We categorize the extracted drum samples into 6 categories which are {kick, snare, hi-hat, ride, crash, toms}. The assignment is based on the filenames of the stems the samples originate from, using keyword dictionaries of typical drum-type expressions (like "hat" for hi-hat or "BD" for kick drum). We split the dataset in train / eval / test set with proportions $0.85/0.05/0.1$ (on a track level, meaning that samples of one track do not spread over different sets). All reported results are computed on the test set.

5. TRAINING

As an encoder, we use the EfficientNet-B4 [29], where we start training from weights that have been pre-trained on the ImageNet dataset. Even though ImageNet is far from the audio spectrum domain, it turned out that using ImageNet weights is crucial in our experiments (see Section 7.1, and it has also been shown in a previous study that cross-domain pre-training can be beneficial [30]).

The 1000 output units are reduced to 256 with a single linear layer. The EfficientNet expects a 2D input, which we provide by converting the audio signals into mel-scaled spectrograms with an STFT window length of 2048, a hop length of 512, and 128 resulting mel bins, considering the whole frequency range (fmax $=$ 22050). Also, we log-scale the values of the resulting mel spectrograms. We input the resulting spectrograms in each of the three RGB input channels of the EfficientNet (after normalization to the ImageNet color statistics).

A positive pair consists of an audio query and a corresponding drum sample. Given a drum sample, we select the stems of the corresponding song and remove the stem from which the drum sample was extracted. Then, we randomly choose at least 2, at most all of the remaining stems (the number of stems is uniformly sampled) and mix them on the fly during training (note that the mel spectrogram transformation is part of the model architecture using the *nnAudio* library).[5] All audio inputs are of length 4 seconds. If a drum sample is shorter than that, it is zero-padded. To obtain a query, we are cutting a 4-second-long snippet from a random position of the mixture.

The encoders are trained by the ADAM optimizer, with a batch size of 190, a learning rate of 3e-4, and a weight decay factor of 3e-5. The temperature parameter τ of the NT-Xent loss is set to 0.2. The factors for the variance and covariance regularization terms γ and δ are set to 1, and the norm regularization factor η is set to 10.

We use some data augmentation on both the queries and the drum samples. First, Gaussian noise is added (up to -12 dB). Furthermore, we perform time-stretch with a ratio between 0.9 and 1.15 of the original tempo. Another augmentation is reducing the gain down to a minimum attenuation of -6 dB, and we perform a time shift of up to 800ms to the right and 200ms to the left (to make the models invariant to the exact onset position of one-shot audios). All these augmentations occur with a probability of 50% for each instance.

[5] `https://github.com/KinWaiCheuk/nnAudio`

6. EXPERIMENTS

In this section, we introduce the conducted experiments to validate our method including objective metrics (Section 6.1), the user study (Section 6.2), and we describe the method of the correlation analysis (Section 6.3).

6.1 Objective Evaluation

As the actual goal of the study is to sort the candidate samples given an audio query $\mathbf{q}_i$, the primary evaluation metric is the rank$_i$ of the ground-truth samples for a given query (i.e., the samples that originate from the same song as the query). For that, we sort all drum samples in descending order according to the cosine similarity to a given encoded query and determine how early in the list a ground-truth sample is located. We also divide the rank$_i$ by N list entries to obtain normalized ranks $\in (0, 1]$. As we can extract several queries (4-second-long random excerpts) for each song in the test set, and as there are several drum samples assigned to each song, we perform this test for every song in the test set 50 times (resulting in $|Q| = 27$k evaluations) and average the resulting ranks. The explanation above results in the Mean Normalized Rank summarized as

$$R_{\text{mn}} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{\text{rank}_i}{N}. \tag{8}$$

In addition, we also report the *Median* Normalized Rank R_{md}, as this is an indicator of how well the model performs *most of the time*, ignoring possible outliers.

Besides also reporting the contrastive loss $\mathcal{X}$ (see Equation 1), we also evaluate how well the model has learned to cluster electronic and acoustic queries (i.e., mixtures) and keys (i.e., drum samples). To that end, we perform a binary (acoustic/electronic) k-nn classification (with $k = 50$) on the encodings separately for both the queries and the keys. From this classification task, we report the likelihood (L_q and L_k, respectively) of the data under the predictions.

In order to better understand the influence of different design choices, we perform ablation studies and report the metrics for each model and training variants. The different ablation scenarios are described in the following.

- `2Enc` denotes our proposed setup with 2 separate encoders instead of a shared encoder for both, queries and samples.
- `PTrain` denotes a variant that starts training from an EfficientNet that was pre-trained on ImageNet, instead of from randomly initialized weights.
- In `Aug`, data augmentation is used.
- `VCReg`, is using the variance and co-variance regularization terms (see Equations 2 and 4).
- In `SMix`, we randomly mix different numbers n of random stems to form a query (where $n > 1$). If `SMix` is not used, we always mix all stems of a song.

- For `QSInv`, in addition to sampling positive pairs from queries and corresponding samples, we also include as positive pairs two queries and two samples originating from the same song.

6.2 User Study

To evaluate the quality of the scoring function, we perform a user study in which we ask 10 experts (with musical education or concerned with music production) to rate the quality of selected drum samples given a musical context. More precisely, we test their preference between drum samples that scored well according to our system and samples that are randomly picked from the dataset. The possible choices are to select one of the two proposed mixtures, or "equal" (if there is no preference for one of the choices), or "skip" (if the samples to be rated are not of the expected class or any other problem occurred). Every participant obtains 12 single comparisons (presenting each of the 6 percussion types twice) and provides 12 ratings accordingly. After that, they can decide if they want to enter a further session (to provide 12 further ratings). Participants are asked not to perform more than 4 sessions to obtain a balanced result.

The procedure to obtain one comparison presented to a participant for rating is as follows. We pick a percussion type (e.g., snare) and a song from the test set that contains such a type as a single stem. We remove the stem containing that type, mix the remaining stems and pick ten 4-second-long excerpts from randomly sampled positions of the resulting mixture. All these excerpts are then fed through the query encoder, and the mean is taken from the resulting encodings. The resulting mean vector acts as the query encoding for the current song. Using this query, the cosine similarities to the latent representations of all 63k drum samples in the data set (obtained as described in Section 6) are computed, and the ten samples with the *highest similarities* are selected. Furthermore, ten drum samples of the corresponding percussion type are *randomly picked* from the data set. The thereby obtained samples are used to generate 20 new versions of the current song (a version for each sample).

To create a new version of a song, we perform an onset detection on the previously removed stem (containing the percussion type in question), position the selected samples in the estimated positions, and mix the resulting stem with the remaining stems (omitting the original stem of the percussion type in question). That way, we replace the original drum track with a track containing the drum sample to be evaluated. For a single user rating, we then contrast the mixtures of a *randomly picked* and a *highly rated* percussion sample and have participants indicate their preference between the two.

6.3 Correlation Analysis

In order to gain some insights into the latent space learned by the encoders, we perform a correlation analysis between audio features of neighboring data points. For that, different perceptual and spectral features are computed for

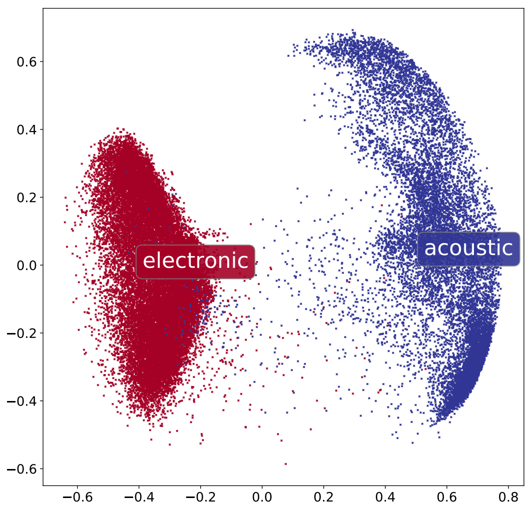

Figure 1: Principal Component Analysis (PCA) of drum sample encodings. Red dots indicate samples originating from electronic music and blue dots indicate samples originating from acoustic music.

the drum samples in the data set using the *Audio Commons timbre models*[6] for perceptual features (e.g., boominess, brightness, depth, hardness, roughness, warmth) and *librosa*[7] for spectral features (e.g., spectral centroid or spectral contrast). We also add an indicator if the respective drum sample originates from an electronic or acoustic song. In the former case, the "electronic" feature value is set to 1, in the latter case to 0.

The drum samples are mapped into the sample encoder's latent space, and k-means clustering is performed using $k = 24$. In each cluster, the mean of each audio feature is used as a specific observation of that audio feature variable. Then, the Pearson correlation coefficient is computed between all such variables (separately for every percussion type). As a result, we can derive statements like "whenever the loudness of a kick drum is high, the boominess of the snare tends to be low". Note that for computing the audio features, we normalize every sample to 0.5 seconds (i.e., cut or pad) so that the audio feature computation is not influenced by the length of the audio file (as some features are computed by averaging several spectrogram frames). To obtain the latent encodings used to compute the clusters, we use the regular 4-second-long samples.

7. RESULTS AND DISCUSSION

7.1 Objective Evaluation

In Table 1, we report the contrastive loss, the Mean and Median Normalized Rank, and the query and sample electronic/acoustic classification likelihood for different archi-

[6] `https://github.com/AudioCommons/ac-audio-extractor`
[7] `https://librosa.org/`

	Queries: full mixtures					Queries: sparse mixtures				
Variant	$\mathcal{X}$	R_{mn}	R_{md}	L_q	L_k	$\mathcal{X}$	R_{mn}	R_{md}	L_q	L_k
`2Enc+PTrain+Aug+VCReg+SMix`	3.614	**0.105**	**0.032**	0.9940	0.9767	**3.761**	**0.124**	0.043	**0.9905**	0.9763
`2Enc+PTrain+Aug+VCReg+SMix+QSInv`	3.718	0.120	0.037	0.9900	0.9782	3.818	0.136	0.047	0.9862	0.9768
`2Enc+PTrain+Aug+VCReg`	4.001	0.124	0.061	0.9825	0.9732	4.389	0.183	0.100	0.9635	0.9724
`2Enc+PTrain+VCReg+SMix`	3.742	0.128	0.037	0.9945	**0.9895**	3.780	0.137	**0.042**	0.9901	**0.9898**
`2Enc+PTrain+Aug+SMix`	**3.575**	0.116	**0.032**	**0.9946**	0.9774	3.812	0.135	0.043	0.9893	0.9790
`2Enc+Aug+VCReg+SMix`	5.235	0.458	0.432	0.7268	0.7629	5.237	0.470	0.451	0.7188	0.7480
`2Enc+Aug+VCReg+SMix+QSInv`	4.174	0.164	0.079	0.9826	0.9566	4.387	0.181	0.091	0.9768	0.9585
`PTrain+Aug+VCReg+SMix`	3.853	0.121	0.047	0.9812	0.9809	4.000	0.137	0.058	0.9768	0.9819
`2Enc+PTrain`	3.883	0.140	0.053	0.9925	0.9795	4.399	0.205	0.089	0.9821	0.9803

Table 1: Ablation study for different architectures and training scenarios tested on queries from full mixtures and queries from sparse mixtures (a sparse mixture is based on a random number n of stems, where $n > 1$). $\mathcal{X}$ denotes the contrastive loss and R_{mn} and R_{md} is the mean and median normalized rank, respectively, of ground truth samples. L_q is the likelihood of the binary k-nn (electronic/acoustic) classification task for query, and L_k for key encodings in the latent space.

tectures and training scenarios (as described in Section 6.1). For each run, we train for 600 epochs and pick the saved checkpoint of the epoch with the best performance (regarding the Mean Normalized Rank R_{mn}) on the evaluation set for evaluating the test set. We report the results when evaluating the models with queries that are drawn from mixtures where all stems of a song are used (denoted as "full mixtures") and mixtures with a random number n of stems (denoted as "sparse mixtures", where $n > 1$). The latter scenario is crucial in music production, where sample selection may occur at an intermediate state of the project, when several instruments are still missing.

The results show that our proposed architecture and training procedure (first row in Table 1) performs best in the mean and median rank metrics (R_{mn} and R_{mn}) for both query scenarios (note that the random guessing baseline is 0.5). It is somewhat surprising to us that `QSInv` (i.e., imposing additional invariance for queries/samples originating from the same song) does not improve but rather worsens the result (cf. row 2 of the table). Apparently, the relaxation of the space caused by not using this regularization helps to perform the main objective.

We can see for `SMix` that it is vital to mix different (numbers of) stems on the fly during training (cf. third row in Table 1 where this has not been done). `SMix` does not only help for the "sparse mixtures" scenario at test time (right part of the table) but also in the "full mixtures" scenario (left part of the table).

Using a pre-trained EfficientNet (`PTrain`) makes a considerable difference in the overall performance and training dynamics. In fact, when not using pre-trained weights, it is necessary to add `QSInv` in our experiments. Otherwise, the encoders do not learn at all (see row 6 in Table 1).

The last row of Table 1 presents the results of a scenario where all training optimizations (augmentation, variance, co-variance regularization, and sparse mixing) are turned off. It can be seen that the performance deteriorates substantially, particularly in the "sparse mixtures" test scenario.

Finally, the results of the classification likelihood (mostly around 99%) show that the models have implicitly learned to separate electronic and acoustic content (cf. Figure 1, showing a PCA of the drum sample encodings). Interestingly, this separation works consistently better if no data augmentation is used (see row 4 in Table 1). Electronic samples are generally "cleaner" than those extracted from acoustic music. Augmentation (e.g., adding noise) may remove some of these characteristics, making it harder to discriminate between the two classes.

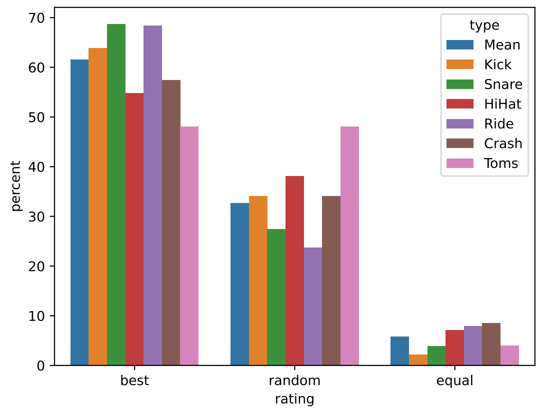

Figure 2: Preference ratings of participants in the user study, separated by percussion type (the blue bar "Mean" shows the mean of all ratings). "best" are mixtures with samples that *scored highest* by our method, and "random" denotes mixtures with *random samples* from the data set. An "equal" rating means no particular preference.

7.2 User Study

Figure 2 shows the results of the listening test (based on 300 ratings in total, omitting skipped ratings). On average ("Mean"), the samples that are ranked "best" by our method were preferred approximately twice as often as random samples (61.57% to 32.64%). It is also interesting that for most percussion types, the human preference for "best" is similar to or better than average (with Snare and Ride being the most salient), while human ratings disagree with our system's choices, particularly often for HiHat and Toms. A hypothesis why HiHats perform worse is that we merged both open and closed hi-hat in one group. When replacing the original hi-hats with the selected ones, there

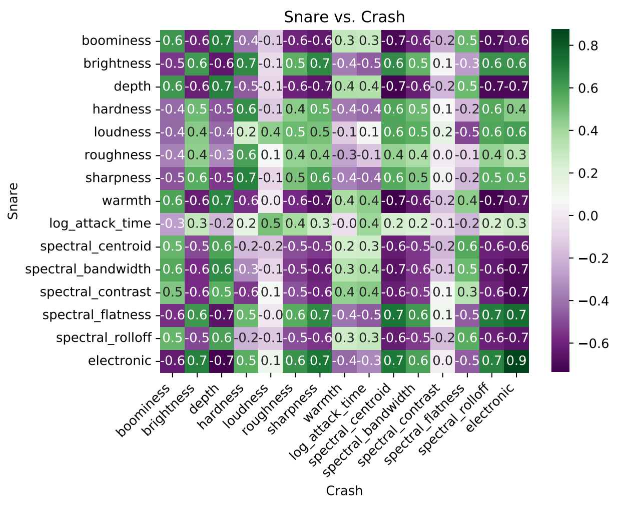

Figure 3: Correlations between perceptual and spectral features (and electronic / acoustic indicator) of Snare and Crash drum samples that are close in the latent space (i.e., scored to fit well in the same musical context).

is an equal chance that an open hi-hat is used for a rhythm that is meant to be instantiated with a closed hi-hat and vice versa. This is particularly problematic in electronic music, where closed hi-hats are often used with minimal temporal intervals. The reason why Toms do not work well is probably due to less available training data (many tracks do not have tom stems). Examples of how the user study was presented can be found on the accompaniment website. [8]

7.3 Correlation Analysis

Figure 3 shows the correlation coefficients between audio features of snare and crash drum samples over clusters in the latent space (as described in Section 6.3, note that the entirety of the percussion-type combinations is shown on the accompaniment website). [8]

To our knowledge, there is no theory about the aesthetic rules of drum sample selection. Therefore, this analysis is particularly informative. It is interesting to see that strong correlations exist at all between the audio features of snare and crash found in the same clusters of the latent space (as is true for most other percussion-type combinations). Consequently, the characteristics that make drum samples fit in the same musical contexts can (also) be explained by such lower-level features. Using such correlation coefficients makes it possible to understand how the learned space is organized. More importantly, it is also possible to derive rules and make the method explainable. For example, it is possible to extract statements like "if the snare of a song sounds warm, the crash should not sound bright" (because "warmth" of the snare and "brightness" of the crash are negatively correlated with a coefficient of -0.6).

When looking at the diagonal of the correlation matrix in Figure 3, we see that the perceptual features tend to be positively correlated (from "boominess" to "warmth"), while the spectral features tend to be negatively correlated (from "log attack time" to "spectral rolloff"). Possibly, as snare and crash have similar perceptual characteristics, they should not get into each other's way regarding their frequency ranges. Whenever the snare occupies the higher frequencies, the crash occupies the lower frequencies and vice versa (according to the "spectral centroid" entry of -0.6 in the diagonal).

Finally, the correlations with the "electronic" indicator are also informative. It shows that a snare (and partly a crash) in electronic music tends to be more intense than in acoustic music, with more brightness, loudness, and sharpness. At the same time (looking at the spectral centroid), in electronic music, a snare tends to occupy rather lower frequencies than in acoustic music, while the opposite is true for crash (where the spectral centroid is positively correlated with electronic snares - having a correlation coefficient of 0.7). The high correlation of "electronic" (0.9) in the diagonal shows that electronic snares and crashes tend to be in the same clusters of the latent space.

8. CONCLUSION AND FUTURE WORK

We introduced a method that automatically scores the fit of drum samples to a given musical context. As the method is thought to be used in a music production context, the audio queries used for training are based on "sparse mixtures", which allows scoring samples at different stages of the music production process. Results show that this form of data augmentation generally improves performance, also when queries are computed from complete mixtures at test time. Critically, the automatic system should agree with human judgment. Therefore, we performed a user study that tests this agreement. It could be shown that the drum samples that are highly rated by our system are also preferred by human experts approximately twice as often as randomly selected samples.

As the architecture and training procedure combine different ideas, we performed an ablation study, where it became clear that starting with a pre-trained model is crucial for a good final performance and that additional choices (like using variance and co-variance regularization, data augmentation, and on-the-fly sparse mixing) improve the results considerably. It was also shown that audio features of drum samples whose encodings are close in the latent space are highly correlated, and we demonstrated that the observed correlations can also be interpreted to derive rules.

Our proposed method is not limited to drum samples but can potentially be used to retrieve different kinds of musical material based on learned aesthetic principles. Also, in the future, we want to scale up the system by using bigger data sets and a more generic pre-processing strategy. Currently, single drum samples are extracted before training to obtain a controlled experiment setup and application scenario. Developing a system that uses stems directly would apply better to audio sources of any type and would therefore greatly increase the applicability of our method in music production.

[8] `https://sites.google.com/view/samplematch`

9. ACKNOWLEDGEMENTS

I want to thank the reviewers for their extraordinarily thorough suggestions to improve this work and my colleagues at Sony CSL that assisted me with technical expertise and emotional support.

10. REFERENCES

[1] E. Deruty, M. Grachten, S. Lattner, J. Nistal, and C. Aouameur, "On the development and practice of AI technology for contemporary popular music production," *Trans. Int. Soc. Music. Inf. Retr.*, vol. 5, no. 1, p. 35, 2022. [Online]. Available: https://doi.org/10.5334/tismir.100

[2] J. Nistal, C. Aouameur, I. Velarde, and S. Lattner, "Drumgan VST: A plugin for drum sound analysis/synthesis with autoencoding generative adversarial networks," in *Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA*, 2022. [Online]. Available: https://arxiv.org/pdf/2206.14723.pdf

[3] J. H. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *8th International Conference on Learning Representations, ICLR, Addis Ababa, Ethiopia, April 26-30*, 2020. [Online]. Available: https://openreview.net/forum?id=B1x1ma4tDr

[4] T. Bazin, G. Hadjeres, P. Esling, and M. Malt, "Spectrogram inpainting for interactive generation of instrument sounds," in *Joint Conference on AI Music Creativity*, 2020. [Online]. Available: https://arxiv.org/abs/2104.07519

[5] S. Lattner and M. Grachten, "High-level control of drum track generation using learned patterns of rhythmic interaction," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA, New Paltz, NY, USA, October 20-23*, 2019, pp. 35–39. [Online]. Available: https://doi.org/10.1109/WASPAA.2019.8937261

[6] G. Hadjeres and L. Crestel, "The piano inpainting application," *CoRR*, vol. abs/2107.05944, 2021. [Online]. Available: https://arxiv.org/abs/2107.05944

[7] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick, "Momentum contrast for unsupervised visual representation learning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, Seattle, WA, USA, June 13-19*, 2020, pp. 9726–9735. [Online]. Available: https://doi.org/10.1109/CVPR42600.2020.00975

[8] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning, ICML, 13-18 July*, 2020, pp. 1597–1607. [Online]. Available: http://proceedings.mlr.press/v119/chen20j.html

[9] J. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. Á. Pires, Z. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent - A new approach to self-supervised learning," in *Annual Conference on Neural Information Processing Systems, NeurIPS, December 6-12*, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/f3ada80d5c4ee70142b17b8192b2958e-Abstract.html

[10] A. Bardes, J. Ponce, and Y. LeCun, "VICReg: Variance-invariance-covariance regularization for self-supervised learning," *CoRR*, vol. abs/2105.04906, 2021. [Online]. Available: https://arxiv.org/abs/2105.04906

[11] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," in *Interspeech, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September*, 2019, pp. 3465–3469. [Online]. Available: https://doi.org/10.21437/Interspeech.2019-1873

[12] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Annual Conference on Neural Information Processing Systems, NeurIPS, December 6-12*, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/92d1e1eb1cd6f9fba3227870bb6d7f07-Abstract.html

[13] D. Jiang, W. Li, M. Cao, W. Zou, and X. Li, "Speech SimCLR: Combining contrastive and reconstruction objective for self-supervised speech representation learning," in *Interspeech, 22nd Annual Conference of the International Speech Communication Association, Brno, Czechia, 30 August - 3 September*, 2021, pp. 1544–1548. [Online]. Available: https://doi.org/10.21437/Interspeech.2021-391

[14] A. Saeed, D. Grangier, and N. Zeghidour, "Contrastive learning of general-purpose audio representations," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, Toronto, ON, Canada, June 6-11*, 2021, pp. 3875–3879. [Online]. Available: https://doi.org/10.1109/ICASSP39728.2021.9413528

[15] J. Spijkervet and J. A. Burgoyne, "Contrastive learning of musical representations," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR, Online, November 7-12*, 2021, pp. 673–681. [Online]. Available: https://archives.ismir.net/ismir2021/paper/000084.pdf

[16] M. Tagliasacchi, B. Gfeller, F. de Chaumont Quitry, and D. Roblek, "Pre-training audio representations with self-supervision," *IEEE Signal Process. Lett.*,

vol. 27, pp. 600–604, 2020. [Online]. Available: https://doi.org/10.1109/LSP.2020.2985586

[17] S. Srivastava, Y. Wang, A. Tjandra, A. Kumar, C. Liu, K. Singh, and Y. Saraf, "Conformer-based self-supervised learning for non-speech audio tasks," *CoRR*, vol. abs/2110.07313, 2021. [Online]. Available: https://arxiv.org/abs/2110.07313

[18] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "BYOL for audio: Self-supervised learning for general-purpose audio representation," in *International Joint Conference on Neural Networks, IJCNN, Shenzhen, China, July 18-22*, 2021, pp. 1–8. [Online]. Available: https://doi.org/10.1109/IJCNN52387.2021.9534474

[19] L. Wang, P. Luc, A. Recasens, J. Alayrac, and A. van den Oord, "Multimodal self-supervised learning of general audio representations," *CoRR*, vol. abs/2104.12807, 2021. [Online]. Available: https://arxiv.org/abs/2104.12807

[20] L. Wang and A. van den Oord, "Multi-format contrastive learning of audio representations," *CoRR*, vol. abs/2103.06508, 2021. [Online]. Available: https://arxiv.org/abs/2103.06508

[21] C. Yeh, C. Hong, Y. Hsu, T. Liu, Y. Chen, and Y. LeCun, "Decoupled contrastive learning," *CoRR*, vol. abs/2110.06848, 2021. [Online]. Available: https://arxiv.org/abs/2110.06848

[22] W. Kim and J. Nam, "Drum sample retrieval from mixed audio via a joint embedding space of mixed and single audio samples," in *Audio Engineering Society Convention 149*. Audio Engineering Society, 2020.

[23] A. Delgado, C. Saitis, E. Benetos, and M. B. Sandler, "Deep conditional representation learning for drum sample retrieval by vocalisation," *CoRR*, vol. abs/2204.04651, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2204.04651

[24] A. Mehrabi, K. Choi, S. Dixon, and M. B. Sandler, "Similarity measures for vocal-based drum sample retrieval using deep convolutional auto-encoders," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, Calgary, AB, Canada, April 15-20*, 2018, pp. 356–360. [Online]. Available: https://doi.org/10.1109/ICASSP.2018.8461566

[25] B. Chen, J. B. L. Smith, and Y. Yang, "Neural loop combiner: Neural network models for assessing the compatibility of loops," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR, Montreal, Canada, October 11-16*, 2020, pp. 424–431. [Online]. Available: http://archives.ismir.net/ismir2020/paper/000225.pdf

[26] J. Huang, J. Wang, J. B. L. Smith, X. Song, and Y. Wang, "Modeling the compatibility of stem tracks to generate music mashups," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, February 2-9*, 2021, pp. 187–195. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16092

[27] G. Bernardo and G. Bernardes, "Leveraging compatibility and diversity in computational music mashup creation," in *AM '21: Audio Mostly, Virtual Event / Trento, Italy, September 1-3*, 2021, pp. 248–255. [Online]. Available: https://doi.org/10.1145/3478384.3478424

[28] M. E. P. Davies, P. Hamel, K. Yoshii, and M. Goto, "AutoMashUpper: Automatic creation of multi-song music mashups," *IEEE ACM Trans. Audio Speech Lang. Process.*, vol. 22, no. 12, pp. 1726–1737, 2014. [Online]. Available: https://doi.org/10.1109/TASLP.2014.2347135

[29] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning, ICML, Long Beach, California, USA, 9-15 June*, 2019, pp. 6105–6114. [Online]. Available: http://proceedings.mlr.press/v97/tan19a.html

[30] A. Guzhov, F. Raue, J. Hees, and A. Dengel, "ESResNet: Environmental sound classification based on visual domain models," in *25th International Conference on Pattern Recognition, ICPR, Virtual Event / Milan, Italy, January 10-15*, 2020, pp. 4933–4940. [Online]. Available: https://doi.org/10.1109/ICPR48806.2021.9413035

A TRANSFORMER-BASED “SPELLCHECKER” FOR DETECTING ERRORS IN OMR OUTPUT

Timothy de Reuse **Ichiro Fujinaga**
Centre for Interdisciplinary Research in Music Media and Technology, McGill University
{timothy.dereuse, ichiro.fujinaga}@mcgill.ca

ABSTRACT

The outputs of Optical Music Recognition (OMR) systems require time-consuming human correction. Given that most of the errors induced by OMR processes appear "non-musical" to humans, we propose that the time to correct errors may be reduced by marking all symbols on a score that are musically unlikely, allowing the human to focus their attention accordingly. Using a dataset of Romantic string quartets, we train a variant of the Transformer network architecture on the task of classifying each symbol of an optically-recognized musical piece in symbolic format as correct or erroneous, based on whether a manual correction of the piece would require an insertion, deletion, or replacement of a symbol at that location. Since we have a limited amount of data with real OMR errors, we employ extensive data augmentation to add errors into training data in a way that mimics how OMR would modify the score. Our best-performing models achieve 99% recall and 50% precision on this error-detection task.

1. INTRODUCTION

An enormous amount of music scores exist only in the form of digital images, where its musical content is not machine-readable. Among the fields of research that that assist with large-scale processing of musical documents is Optical Music Recognition (OMR), which investigates how to transform images of music scores into symbolic music files (e.g., MusicXML files) that can be searched, played back, and edited. OMR has the potential to be a valuable tool for music archives and libraries, but it is not yet reliable enough; all methods require a human correction step after the OMR process, as the raw outputs of currently-available OMR algorithms contain too many errors to be acceptable for musicians or music researchers. This is an issue when using OMR on printed Western music notation, and is worse for other types of music notation, especially when training data is scarce. While custom interfaces exist for the purpose of facilitating human correction, such as the correction interface of MuRET [1], this is still a time-consuming task even for experts. While OMR speeds up symbolic encoding in many cases [2], the process of scanning documents, performing OMR, and correcting OMR is not always faster than that of transcribing pieces from scratch on complex, polyphonic scores [3].

We observe that OMR systems tend to introduce musically unlikely phenomena into scores. A typical example of the errors induced by the OMR process of PhotoScore [1] (a leading proprietary, commercial OMR software) is illustrated in Figure 1. In the OMR'ed score, consider the pattern of missing staccato articulations in measure 1, the eighth note misidentified as a sixteenth note in measure 2, the strangely placed grace note in measure 3, and the accidental misidentified as a grace note in measure 6. Also note how the slurs in each instrument of the OMR'ed score do not match up with each other, though the correct phrasing could probably be guessed from the melodic content alone. While not all of these phenomena are necessarily impossible in a string quartet, a human looking over the score would notice their presence.

Given the musical “incorrectness” of many of the errors introduced by OMR, we propose that a machine-learning algorithm could flag most OMR errors automatically given information on the conventions of the genre. This could reduce the correction time of a score; instead of checking every measure of OMR output against the corresponding measure in the score image, the user would only have to check those regions flagged by the algorithm as possibly erroneous. Even if the algorithm flags some non-erroneous regions of the score, it could potentially still exclude large portions of it from needing human review.

The reader may question why we aim for *detection* of errors as opposed to *correction* of errors outright; this could further reduce the time to correct OMR output even further, and there exists a large body of work on error correction for natural language that we could draw on. We posit, however, that any amount of correction that is not guaranteed to address nearly *all* errors in an OMR output would not significantly save time on the part of the human corrector, as they would still be required to manually review the result anyway. Current studies on correcting errors in polyphonic music (discussed in Section 2) do not attain high enough performance to meet this bar.

[1] /www.neuratron.com/photoscore.htm

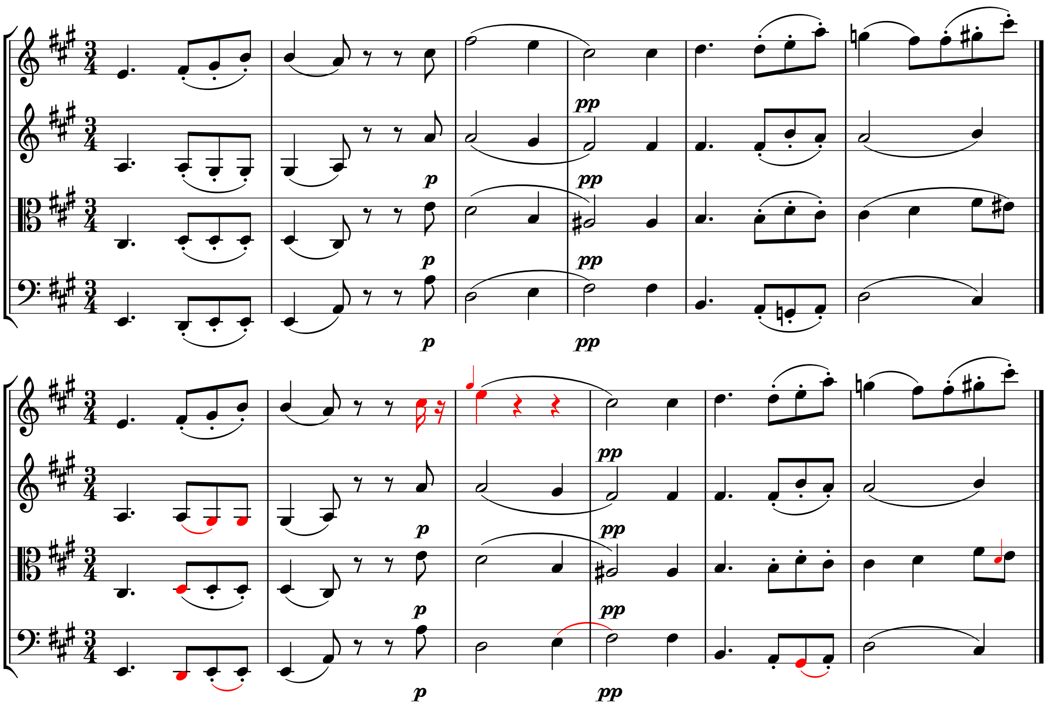

Figure 1: Two versions of Felix Mendelssohn's Quartet No. 2 in A Major, Op. 13, mm. 4-9. The upper system is an engraving of the correct score. The lower system is an output from PhotoScore's OMR Process. Some prominent errors are highlighted in red.

2. PREVIOUS WORK

There is little other research that seeks to detect or correct errors in symbolic music in the context of OMR. Most studies that discuss errors made by OMR systems seek to characterize them as a part of a method of evaluating their performance [4, 5]. As part of an end-to-end OMR system, Rossant and Blach [6] explicitly encode some musical rules (such as ensuring the number of beats in a measure matches the most recent time signature) that let them highlight areas that are likely incorrect, and also use confidence measures of detection steps from earlier in their OMR process to highlight areas that may have been detected incorrectly. More research focuses on error correction in music for other applications. McLeod et al. [7] focus on the task of analyzing errors in the output of Automatic Music Transcription (AMT) methods, introducing rule-based and machine leaning-based models to solve error detection and correction tasks, though these are intended as proof-of-concept baselines to define the tasks. Ycart et al. [8] compare a number of machine-learning models on the closely related task of Symbolic Music transduction, which involves processing the raw per-note probabilities output by an AMT model and producing a valid symbolic music file as output. Sidorov et al. [9] analyse music using formal grammars, taking the assumption that "correct" music tends to be highly compressible, and so any note whose presence alone significantly increases the amount of information in a piece is likely to be an error.

Significantly more literature exists in the fields of Grammar Error Detection (GED) and Grammar Error Correction (GEC) in natural language. The fields are far too broad to cover here; we direct readers to reviews on these subjects by Wang et al. [10], covering GEC, and Madi and Al-Khalifa [11], covering GED. Early models in both of these fields were rule-based, wherein every type of error had to be manually codified [12]. The current state of the art in these areas is in sequence-to-sequence deep-learning models similar to those used for machine translation, reaching human-level performance on common English-language GEC benchmarks [13]. To achieve this level of performance, however, a large amount of training data is required; there is much less research on GEC and GED in low-resource languages, the vast majority being on either English or Chinese [11]. Our task faces a similar issue; there exists orders of magnitude fewer training data for any one genre of music than there exists for the English language.

Data augmentation is commonly used in GEC and GED where data availability is low. Errors are added to correct examples and the models are then directed to correct those errors; we refer to errors created in this way as *synthetic errors* and errors created by the process of interest (e.g. errors induced by an OMR process) as *natural errors*. Early work on this process used simple statistics about distributions and types of errors to add errors semi-randomly [14, 15]. Later work, inspired by back-translation techniques from machine translation, trained models that *add* synthetic errors simultaneously with ones that *remove* them. Htut and Tetreault [16] evaluate a number of these models, finding that they provide significant improvements to training on natural errors when the dataset containing natural errors is small. Notably, they