CBMI 2014, Klagenfurt, Austria, June 18-20, 2014*. IEEE, 2014, pp. 1–6. [Online]. Available: https://doi.org/10.1109/CBMI.2014.6849814

[35] Z. J. Guzman-Zavaleta, C. F. Uribe, A. Menendez-Ortiz, and J. J. Garcia-Hernandez, "A robust audio fingerprinting method using spectrograms saliency maps," in *9th International Conference for Internet Technology and Secured Transactions, ICITST 2014, London, United Kingdom, December 8-10, 2014*. IEEE, 2014, pp. 47–52. [Online]. Available: https://doi.org/10.1109/ICITST.2014.7038773

[36] NIST-TRECVID, "Trecvid 2011 - content-based copy detection task," https://www-nlpir.nist.gov/projects/

tv2011/index.html#ccd, 2010, [Accessed August 2022].

[37] J. D. of Computational Perception, "Audio fingerprinting data sets," http://www.cp.jku.at/datasets/fingerprinting, 4 2017, [Accessed August 2022].

[38] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: A dataset for music analysis," in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, S. J. Cunningham, Z. Duan, X. Hu, and D. Turnbull, Eds., 2017, pp. 316–323. [Online]. Available: https://ismir2017.smcnus.org/wp-content/uploads/2017/10/75_Paper.pdf

[39] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. M. Wallach, H. D. III, and K. Crawford, "Datasheets for datasets," *Commun. ACM*, vol. 64, no. 12, pp. 86–92, 2021. [Online]. Available: https://doi.org/10.1145/3458723

[40] Tensorflow, "Sound classification with yamnet," https://www.tensorflow.org/hub/tutorials/yamnet, 2022, [Accessed August 2022].

[41] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*. IEEE, 2017, pp. 776–780. [Online]. Available: https://doi.org/10.1109/ICASSP.2017.7952261

[42] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audioset ontology," https://research.google.com/audioset/ontology/index.html, 2017, [Accessed August 2022].

[43] MIREX, "2019: Music detection results," https://www.music-ir.org/mirex/wiki/2019:Music_Detection_Results, 2019, [Accessed August 2022].

[44] B. Meléndez-Catalán, "Relative music loudness estimation using temporal convolutional networks and a cnn feature extraction front-end," in *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx-20)*, vol. 5, 2020, pp. 273–280.

[45] B. Meléndez-Catalán, E. Molina, and E. Gómez, "Open broadcast media audio from TV: A dataset of TV broadcast audio with relative music loudness annotations," *Trans. Int. Soc. Music. Inf. Retr.*, vol. 2, no. 1, pp. 43–51, 2019. [Online]. Available: https://doi.org/10.5334/tismir.29

[46] J. L. Fleiss, "Measuring nominal scale agreement among many raters," *Psychological bulletin*, vol. 76, no. 5, p. 378, 1971.

[47] A. J. Viera, J. M. Garrett *et al.*, "Understanding interobserver agreement: the kappa statistic," *Fam med*, vol. 37, no. 5, pp. 360–363, 2005.

[48] H. Son, S. W. Byun, and S. Lee, "Illegal audio copy detection using fundamental frequency map," in *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications, ICETE 2019 - Volume 1: DCNET, ICE-B, OPTICS, SIGMAP and WINSYS, Prague, Czech Republic, July 26-28, 2019*, M. S. Obaidat, C. Callegari, M. van Sinderen, P. Novais, P. G. Sarigiannidis, S. Battiato, Á. S. S. de León, P. Lorenz, and F. Davoli, Eds. SciTePress, 2019, pp. 356–361. [Online]. Available: https://doi.org/10.5220/0008113403500355

[49] R. Sonnleitner and G. Widmer, "Quad-based audio fingerprinting robust to time and frequency scaling," in *Proceedings of the 17th International Conference on Digital Audio Effects, DAFx-14, Erlangen, Germany, September 1-5, 2014*, S. Disch, J. Herre, R. Rabenstein, B. Edler, M. Müller, and S. Turowski, Eds., 2014, pp. 173–180. [Online]. Available: http://www.dafx14.fau.de/papers/dafx14_reinhard_sonnleitner_quad_based_audio_fingerpr.pdf

[50] L. Lalinský, "Chromaprint," https://acoustid.org/chromaprint, [Accessed August 2022].

[51] Y. Ke, D. Hoiem, and R. Sukthankar, "Computer vision for music identification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA*. IEEE Computer Society, 2005, pp. 597–604. [Online]. Available: https://doi.org/10.1109/CVPR.2005.105

[52] M. Ramona and G. Peeters, "Audioprint: An efficient audio fingerprint system based on a novel cost-less synchronization scheme," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*. IEEE, 2013, pp. 818–822. [Online]. Available: https://doi.org/10.1109/ICASSP.2013.6637762

CADENCE DETECTION IN SYMBOLIC CLASSICAL MUSIC USING GRAPH NEURAL NETWORKS

Emmanouil Karystinaios[1] **Gerhard Widmer**[1,2]
[1] Institute of Computational Perception, Johannes Kepler University Linz, Austria
[2] LIT AI Lab, Linz Institute of Technology, Austria
firstname.lastname@jku.at

ABSTRACT

Cadences are complex structures that have been driving music from the beginning of contrapuntal polyphony until today. Detecting such structures is vital for numerous MIR tasks such as musicological analysis, key detection, or music segmentation. However, automatic cadence detection remains challenging mainly because it involves a combination of high-level musical elements like harmony, voice leading, and rhythm. In this work, we present a graph representation of symbolic scores as an intermediate means to solve the cadence detection task. We approach cadence detection as an imbalanced node classification problem using a Graph Convolutional Network. We obtain results that are roughly on par with the state of the art, and we present a model capable of making predictions at multiple levels of granularity, from individual notes to beats, thanks to the fine-grained, note-by-note representation. Moreover, our experiments suggest that graph convolution can learn non-local features that assist in cadence detection, freeing us from the need of having to devise specialized features that encode non-local context. We argue that this general approach to modeling musical scores and classification tasks has a number of potential advantages, beyond the specific recognition task presented here.

1. INTRODUCTION

Graph Neural Networks (GNNs) have recently seen staggering successes in various fields. The MIR community has also experienced the influence of GNNs, principally in the field of recommender systems [1]. However, other sub-branches of MIR could potentially enjoy the graph representation and the benefits of graph deep learning.

Modeling musical scores in all their complexity has been challenging, with many approaches resorting to piano rolls [2], note arrays [3], or custom descriptors [4]. In this paper, we present a new representation of the score as a homogeneous graph with note-wise features to model aspects of the score. We use this representation to address the cadence detection task using graph neural networks, treating the task as a node classification problem. More specifically, our contribution is two-fold: a simple graph representation of scores extended with local features, and a Graph Convolutional Network (GCN) model to tackle heavily imbalanced classification tasks such as Cadence Detection. *Score modeling* itself has two aspects: (1) the construction of the graph, i.e., what are the nodes, and which connections do we define between them; and (2) the choice of score features, and how these relate to their respective graph nodes. The *classification model* is an adapted version of GraphSMOTE [5], a Graph Convolutional Network designed to deal with imbalanced classification problems, which we modified to deal with larger graphs and apply stochastic training. Henceforth, we call this model *Stochastic GraphSMOTE*. We employ this model on top of our score modeling with the intention of solving the Cadence Detection task.

The cadence detection setting is binary, i.e., there is a cadence (maybe of a specific type) or not. The current state of the art [4] uses an Support Vector Machine (SVM) classifier on a set of custom-designed cadence-specific features, based on three defined "cadence anchor points", and performs score/feature modeling and cadence classification at the level of beats. The model was tested on two annotated datasets: 24 Bach fugues and 42 Haydn string quartet expositions. Our new model proposed here will be shown to achieve comparable overall results; however, we will argue that it makes fewer task-related and musical assumptions, resulting in more general applicability. In particular, our empirical results suggest that by providing local features and applying a Graph Neural Network with neighbor convolution, we can learn nonlocal aspects that help improve prediction. This gives a more general approach for a variety of tasks where features are provided at the level of notes, but prediction may be note-wise, onset-wise, or beat-wise.

The rest of the paper is structured as follows. Section 2 discusses related work on cadence detection and music score modeling. Section 3 describes the score model and the graph construction from the score, section 4 introduces the corpora, and section 5 presents the proposed learning algorithm. Section 6 presents a series of three experiments and also takes a qualitative look at some examples. Finally, section 7 summarizes and concludes.

2. RELATED WORK

Graphs have emerged as a natural representation of music since the development of Tonnetz by Euler. Since then, there have been various proposals to use graph representations for addressing music analysis and MIR tasks. For instance (to name just two), [6] introduced relational *Klumpenhouwer networks* for music analysis, and [7] used *Tonnetz trajectories* for composer classification. One can distinguish between *heterogeneous* and *homogeneous* graphs [8]. Heterogeneous graphs may have multiple types of edges and nodes, while homogeneous graphs are simpler, containing only a single edge and node type. Recently, the creators of VirtuosoNet, a computational model for generating piano performances, used a heterogeneous graph representation of the score and trained their system using a Graph Neural Network [9]. However, in later publications, they reverted to a model without using graphs which achieved better performance [10]. In the present paper, we wish to show that a simple, homogeneous graph representation can form a natural and general basis for modeling a non-trivial music analysis task.

Automatic *cadence detection* is a challenging task. Although cadences are well established concepts, their definition or annotation in music can cause disagreements among musicologists. Previous work on automatic cadence detection has been done by [11] on Bach fugues and by [12] for a generalized classical music analysis system. A feature-based approach using standard Machine Learning classifiers is presented in [4] which represents the current state of the art. Recently, Sears and Widmer [13] highlighted the difficulty of detecting textbook voice leading schemata that occur near cadences in written music. However, to our knowledge, there exists no method employing deep learning models to solve the task.

3. MODELING SCORES AS A GRAPH

We model a score as a graph with individual notes and rests as nodes and simple temporal relations as edges. In addition, each graph node is associated with a vector of feature values that represent some basic properties of a note and its immediate context. Formally, let $G = (V, E)$ be a graph, where V is the set of nodes and $E \subseteq V \times V$ the set of edges and let A be the adjacency matrix of G. Each note and each rest in a score are represented as a node in the graph. We create three types of undirected connections between notes/rests: edges E_{on} between notes that occur on the same onset; edges E_{cons} between consecutive notes, and edges E_{dur} between a note of longer duration and notes whose onsets occur during this time:

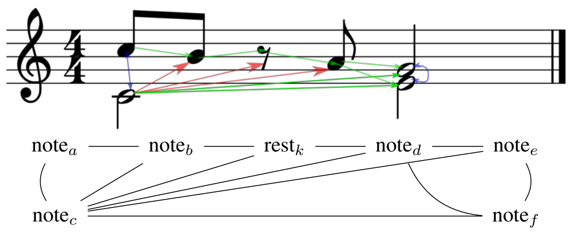

Figure 1. Example graph creation from a score following the process described in the text. E_{on} is denoted in blue, E_{cons} in green, and E_{dur} in red. Global attributes such as time and key signatures are added as node features.

$$\begin{aligned}
E_{on} &= \{(i,j) \mid on(n_i) = on(n_j)\} \\
E_{cons} &= \{(i,j) \mid on(n_i) + dur(n_i) = on(n_j)\} \\
E_{dur} &= \{[(i,j) \mid on(n_i) + dur(n_i) > on(n_j)] \wedge \\
&\qquad [on(n_i) < on(n_j)]\} \\
E &= E_{on} \cup E_{cons} \cup E_{dur}
\end{aligned}$$

where n_i is the i^{th} note. on denotes the onset of a note, dur the duration. All edges in E are undirected.

3.1 Feature Overview

We use three types of features to further describe a note: [1] general-purpose note-level features to describe a note and its immediate rhythmic/melodic context; general graph topology features to capture aspects of local connectivity; and cadence-specific note features inspired by [4]. The third feature category is the only one that is designed with the specific classification target in mind; however, in contrast to [4], we restrict these to only consider the immediate local context of a note instead of using positional features relating to predefined past "cadence anchor points". In this way, we wish to demonstrate the generality of our representation and learning approach, which will hopefully learn more long-distance aspects automatically, as needed.

The first category, *general note-wise features*, is the largest one. For each note in the score, we extract onset time expressed in score-relative beats, duration in beats, and MIDI pitch, using the partitura package [14]. Furthermore, we translate global attributes such as time signature and assign them to each note. Also using partitura, we extract a set of generic note-wise features as defined in [15]. Finally, we extract features summarizing intervallic information at the time of onset of each note. These include *interval vectors* [16] and binary features activated when intervallic content is identical to the interval set corresponding to particular chord types, i.e., major, minor, diminished, etc.

Second, we add *graph-aware features* using the first 20 eigenvectors from the Laplacian of the adjacency matrix [17].

The final category contains *note-wise cadence-related features* similar to those in [4], such as voice leading information and voicing. However, our features are calculated at the note level only, considering the time of onset for each note and its immediate neighbors, such as adjacent past onsets or simultaneous onsets. In particular, we do not use any information about events that occur on previous beats. While these features are more restricted compared to [4] they are also more general, since we make no assumptions on and reference to "cadence anchor points" (e.g.,

[1] Code and a complete specification of all features is available on `https://github.com/manoskary/cadet`.